Оглавление

Об авторе

Эдриан Прутяну (Adrian Pruteanu) – опытный консультант по вопросам безопасности и специалист, работающий в основном в области наступательной безопасности. За свою более чем десятилетнюю карьеру он создал бесчисленное количество упражнений для тренировок Красной команды и заданий, связанных с тестированием на проникновение и с оценкой безопасности приложений. Он регулярно работает с компаниями из списка «Fortune 500», помогая им защитить свои системы, выявляя уязвимости или проводя обратную разработку образцов вредоносного программного обеспечения. Эдриан является обладателем нескольких сертификатов, в том числе CISSP, OSCE, OSCP, GXPN, GREM, а также нескольких сертификатов от компании Microsoft. Будучи сертифицированным тренером Microsoft, он проводил в том числе индивидуальные занятия для различных клиентов в прошлом.

В свободное время Эдриану нравится разрабатывать новые инструменты и программное обеспечение, которые помогают проводить тестирование на проникновение или просто гарантируют безопасность пользователей в интернете. Иногда он получает награду за одну или две ошибки, и ему нравится исследовать и раскрывать уязвимости.

«Я хотел бы поблагодарить мою удивительную жену, чья поддержка и понимание помогли мне написать эту книгу. На исследование темы и создание книги уходит много времени, но ее постоянная поддержка давала мне стимул работать.

Также я хотел бы выразить отдельную благодарность семье и друзьям за их поддержку и наставничество. Благодарю и своих родителей за то, что они принесли домой компьютер Siemens и показали мне BASIC, тем самым у меня зародилась любовь к компьютерам в юном возрасте. Они всегда поддерживали мою одержимость технологиями, и за это я всегда буду им благодарен».

О рецензентах

Бабак Эсмаейли (Babak Esmaeili) работает в сфере кибербезопасности более 15 лет. Он начинал как реверс-инженер и продолжил карьеру в области тестирований на проникновение.

Специалист выполнил множество тестов на проникновение, он проводил большое количество консультаций для ИТ-инфраструктур. Поработав старшим тестировщиком в ряде компаний, Бабак приступил к исследованиям, касающимся сочетания стеганографии и криптографии затем к разработке программы с возможностью скрывать и шифровать бесконечные блокчейн-узлы разных файлов в один файл.

Кроме того, Бабак написал много статей о практическом тестировании на проникновение и о защите программного обеспечения. Сейчас он фрилансер, разрабатывает универсальную защищенную базу данных с новой технологией хранения цифровых данных, идея которой пришла к нему из его программного обеспечения. Бабак считает, что каждый должен разбираться в информационных технологиях, поскольку новый мир будет цифровым.

Он также советует всем узнавать как можно больше о том, как сохранить свои данные в этом новом цифровом мире.

«Я хочу поблагодарить всех, кто оказал помощь в написании этой книги, а также моих любимых родителей и дорогих друзей за их поддержку».

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте **www.dmkpress.com**, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу **dmkpress@gmail.com**, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу **http://dmkpress.com/authors/publish_book/** или напишите в издательство по адресу **dmkpress@gmail.com**.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу **dmkpress@gmail.com**, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Packt очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли применить санкции.

Пожалуйста, свяжитесь с нами по адресу электронной почты **dmkpress@gmail.com** со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую предоставлять вам качественные материалы.

Предисловие

Книга «*Как стать хакером*» научит вас, как подходить к тестированию на проникновение с точки зрения злоумышленника. Несмотря на то что тестирование производительности веб-приложений – это обычное дело, постоянно меняющийся ландшафт угроз делает тестирование защищенности гораздо более сложным.

Многие инструменты веб-приложений утверждают, что предоставляют полный обзор и защиту от угроз, но их необходимо анализировать в соответствии с потребностями безопасности каждого веб-приложения или каждой службы. Мы должны понимать, как злоумышленник подходит к веб-приложению и каковы последствия нарушения его защиты.

В первой части книги автор познакомит вас с часто встречающимися уязвимостями и способами их использования. В последней части книги только что освоенные методы применяются на практике, рассматриваются сценарии, в которых объектом атаки может стать популярная система управления контентом или контейнерное приложение и его сеть.

Книга «*Как стать хакером*» – четкое руководство по безопасности веб-приложений с точки зрения злоумышленника. В выигрыше останутся обе стороны.

Кому адресована эта книга

Читатель должен иметь базовый опыт в области безопасности. Например, хорошо бы он работал в сети или у него возникли проблемы с безопасностью во время разработки приложений. Формальное образование в области безопасности полезно, но не обязательно. Книга подходит тем, кто имеет не менее чем двухлетний опыт разработки, организации работы сети или DevOps, а также тем, кто интересуется вопросами информационной безопасности.

О чем идет речь в книге

Глава 1 «Атаки на веб-приложения. Введение» знакомит с инструментами, средами и минимальным набором правил ведения боя, которому мы должны следовать во время выполнения заданий. Рассмотрим также инструментарий специалиста, проведящего тестирования на проникновения, и исследуем облако в качестве нового инструмента тестировщика.

Глава 2 «Эффективное обнаружение» расскажет о повышении эффективности с точки зрения сбора информации о цели.

В главе 3 «Легкая мишень» разъясняется тот факт, что средствам защиты очень трудно постоянно обеспечивать безопасность, и многие простые уязвимости часто просачиваются сквозь бреши.

В главе 4 «Передовые атаки с использованием метода полного перебора» подробно рассматривается метод полного перебора, а также пара способов, которые позволяют оставаться незамеченным при проведении данного типа атак.

Глава 5 «Включение файлов» поможет изучить уязвимости, связанные с включением файлов. Рассмотрим также ряд методов использования базовой файловой системы приложения в своих интересах.

