

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Государственный университет морского и речного флота имени адмирала С.О. Макарова»**

**(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)**

Институт (факультет): Институт водного транспорта

Кафедра: Комплексного обеспечения информационной безопасности

Направление подготовки (специальность): Информационная безопасность   
автоматизированных систем 10.05.03

(код, наименование направления подготовки/специальности) Форма обучения: Очная

«**К ЗАЩИТЕ ДОПУЩЕН**»

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО)

«\_\_\_»\_\_\_\_\_\_\_\_\_ 2021г.

**Выпускная квалификационная работа**

Обучающегося: Котова Александра Дмитриевича

(фамилия, имя, отчество)

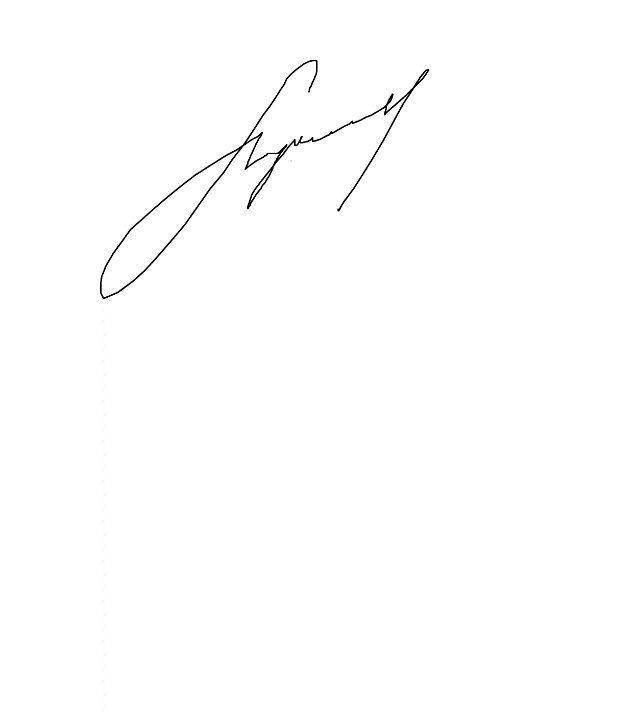
Вид работы: Выпускная квалификационная работа специалиста

(выпускная квалификационная работа бакалавра, специалиста, магистра)

**Пояснительная записка**

Тема: Исследование базовых методик тестирования на проникновение веб-приложений

(полное название темы квалификационной работы, в соответствии с приказом об утверждении тематики ВКР)

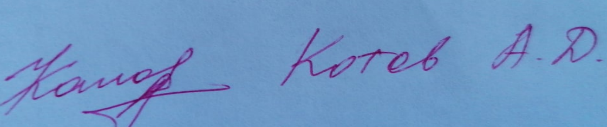
****

Руководитель работы: Ли Изольда Валерьевна, доцент кафедры Комплексного

обеспечения информационной безопасности, к.т.н.

(должность, подпись, фамилия, инициалы, дата)

Обучающийся: Котов Александр Дмитриевич



(подпись, фамилия, инициалы, дата)

Санкт-Петербург

2021 г.

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Государственный университет морского и речного флота имени адмирала С.О. Макарова»**

**(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)**

Институт (факультет): Водного транспорта

Кафедра: Комплексного обеспечения информационной безопасности

Направление подготовки (специальность): Информационная безопасность   
автоматизированных систем 10.05.03

(код, наименование направления подготовки/специальности)

Форма обучения: Очная

«**УТВЕРЖДАЮ**»

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_2021г.

**Задание**

**на выпускную квалификационную работу**

Вид работы: ВКР специалиста

(ВКР бакалавра, специалиста, магистра)

Обучающемуся: Котову Александру Дмитриевичу

(фамилия, имя, отчество)

Тема: Исследование базовых методик тестирования на проникновение веб-приложений

Утверждена приказом ректора Университета от «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2021 года, № \_\_\_\_\_\_\_

Срок сдачи законченной работы «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

Исходные данные (или цель ВКР):

Рассмотрение базовых методологий тестирования на проникновение веб-приложений для предотвращения угроз информационной безопасности в веб-приложениях

Перечень подлежащих исследованию, разработке, проектированию вопросов (краткое содержание ВКР):

Глава 1. ПРОБЛЕМА УГРОЗ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ

* 1. Тестирование на проникновение - в разделе описывается что из себя представляет тестирования на проникновение важность использования тестирования на проникновение
  2. Этапы тестирования на проникновение – в разделе описываются этапы тестирования на проникновения
  3. Инструменты для тестирования на проникновение – в разделе описываются инструменты тестирования на проникновение
  4. Правовые аспекты тестирования на проникновение – в разделе описываются предписания к тестированию с точки зрения правовых актов
  5. Методологии тестирования на проникновение - в разделе описываются выбранные методологии тестирования на проникновение

Глава 2. OWASP WEB SECURITY TESTING GUIDE 4.2

2.1. Комплексный подход к тестированию безопасности веб-приложений по OWASP Web Security Testing Guide – в разделе описывается комплексный подход к тестированию веб-приложений

2.2. Глава Introduction (Введение) – в разделе описываются основные положения главы Introduction OWASP Web Security Testing Guide

2.3. Глава The OWASP Testing Framework – в разделе описываются основные положения главы The OWASP Testing Framework OWASP Web Security Testing Guide

2.4. Глава Web Application Security Testing (Тестирование безопасности веб-приложений) - в разделе описываются основные положения главы Web Application Security Testing The OWASP Testing Framework OWASP Web Security Testing Guide

2.5. Методы тестирования на проникновение веб-приложений - в разделе описываются основные методы тестирования на проникновение веб-приложений, выбранные по OWASP top ten

ГЛАВА 3. ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ МЕТОДОВ OWASP WEB SECURITY TESTING GUIDE V4.2 – в главе описываются методы тестирования на проникновение веб-приложений, примененные к тестовому стенду.

Перечень графического материала (или презентационного материала):

1. Презентация - 19 слайдов

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.

