ТВЕРСКОЙ КОЛЛЕДЖ ИМ. А.Н. КОНЯЕВА

УЯЗВИМОСТИ ВЕБ-ПРИЛОЖЕНИЙ

г. Тверь
2022 год

УЯЗВИМОСТИ ВЕБ-ПРИЛОЖЕНИЙ

Уязвимости веб-приложений возникают тогда, когда разработчики добавляют небезопасный код в веб-приложение. Это может происходить как на этапе разработки, так и на этапе доработки или исправления найденных ранее уязвимостей. Недостатки часто классифицируются по степени критичности и их распространенности. Объективной и наиболее популярной классификацией уязвимостей считается OWASP Top 10. Рейтинг составляется специалистами OWASP Project и актуализируется каждые 3-4 года.

OWASP TOP 10

OWASP (Open Web Application Security Project) — это открытый некоммерческий фонд, онлайн-сообщество, которое занимается вопросами обеспечения безопасности веб-приложений, выпускает статьи на тему безопасности веб-приложений, а также документацию, различные инструменты и технологии.

OWASP Топ-10 — это отчет или информационный документ, в котором перечислены основные проблемы, связанные с безопасностью веб-приложений. Он регулярно обновляется, чтобы постоянно отображать 10 наиболее серьезных рисков, с которыми сталкиваются организации. Благодаря OWASP Топ-10 пользователи осведомлены о наиболее критичных рисках и угрозах, их последствиях и мерах противодействия. Обновляется список OWASP каждые три-четыре года

OWASP TOP 10

A1 Инъекции — Уязвимости, связанные с внедрением SQL, NoSQL, OS и LDAP. Возникают, когда непроверенные данные отправляются интерпретатору в составе команды или запроса. Вредоносные данные могут заставить интерпретатор выполнить непредусмотренные команды или обратиться к данным без прохождения соответствующей авторизации.

A2 Недостатки аутентификации — Функции приложений, связанные с аутентификацией и управлением сессиями, часто некорректно реализуются, позволяя злоумышленникам скомпрометировать пароли, ключи или сессионные токены, а также эксплуатировать другие ошибки реализации для временного или постоянного перехвата учетных записей пользователей.

A3 Разглашение конфиденциальных данных — Многие веб-приложения и API имеют плохую защиту критичных финансовых, медицинских или персональных данных. Злоумышленники могут похитить или изменить эти данные, а затем осуществить мошеннические действия с кредитными картами или персональными данными. Конфиденциальные данные требуют дополнительных мер защиты, например их шифрования при хранении или передаче, а также специальных мер предосторожности при работе с браузером.

A4 Внедрение внешних сущностей XML — Старые или плохо настроенные XML-процессоры обрабатывают ссылки на внешние сущности внутри документов. Эти сущности могут быть использованы для доступа к внутренним файлам через обработчики URI файлов, общие папки, сканирование портов, удаленное выполнения кода и отказ в обслуживании.

A5 Недостатки контроля доступа — Действия, разрешенные аутентифицированным пользователям, зачастую некорректно контролируются. Злоумышленники могут воспользоваться этими недостатками и получить несанкционированный доступ к учетным записям других пользователей или конфиденциальной информации, а также изменить пользовательские данные или права доступа.

A6 Некорректная настройка параметров безопасности — Некорректная настройка безопасности является распространенной ошибкой. Это происходит из-за использования стандартных параметров безопасности, неполной или специфичной настройки, открытого облачного хранения, некорректных HTTP-заголовков и подробных сообщений об ошибках, содержащих критичные данные. Все ОС, фреймворки, библиотеки и приложения должны быть не только настроены должным образом, но и своевременно корректироваться и обновляться.

A7 Межсайтовое выполнение сценариев (Межсайтовый скриптинг) — XSS имеет место, когда приложение добавляет непроверенные данные на новую вебстраницу без их соответствующей проверки или преобразования, или когда обновляет открытую страницу через API браузера, используя предоставленные пользователем данные, содержащие HTML- или JavaScript-код. С помощью XSS злоумышленники могут выполнять сценарии в браузере жертвы, позволяющие им перехватывать пользовательские сессии, подменять страницы сайта или перенаправлять пользователей на вредоносные сайты.

A8 Небезопасная десериализация (Insecure Deserialization) — Небезопасная десериализация часто приводит к удаленному выполнению кода. Ошибки десериализации, не приводящие к удаленному выполнению кода, могут быть использованы для атак с повторным воспроизведением, внедрением и повышением привилегий.

A9 Использование компонентов с известными уязвимостями — Компоненты, такие как библиотеки, фреймворки и программные модули, запускаются с привилегиями приложения. Эксплуатация уязвимого компонента может привести к потере данных или перехвату контроля над сервером. Использование приложениями и API компонентов с известными уязвимостями может нарушить защиту приложения и привести к серьезным последствиям.

A10 Недостатки журналирования и мониторинга — Недостатки журналирования и мониторинга, а также отсутствие или неэффективное использование системы реагирования на инциденты, позволяет злоумышленникам развить атаку, скрыть свое присутствие и проникнуть в другие системы, а также изменить, извлечь или уничтожить данные. Проникновение в систему обычно обнаруживают только через 200 дней и, как правило, сторонние исследователи, а не в рамках внутренних проверок или мониторинга.

Инъекции — Injections

Все данные, как правило, хранятся в специальных базах, обращения к которым строятся в виде запросов, чаще всего написанных на специальном языке запросов SQL (Structured Query Language – структурированный язык запросов).

Приложения используют SQL-запросы для того, чтобы получать, добавлять, изменять или удалять данные, например при редактировании пользователем своих личных данных или заполнении анкеты на сайте. При недостаточной проверке данных от пользователя, злоумышленник может внедрить в форму Web-интерфейса приложения специальный код, содержащий кусок SQL-запроса.

Такой вид атаки называется инъекция, в данном случае самый распространённый — SQL-инъекция. Это опаснейшая уязвимость, позволяющая злоумышленнику получить доступ к базе данных и возможность читать/изменять/удалять информацию, которая для него не предназначена.

Например, изменить вместе с именем и фамилией баланс своего счета, посмотреть баланс чужого счета, или же, похитить конфиденциальные личные данные.

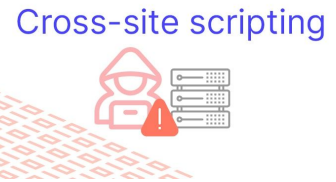
Эта уязвимость является следствием недостаточной проверки данных, поступающих от пользователя. Это позволяет злоумышленнику «подсунуть», например, в веб-формы, специально подготовленные запросы, которые «обманут» приложение и позволят прочитать или записать нелегитимные данные.

Недочеты системы аутентификации и хранения сессий (Broken Authentication and Session Management)

Для того, чтобы отличать одного пользователя от другого, web-приложение использует так называемые сессионные куки. После того, как Вы ввели логин и пароль и приложение вас авторизовало, в хранилище браузера сохраняется специальный идентификатор, который браузер в дальнейшем предъявляет серверу при каждом запросе страницы вашего web-приложения. Именно так web-приложение понимает, что Вы это именно Вы.

В случае, если ваш идентификатор украдет злоумышленник, а в системе не были реализованы проверки, скажем IP-адреса сессии, или проверки наличия более одного соединения в одной сессии, злоумышленник сможет получить доступ в систему с правами вашего аккаунта. А если это интернет-банк или кабинет платежной системы, о последствиях такого несанкционированного доступа Вы можете легко догадаться сами.

Межсайтовый скриптинг – XSS (Cross Site Scripting)



Межсайтовый скриптинг – еще одна ошибка валидации пользовательских данных, которая позволяет передать JavaScript код на исполнение в браузер пользователя. Атаки такого рода часто также называют HTML-инъекциями, ведь механизм их внедрения очень схож с SQL-инъекциями, но в отличие от последних, внедряемый код исполняется в браузере пользователя. Чем это чревато?

Во-первых, злоумышленник может украсть вашу сессионную cookie, последствия чего были описаны во втором пункте, буквально парой абзацев выше. Нужно отметить, что далеко не все серверы приложений уязвимы к данному типу атак.

Во-вторых, могут быть украдены данные, вводимые в формы на зараженной странице. А это могут быть конфиденциальные персональные данные, или, что еще хуже, данные кредитной карты вместе с CVC-кодом.

В третьих, через JavaScript можно изменять данные, расположенные на странице, например, там могут быть реквизиты для банковского перевода, которые злоумышленник с удовольствием подделает и заменит подставными.

Небезопасные прямые ссылки на объекты (Insecure Direct Object References)

Данный вид уязвимости является также следствием недостаточной проверки пользовательских данных. Суть ее заключается в том, что при выводе каких-либо конфиденциальных данных, например личных сообщений или учетных карточек клиентов, для доступа к объекту используется идентификатор, который передается в открытом виде в адресной строке браузера, И не реализована проверка прав доступа к объектам. Например, есть страница, которая отображает личное сообщение и она имеет адрес вида:

mysite.ru/read_message.jsp?id=123654

Перебирая число после "id=" можно будет читать чужие личные сообщения. Эксплуатация данной уязвимости очень проста и не требует вообще никаких специальных навыков – достаточно лишь перебирать число в адресной строке браузера и наслаждаться результатом. Как ни парадоксально, но этой детской болезни, порой были подвержены достаточно крупные европейские платежные системы.