В главе 6 «Обнаружение и эксплуатация уязвимостей в приложениях с помощью внешних сервисов» рассматривается обнаружение уязвимости внеполосными (то есть по отклику на машине, не являющейся ни клиентом, ни сервером) методами, эксплуатация уязвимостей приложений и настройка инфраструктуры управления и контроля в облаке.

Глава 7 «Автоматизированное тестирование» помогает автоматизировать эксплуатацию уязвимостей, включая использование Collaborator от Burp, чтобы упростить внеполосное обнаружение.

В главе 8 «Вредоносная сериализация» подробно рассматриваются атаки, связанные с десериализацией. Рассмотрим данный тип уязвимости и практические эксплойты.

Глава 9 «Практические атаки на стороне клиента» содержит информацию, касающуюся атак на стороне клиента. Будут рассмотрены три типа межсайтового скриптинга: отраженный, хранимый и DOM-модель, а также межсайтовые подделки запроса и объединение атак в цепочку. В ней также рассказывается о правиле ограничения домена и о том, как оно влияет на загрузку стороннего контента или кода атаки на страницу.

Глава 10 «Практические атаки на стороне сервера» рассказывает, как атаковать сервер с помощью XML, а также использовать подделку запроса на стороне сервера для объединения атак в цепочку и дальнейшего проникновения в сеть.

Глава 11 «Атака на API» фокусирует наше внимание на API и на том, как эффективно тестировать и атаковать их. Вам пригодятся навыки, полученные к этому моменту.

Глава 12 «Атака на CMS» изучает уязвимости CMS.

Глава 13 «Взлом контейнеров» помогает понять, как безопасно настроить контейнеры Docker перед развертыванием на примере того, как компрометация CMS привела к еще одной уязвимости контейнера, из-за чего в конечном итоге случилась компрометация всего хоста.

Как извлечь максимум из книги?

1. Вы должны иметь базовые знания об операционных системах, в том числе о Windows и Linux. Мы будем активно использовать инструменты Linux и оболочку на протяжении всей книги, и было бы неплохо, если бы вы были знакомы с этой средой.

2. Опыт написания сценариев определенно сыграет вам на руку, но это не обязательно. В книге будут встречаться сценарии, написанные на языках Python, JavaScript и PHP.
3. Мы будем исследовать командно-контрольные серверы в облаке, поэтому настоятельно рекомендуем создать бесплатную учетную запись на одном из основных поставщиков, чтобы подготовиться к выполнению действий, описанных в книге.
4. Виртуальная машина или хост, работающий под управлением Kali либо выбранного вами дистрибутива для проведения тестирований на проникновение, помогут вам в полной мере опробовать некоторые сценарии, описанные в книге.
5. Мы регулярно скачиваем код из проектов с открытым исходным кодом на GitHub, и хотя основательные знания в области Git, безусловно, помогут, иметь их не обязательно.

Загрузка примеров

Вы можете скачать файлы с примерами для этой книги, используя свою учетную запись на сайте http://www.packt.com. Если вы приобрели издание в другом месте, то можете зайти на страницу http://www.packt.com/support и зарегистрироваться там, чтобы получить файлы по электронной почте.

Можно скачать файлы, выполнив следующие действия:

- войдите или зарегистрируйтесь на сайте http://www.packt.com;
- выберите вкладку **SUPPORT** (Поддержка);
- нажмите кнопку **Code Downloads & Errata** (Загрузки кода и опечатки);
- введите название книги в поле поиска и следуйте инструкциям на экране.

После скачивания файла убедитесь, что вы распаковали или извлекли содержимое папки, используя последнюю версию:

1. WinRAR / 7-Zip для Windows;
2. Zipeg / iZip / UnRarX для Mac;
3. 7-Zip / PeaZip для Linux.

Код, приведенный в книге, также размещен на GitHub: https://github.com/PacktPublishing/Becoming-The-Hacker. В случае обновления кода это отразится в имеющемся репозитории GitHub.

У нас также есть другие пакеты кода из нашего богатого каталога книг и видео, доступного на странице https://github.com/PacktPublishing/. Посмотрите их!

Загрузка цветных изображений

Кроме того, мы предоставляем PDF-файл с цветными изображениями скриншотов и диаграмм, используемых в книге. Вы можете скачать его на страни-

це https://www.packtpub.com/sites/default/files/downloads/9781788627962_ColorImages.pdf.

Условные обозначения

В книге используется ряд текстовых обозначений.

`КодВТексте`: указывает кодовые слова в тексте, имена таблиц базы данных, имена папок и файлов, расширения файлов, пути, фиктивные URL-адреса, ввод пользователя и маркеры Twitter. Например, это будет выглядеть так: «Смонтируйте загруженный файл образа диска `WebStorm-10*.dmg` в качестве еще одного диска в вашей системе».

Блок кода будет выглядеть следующим образом:

```
[default]
exten => s,1,Dial(Zap/1|30)
exten => s,2,Voicemail(u100)
exten => s,102,Voicemail(b100)
exten => i,1,Voicemail(s0)
```

Если мы хотим обратить внимание на определенную часть блока кода, соответствующие строки или элементы выделяются жирным шрифтом:

```
[default]
exten => s,1,Dial(Zap/1|30)
exten => s,2,Voicemail(u100)
exten => s,102,Voicemail(b100)
exten => i,1,Voicemail(s0)
```

Ввод или вывод командной строки записывается так:

```
# cp /usr/src/asterisk-addons/configs/cdr_mysql.conf.sample
     /etc/asterisk/cdr_mysql.conf
```

Жирным шрифтом выделяется новый термин, важное слово или слова, которые вы видите на экране, например в меню или диалоговых окнах. Вы увидите, к примеру, следующее: «Выберите надпись **Системная информация** на панели **Администрирование**».

Так будут оформляться предупреждения и важные примечания.

Так будут оформляться советы и рекомендации.