Дата выдачи задания: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

Задание согласовано и принято к исполнению: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель ВКР: | Ли Изольда Валерьевна,  доцент кафедры Комплексного  обеспечения информационной безопасности, к.т.н. |  |  |
|  | (должность, ученая степень, ученое звание, ФИО) |  | (подпись) |
| Обучающийся: | Котов Александр Дмитриевич  ИБ-51 10.05.03 |  |  |
|  | (учебная группа, ФИО) |  | (подпись) |

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Государственный университет морского и речного флота имени адмирала С.О. Макарова»**

**(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)**

**ОТЗЫВ**

**руководителя выпускной квалификационной работы**

Обучающегося: Котова Александра Дмитриевича

На тему: Исследование базовых методик тестирования на проникновение веб-приложений

Направление подготовки (специальность): Информационная безопасность автоматизированных систем 10.05.03

Институт (факультет): Институт водного транспорта

Кафедра: Комплексного обеспечения информационной безопасности

Руководитель выпускной квалификационной работы:

Ли Изольда Валерьевна, доцент кафедры Комплексного обеспечения информационной безопасности, к.т.н.

(ВСТАВИТЬ ОТЗЫВ К ДИПЛОМУ)

Выпускная квалификационная работа Котова А.Д. удовлетворяет требованиям, предъявляемым к ВКР специалиста, может быть допущена к защите и заслуживает оценки **ОТЛИЧНО**, а автор, **Котов Александр Дмитриевич**, заслуживает присвоения квалификации «**специалист**» по направлению 10.05.03 – **Информационная безопасность автоматизированных систем**.

|  |  |
| --- | --- |
|  | И.В. Ли |
|  | «01» июня 2021 г. |

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 11](#_Toc75291502)

[ГЛАВА 1. ПРОБЛЕМА УГРОЗ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ 12](#_Toc75291503)

[1. Тестирование на проникновение 15](#_Toc75291504)

[1.1.Определение тестирования на проникновение 15](#_Toc75291505)

[1.2.Важность тестирования на проникновение 16](#_Toc75291506)

[2. Этапы тестирования на проникновение 18](#_Toc75291507)

[2.1. Планирование и подготовка 18](#_Toc75291508)

[2.2 Сбор информации 19](#_Toc75291509)

[2.3. Попытки проникновения и эксплуатация 19](#_Toc75291510)

[2.4. Анализ и отчетность 19](#_Toc75291511)

[2.5. Очистка и исправление 19](#_Toc75291512)

[2.6. Повторное тестирование 20](#_Toc75291513)

[3. Инструменты для тестирования на проникновение 20](#_Toc75291514)

[4. Правовые аспекты тестирования на проникновение 20](#_Toc75291515)

[4.1. ФСТЭК 20](#_Toc75291516)

[5. Методологии тестирования на проникновение 23](#_Toc75291517)

[5.1. OSSTMM 3 – The Open Source Security Testing Methodology Manual 23](#_Toc75291518)

[5.2. PTES (Penetration Testing Execution Standard) 25](#_Toc75291519)

[5.3. OWASP Web Security Testing Guide 26](#_Toc75291520)

[5.4. Выбор методологии 27](#_Toc75291521)

[ГЛАВА 2. OWASP WEB SECURITY TESTING GUIDE 4.2 28](#_Toc75291522)

[1. Комплексный подход к тестированию безопасности веб-приложений по OWASP Web Security Testing Guide 28](#_Toc75291523)

[2. Глава Introduction (Введение) 29](#_Toc75291524)

[3. Глава The OWASP Testing Framework 33](#_Toc75291525)

[4. Глава Web Application Security Testing (Тестирование безопасности веб-приложений) 35](#_Toc75291526)

[5. Методы тестирования на проникновение веб-приложений 38](#_Toc75291527)

[5.1.Сбор информации 38](#_Toc75291528)

[5.2. Тестирование конфигурации и управления развертывания 40](#_Toc75291529)

[5.3. Тестирование управления идентификацией 42](#_Toc75291530)

[5.4. Тестирование аутентификации 43](#_Toc75291531)

[5.5. Тестирование авторизации 44](#_Toc75291532)

[5.6. Тестирование управления сессиями(сеансом) 45](#_Toc75291533)

[5.7. Тестирование проверки входных данных 47](#_Toc75291534)

[5.8. Тестирование на обработку ошибок 52](#_Toc75291535)

[5.9. Тестирование слабой криптографии 53](#_Toc75291536)

[5.10. Тестирование бизнес-логики 54](#_Toc75291537)

[5.11.Тестирование на стороне клиента 55](#_Toc75291538)

[ГЛАВА 3. ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ МЕТОДОВ OWASP WEB SECURITY TESTING GUIDE V4.2 56](#_Toc75291539)

[1. Тестовый стенд 56](#_Toc75291540)

[2. Сбор информации 57](#_Toc75291541)

[2.1. Определение веб-сервера 57](#_Toc75291542)

[2.2. Проверка содержимого веб-страницы на предмет утечки информации 58](#_Toc75291543)

[2.3. Составление карты веб-приложения 60](#_Toc75291544)

[3. Тестирование конфигурации и управления развертывания 63](#_Toc75291545)

[3.1. Тестирование HTTP методов 63](#_Toc75291546)

[3.2. Тестирование метода PUT 63](#_Toc75291547)

[3.3. Тестирование потенциальной межсайтовой трассировки 64](#_Toc75291548)

[4. Тестирование управления учетными записями 65](#_Toc75291549)

[4.1. Тестирование ролей пользователей 65](#_Toc75291550)

[4.2. Тестирование процесса регистрации пользователей 67](#_Toc75291551)

[5. Тестирование аутентификации 71](#_Toc75291552)

[5.1. Тестирование учетных данных, передаваемых по зашифрованному каналу 71](#_Toc75291553)

[5.2. Проверка слабого механизма блокировки 72](#_Toc75291554)

[5.3. Тестирование ненадежной политики генерации паролей 75](#_Toc75291555)

[6. Тестирование авторизации 77](#_Toc75291556)

[6.1. Тестирование повышения привилегий 77](#_Toc75291557)

[7. Тестирование управления сессиями(сеансом) 78](#_Toc75291558)

[7.1. Тестирование тайм-аутов сессии 78](#_Toc75291559)

[7.2. Тестирование функции выхода из системы 78](#_Toc75291560)

[8. Тестирование валидации входных данных 81](#_Toc75291561)

[8.1. Тестирование отраженных межсайтовых сценариев 81](#_Toc75291562)

[8.2. Тестирование хранимых межсайтовых сценариев 82](#_Toc75291563)