Небезопасная конфигурация (Security Misconfiguration)

Безопасность Web-приложения требует наличия безопасной конфигурации всех компонентов инфраструктуры: компонентов приложения (таких как фреймворки – frameworks), веб-сервера, сервера баз данных и самой платформы. Настройки компонентов сервера по-умолчанию зачастую небезопасны и открывают возможности к атакам. Например, кража сессионной cookie через JavaScript при XSS-атаке становится возможна благодаря выключенной по умолчанию настройке cookie_http only.

При правильной настройке сервера и включенной опции cookie_httponly, получить сессионную cookie через JavaScript невозможно, но зачастую эта простая и важная настройка отсутствовала в таких критично важных местах, как личные кабинеты платежных систем.

Еще один пример детской уязвимости – использование настроек по умолчанию в серверах баз данных, таких как Redis, Memcached и других – закрытая служба может быть доступна на публичном IP-адресе сервера, и/или использовались пароли, установленные производителем по-умолчанию. Это позволяет злоумышленнику запросто читать и изменять данные, в числе которых, нередко бывают и сессионные cookies (чем это чревато – мы уже знаем) и выводимые пользователям в браузер данные (что позволяет еще и XSS-атаку применить).

Кроме того, программное обеспечение должно быть в актуальном состоянии: уязвимости находят каждый день в самых различных программных компонентах – операционной системе, web-серверах, серверах баз данных, почтовых серверах и т.д. И даже если ваше приложение правильно написано и тщательно проверяет все входящие данные, и вообще, хорошо защищено, это не означает что в один прекрасный момент не найдется уязвимость в вашей ОС или Web-сервере.

Незащищенность критичных данных (Sensitive Data Exposure)

Многие веб-приложения не защищают конфиденциальные данные, такие как кредитные карты и учетные данные для аутентификации. Злоумышленники могут украсть или модифицировать такие слабо защищенные данные для использования в своих корыстных целях.

Самый простой пример – передача конфиденциальных данных по протоколу HTTP. Такие данные должны передаваться исключительно по протоколу HTTPS, о чем должна гласить соответствующая надпись в адресной строке браузера:

Еще одна задача SSL-сертификата (а именно так называется специальный ключ, при помощи которого осуществляется проверка подлинности и шифрование в HTTPS) – подтвердить, что он выдан именно для данного сайта. В случае, если сертификат просрочен или подделан, вы должны увидеть соответствующее предупреждение.

Другой пример – отсутствие шифрования критичных данных, таких как пароли или номера кредитных карт. В случае, если данные зашифрованы, то даже в случае получения несанкционированного доступа на сервер, злоумышленник не сможет украсть критичные данные. К паролям, в частности, должна применяться необратимая хеш-функция – расшифровать шифрограмму при этом не возможно и проверка пароля происходит путем формирования шифрограммы введенного пароля и сравнения ее с имеющейся в базе.

Отсутствие функций контроля доступа (Missing Function Level Access Control)

Суть уязвимости, как следует из названия, заключается в отсутствии проверки наличия надлежащего доступа к запрашиваемому объекту.

Большинство веб-приложений проверяют права доступа, прежде чем отобразить данные в пользовательском интерфейсе. Тем не менее, приложения должны выполнять те же проверки контроля доступа на сервере при запросе любой функции. Ведь есть еще множество вспомогательных служебных запросов, которые, зачастую отправляются в фоновом режиме асинхронно, при помощи технологии AJAX.

Если параметры запроса не достаточно тщательно проверяются, злоумышленники смогут подделать запрос для доступа к данным без надлежащего разрешения.

Частный случай данной уязвимости – отсутствие проверки пользователя в личных сообщениях.

Межсайтовая подделка запроса (Cross-Site Request Forgery, CSRF/XSRF)

Вектор атаки CSRF, также известный как XSRF, позволяет злоумышленнику выполнять от имени жертвы действия на сервере, где не реализованы дополнительные проверки.

Например, в некоторой платежной системе для перевода средств на другой аккаунт, есть страница вида:

demobank.com/transfer_money.jsp?transfer_amount=1000&transfer_account=123456789

где transfer_amount – сумма для перевода и transfer_account – номер аккаунта, куда должны быть переведены средства.

Если жертва заходит на сайт, созданный злоумышленником, от её лица тайно отправляется запрос на вышеуказанную страницу платежной системы. Как результат – деньги уйдут на счет злоумышленника, после чего, вероятно, будут оперативно обменяны на Bitcoin или переведены в другую безвозвратную платежную систему, и получить их назад уже не получится.

Предполагается, что жертва должна была предварительно пройти аутентификацию в платежной системе и должна быть открыта активная сессия (скажем, страница платежной системы открыта в другой вкладке браузера).

Использование компонентов с известными уязвимостями (Using Components with Known Vulnerabilities)

Зачастую web-приложения написаны с использованием специальных библиотек или «фреймворков» (англ – framework), которые поставляются сторонними компаниями. В большинстве случаев эти компоненты имеют открытый исходный код, а это означает, что они есть не только у вас, но и у миллионов людей во всем мире, которые штудируют их исходный код, в том числе, и на предмет уязвимостей. И нужно отметить, что делают они это отнюдь не безуспешно.

Также уязвимости ищут (и находят) в более низкоуровневых компонентах системы, таких как сервер базы данных, web-сервер, и наконец, компоненты операционной системы вплоть до ее ядра.

Очень важно использовать последние версии компонентов и следить за появляющимися известными уязвимостями на сайтах типа securityfocus.com.

Непроверенные переадресации и пересылки (Unvalidated Redirects and Forwards)

Web-приложения зачастую переадресуют пользователя с одной страницы на другую. В процессе могут использоваться ненадлежащим образом проверяемые параметры с указанием страницы конечного назначения переадресации.

Без соответствующих проверок, атакующий может использовать такие страницы для переадресации жертвы на подложный сайт, который, к примеру, может иметь очень схожий или неотличимый интерфейс, но украдет ваши данные кредитной карты или другие критичные конфиденциальные данные.

Этот вид уязвимостей, также как и многие другие перечисленные выше, является разновидностью ошибок проверки входящих данных (input validation).

Язык SQL

SQL — это язык структурированных запросов (Structured Query Language), позволяющий хранить, манипулировать и извлекать данные из реляционных баз данных (далее — РБД, БД). SQL позволяет:

получать доступ к данным в системах управления РБД
описывать данные (их структуру)
определять данные в БД и управлять ими
взаимодействовать с другими языками через модули SQL, библиотеки и предварительные компиляторы
создавать и удалять БД и таблицы
создавать представления, хранимые процедуры (stored procedures) и функции в БД
устанавливать разрешения на доступ к таблицам, процедурам и представлениям

Язык SQL

SQL — это язык структурированных запросов (Structured Query Language), позволяющий хранить, манипулировать и извлекать данные из реляционных баз данных (далее — РБД, БД). SQL позволяет:

получать доступ к данным в системах управления РБД
описывать данные (их структуру)
определять данные в БД и управлять ими
взаимодействовать с другими языками через модули SQL, библиотеки и предварительные компиляторы
создавать и удалять БД и таблицы
создавать представления, хранимые процедуры (stored procedures) и функции в БД
устанавливать разрешения на доступ к таблицам, процедурам и представлениям

ОПЕРАТОРЫ ЯЗЫКА SQL

Операторы определения данных (язык DDL).

Соответствующие операторы предназначены для создания, удаления, изменения основных объектов модели данных реляционных СУБД: таблиц, представлений, индексов.

CREATE TABLE <имя> - создание новой таблицы в базе данных.

DROP TABLE <имя> - удаление таблицы из базы данных.

ALTER TABLE <имя> - изменение структуры существующей таблицы или ограничений целостности, задаваемых для данной таблицы.

ОПЕРАТОРЫ ЯЗЫКА SQL

Использование языка SQL для выбора информации из таблицы

Выборка данных осуществляется с помощью оператора SELECT, который является самым часто используемым оператором языка SQL. Синтаксис оператора SELECT имеет следующий вид:

SELECT [ALL/DISTINCT] <список атрибутов>/*
FROM <список таблиц>
[WHERE <условие выборки>]
[ORDER BY <список атрибутов>]
[GROUP BY <список атрибутов>]
[HAVING <условие>]
[UNION<выражение с оператором SELECT>]

ПРИМЕРЫ SQL ИНЪЕКЦИЙ

SQL-инъекции — это атака на базу данных, позволяющая выполнить некоторое действие, которое не планировалось разработчиками приложения

Атака основана на внедрении **вредоносного** кода в исходный SQL-код (скрипт). Например, на взламываемом сервере баз данных злоумышленник может:

- получить доступ к таблице, доступ к которой ограничен
- удалять данные из таблиц, удалять сами таблицы
- изменять/добавлять данные в таблицах

ПРИМЕРЫ SQL ИНЪЕКЦИЙ

Основная ошибка при которой возможно осуществление атаки – отсутствие фильтрации входных данных. Данная атака основана на внедрении вредоносного кода в исходный код или скрипт.

Данные для запроса часто принимаются от пользователя при заполнении форм, а также из get запроса текст которого формируется в url адресе.