Глава 1

Атаки на веб-приложения. Введение

Веб-приложения можно встретить повсюду. Они являются частью структуры общества, и мы зависим от них во многих аспектах нашей жизни. В настоящее время их легко разрабатывать, их можно быстро развертывать, они доступны для любого, у кого есть интернет.

Технология, созданная, чтобы помочь при разработке и развертывании веб-приложений, также существенно улучшилась. Новые фреймворки, совершенствующие функциональность и удобство использования, выпускаются ежедневно. Компании передали полномочия разработчикам, и последние стали более гибкими, быстрее создают веб-приложения.

На рисунке дано представление о наиболее популярных фреймворках и средах разработки, которые покорили мир разработки приложений. Node.js перенес язык сценариев клиента браузера JavaScript на серверную сторону в комплекте с обширной библиотекой модулей, помогающих в быстрой разработке приложений. JavaScript, ранее редко использовавшийся язык сценариев для браузера, получает дополнительную производительность на стороне клиента с помощью фреймворков **React** и **Angular** и даже доступен для кросс-платформенной разработки с использованием **Electron** и **Chromium**.

Рис. 1.1. Мир изменился с тех пор, как Netscape правил в сети, и на этом рисунке демонстрируются технологии, доминирующие в интернете

GitHub стал единым пространством для библиотек с открытым исходным кодом, приложений и всего того, чем разработчик захочет поделиться с ми-

ром. Любой может загрузить туда все, что захочет, все могут принять участие в развитии GitHub: вносить изменения в код или поддерживать умирающую кодовую базу, создавать новые ветки проекта и продолжать разработку на локальном уровне. Разумеется, не только GitHub имеет подобные возможности. Существуют аналогичные репозитории для модулей Node.js, Python и PHP.

В центре внимания разработчика всегда находится поставка продукта, будь то простая реализация свойства во внутреннем веб-приложении, используемом отделом маркетинга, или самый передовой и лучший интерфейс для интернет-банкинга. Инфраструктура, необходимая для поддержки подобного рода приложений, также развивается, и разработчики изо всех сил пытаются интегрировать безопасность в свой рабочий процесс. Однако ошибки в безопасности не всегда происходят от невежества. Чаще всего в этом виноваты временны́е ограничения.

Цель книги – показать, как злоумышленники видят веб-приложения и как используют слабые места в коде приложений и инфраструктуре. Мы рассмотрим все распространенные ошибки, допускаемые в процессе разработки, которые используются для получения нужного доступа, а также практические атаки, извлекая максимум информации при работе с распространенными уязвимостями.

Итак, мы сделали некоторые предположения относительно вашего уровня знаний. Чтобы извлечь максимальную пользу из книги, необходимо иметь базовые знания о безопасности приложений. Читатели не обязательно должны быть экспертами в области тестирования на проникновение или безопасности приложений, но они должны иметь представление о том, что такое **межсайтовый скриптинг** (XSS) или **SQL-инъекции**. Мы не будем посвящать отдельную главу стандартному примеру Hello World для XSS, а покажем влияние эксплуатации данной уязвимости. Читатель также должен быть знаком с командной строкой Linux и распространенными консольными инструментами, такими как curl, git и wget. Знакомство с программированием, безусловно, будет полезно, но это не является жестким требованием.

В первой главе рассматриваются следующие темы:

- типичные правила применения оружия при проведении теста;
- инструментарий тестировщика;
- атака на прокси;
- как облако помогает при выполнений заданий.

Правила применения оружия

Прежде чем двигаться дальше, вспомним о правилах применения оружия (англ. *rules of engagement*) при проведении атаки. Это терминология, принятая в вооруженных силах. Правила обычно записаны в техническом задании, и все тестировщики должны их соблюдать. Они обрисовывают в общих чертах ожи-

дания тестировщика и устанавливают ряд ограничений на действия, которые можно совершать во время работы.

Хотя цель типичного теста на проникновение состоит в том, чтобы смоделировать реальную атаку и найти слабые места в инфраструктуре или приложении, существует множество ограничений, и на то есть веские причины. Нельзя стрелять из огнестрельного оружия и наносить больше урона, чем наносит настоящий противник. Цель (клиент), будь то третья сторона или внутренняя группа, не должна испытывать неудобств, позволяя профессиональным хакерам справляться со своими задачами.

Обмен данными

Хороший обмен данными – ключ успешного взаимодействия. Установочные встречи и собрания, посвященные закрытию проекта, чрезвычайно важны для обеих сторон. Клиент должен быть хорошо осведомлен относительно того, кто выполняет упражнение, как с ними связаться или как сделать резервную копию на случай чрезвычайной ситуации.

Установочная встреча – это шанс обсудить все аспекты теста, в том числе обзор масштаба проекта, критичности систем, всех предоставленных учетных данных и контактной информации. Если повезет, вся эта информация уже может быть включена в обзорный документ. Цель документа – четко определить, какие части инфраструктуры или приложений должны быть протестированы. Это может быть комбинация диапазонов IP-адресов, приложения, определенные домены или URL-адреса. Такой документ обычно пишется с учетом пожеланий клиента задолго до начала теста. Однако все может измениться, и установочная встреча – неплохая возможность рассмотреть все еще раз.

Вот несколько полезных вопросов, которые следует задать во время установочной встречи.

1. Изменилась ли область действия с момента последней редакции документа? Изменился ли список целей? Следует ли избегать определенных частей приложения или сети?
2. Существует ли окно тестирования, которого вы должны придерживаться?
3. Целевые приложения находятся в эксплуатации или в среде разработки? Они ориентированы на клиентов или предназначены только для внутреннего использования?
4. Контактные данные, которые можно использовать при возникновении экстренных ситуаций, по-прежнему актуальны?
5. Если были предоставлены учетные данные, они все еще актуальны? Сейчас самое время проверить это.
6. Существует ли брандмауэр приложения, который может препятствовать тестированию?