[8.3. Тестирование SQL-инъекций 82](#_Toc75291564)

[8.4. Тестирование инъекций заголовка Host 85](#_Toc75291565)

[9. Тестирование на обработку ошибок 87](#_Toc75291566)

[9.1. Тестирование на неправильную обработку ошибок 87](#_Toc75291567)

[10. Тестирование слабой криптографии 89](#_Toc75291568)

[10.1. Тестирование отправки конфиденциальной информации по незашифрованному каналу 89](#_Toc75291569)

[11. Тестирование бизнес-логики 91](#_Toc75291570)

[11.1. Тестирование возможности подделки запросов 91](#_Toc75291571)

[12. Тестирование на стороне клиента 96](#_Toc75291572)

[12.1. Тестирование межсайтового скриптинга на основе DOM 96](#_Toc75291573)

[12.2. Тестирование HTML инъекций 97](#_Toc75291574)

[ЗАКЛЮЧЕНИЕ 98](#_Toc75291575)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 99](#_Toc75291576)

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ**

|  |  |
| --- | --- |
| SDLC – | Жизненный цикл разработки программного обеспечения; |
| ФСТЭК – | Федеральная служба по техническому и экспортному контролю; |
| API – | Программный интерфейс приложения; |
| СУБД – | Система управления базами данных; |
| HTML – | Язык гипертекстовой разметки; |
| HTTP – | Протокол передачи гипертекста; |
| XSS – | Межсайтовый скриптинг; |
| DOM – | Объектная модель документа; |
| HTTPS – | Безопасный протокол передачи гипертекста; |
| JSON – | JavaScript Object Notation; |
| IP – | Интернет протокол; |
| SGML – | Стандартный обобщенный язык разметки; |
| XML – | Расширяемый язык разметки; |
| XXE– | XML External entity; |
| XST – | Cross-Site Tracing; |
| SQL – | Язык структурированных запросов; |
| SSL – | Уровень защищенных сокетов; |

# ВВЕДЕНИЕ

В связи с повсеместным использованием Интернета, в частности, веб-приложений и электронных ресурсов (интернет-магазины, медицинские услуги и т.д.), они представляют собой огромную ценность, как для частных лиц, так и для целых государств, что несомненно вызывает сильное проявление интереса у злоумышленников. Своей незаконной деятельностью они пытаются нарушить целостность, доступность и конфиденциальность информации, путем использования множества методов и средств атак на веб-приложения, таких как SQL-инъекции, межсайтовый скриптинг и прочее. Однако, эти угрозы возможны благодаря уязвимостям веб-приложений и при должном и решении проблем безопасности веб-приложений большинство угроз и атак возможно отразить или избежать. В современном подходе к созданию приложений неотъемлемой частью является процесс тестирования. Помимо проверки функциональности приложения, необходимо проводить и тестирование безопасности, а именно тестирование на проникновение.

# ГЛАВА 1. ПРОБЛЕМА УГРОЗ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ

На сегодняшний день, в эпоху глобальной информатизации, Интернет стал неотъемлемой частью жизни человека. Среднестатистический житель планеты проводит в сети ежедневно порядка 7 часов. Такому повсеместному использованию способствует колоссальное количество Интернет-ресурсов, практически каждая компания имеет свой веб-сайт, а суммарно на 2020 год во всем мире насчитывается около 1.8 млрд. веб-приложений. Спектр предоставления услуг огромен: социальные сети, электронные библиотеки, торговые площадки, медицинские, новостные порталы и т.д.

Несомненно, такое обилие и разносторонность Интернет-ресурсов, хранящих терабайты пользовательских данных, имеющих ценность, создает благоприятные условия для роста кибер-преступности.

Как правило, атаки злоумышленников связаны с уязвимостями веб-приложений, которые были допущены разработчиками на этапе создания приложений, не уделившим должного внимания безопасности своего продукта.

Компании, находящиеся в зоне риска, можно условно разделить на три типа:

1. Компании, чей сайт уже был подвержен кибер-атакам и успешно взломан;
2. Компании, чей сайт еще не был подвержен кибер-атакам;
3. Компании, которые рассмотрели основные векторы атак злоумышленников и предприняли меры по защите своих веб-приложений.

Очевидно наилучшим вариантом развития события – является третий вариант, когда разработчики веб-приложений тщательно изучают основные векторы угроз атак на этапе разработки, а также на этапе тестирования моделируют угрозы кибер-преступников, для дальнейшего предотвращения реальных атак на этапе эксплуатации.

В действительности, злоумышленники обладают большим количеством средств взлома электронных ресурсов, и эти средства с каждым днем прогрессируют и предоставляют кибер-преступникам большие возможности. В следствие такой ситуации, при разработке и тестировании предусмотреть все уязвимости безопасности просто нецелесообразно, так как при такой постановки задачи, конечное приложение, скорее всего, так и не дойдет до пользователей.

Встает острый вопрос, от чего же надо стараться защитить свое приложение, а какие уязвимости можно считать незначительными. Для решения такой проблемы, появились организации, занимающиеся сбором статистики и анализом основных направлений атак злоумышленников на веб-сайты, а также разрабатывающие методики тестирования на проникновение (пентестинг) с целью предотвращения атак.

Пентестинг (тестирование на проникновение) - включает в себя серию тестов на проникновение, основанных на атаках ИТ-систем для выявления их слабых мест или уязвимостей. Они предназначены для классификации и определения масштабов брешей безопасности, а также их степени влияния. В результате таких тестов вы можете получить достаточно четкое представление об опасностях для вашей системы и эффективности вашей защиты.

Существует множество методик тестирования на проникновение веб-приложений[1]:

* OSSTMM;
* ISSAF;
* OWASP;
* PTES;
* NIST SP 800-115;
* BSI;
* PETA;
* PTF;
* Positive Technologies

Однако в своей работе я акцентирую своё внимание на The Open Web Application Security Project («OWASP»), а именно OWASP Web security testing guide (Version 4.2), а также рассмотрены альтернативные методологии такие как:

* Open Source Security Testing Methodology Manual («OSSTMM»)
* Penetration Testing Execution Standard («PTES»)

Выбор именно этих методик обусловлен следующим: по мнению ряда экспертов, хорошими типами пентеста являются OWASP и PTES, так как эти методы структурированы. В дополнение к тому, что PTES, принят многими авторитарными специалистами, уже является моделью, используемой в учебных пособиях для таких систем пентестинга, как Rapid7 Metasploit. OSSTMM же, несмотря на то, что тесты не особо новаторские, стал стандартом и является одним из первых подходов к универсальной структуре концепции безопасности. Сегодня OSSTMM служит ориентиром не только для организаций, которые хотят разрабатывать качественный, организованный и эффективный пентестинг, но и для целого ряда компаний.