Ссылка на сайт: site.ru/test.php?id=1

- id=1 – подгружается элемент 1
- id=2 – подгружается элемент 2
- id=3 – подгружается элемент 3

Ссылка на сайт со статьей site.ru/test.php?id=1. Файл test.php – общий шаблон вывода некоего контента. В зависимости от значения id происходит наполнение контентом из базы, которое ищется в базе по id. Формируется запрос к БД

Запрос к БД: SELECT * FROM test WHERE id = $id

Атакующий может изменить значение id запроса, получить доступ к таблицам и извлечь данные о пользователях и паролях. Перед выполнением запроса злоумышленник узнает структуру базы данных имена таблиц, полей и типы данных в таблицах

SQL инъекция: SELECT * FROM test WHERE id = 1 UNION SELECT * FROM admin

Выбрать из таблицы тест все данные где id =1 а также выбрать все данные из таблицы admin

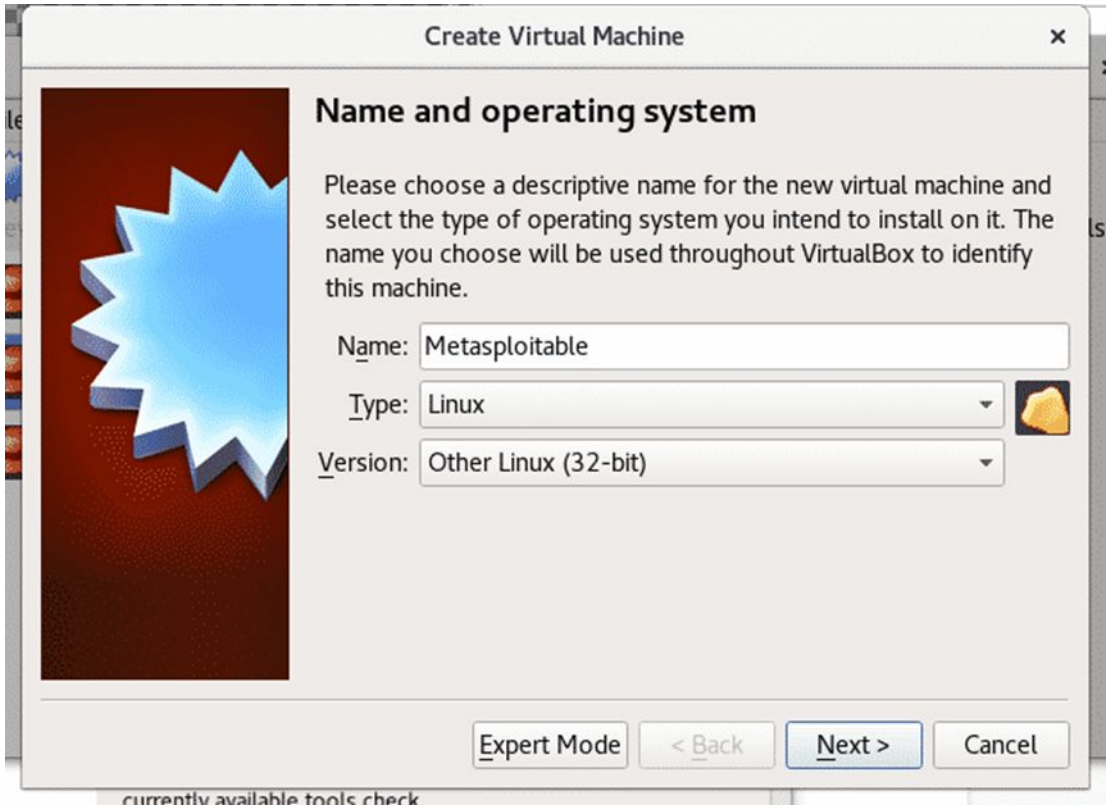
ПРИМЕРЫ SQL ИНЪЕКЦИЙ

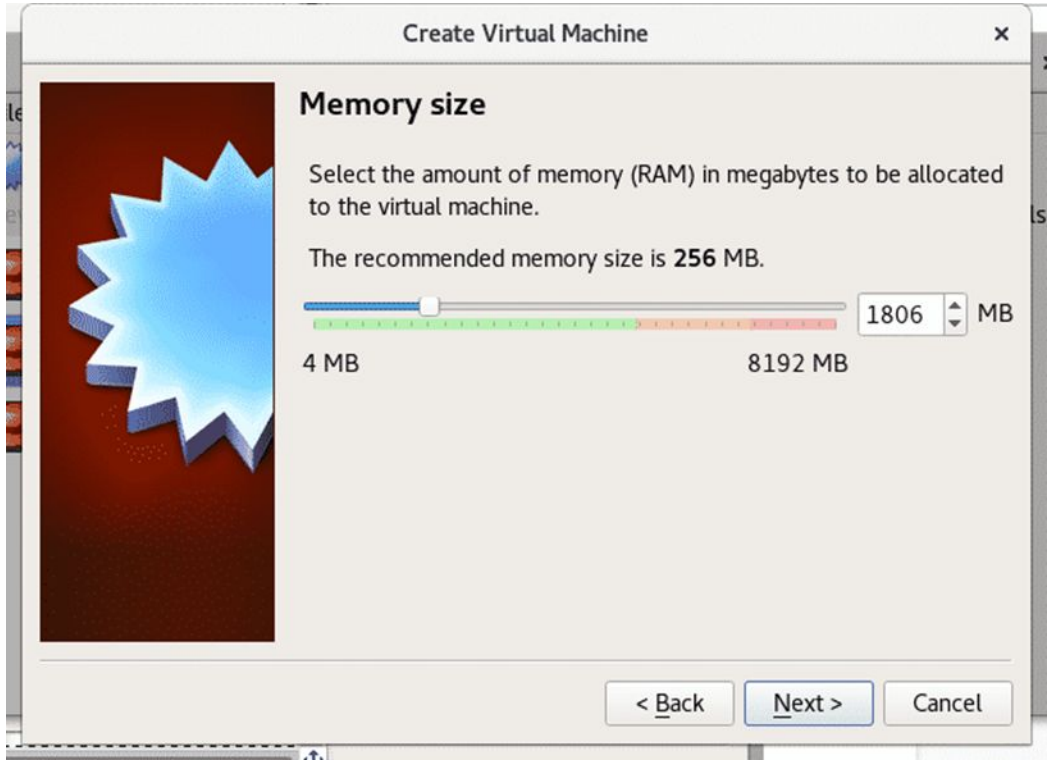
Команда	Описание
sqlmap -u «http://site.com/login.php»	Простая проверка для определенного URL
sqlmap -u «http://site.com/login.php» --dbs	Получить имена все баз данных, найденных в случае успешной эксплуатации
sqlmap -u «http://site.com/login.php» -D site_db --tables	Получить таблицы, которые находятся в базе данных (здесь site_db)
sqlmap -u «http://site.com/login.php» -D site_db -T users --dump	Получить содержимое определенной таблицы (здесь таблицы users в БД site-db)
sqlmap -u «http://site.com/login.php» -D site_db -T users --columns	Вернуть все столбцы в таблице
sqlmap -u «http://site.com/login.php» -D site_db -T users -C имя пользователя, пароль –dump	Возврат определенного содержимого столбца
sqlmap -u «http://site.com/login.php» –метод «POST» --data «username = admin & password = admin & submit = Submit» -D social_mccodes -T users –dump	Возврат таблицы, когда у нас есть данные для входа администратора

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

Установка Metasploitable 2 наVirtualbox

Скачиваем Metasploitable 2, создаем VM



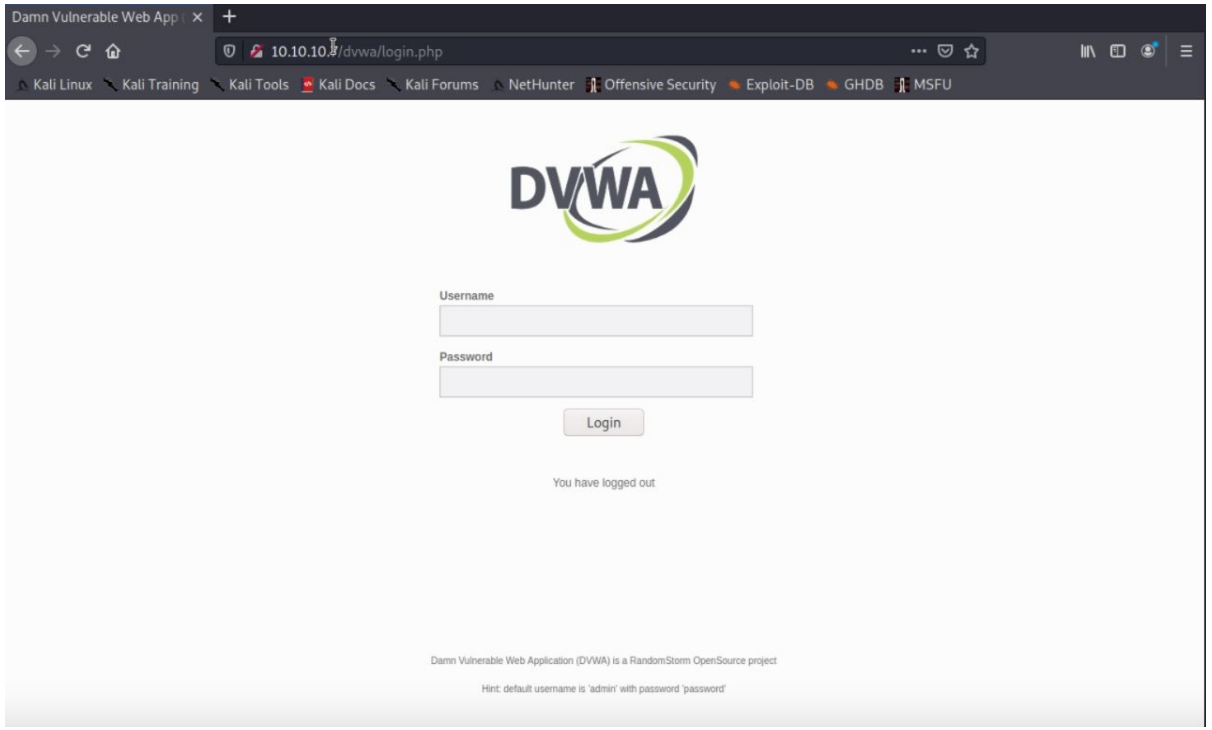


Настраиваем сеть. Kali Linux и Metasploitable 2 должны быть в одной сети. На главном экране Virtualbox переходим в Settings> System> Processor и включаем PAE / NX, нажмаем «OK»