Как правило, целью является тестирование приложения, а не защита третьей стороны. У специалистов, выполняющих тестирование на проникновение, есть ограничение по времени, а у злоумышленников – нет.

При тестировании приложения на наличие уязвимостей рекомендуется попросить клиента внести в белый список IP-адреса в любых сторонних брандмауэрах веб-приложений. Эти брандмауэры проверяют трафик, достигающий защищенного приложения, и отбрасывают запросы, соответствующие известным сигнатурам или шаблонам атак. Некоторые клиенты предпочитают оставлять работать брандмауэры в принудительном режиме, поскольку их целью может быть симуляция реальной атаки. Именно в этот момент следует напомнить клиентам, что брандмауэры могут вносить задержки в оценку фактического приложения, так как тестировщику, возможно, придется потратить дополнительное время, пытаясь обойти средства защиты. Кроме того, поскольку для большинства заданий существует ограничение по времени, окончательный отчет может не совсем точно отражать состояние безопасности приложения.

Ни один менеджер не хочет слышать, что его критически важное приложение может отключиться во время теста, но такое иногда случается. Некоторые приложения не справятся с увеличенной рабочей нагрузкой обычного сканирования и выполнят аварийное переключение. Определенные полезные нагрузки также могут нарушить плохо спроектированные приложения или инфраструктуру и привести к остановке производительности.

Если во время теста приложение перестает отвечать на запросы, полезно позвонить основному контактному лицу и как можно скорее поставить его в известность о произошедшем, особенно если приложение представляет собой критически важную производственную систему. Если клиент недоступен по телефону, обязательно отправьте оповещение как минимум по электронной почте.

Собрания, посвященные закрытию проекта, или разбор полетов также очень важны. Особенно успешное тестирование с большим количеством найденных критических уязвимостей может обрадовать тестировщика, но огорчить клиента, поскольку ему придется отчитываться перед своим начальством. Настало время встретиться с клиентом, проанализировать каждую обнаруженную уязвимость и четко объяснить, как произошло нарушение безопасности и что

можно сделать для его устранения. Помните о людях и рассказывайте о проблемах на общедоступном языке, не назначая кого-либо виновным и никого не высмеивая.

Вопросы конфиденциальности данных

В ходе тестирования, в которое входят любые виды социальной инженерии или взаимодействия между людьми, такие как **фишинговые** упражнения, следует быть аккуратными. Во время фишинговой атаки пользователя обманом пытаются заставить перейти по ссылке в электронном письме на страницу похитителя учетных данных или открыть вредоносное вложение, и некоторые сотрудники чувствуют себя некомфортно, когда их вовлекают в подобные действия.

Например, перед отправкой фишинговых писем тестировщики должны удостовериться, что клиента устраивает, когда его сотрудники неосознанно участвуют в тестировании. Это должно быть зафиксировано в письменной форме, как правило, в техническом задании. Установочная встреча – подходящее место, чтобы синхронизировать свои действия с клиентом и его ожиданиями.

Если у вас нет явного письменного разрешения от клиента, избегайте следующего:

- не выполняйте атаки с применением методов социальной инженерии, которые считаются аморальными, например не собирайте данные о семье жертвы, чтобы побудить их нажать на ссылку;
- не используйте медицинские записи или конфиденциальные данные пользователя;
- не делайте скриншоты компьютеров пользователей;
- не показывайте учетные данные в личных электронных письмах пользователя, социальных сетях или других аккаунтах.

Некоторые веб-атаки, такие как внедрение SQL-кода или XML External Entity (XXE), могут привести к утечке данных. В этом случае нужно сообщить клиенту об уязвимости как можно скорее и уничтожить без возможности восстановления все, что уже загружено.

Хотя большинство тестов проводится в соответствии с соглашениями о соблюдении конфиденциальности, по возможности следует избегать обработки конфиденциальных данных. Нет особых причин хранить медицинские записи или данные кредитной карты после проведения тестирования. На самом деле, накапливая эти данные, клиент нарушает нормативные требования, кроме того, это незаконно. Такой тип данных обычно не предоставляет како-

го-либо рычага при попытке эксплуатации дополнительных приложений. При вводе доказательства в итоговый отчет необходимо проявлять особую осторожность, чтобы удостовериться, что из улик была удалена не подлежащая разглашению информация и что в них содержится только необходимый контекст для подтверждения результатов.

> *«Данные – это токсичный ресурс. Мы должны начать рассматривать их в таком виде и относиться к ним так же, как к любому другому источнику токсичности. Вести себя иначе – значит рисковать своей безопасностью и конфиденциальностью».*
>
> *Брюс Шнайер* (*Bruce Schneier*)

Предыдущая цитата, как правило, ориентирована на компании, занимающиеся сомнительной практикой, когда речь идет о личных данных пользователя, но она применима и к тестировщикам. В работе мы часто сталкиваемся с конфиденциальными данными.

Очистка

Успешный тест на проникновение или оценка приложения, несомненно, оставят после себя множество следов активности. Записи в журнале покажут, каким образом было осуществлено вторжение, а файл истории командной строки подскажет, как злоумышленник выполнял дальнейшее распространение по сети. Однако такие следы дают некоторое преимущество. Специалисты по защите, также именуемые Синей командой, могут анализировать действия во время или после тестирования и оценить эффективность защиты. Записи журнала предоставляют ценную информацию о том, как злоумышленник обошел защитные средства системы и выполнил код, чтобы проникнуть в сеть и похитить данные или иным образом взломать сеть.

Существует множество инструментов для очистки журналов после эксплуатации, но если клиент явно не разрешил эти действия, такой практики следует избегать. В некоторых случаях Синей команде может понадобиться проверить устойчивость своей инфраструктуры SIEM[1] (централизованная система сбора и анализа журналов), поэтому журналы можно очищать, но это должно быть явно разрешено в документах, касающихся задания.