## Тестирование на проникновение

Перед тем, как приступить к изучению методологий тестирования на проникновение веб-приложений, необходимо понять, что из себя представляет тестирование на проникновение.

### 1.1.Определение тестирования на проникновение

Тестирование на проникновение, или penetration test, - это попытка оценить безопасность IT-инфраструктуры путем безопасного использования уязвимостей. Эти уязвимости могут существовать в операционных системах, службах и приложениях, неправильных конфигурациях или опасном поведении конечного пользователя. Такие оценки также полезны для проверки эффективности защитных механизмов, а также соблюдения конечным пользователем политик безопасности [2].

Тестирование на проникновение обычно выполняется с использованием ручных или автоматизированных технологий для систематического взлома серверов, конечных точек, веб-приложений, беспроводных сетей, сетевых устройств, мобильных устройств и других потенциальных точек воздействия. После того, как уязвимости были успешно использованы в конкретной системе, тестировщики могут попытаться использовать скомпрометированную систему для запуска последующих эксплойтов на других внутренних ресурсах, в частности, пытаясь постепенно достичь более высоких уровней безопасности и более глубокого доступа к электронным активам и информации посредством повышения привилегий.

Информация о любых уязвимостях безопасности, успешно использованных с помощью тестирования на проникновение, обычно собирается и представляется ИТ-менеджерам и администраторам сетевых систем, чтобы помочь этим профессионалам сделать стратегические выводы и расставить приоритеты в соответствующих усилиях по исправлению. Основная цель тестирования на проникновение - измерить возможность компрометации систем или конечного пользователя и оценить любые связанные последствия, которые такие инциденты могут иметь для задействованных ресурсов или операций

### 1.2.Важность тестирования на проникновение

#### 1.2.1.Выявление и приоритезация рисков безопасности

Тестирование на проникновение оценивает способность организации защитить свои сети, приложения, конечные точки и пользователей от внешних или внутренних попыток обойти меры безопасности и получить несанкционированный или привилегированный доступ к защищенным активам.

Для всех организаций важно периодически оценивать и тестировать уязвимости системы безопасности, чтобы лучше оценивать риски и быть готовыми к обнаружению, предотвращению и реагированию на угрозы по мере их возникновения, что в целом, улучшает общее состояние безопасности.

В ходе тестирования на проникновение специалисты имитирую реальную атаку, с целью достичь заранее поставленных целей, например, получение доступа к несанкционированной информации посредством украденных учетных данных пользователя.

Организации могут использовать тестирование на проникновение с различными целями:

* Управление уязвимостями;
* Избежание затрат на простой сети;
* Соответствие нормативным требованиям;
* Сохранение корпоративного имиджа и лояльности клиентов.

#### 1.2.2. Управление уязвимостями

Тесты на проникновение предоставляют подробную информацию о реальных, эксплуатируемых угрозах безопасности.

При выполнении тестирования на проникновение, специалист имеет возможность заранее определить, какие уязвимости являются наиболее критическими, а какие менее значительными. Это позволяет организациям более разумно определять приоритеты исправлений уязвимостей, применять необходимые меры и эффективно распределять ресурсы безопасности.

Тесты на проникновение добавляют больше бизнес-контекста, проверяя потенциальную угрозу, ранжируя опасность уязвимостей, на основе которых можно фактически использовать для получения доступа в вашей среде.

Управление атаками использует данные сканирования и тестирования на проникновение, чтобы определить, как злоумышленник может перемещаться по системе, используя различные уязвимости для получения доступа к критически важным бизнес-активам.

#### 1.2.3. Использование упреждающего подхода к безопасности

В наши дни нет единого решения для предотвращения взлома. Организации теперь должны иметь набор защитных механизмов и инструментов безопасности, включая криптографию, антивирус, решения SIEM и программы IAM, и это лишь некоторые из них. Однако даже с этими жизненно важными инструментами безопасности сложно найти и устранить каждую уязвимость в ИТ-среде. Тестирование на проникновение использует упреждающий подход, выявляя слабые места, чтобы организации знали, какое исправление необходимо, и следует ли внедрять дополнительные уровни зашиты.

#### 1.2.4. Проверка работоспособности и выявление сильных и слабых сторон своей программы безопасности

Без должного понимания и видимости среды в целом, изменение системы безопасности может повлечь за собой уничтожение тех компонентов системы, которые не были проблематичными.

Тесты на проникновения позволяют определить не только то, что не работает, но и служат с целью проверки качества, показывая, какие политики наиболее эффективны и какие инструменты обеспечивают максимальную рентабельность инвестиций. Благодаря этому, организации могут разумно распределять ресурсы безопасности, обеспечивая их доступность там, где они в большей степени необходимы.

#### 1.2.5. Повышение уверенности в стратегии безопасности

Регулярная проверка инфраструктуры безопасности и команды безопасности позволяет организациям не задумываться о том, как будет выглядеть атака на их ресурс и как им придется реагировать на эту атаку. Тестирование на проникновение позволяет опробовать ряд этих атак и выстроить стратегию безопасности организации, при которой она будет готова к подобным атакам.

#### 1.2.6. Соответствие нормативным требованиям

Тестирование на проникновение помогает организациям решать общие аспекты аудита и соблюдения нормативных требований и передовых отраслевых практик. Используя инфраструктуру организации, тестирование на проникновение может продемонстрировать, как злоумышленник может получить доступ к конфиденциальным данным. По мере роста и развития стратегий атак периодическое обязательное тестирование гарантирует, что организации могут оставаться на шаг впереди, выявляя и исправляя слабые места безопасности, прежде чем они могут быть использованы.

Кроме того, для аудиторов эти тесты могут также подтвердить, что другие обязательные меры безопасности действуют или работают должным образом. Подробные отчеты, которые генерируют тесты на проникновение, могут помочь организациям проиллюстрировать текущую должную осмотрительность для поддержания необходимых мер безопасности.