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

В адресной строке браузера Kali Linux вводим адрес Metasploitable 2

Выбираем платформу DVWA



имя пользователя и пароль: msfadmin

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

Уязвимость DVWA. SQL-Injection (уровень Low)

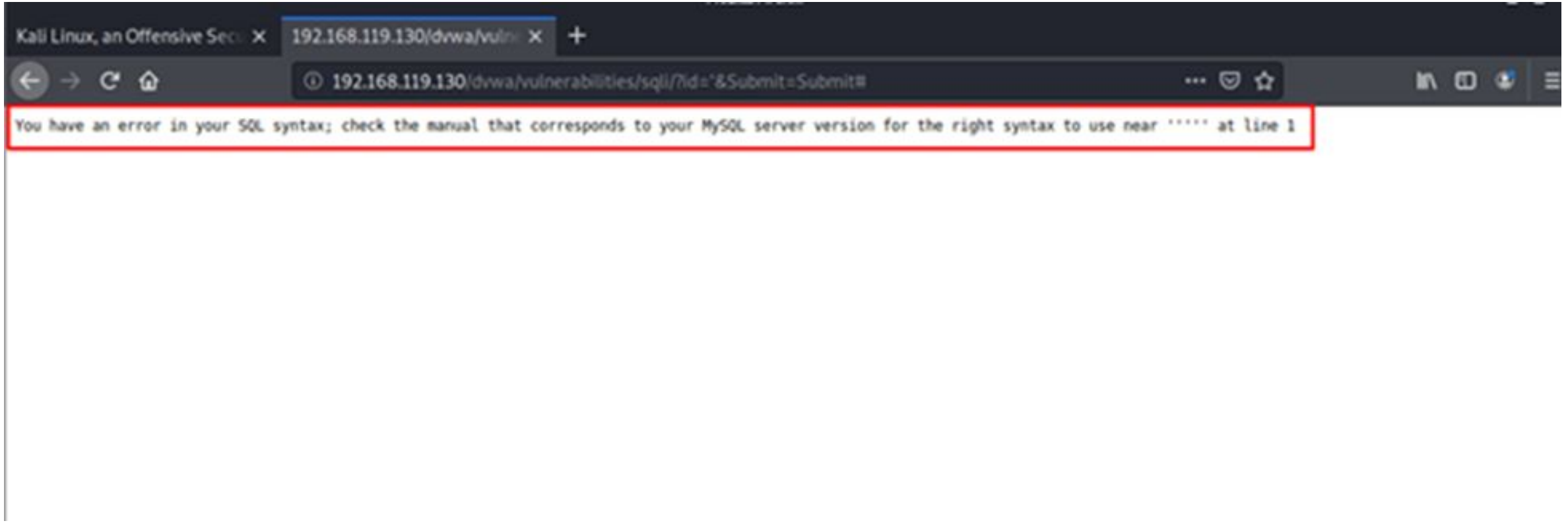
Передим на вкладку «SQL-Injection», и вводим цифру 1

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

Вывод информации говорит о том, что страничка работает корректно, но в ней есть уязвимость. Пробуем ввести цифру 2, и получаем тот же самый вывод

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

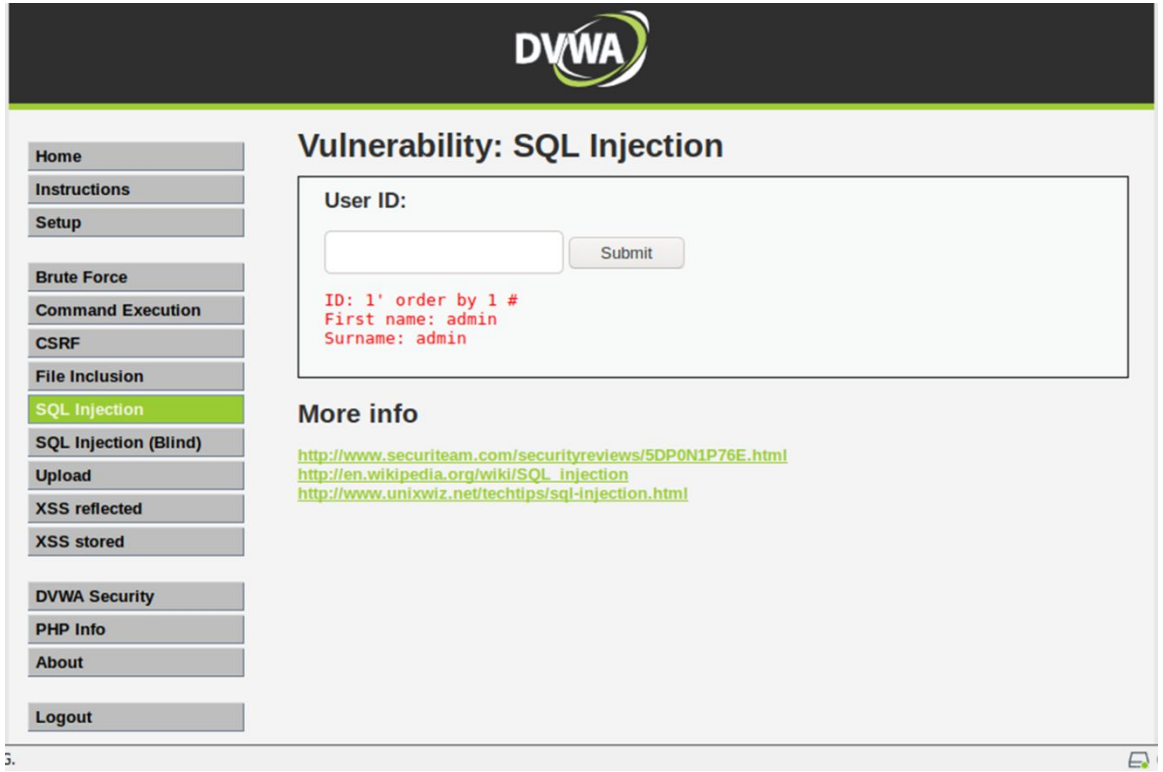
Для проверки уязвимости данной страница к SQL-инъекциям вводим одинарную кавычку

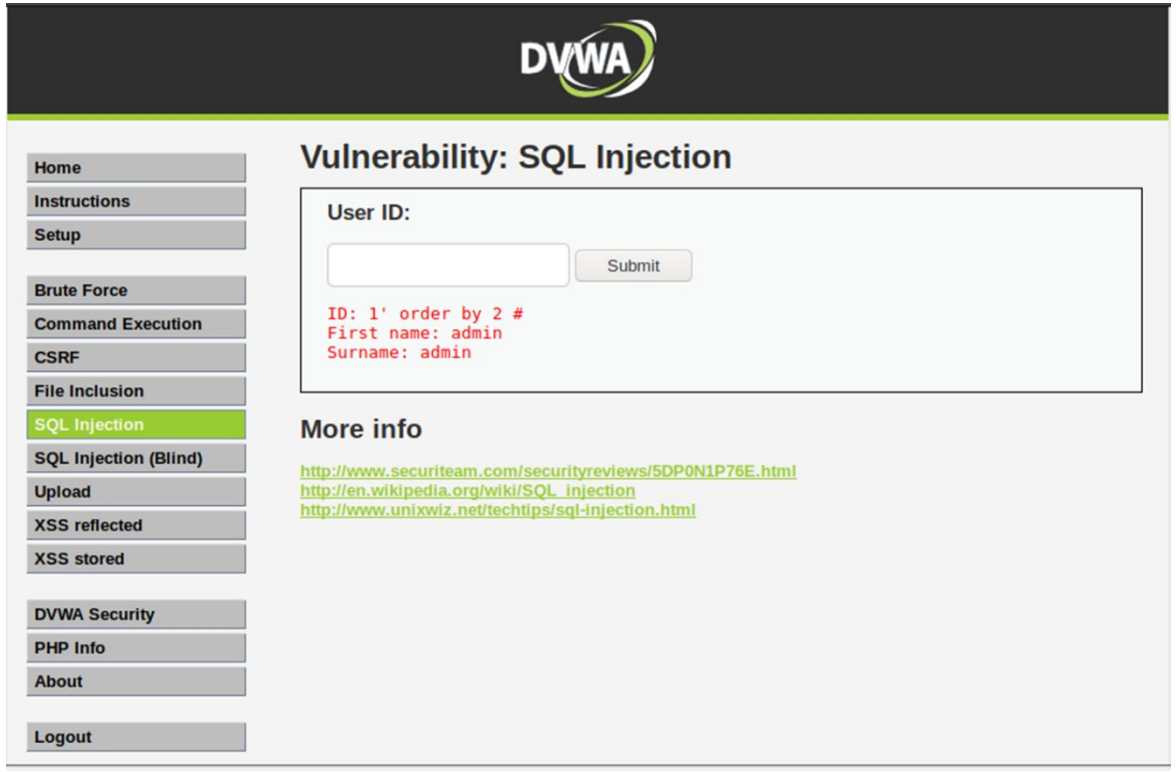


Получаем ошибку синтаксиса, а это значит, что уязвимость действительно есть.