Существуют определенные артефакты, которые почти всегда следует полностью удалять из систем или баз данных приложений после завершения задания. Артефакты, указанные ниже, могут подвергнуть клиента ненужному риску даже после исправления уязвимостей:

[1] SIEM (Security information and event management) – объединение двух терминов, обозначающих область применения программного обеспечения: SIM (Security information management) – управление информационной безопасностью и SEM (Security event management) – управление событиями безопасности. – *Прим. перев.*

- веб-оболочки, обеспечивающие доступ к **операционной системе**;
- дропперы, обратное подключение (reverse shell) и вредоносное программное обеспечение, используемое для повышения привилегий;
- вредоносные программы в виде Java-апплетов, развернутые с помощью сервера Tomcat;
- модифицированные или пораженные бэкдором компоненты приложения или системы:
    - пример: перезаписать двоичный файл пароля, воспользовавшись уязвимостью race condition, и не восстанавливать резервную копию перед выходом из системы;
- сохраненное вредоносное программное обеспечение, используемое для XSS-атак. Больше всего неудобств оно причиняет пользователям в производственных системах.

Не все вредоносные программы, с которыми вы имели дело во время теста, могут быть удалены тестировщиком. Для очистки требуется обратиться к клиенту.

Запишите все вредоносные файлы, пути и полезные нагрузки, используемые при тестировании. В конце попытайтесь удалить как можно больше элементов, приведенных в данном списке. Если что-то осталось, сообщите об этом основному контактному лицу, предоставив детали и подчеркнув важность удаления артефактов.

Пометка полезных нагрузок с помощью уникального ключевого слова поможет определить фиктивные данные во время очистки. Например, напишите следующее: «Пожалуйста, удалите из базы данных любые записи, содержащие ключевое слово 2017Q3TestXyZ123».

Хорошо, если клиент пришлет электронное письмо, подтверждающее, что он удалил все оставшиеся вредоносные программы или артефакты.

Инструментарий тестировщика

Профессионалы используют разные инструменты тестирования на проникновение. Инструментальные средства и методы тестирования ежедневно меняются, и нужно идти в ногу с изменениями. Несмотря на то что практически невозможно составить исчерпывающий список инструментов для каждого конкретного сценария, существуют проверенные программы, методы и среды, которые, несомненно, помогут злоумышленнику достичь своей цели.

Kali Linux

Kali Linux, ранее известный как BackTrack, уже на протяжении многих лет является популярным дистрибутивом Linux среди специалистов, проводящих тесты на проникновение. Его ценность трудно оспорить, поскольку в него входят почти все инструменты, необходимые для проверки приложений и сетей. Команда Kali Linux также регулярно обновляет не только операционную систему, но и средства атаки.

Kali Linux легко развернуть практически везде. Он поставляется во множестве форматов. Существуют 32-разрядные и 64-разрядные версии, пакеты переносимых виртуальных машин и даже версия, которая работает на операционной системе Android.

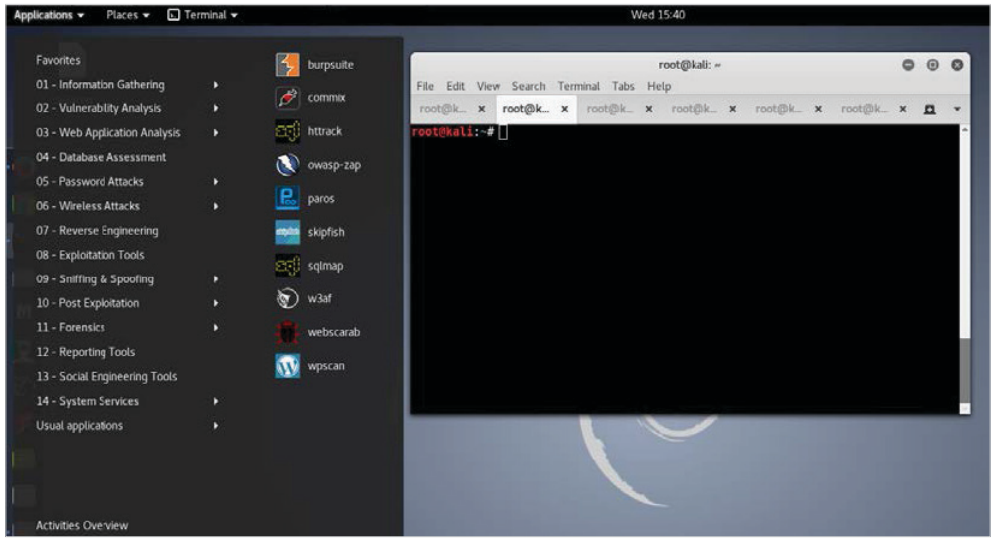

Рис. 1.2. Свежий экземпляр экрана Kali Linux

Альтернативы Kali Linux

Одной из альтернатив или дополнений для Kali Linux является **Penetration Testing Framework** (PTF) от команды TrustedSec. Он написан на Python и представляет собой модульный фреймворк, позволяющий превратить выбранную вами среду Linux в набор инструментов для тестирования на проникновение. На данный момент уже доступны сотни модулей PTF, и можно быстро создавать новые. PTF также можно запустить на Kali, чтобы оперативно собрать инструменты в одном месте.