## 2. Этапы тестирования на проникновение

Тестирование на проникновение - это тщательный, хорошо продуманный проект, состоящий из нескольких этапов:

### 2.1. Планирование и подготовка

Перед началом проведения тестирования на проникновение, клиентам и тестировщикам необходимо согласовать цели теста, что позволит последним выстроить критерии тестирования: типы тестов, осведомленность о проведении тестов на проникновение, исходные данные для проведения тестов и прочие. Это обеспечивает корректное выполнение теста должным образом.

### 2.2 Сбор информации

На этом этапе команды проводят различные виды разведки своей цели. С технической стороны такая информация, как IP-адреса, может помочь определить информацию о брандмауэрах и других соединениях. С личной точки зрения такие простые данные, как имена, должности и адреса электронной почты, могут иметь большую ценность.

Кроме того, перед эксплуатацией системы группы тестирования на проникновение должны искать слабые места в среде. Эта фаза обнаружения, часто называемая следом, включает сбор как можно большего количества информации о целевых системах, сетях и их владельцах без попытки проникнуть в них.

### 2.3. Попытки проникновения и эксплуатация

После получения информации о цели, пен-тестеры могут начать предприниматься попытки проникнуть в среду, используя слабые места безопасности и демонстрируя, насколько глубоко они могут проникнуть в систему.

### 2.4. Анализ и отчетность

Пен-тестеры должны создать отчет, включающий подробную информацию о каждом этапе процесса, с указанием того, что было использовано для успешного проникновения в систему, какие недостатки безопасности были обнаружены, обнаружена другая соответствующая информация и рекомендации по исправлению.

### 2.5. Очистка и исправление

После осуществления тестирования на проникновения пен-тестерам необходимо не оставлять следы тестирования и удалять любые артефакты, которые были использованы в тесте, по причине того, что они могут быть использованы настоящим злоумышленником в будущем. Организации на данном этапе могут начать вносить необходимые исправления для устранения уязвимостей в своей инфраструктуре безопасности.

### 2.6. Повторное тестирование

Для проверки успешности и эффективности внесенных изменений, лучшим вариантом выступает проведение повторного тестирования, что также, возможно, может выявить новые уязвимые места системы.

## 3. Инструменты для тестирования на проникновение

Для взлома систем, чтобы сделать свои попытки наиболее успешными злоумышленники используют различные инструменты. Аналогичная ситуация обстоит и пен-тестерами. Программное обеспечение для тестирования позволяет тестировщикам сосредоточиться на нестандартном мышлении при взломе систем, беря на себя задачи, требующие больших временных затрат.

Тестирование на проникновение обычно выполняется с использованием набора инструментов, которые предоставляют множество функций. Часть из них имеют открытый исходный код, а другие являются коммерческими. Некоторые из этих инструментов аналогичны инструментам, используемым злоумышленниками, что позволяет точно воспроизвести атаку. Другие подчеркивают потребности этичного хакера, позволяя сделать больший акцент на функциях, которые отдают приоритет конечной цели проверки слабых мест безопасности без воздействия на производственную среду и уделяют приоритетное внимание исправлению.

Тестирование на проникновение веб-приложений

Тесты на проникновение веб-приложений исследуют общую безопасность и потенциальные риски веб-приложений, включая ошибки кодирования, нарушение аутентификации или авторизации, а также уязвимости, связанные с внедрением. Все вышеперечисленные этапы тестирования на проникновение будут рассмотрены на примере выбранной методологии с использованием практических методов пентестинга.

## 4. Правовые аспекты тестирования на проникновение

### 4.1. ФСТЭК

* В приказе ФСТЭК России от 18.02.2013 N 21 (ред. от 14.05.2020) "Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных" в пункте 11 главы «Состав и содержание мер по обеспечению безопасности персональных данных» содержится, касательная тестирования на проникновение: «В случае определения в соответствии с Требованиями к защите персональных при их обработке в информационных системах персональных данных, утвержденными постановлением Правительства Российской Федерации от 1 ноября 2012 г. N 1119, в качестве актуальных угроз безопасности персональных данных 1-го и 2-го типов дополнительно к мерам по обеспечению безопасности персональных данных, указанным в пункте 8 настоящего документа, могут применяться следующие меры: проверка системного и (или) прикладного программного обеспечения, включая программный код, на отсутствие недекларированных возможностей с использованием автоматизированных средств и (или) без использования таковых; тестирование информационной системы на проникновения; использование в информационной системе системного и (или) прикладного программного обеспечения, разработанного с использованием методов защищенного программирования.» [3].
* Приказ ФСТЭК РОССИИ от 25 декабря 2017 г. N 239 «Об утверждении требований по обеспечению безопасности значимых объектов критической информационной инфраструктуры Российской Федерации», раздел «Внедрение организационных и технических мер по обеспечению безопасности значимого объекта и ввод его в действие» пункт 12.6: «Анализ уязвимостей значимого объекта проводится в целях выявления недостатков (слабостей) в подсистеме безопасности значимого объекта и оценки возможности их использования для реализации угроз безопасности информации. При этом анализу подлежат уязвимости кода, конфигурации и архитектуры значимого объекта. Анализ уязвимостей проводится для всех программных и программно-аппаратных средств, в том числе средств защиты информации, значимого объекта. При проведении анализа уязвимостей применяются следующие способы их выявления… д) тестирование на проникновение в условиях, соответствующих возможностям нарушителей, определенных в модели угроз безопасности информации. Применение способов и средств выявления уязвимостей осуществляется субъектом критической информационной инфраструктуры с учетом особенностей функционирования значимого объекта. Допускается проведение анализа уязвимостей на макете (в тестовой зоне) значимого объекта или макетах отдельных сегментов значимого объекта. Анализ уязвимостей значимого объекта проводится до ввода его в эксплуатацию на этапах, определяемых субъектом критической информационной инфраструктуры. В случае выявления уязвимостей значимого объекта, которые могут быть использованы для реализации (способствовать возникновению) угроз безопасности информации, принимаются меры, направленные на их устранение или исключающие возможность использования (эксплуатации) нарушителем выявленных уязвимостей. По результатам анализа уязвимостей должно быть подтверждено, что в значимом объекте, отсутствуют уязвимости, как минимум содержащиеся в банке данных угроз безопасности информации ФСТЭК России, указанном в пункте 11.1 настоящих Требований, или выявленные уязвимости не приводят к возникновению угроз безопасности информации в отношении значимого объекта.» [4].