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

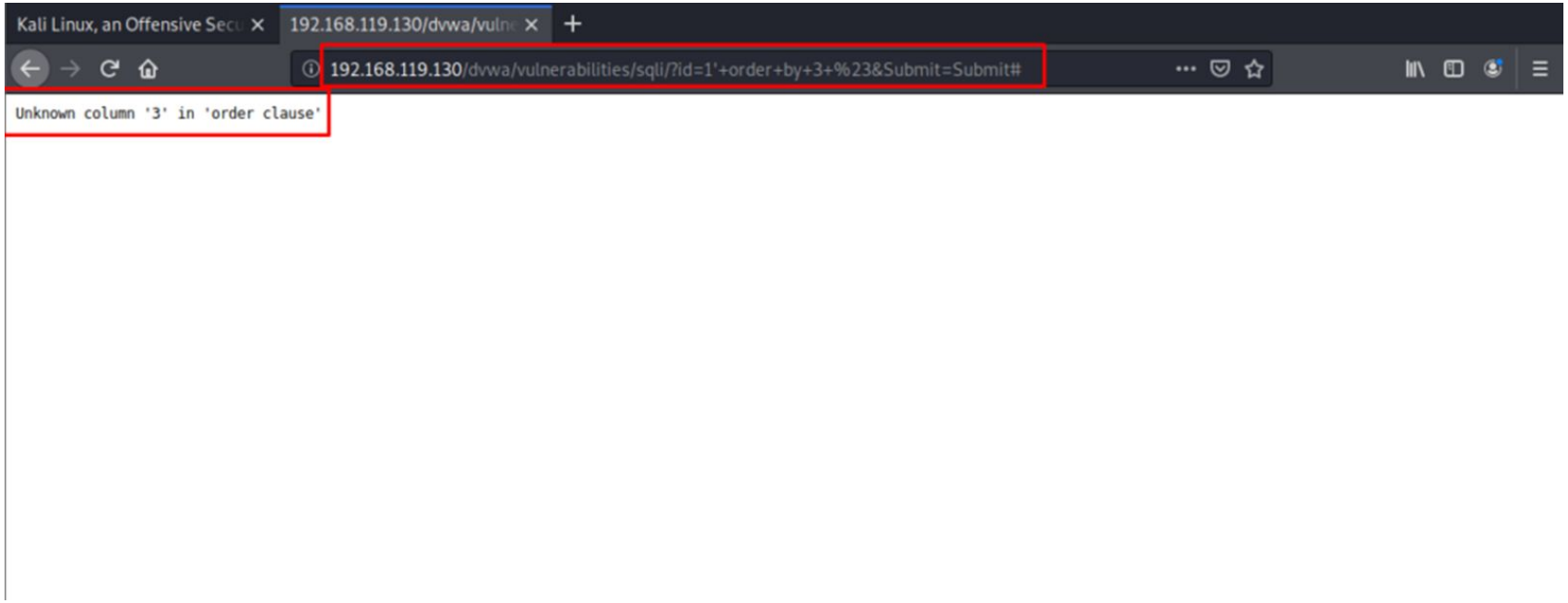
Проверим количество столбцов с помощью SQL-запросов. Команды будут выглядеть как: «1’ order by 1 #», «1’ order by 2 #»





ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

Введем команду «1’ order by 3 #»:



Исходя из полученной записи делаем вывод, что в таблице 2 столбца.

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

Проверим версию базы данных, с помощью запроса «1’ union select null, version() #»:

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

Проверим версию базы данных, с помощью запроса «1’ union select null, version() #»:

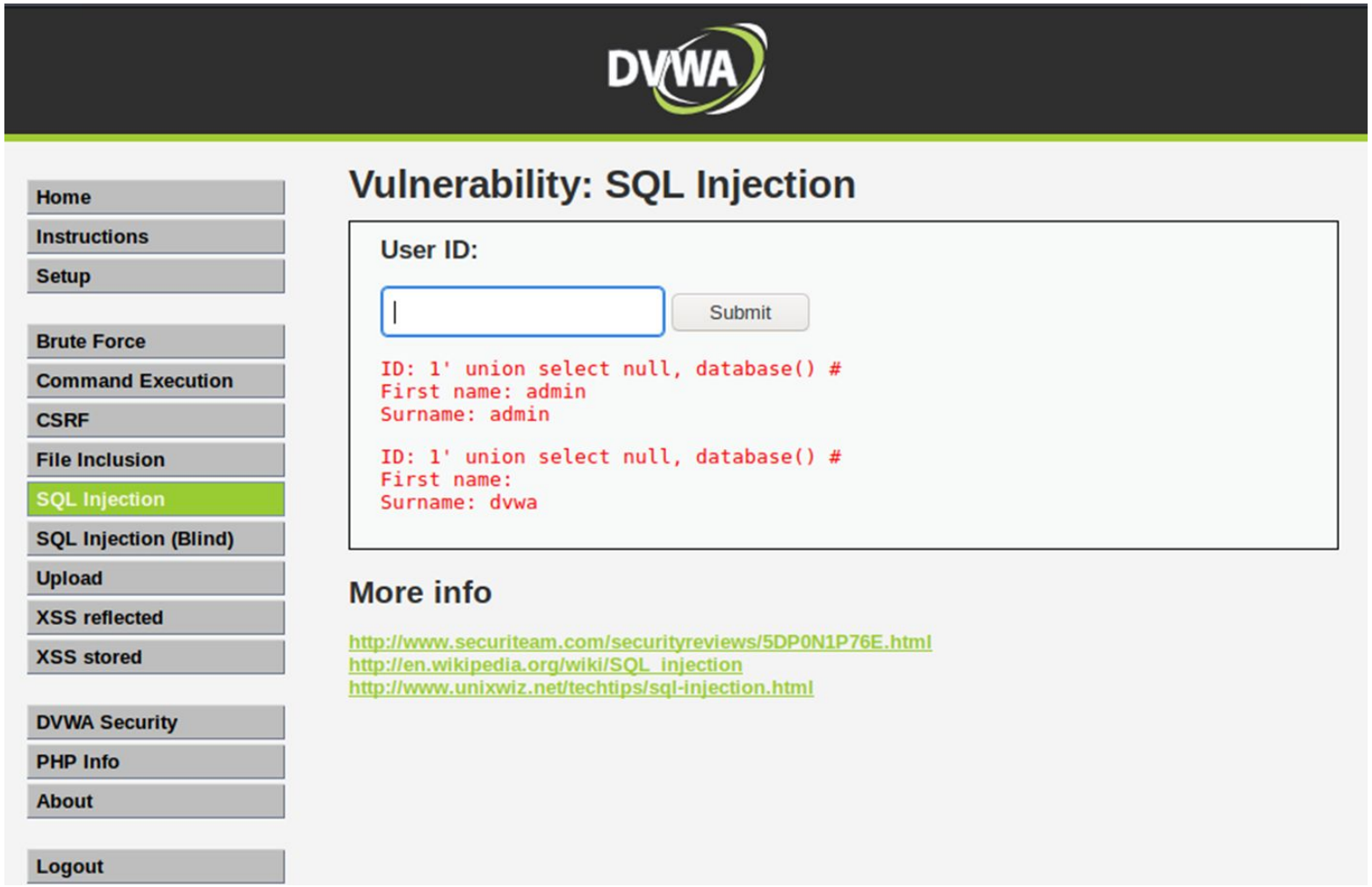
Версия базы данных (5.0.51a-3ubuntu5)

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

Проверим имя пользователя, с помощью команды «1’ union select null, user() #»:

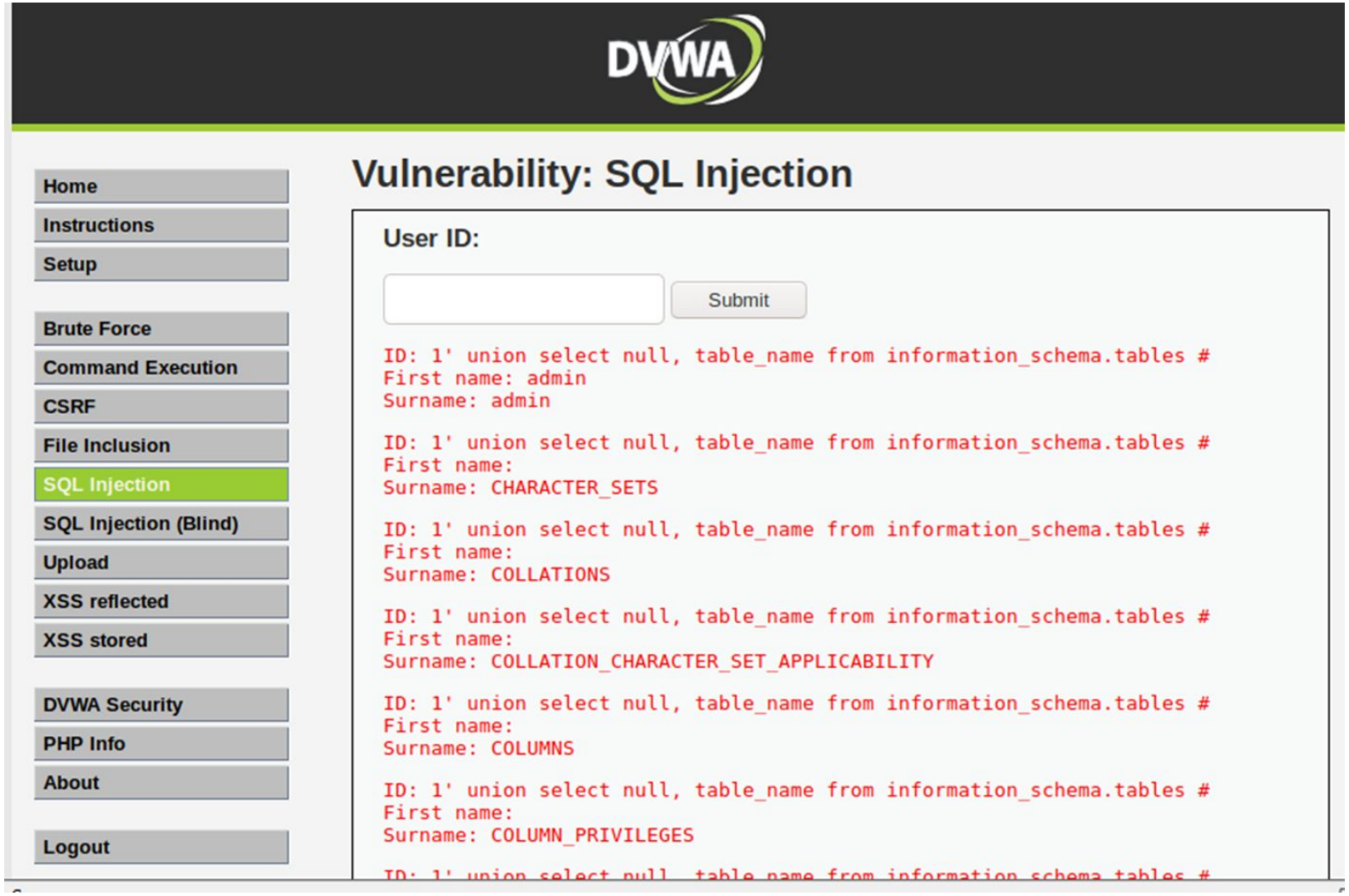
ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