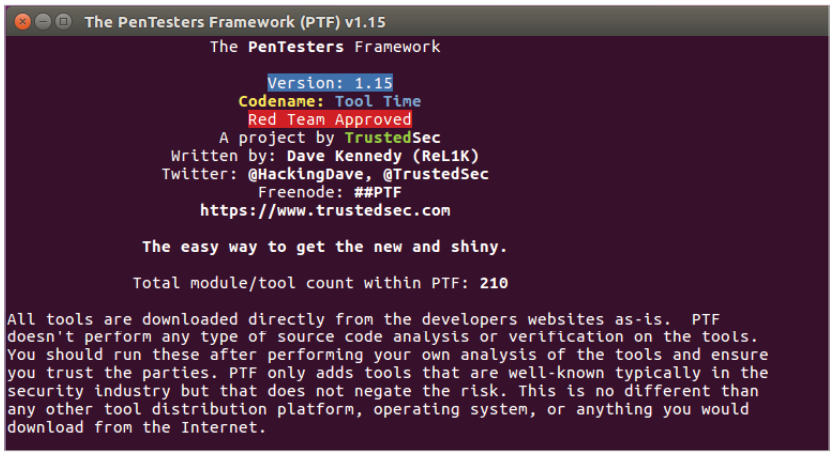

Рис. 1.3. Интерактивная консоль PTF

Еще одна хорошо зарекомендовавшая себя альтернатива Kali Linux – это **BlackArch**, дистрибутив на базе **Arch Linux**, в составе которого множество инструментов, входящих в состав и других дистрибутивов, используемых для тестирования на проникновение. В BlackArch есть много инструментов, с которыми знакомы тестировщики. BlackArch используется для тестирования сети или приложений, и он регулярно обновляется, как и Kali Linux. Для поклонников Arch Linux это приятная альтернатива дистрибутиву Kali на базе Debian.

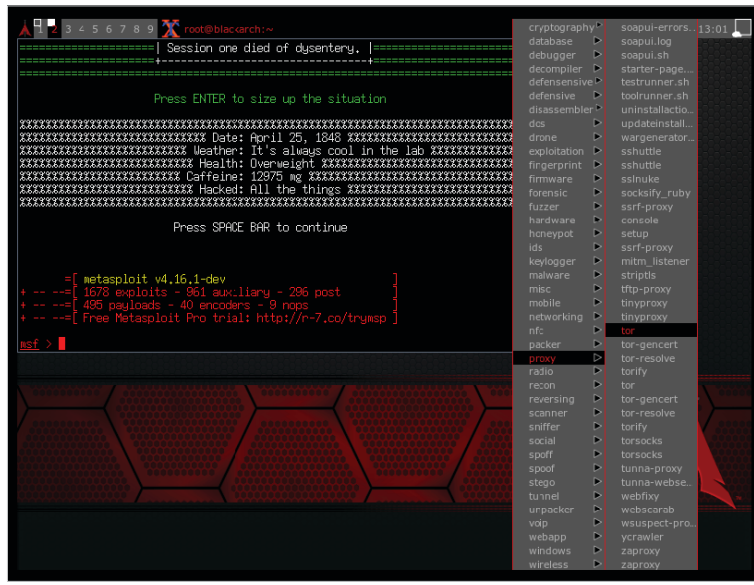

Рис. 1.4. Основной экран BlackArch

BlackArch доступен во многих форматах на странице https://blackarch.org.

Прокси-сервер

При тестировании приложений манипулирование и запись трафика неоценимы. Основные инструментальные средства, которые можно встретить на этом рынке, также можно расширить, что позволяет сообществу исследователей улучшать их функциональность с помощью бесплатных дополнений. Надежные и поддерживаемые прокси-серверы являются мощным оружием в арсенале злоумышленника.

Burp Suite

Burp Suite, пожалуй, можно назвать королем, когда дело доходит до атаки с применением прокси-сервера. Он позволяет перехватывать, изменять, воспроизводить и записывать трафик прямо из коробки. Burp Suite обладает широкими возможностями расширения, имея в арсенале мощные подключаемые модули, предоставляемые сообществом, которые интегрируются с **sqlmap** (де-факто средством эксплуатации SQLi), проводит автоматическое тестирование с целью повышения привилегий и предлагает другие полезные модули:

- **Proxy**: осуществляет запись, перехват и изменение запросов на лету;
- **Spider**: используется для автоматического сканирования веб-приложений;
- **Decoder**: быстро расшифровывает закодированные данные;
- **Intruder**: настраиваемый модуль, используемый для атак методом полного перебора;
- **Repeater**: разрешает воспроизведение любого ранее записанного запроса с возможностью изменения любой части самого запроса;
- **Scanner** (только в профессиональной версии): сканер уязвимостей, который интегрируется с Burp Collaborator, чтобы найти скрытые уязвимости;
- **Collaborator**: помогает в обнаружении скрытых уязвимостей, которые традиционные сканеры обычно пропускают.

Существует бесплатная версия Burp Suite, но профессиональная версия намного эффективнее, она стоит того, чтобы в нее вложиться. Бесплатная версия отлично подходит для проведения быстрых тестов, но у нее есть ограничения. Примечательно, что модуль Intruder ограничен по времени, что делает его бесполезным при больших нагрузках. Модуль Scanner также доступен только в профессиональной версии, но и его стоит приобрести. Scanner может быстро найти легкую добычу и даже автоматически использовать Collaborator для поиска уязвимостей внеполосных данных. Бесплатная версия по-прежнему может перехватывать, проверять и воспроизводить запросы, а также оповещать о любых уязвимостях, обнаруженных пассивно.