В Положении Банка России от 09.06.2012 N 382-П (ред. от 07.05.2018) "О требованиях к обеспечению защиты информации при осуществлении переводов денежных средств и о порядке осуществления Банком России контроля за соблюдением требований к обеспечению защиты информации при осуществлении переводов денежных средств" (Зарегистрировано в Минюсте России 14.06.2012 N 24575) (с изм. и доп., вступ. в силу с 01.01.2020) во второй главе «Требования к обеспечению защиты информации при осуществлении переводов денежных средств» включает в себя следующую информацию:

«Оператору по переводу денежных средств, оператору услуг платежной инфраструктуры на стадиях создания и эксплуатации объектов информационной инфраструктуры необходимо обеспечить: …ежегодное тестирование на проникновение и анализ уязвимостей информационной безопасности объектов информационной инфраструктуры.» [5].

Для проведения тестирования на проникновение между организацией и субподрядчиком должен быть составлен договор, который на основании ст. 432 ГК РФ, считается заключенным, если сторонами достигнуто соглашение по всем существующим условия. Наиболее важными условиями договора о тестировании на проникновение является четкое описание объекта тестирования и перечня действий, которые необходимо произвести и которые можно производить с учетом всех правовых норм, а также условия использования инструментария тестирования на проникновение с учетом статьи 273 УК РФ «Создание, использование и распространение вредоносных компьютерных программ». Помимо этого, между заказчиком и исполнителем тестирования на проникновение в случае необходимости может заключаться дополнительное соглашение о неразглашении конфиденциальной информации. В случае банковской тайны и персональных данных клиента банка, организатору и исполнителю тестирования на проникновение необходимо получить разрешение от клиентов банка, чья конфиденциальная информация может стать известна третьей стороне [6].

## 5. Методологии тестирования на проникновение

Как в большинстве сфер жизни, так и в тестировании на проникновение систематичность, а именно стандартная методология испытаний, является одним из ключевых факторов. Отсутствие методики означает отсутствие однотипности, что вызывает хаотичность использования в тестировании различных инструментов взлома, а также сумбурность в ожидаемых целях и результатах.

Методология, в контексте пентестинга - набор правил, действий, процедур, которые используются при работе с программами для проверки информационной безопасности.

### 5.1. OSSTMM 3 – The Open Source Security Testing Methodology Manual

OSTMM 3 - методология тестирования операционной безопасности физических объектов, взаимодействия людей и всех форм связи, таких как беспроводная, проводная, аналоговая и цифровая, разработанная при поддержке Института безопасности и открытых методологий (ISECOM).

Методология OSSTMM включает в себя следующие разделы:

1. Исходные сведения о документе
2. Определение тестирования на проникновение, границ тестирования, процессов и ролей
3. Анализ безопасности объектов
4. Показатели безопасности тестируемого объекта
5. Анализ социальных процессов в персонале объекта тестирования
6. Этап тестирования
7. Тестирование устойчивости персонала к социальной инженерии
8. Тестирование безопасности физической инфраструктуры
9. Тестирования безопасности беспроводных технологий
10. Тестирование безопасности телекоммуникационных технологий
11. Тестирование безопасности данных
12. Рекомендации по следованию национальным стандартам и соглашениям
13. Подготовка отчета о тестировании

В методологии описывается визуальный показатель безопасности, указывающий основные категории информационной безопасности, которые должны быть проверены на соответствие требованиям методики:

* Информационная безопасность;
* Безопасность социальных процессов;
* Безопасность информационных процессов;
* безопасность Интернет-технологий;
* безопасность каналов связи;
* безопасность беспроводных технологий;
* безопасность физической инфраструктуры;

Методология OSSTMM не предоставляет как таковую классификацию уязвимостей, понятие «уязвимость» расценивается как ограничение безопасности – ошибка или дефект, в результате которой происходит нарушение предоставления доступа авторизированным и неавторизированным пользователям к ресурсам.

К плюсам методологии можно отнести высокую степень формализованности и структурности документа, определяющего регламент практически всех аспектов тестирования на проникновение преимущественно компьютерных сетей[7].

### 5.2. PTES (Penetration Testing Execution Standard)

PTES – стандарт тестирования на проникновение, разработанный международной группой независимых экспертов в области информационной безопасности. Стандарт проведения тестирования на проникновение состоит из семи 7 основных разделов, охватывающих все, что связано с тестом на проникновение - от первоначального общения и обоснования пентеста до этапов сбора информации и моделирования угроз, исследования уязвимостей, эксплуатации, пост-эксплуатации и отчетности.

1. Первоначальное общение – раздел дает представление и объяснение о доступных методах и инструментах для успешного предварительного этапа тестирования на проникновение.
2. Сбор информации – раздел описывает основные действия по сбору данных, которые могут быть использованы тестировщиком при создании плана по тестированию на проникновение.
3. Моделирование угроз – в разделе определяется подход к моделированию угроз, необходимый для правильного выполнения тестирования на проникновение. С целью возможности общего применения стандарт не использует конкретную модель угроз, а фокусируется на двух ключевых элементах традиционного моделирования угроз – активах и злоумышленнике, каждый из которых разбит на бизнес-активы, бизнес-процессы, категории угроз и их возможности.
4. Анализ уязвимостей – раздел, описывающий различные подходы по обнаружению и поиску уязвимостей
5. Эксплуатация – в разделе описываются подходы к определению основной точки входа в организацию для выявления ценных целевых активов, а также методы контрмер.
6. Пост-эксплуатация – раздел описывает методы, предназначенные для помощи тестировщику идентифицировать и документировать конфиденциальные данные, определять параметры конфигурации, каналы связи и отношения с другими сетевыми устройствами, которые могут использоваться для получения дальнейшего доступа к сети
7. Составление отчетов – раздел описывает базовые критерии отчетности о тестировании на проникновение

Также к данному стандарту прикладывается техническое руководство PTES Technical Guidelines по использованию различных методов и инструментов при тестировании, имеющий следующую структуру:

1. Инструменты тестирования:
2. Сбор информации об объекте тестирования
3. Анализ уязвимостей
4. Эксплуатация уязвимостей
5. Пост-эксплуатация
6. Отчетность

Стандарт PTES может быть использован как на этапе предварительной оценки защищенности готовых объектов, так и на этапе разработки объектов [8].