узнаем имя базы данных с помощью команды: «1’ union select null, database() #»:



База данных называется «dvwa»

узнаем имя таблицы от information_schema.tables. Команда: «1’ union select null, table_name from information_schema.tables #»:



ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

узнаем захешированные пароли и имена пользователей. Команда «1′ union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users#»:

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

узнаем захешированные пароли и имена пользователей. Команда «1′ union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users#»:

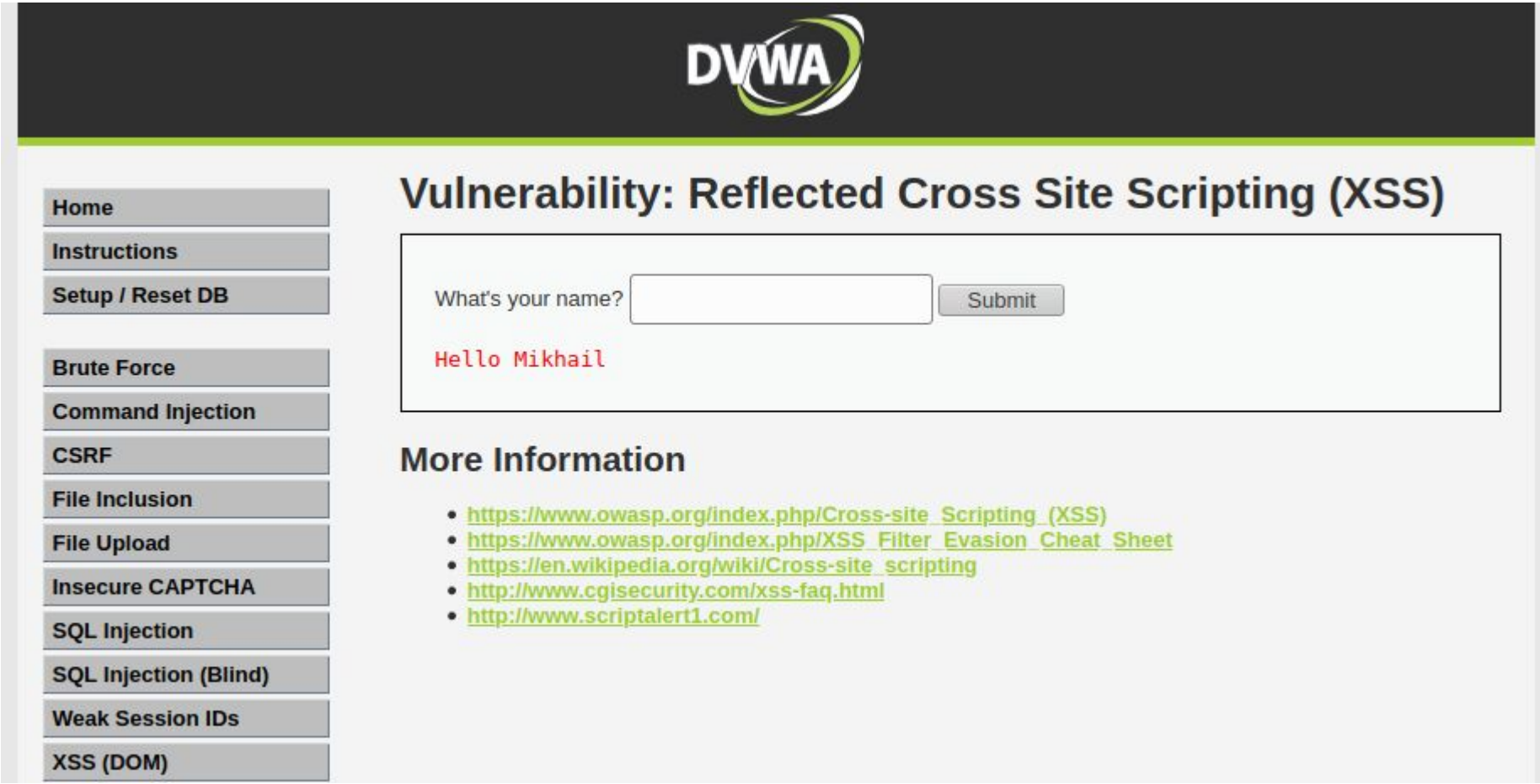
XSS (Cross Site Scripting — межсайтовый скриптинг) — широко распространённая уязвимость, затрагивающая множество веб-приложений. Она позволяет злоумышленнику внедрить вредоносный код в веб-сайт таким образом, что браузер пользователя, зашедшего на сайт, выполнит этот код

Для эксплуатации подобной уязвимости требуется определённое взаимодействие с пользователем: либо его заманивают на заражённый сайт при помощи социальной инженерии, либо просто ждут, пока тот сам посетит данный сайт

Межсайтовый скриптинг предоставляет злоумышленнику практически полный контроль над браузером

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

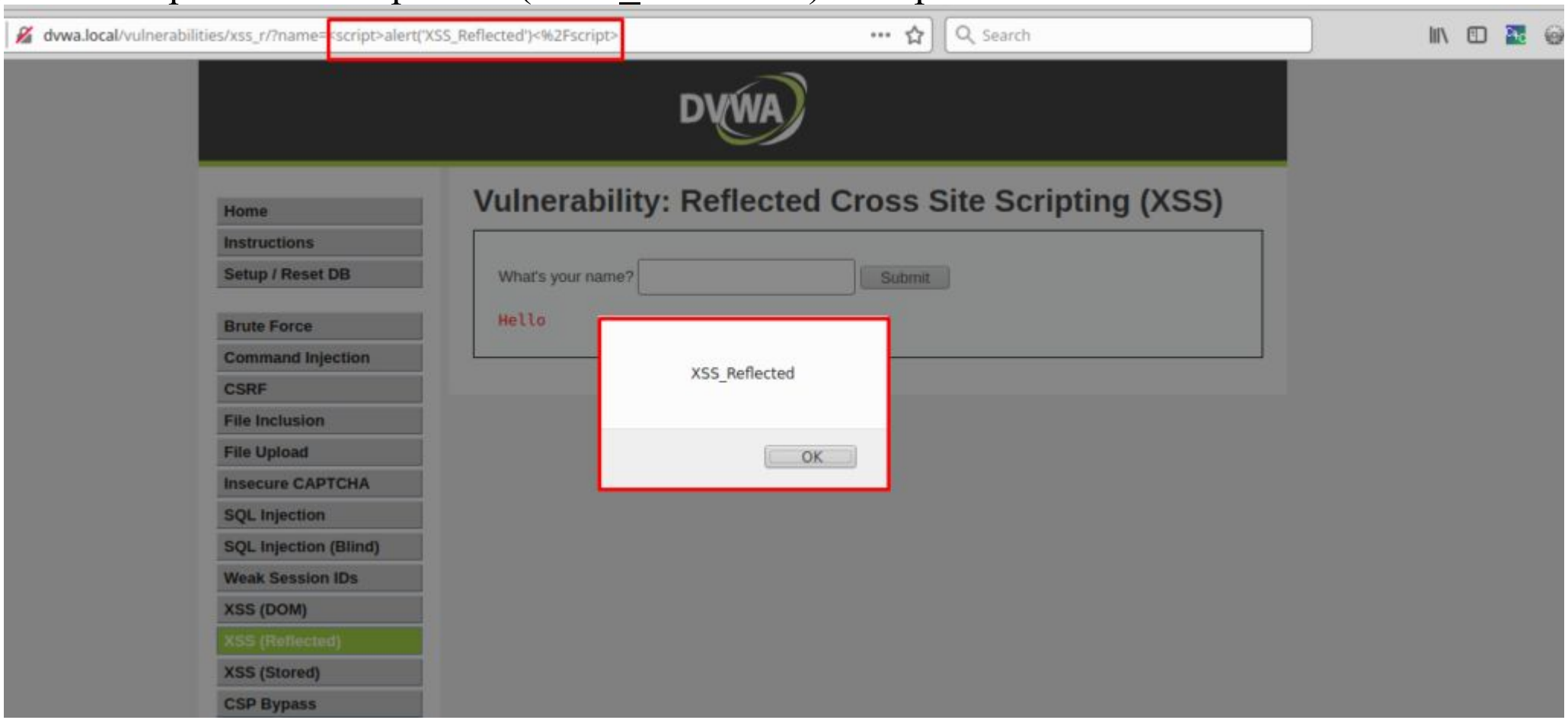
Уязвимость DVWA. XSS (Reflected) — (уровень Low)



Переходим на страницу «XSS (Reflected)», и видим поле для ввода имени, где при нажатии кнопки «Submit», происходит отображение введенного имени и приветствие, в виде слова «Hello»

ПРАКТИКА ПО ОБНАРУЖЕНИЮ УЯЗВИМОСТЕЙ WEB ПРИЛОЖЕНИЙ

Пробуем ввести скрипт, на языке JavaScript, который выглядит следующим образом: «<script>alert('XSS_Reflected')</script>»:



данные, которые были введены, передаются методом GET в адресной строке URL. Мы успешно проэксплуатировали данную уязвимость.