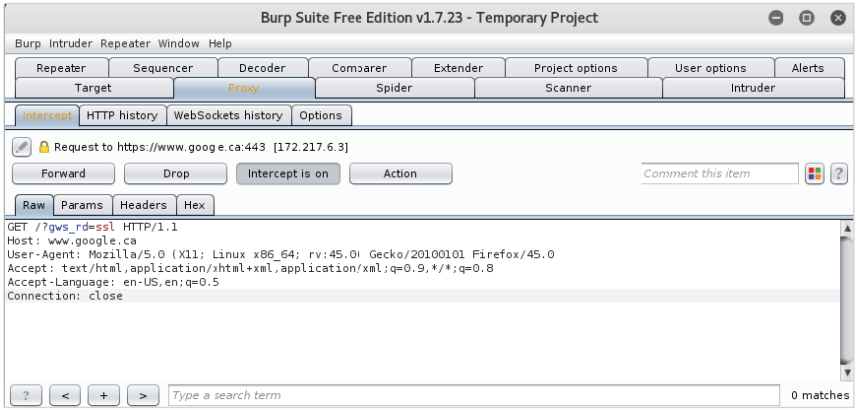

Рис. 1.5. Основной экран бесплатной версии Burp Suite

Zed Attack Proxy

Zed Attack Proxy (ZAP) от сообщества OWASP – еще одно замечательное средство для атаки на прокси-сервер. Он расширяем и прост в использовании, однако в нем отсутствуют некоторые функции, имеющиеся в Burp Suite. Например, ZAP не обладает обширными возможностями активного сканирования уязвимостей Burp Suite Pro, а также автоматической системы внеполосного обнаружения уязвимостей, какие есть у Collaborator.

Однако у ZAP нет ограничений по времени, и все его функции доступны из коробки. ZAP имеет открытый исходный код, и над ним активно работают сотни добровольцев.

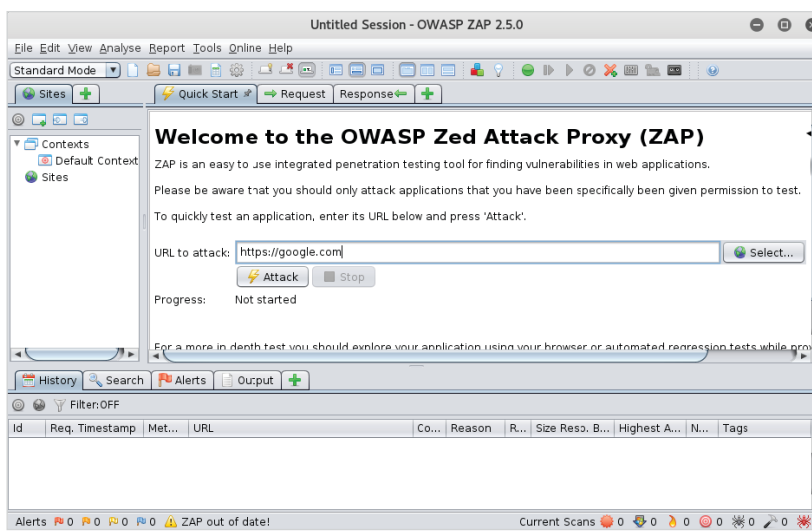

Рис. 1.6. Основной экран ZAP

Облачная инфраструктура

При тестировании злоумышленник обычно использует командно-контрольные серверы. Целью большинства таких серверов является передача команд вредоносным программам, работающим в скомпрометированной среде.

Злоумышленники могут велеть вредоносному программному обеспечению осуществлять перехват и кражу данных, запускать кейлоггеры, выполнять произвольные команды или шелл-код и многое другое. В последующих главах мы будем в первую очередь использовать облачный командно-контрольный сервер для кражи данных и внеполосного обнаружения уязвимостей.

Командно-контрольный сервер, доступный из любой точки мира, – универсальное средство в любом тестировании. Облако является идеальным местом для размещения его инфраструктуры, делает возможными быстрые и программируемые развертывания, доступные из любой точки мира. Некоторые облачные провайдеры даже поддерживают HTTPS, что позволяет быстро запустить сервер, не беспокоясь о приобретении доменов или сертификатов и об управлении ими.

Популярным выбором специалистов, проводящих тестирование на проникновение, является **Amazon Web Services** (AWS), лидер в облачном пространстве. Его услуги довольно недороги, есть и бесплатная версия.

Среди других приемлемых облачных провайдеров можно упомянуть следующие:

- Microsoft Azure: `https://portal.azure.com`;
- Google Cloud Platform: `https://cloud.google.com`;
- DigitalOcean: `https://www.digitalocean.com`;
- Linode: `https://www.linode.com`.

У Microsoft Azure есть уровень бесплатного использования по модели SaaS, который позволяет автоматически развертывать командно-контрольный сервер из репозитория GitHub. Он также обеспечивает поддержку HTTPS из коробки, облегчая сокрытие данных сервера от посторонних глаз, позволяя им сливаться с обычным пользовательским трафиком.

Всегда получайте разрешение (в письменном виде) от облачного провайдера, прежде чем приступать к тестированию с использованием его инфраструктуры, даже если это нечто совсем простое, как, например, размещение вредоносного файла JavaScript на временной виртуальной машине.

У провайдеров **облачных интернет-услуг** (ISP) должна быть форма, которую вы можете заполнить. В ней будет подробно описан предстоящий тест на

проникновение с использованием их инфраструктуры. Скорее всего, вам потребуется предоставить тестировочное окно и контактную информацию.

Независимо от того, что вы используете, – облако для размещения командно-контрольного сервера для тестирования или атаки на приложения, размещенные в облаке, всегда следует уведомлять клиента о действиях, связанных с тестированием на проникновение.