### 5.3. OWASP Web Security Testing Guide

Методология разработана некоммерческим фондом OWASP, который занимается повышением безопасности программного обеспечения. Фонд OWASP, за счет проектов с открытым исходным кодом, огромного количества заинтересованных людей и ведущих образовательных и учебных конференция, является одним из главных источников по обеспечению безопасности Интернета для всех специалистов, принимающих участие в проектировании, разработке, развертывании и проверки на безопасность веб-сервисов и веб-приложений.

Проект тестирования OWASP Testing Guide разрабатывался продолжительное количество лет с целью помочь людям разобраться, что, как, почему, где и когда тестировать веб-приложения. Проект предоставляет полную структуру тестирования веб-приложений, позволяющий тестировщикам использовать ее в качестве шаблона для создания собственных программ тестирования или оценки процессов других людей. Руководство описывает как общую структуру тестирования, так и методы, необходимые для реализации этой структуры на практике.

Руководство организовано следующим образом:

* Введение – охватывает предварительные условия тестирования веб-приложений и объем тестирования, а также принципы успешного тестирования, методы тестирования, методы составления отчетов и бизнес-кейсы для тестирования безопасности
* The OWASP Testing Framework (тестовый фреймворк owasp) – в этой главе представлена структура тестирования OWASP, объясняются ее методы и задачи в отношение различных фаз жизненного цикла разработки программного обеспечения.
* Web Application Security Testing (Тестирование безопасности веб-приложений) – в этой главе описываются конкретные методы тестирования определенных уязвимостей с помощью проверки кода и тестирования на проникновение [9].

### 5.4. Выбор методологии

Все описанные выше методологии являются общепринятыми при тестировании на проникновение, однако в рамках этой выпускной квалификационной работы, ряд из них имеет свои недостатки:

OSSTMM - К минусам методологии OSTMM можно отнести недостаток информации, касательно практических методов и инструментов тестирования на проникновение, а также отсутствия строгой классификации уязвимостей.

PTES - В целом PTES имеет хорошую структуру и большой набор методов тестирования, однако в отличие от OWASP Web Security Testing Guide не заточена под специфику тестирования веб-приложений. В связи с тем, что OWASP Web Security Tetsting Guide был разработан конкретно под тестирование на проникновение веб-приложений, то для более детально рассмотрения была выбрана именно эта методология.

# ГЛАВА 2. OWASP WEB SECURITY TESTING GUIDE 4.2

## Комплексный подход к тестированию безопасности веб-приложений по OWASP Web Security Testing Guide

Прежде, чем приступить к детальному разбору конкретных методов и методологии тестирования на проникновение, хотелось мы отметить, что процесс тестирования безопасности веб-приложения – это не только пентестинг, это обширный процесс, включающий в себя ряд различных работ, которые должны быть выполнены на разных этапах жизненного цикла разработки и эксплуатации приложения. Только в таком случае тестирование на проникновение оказывает должный эффект, и в целом, тестирование безопасности позволяет сделать полную и корректную оценку безопасности приложения, что позволяет организациям в дальнейшем правильно использовать свои ресурсы и предпринимать меры по устранению уязвимостей.

В силу такого подхода, считаю необходимым дать краткий обзор комплекса мер и решений, которые требуется выполнить на всех этапах жизненного цикла разработки веб-приложения, по методологии OWASP Web Security Testing Guide.

Начать, следует с определений.

Перед тем, как тестировать какой-либо продукт, необходимо сначала понять, что такое тестирование, какие у него цели, и что конкретно требуется тестировать.

Тестирование – процедура, предназначенная для определения качества производительности или надежности чего-либо. В рамках руководства OWASP Testing Guideтестирование определяется как процесс сравнения состояния системы или приложения с набором критериев.

Документ позволяет организациям понять из чего состоит программа тестирования и какие шаги необходимо предпринять для создания и эксплуатации современной комплексной программы тестирования веб-приложений.

Эффективная программа тестирования должна включать в себя следующие компоненты, проверяющие следующее:

* Люди - для обеспечения адекватного образования и осведомленности;
* Процесс - для обеспечения наличия адекватных политик и стандартов, а также того, что люди умеют следовать этим политикам;
* Технология - чтобы гарантировать, что процесс был эффективным при его реализации

Целостный подход позволяет выявить тестировщикам управляющие и операционные уязвимости, которые могут присутствовать в системе, на раннем этапе, до проявления этих уязвимостей в дефектах технологии и дать полную и относительно точную оценку состояния безопасности веб-приложения.

## 2. Глава Introduction (Введение)

Первая глава дает специалистам базовые понятия, используемые при тестировании веб-приложений, от которых следует отталкиваться.

Глава состоит из следующих разделов:

1. Principles of Testing (Принципы тестирования) –

Дает базовые представления о принципах тестирования безопасности таких как:

* Программное обеспечение для оценки безопасности веб-приложений полезно, но является неэффективным и незрелым для глубокой оценки и обеспечения хорошего тестового покрытия.
* Интегрирование безопасности на каждом этапе жизненного цикла разработки веб-приложения обеспечивает целостный подход к безопасности приложения
* Тестирование безопасности необходимо проводить часто и заблаговременно
* Использование автоматизации тестирования
* Должное понимание необходимого уровня безопасности для проекта позволяет правильно использовать ресурсы безопасности
* Хорошее тестирование безопасности требует выхода за рамки ожидаемого и предполагает у тестировщика способности мышления, как у злоумышленника.
* Хорошая программа безопасности должна включать точную документацию приложения
* Использование правильных инструментов тестирования для различных задач
* Поверхностная оценка безопасности приложения, также опасна, как и отсутствие оценки безопасности. Детали важны.
* Использование исходного кода приложения, когда он доступен
* Важно отслеживать результаты заданий по тестированию и разрабатывать показатели, которые позволят выявить тенденции безопасности приложений
* Документирование отчета о проведенных испытаниях

1. Testing Techniques Explained (Объяснение методов тестирования)

В разделе представлен общий обзор различных методов тестирования, которые можно использовать при построении программы тестирования безопасности. Раздел позволяет выявить преимущества и недостатки методов, которые необходимо учитывать.