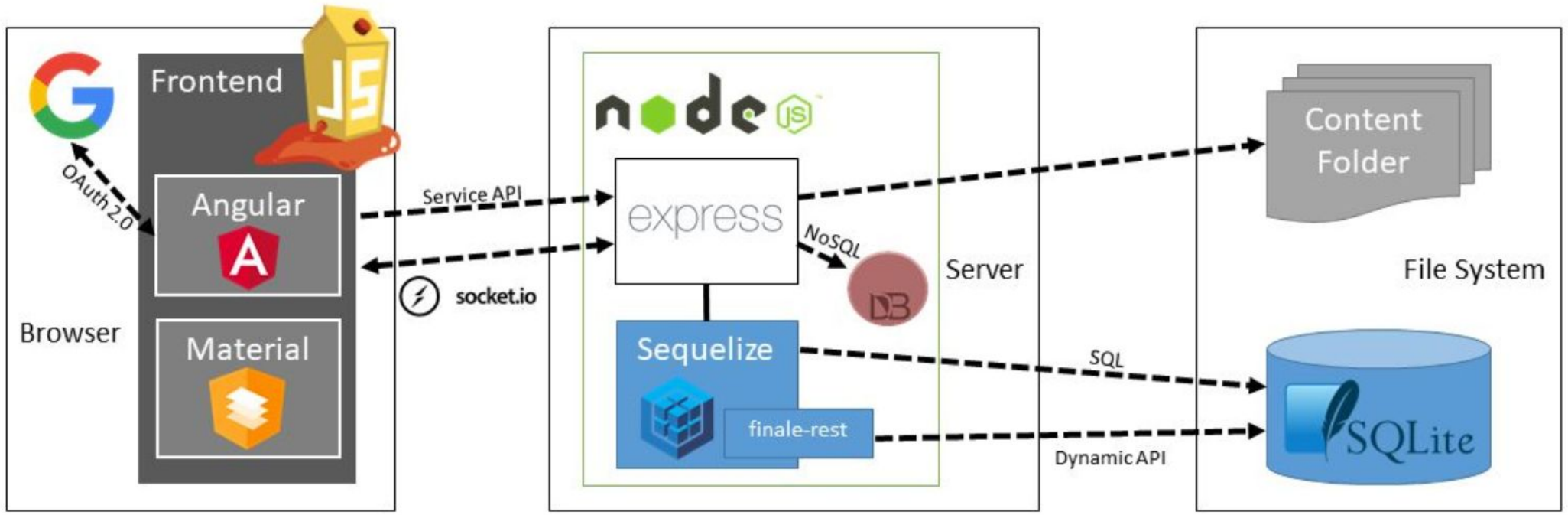
Уязвимые среды для тестирования на проникновение

В Kali Linux 2022.3: добавлены уязвимые среды для тестирования на проникновение, в репозитории дистрибутива добавлены две уязвимые платформы для изучения методов взлома сайтов:
DVWA - Damn Vulnerable Web Application
Juice Shop - OWASP Juice Shop

OWASP Juice Shop

Для того, чтобы попасть в панель администратора, нам необходимо с использованием хитрой техники ввести верные учётные данные, чтобы база данных, работающая на бэкенде, подумала, что мы ввели всё верно.
Первым делом начинаем с кавычки в поле ввода, дабы проверить отсутствие фильтрации значений, и мы должны получить ошибку — Security Misconfiguration.
Далее формируем запрос для базы данных.

OWASP Juice Shop



Проект Juice Shop призван продемонстрировать самые часто встречаемые уязвимости в приложениях и «худшие практики» веб-разработки. Этот магазинчик является полигоном для тренировки и изучения пентеста реальных приложений

УСТАНОВКА HEROKU

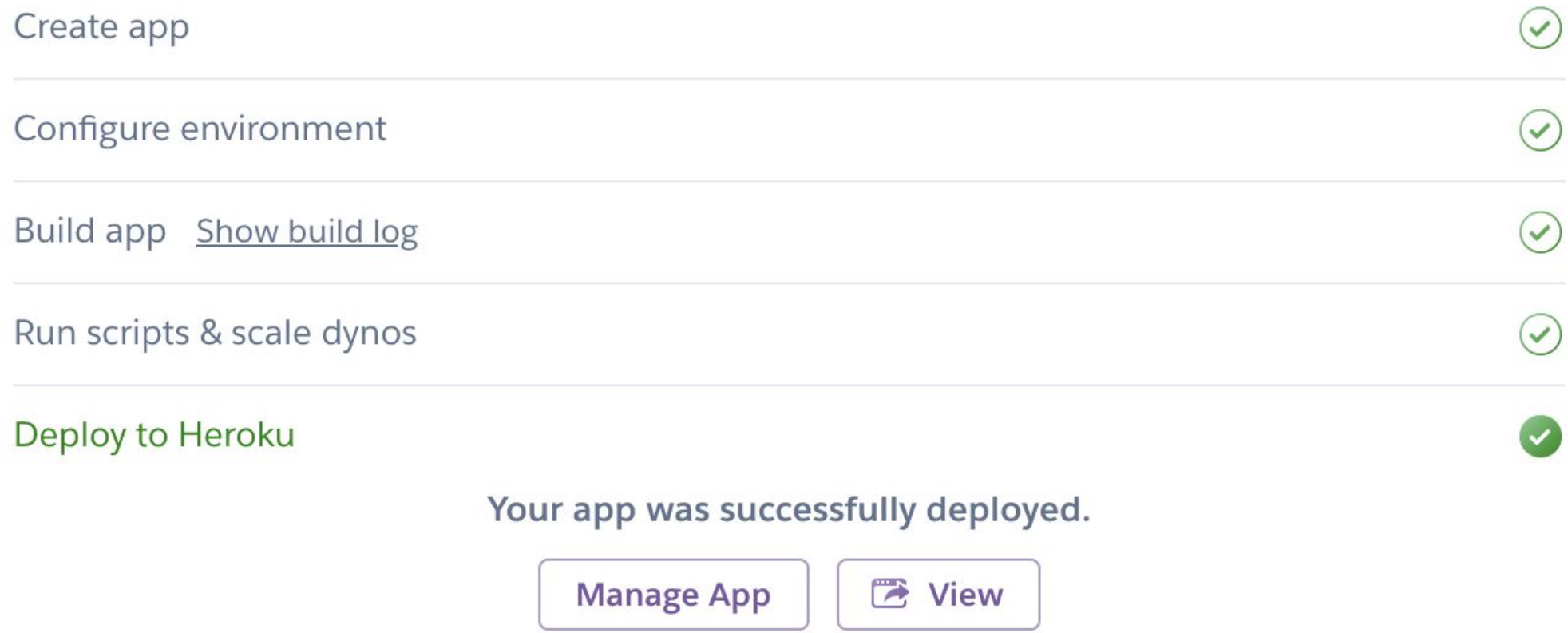
Разверните собственное веб-приложение. Для этого:
1. Зарегистрируйтесь на бесплатном ресурсе Heroku: Cloud Application Platform.
2. Зайдите в Heroku с вашего нового аккаунта и введите в адресную строку браузера следующее:

https://dashboard.heroku.com/new?button-url=https%3A%2F%2Fgithub.com%2Fbkimminich%2Fjuice-shop&template=https%3A%2F%2Fgithub.com%2Fbkimminich%2Fjuice-shop

Так вы укажете системе, откуда брать данные для установки приложения.

УСТАНОВКА HEROKU

3. Выберите уникальное имя собственного приложения, затем Deploy app:
4. Дождитесь развертывания (занимает около 15–20 минут). В конце должно получиться следующее:



5. Нажмите на кнопку View и посмотрите на ваше приложение.
Если приложением долго не пользуются, оно уходит в сон, тогда его нужно будет включить через Open app при входе в личный кабинет Heroku:

ЗАЛАНИЕ 1. ТАБЛО ПРОГРЕССА (Прогрессбар)

1. Найдите путь к панели прогресса (панели администратора) веб-приложения Juice Shop. Попробуйте прочитать код сайта и поискать в модуле main*.js м через на вкладку Sources. Найдите в коде слова, содержащие «score» (счет). (браузер Google Chrome Поиск Ctrl + f)
2. Если вы найдёте путь верно (снова потребуются знания английского языка), должна возникнуть ошибка 403, так как у вас не хватит прав доступа — не получится войти в личный кабинет ад министратора.