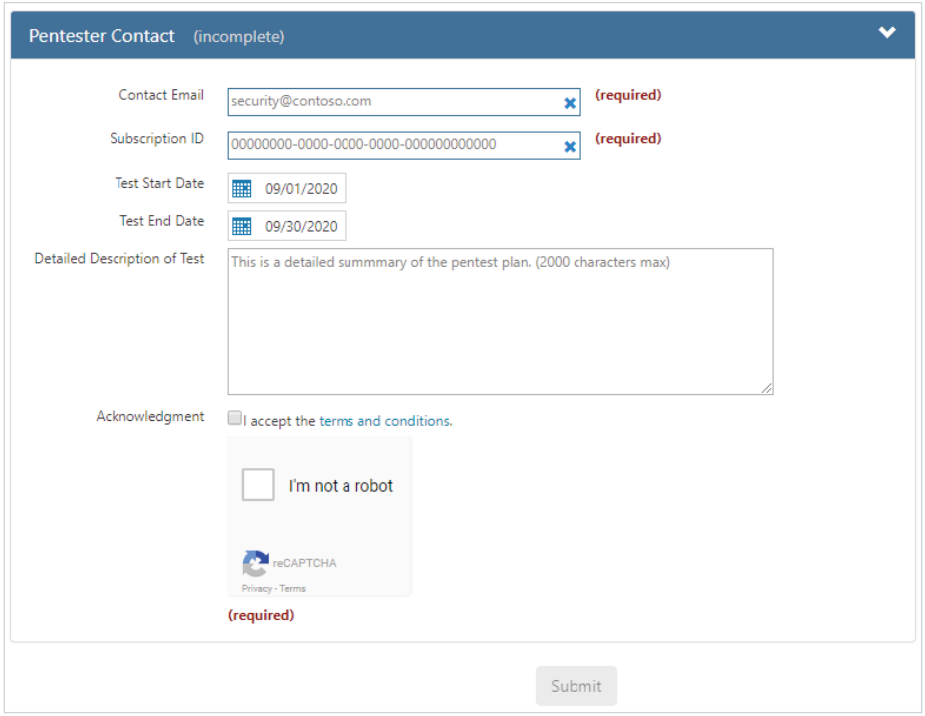

Рис. 1.7. Типичная форма уведомления о проведении теста на проникновение

Дополнительные источники

Воспользуйтесь указанными ресурсами для получения дополнительной информации об инструментах и методах тестирования на проникновение:

- **Penetration Testers Framework (PTF)**: https://github.com/trustedsec/ptf;
- **BlackArch**: https://blackarch.org;
- **Burp Suite**: https://portswigger.net/burp/;
- **OWASP ZAP**: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project;
- **Amazon Web Services**: https://aws.amazon.com;

- **Microsoft Azure**: https://portal.azure.com;
- **Google Cloud Platform**: https://cloud.google.com;
- **DigitalOcean**: https://www.digitalocean.com;
- **Linode**: https://www.linode.com.

Упражнения

Выполните упражнения, чтобы лучше ознакомиться с инструментарием хакера и инструментами, которые мы будем использовать в книге.

1. Скачайте и установите предпочитаемый дистрибутив для тестирования на проникновение: Kali, BlackArch – или поэкспериментируйте с PTF.
2. Используйте Burp Suite Free или ZAP для перехвата, проверки и модификации трафика на свой любимый сайт.
3. Создайте бесплатную учетную запись у выбранного вами поставщика облачных вычислений и примените уровень бесплатного использования для запуска экземпляра виртуальной машины Linux.

Резюме

В этой главе мы рассмотрели инструменты, среды и минимальный набор правил, которому вы должны следовать во время тестирования. Мы объяснили, насколько важен обмен сведениями и как важно учитывать конфиденциальность клиента при тестировании. Мы не плохие парни, и мы не можем действовать безнаказанно. Мы также рассмотрели процесс очистки. Очень важно не оставлять никаких артефактов, только если того не требует клиент. То, что осталось после нас, не должно стать причиной будущего взлома.

Мы также рассмотрели инструментарий, используемый специалистом, который проводит тестирование на проникновение: Kali, дистрибутив Linux «все-в-одном» и парочку его альтернатив. Возможно, более важной частью инструментария для взлома веб-приложения являются средства атаки на прокси-серверы, из которых мы выделили два: Burp Suite и ZAP. Наконец, мы упомянули облако как новый полезный инструмент для тестировщика веб-приложений.

Злоумышленнику всегда легче работать, чем защищающему. Это подтвердит любой профессиональный хакер с опытом работы в корпоративном мире. Злоумышленнику нужно всего одно слабое звено в цепочке (даже если оно является временным), чтобы полностью овладеть средой.

Когда речь идет о безопасности, трудно все сделать правильно с первого раза и еще сложнее поддерживать ее в исходном состоянии все время. Часто возникают проблемы: недостает ресурсов, знаний или неправильно расставлены приоритеты. Приложения должны быть пригодными для использования – до-

ступными и предоставлять полезные функции. Кажется, что времени, чтобы протестировать код правильно, всегда мало, не говоря уже о том, чтобы проверить его на наличие ошибок безопасности.

Из-за текучки кадров неопытные разработчики будут предоставлять недостаточно протестированный код. Сотрудники службы безопасности ежедневно сталкиваются с инцидентами, не говоря уже о том, что они беспокоятся по поводу проверок безопасного кода. Когда речь идет о тестировании, то тут нет универсального средства, а в бюджете нередко не хватает на это средств. В данной головоломке много частей и множество факторов, которые влияют на полностью защищенное приложение и базовую инфраструктуру.

Именно здесь может проявить себя профессиональный хакер, который в курсе этих ограничений. Имея доступ к командной строке на сервере, можно найти потенциальный эксплойт для повышения привилегий, попытаться заставить его работать и после ряда проб и ошибок получить полноценный доступ. В качестве альтернативы можно воспользоваться тем фактом, что обмен данными между серверами является распространенным требованием для системного администратора. Это означает, что либо соединения между серверами не защищены паролем, либо пароль хранится где-то на том же сервере. Нередко можно найти незащищенные закрытые ключи в глобально читаемых каталогах, что позволяет получить доступ ко всем остальным серверам в инфраструктуре. Закрытые ключи протокола **Secure Shell** (SSH), часто используемые при автоматизации SSH-соединений, не защищены паролем, поскольку защита секретного ключа паролем нарушит сценарий автоматизации, который его использует.

В последующих главах будем использовать эти прискорбные истины, касающиеся разработки и развертывания приложений, в своих интересах.