Подсказка 1
Используйте «Инструменты разработчика» в браузере (браузер Google Chrome: контекстное меню: просмотр кода страницы , Firefox F12.) для чтения кода веб-приложения.
Подсказка 2
Помимо стандартного HTML-кода, страница подгружает файлы. Обратите внимание на вкладку Sources в панели инструментов разработчика в браузере.
Подсказка 3
Прочтите код файла main*.js (удобно использовать при чтении кода Pretty View в составе Chrome). Найдите ключевое слово admin в его строках. В одном из вхождений будет указан путь (path) к странице счёта — administration.
В ответе на это задание укажите путь/страницу, которую нужно добавить к основному домену (после символа #/), чтобы перейти на страницу входа в панель администратора.

ЗАЛАНИЕ 2. ВХОД ИЗ ПОД АДМИНИСТРАТОРА (SQL-ИНЪЕКЦИЯ)

Для того, чтобы попасть в панель администратора необходимо с использованием определенной техники ввести верные учётные данные, чтобы база данных, работающая на бэкенде, подумала, что мы ввели всё верно.

Проверка наличия уязвимостей начинается со ввода спецсимволов. Например с кавычки в поле ввода, чтобы проверить отсутствие фильтрации значений. Если мы получаем ошибку сервера это свидетельствует о наличии уязвимости.

Далее после кавычки формируем запрос для базы данных, содержащий истинное утверждение , а остальная часть запроса не читалась базой (закоментирована).

Например в поле «Логин» попробуйте ввести

‘or TRUE --

ЗАЛАНИЕ 3. ВХОД ИЗ ПОД ДРУГОГО ЗАРЕГИСТРОРОВАННОГО ПОЛЬЗОВАТЕЛЯ, ПРОСМОТР СОДЕРЖИМОГО ЧУЖОЙ КОРЗИНЫ

Контроль доступа обеспечивает соблюдение политики, так что пользователи не могут действовать за пределами своих предполагаемых разрешений, когда это нарушается, такого рода уязвимость называется **Broken Access Control**.

У каждого пользователя OWASP Juice Shop есть своя корзина товаров, чтобы подглядеть в чужую, вовсе не обязательно логиниться в каждую из учёток после создания собственной учётной записи, попробуем посмотреть чужую корзину, не выходя из своего аккаунта.

ЗАЛАНИЕ 3. ВХОД ИЗ ПОД ДРУГОГО ЗАРЕГИСТРОРОВАННОГО ПОЛЬЗОВАТЕЛЯ, ПРОСМОТР СОДЕРЖИМОГО ЧУЖОЙ КОРЗИНЫ

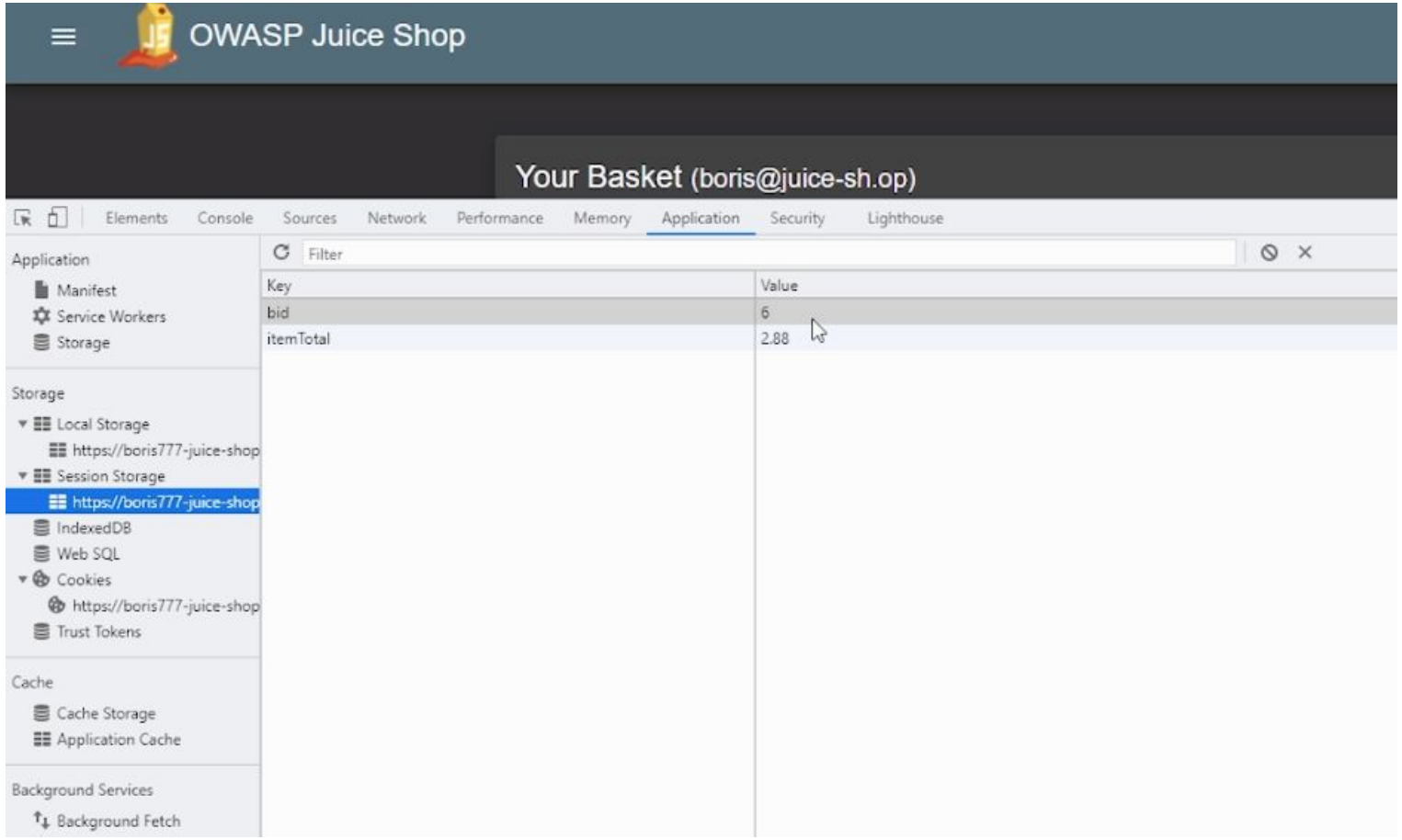
Создайте свою учетную запись в Juice Shop. Ознакомьтесь у устройством магазина. Попробуйте найти параметр, определяющий принадлежность корзины пользователю.

Попробуйте зайти в корзину, используя другой basket id. Система должна сообщить о том, что для этой операции у вас недостаточно прав.

Попробуйте зайти с другой стороны и посмотреть cookie. Зайдите в свою корзину, далее в панели администратора браузера вкладка Application Посмотрите какие параметры хранятся в рамках сессии.

ЗАЛАНИЕ 3. ВХОД ИЗ ПОД ДРУГОГО ЗАРЕГИСТРОРОВАННОГО ПОЛЬЗОВАТЕЛЯ, ПРОСМОТР СОДЕРЖИМОГО ЧУЖОЙ КОРЗИНЫ

Найдите ключ bid, соответствующий номеру вашей корзины.



В рамках сессии попробуйте изменить ключ bid на соответствующей номеру чужой корзины и обновить страницу

ЗАЛАНИЕ 3. ВХОД ИЗ ПОД ДРУГОГО ЗАРЕГИСТРОРОВАННОГО ПОЛЬЗОВАТЕЛЯ, ПРОСМОТР СОДЕРЖИМОГО ЧУЖОЙ КОРЗИНЫ

Создайте свою учетную запись в Juice Shop. Ознакомьтесь у устройством магазина. Попробуйте найти параметр, определяющий принадлежность корзины пользователю.

Попробуйте зайти в корзину, используя другой basket id. Система должна сообщить о том, что для этой операции у вас недостаточно прав.

Подсказка
Вкладка Network параметр basket

В ответе на это задание укажите итоговую сумму заказа корзины пользователя.

Так как имя пользователя нам известно, нам необходимо, чтобы часть проверки пароля отсеивалась. Для этого в поле логина нужно использовать уже известную комбинацию символов.

ЗАДАНИЕ 3. ОСТАВЛЯЕМ НУЛЕВОЙ ФИДБЭК

Оставим негативный фидбэк этому магазину! Зайдите в вашу ранее созданную учётную запись в магазине соков либо создайте новую, затем в меню слева выберите Customer Feedback.
Ваша задача — выставить оценку с нулём звезд. Сложность в том, что такого параметра нет (ноль звезд). Такого рода уязвимости приложений называются Improper Input Validation.
Внимательно изучите код до и после ввода значений. Как вы видите, изначально рейтинг равняется нулю:
Но кнопка без выставления оценки не нажимается. Ваша задача — исправить это. В ответе на это задание укажите, какой именно класс CSS нужно удалить, чтобы кнопка нажалась и опубликовался отзыв.
Класс(class) - это правило css, которое отвечает за свойства(размер, цвет... и др.) элемента, которому принадлежит данный класс. CSS (Cascading Style Sheets) — это код, который вы используете для стилизации вашей веб-страницы, формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML).

Подсказка 1

Какие теги и классы появляются или пропадают?
Код кнопки до выставления рейтинга:
Код после выставления рейтинга:

Подсказка 2

Обратите внимание на переменную кнопки <button … disabled=...>.

Подсказка 3

Нужно избавиться от ненужного класса CSS для кнопки.