* 1. Manual Inspections and Reviews (Ручные проверки и обзоры)

Ручные проверки - это проверки, проводимые людьми, которые обычно проверяют влияние людей, политик и процессов на безопасность. Ручные проверки могут также включать проверку технологических решений, таких как архитектурные проекты. Обычно они проводятся путем анализа документации или собеседований с разработчиками или владельцами системы.

*Преимущества:*

* Не требует поддерживающих технологий
* Может применяться в самых разных ситуациях
* Гибкий
* Способствует совместной работе
* В начале SDLC

*Недостатки*

* Может занять много времени
* Вспомогательный материал не всегда доступен
* Требует значительного опыта и навыка у специалиста, чтоб быть эффективной
  1. Threat Modeling (Моделирование угроз)

Позволяет разработчикам систем задуматься об угрозах безопасности, с которыми могут столкнуться системы и приложения, что делает моделирование угроз некой оценкой рисков для приложения и позволяет проектировщикам создать стратегии смягчения потенциальных уязвимостей. Для разработки модели угроз мы рекомендуем использовать простой подход, соответствующий стандарту NIST 800-30 для оценки рисков. Этот подход включает в себя:

* Декомпозиция приложения – использование процесса ручных проверок для понимания работы приложения, его активов, функциональности и возможности подключения
* Определение и классификация активов - классификация активов на материальные и нематериальные активы и по важности для бизнеса.
* Изучение потенциальных уязвимостей.
* Изучение потенциальных угроз – разработка реалистичного представления о потенциальных векторах атак с точки зрения злоумышленника, с использованием сценариев угроз или деревья атак.
* Создание стратегий смягчения – разработка элементов управления смягчением для каждой из угроз, которые считаются реалистичными.

*Преимущества:*

* Практический взгляд злоумышленника на систему
* Гибкий
* В начале SDLC

*Недостатки:*

* Хорошие модели угроз не означают автоматически хорошее программное обеспечение
  1. Source Code Review (Обзор исходного кода) – это процесс ручной проверки исходного кода веб-приложения на наличие проблем с безопасностью.

*Преимущества:*

* Полнота и эффективность
* Точность
* Быстро (для грамотных рецензентов)

*Недостатки:*

* Требуются высококвалифицированные разработчики, осведомленные о безопасности.
* Может пропустить проблемы в скомпилированных библиотеках
* Невозможно легко обнаружить ошибки времени выполнения
* Фактически развернутый исходный код может отличаться от анализируемого
  1. Penetration Testing (Тестирование на проникновение) – тестирования системы или приложения для обнаружения уязвимостей безопасности без знания внутренней работы самой цели.

*Преимущества:*

* Может быть быстрым (и, следовательно, дешевым)
* Требует относительно меньшего набора навыков, чем проверка исходного кода
* Проверяет код, который на самом деле отображается

*Недостатки:*

* Слишком поздно в SDLC
* Только испытания на лобовое столкновение

1. The Need for a Balanced Approach (Необходимость сбалансированного подхода) – В разделе описывается преимущество использования сбалансированного подхода в тестировании, в котором используются все методы тестирования.
2. Deriving Security Test Requirements (Получение требований к тестированию безопасности) – В разделе описывается, каким образом следует документировать требования к тестированию безопасности и расчета применимых стандартов и правил, положительных и отрицательных требований к приложению, а также и как данные тестирования безопасности могут быть использованы для эффективного управления рисками безопасности программного обеспечения.
3. Security Tests Integrated in Development and Testing Workflows (Тесты безопасности, интегрированные в рабочие процессы разработки и тестирования) – В разделе описывается, каким образом следует проводить тестирование после сборки приложения в целом
4. Security Test Data Analysis and Reporting (Анализ данных тестирования безопасности и составление отчетов) – В разделе рассказывается о целях для показателей и измерений тестирования безопасности, а также о требованиях к составлению отчетности

## 3. Глава The OWASP Testing Framework

Во второй главе The OWASP Testing Framework (Платформа тестирования OWASP) описывается эталонная структура тестирования, состоящая из методов и задач, подходящих на различных этапах жизненного цикла разработки программного обеспечения, которую можно создать в организации. Структура считается гибкой, позволяющей конкретной организации расширить и адаптировать ее под свои решения.

Глава включает в себя следующие разделы:

1. Phase 1 Before Development Begins - Фаза 1, До начала разработки

В разделе описываются действия, которые необходимо выполнить перед разработкой приложения:

* Определение SDLC
* Проверка политик и стандартов
* Разработка критериев измерения и показателей и обеспечение отслеживаемости

1. Phase 2 During Definition and Design - Фаза 2, В процессе описания и проектирования.

В разделе описываются действия, которые необходимо предпринять на этапе проектирования:

* Проверка требований безопасности
* Обзор дизайна и архитектуры
* Создание и проверка моделей UML
* Создание и проверка моделей угроз

1. Phase 3 During Development – Фаза 3, В процессе разработки.

В разделе описываются действия, которым рекомендуется следовать в процессе разработки приложения:

* Прохождение кода **-** пошаговое руководство по коду с разработчиками, во время которого разработчики объясняют логику и последовательность реализованного кода
* Проверка кода

1. Phase 4 During Deployment – Фаза 4, в процессе развертывания приложения.

В разделе описываются действия, предпринимаемые на этапе развертывания приложения:

* Тестирование приложений на проникновение
* Тестирование управления конфигурацией

1. Phase 5 During Maintenance and Operations - Этап 5, Во время технического обслуживания и эксплуатации.

В главе описываются действия, необходимые на этапе обслуживания и эксплуатации приложения:

* Проведение анализов оперативного управления
* Проведение периодических проверок работоспособности
* Обеспечение проверки изменений

## 4. Глава Web Application Security Testing (Тестирование безопасности веб-приложений)

**4.1. Обоснование выбранных методов**

Методология OWASP testing guide включает в себя порядка 110 различных методов сбора информации и тестирования на проникновение веб-приложений, разделенных на следующие категории:

1. Сбор информации
2. Тестирование управления конфигурацией и развертыванием
3. Тестирование управления идентификацией
4. Проверка аутентификации
5. Проверка авторизации
6. Тестирование управления сеансом
7. Проверочное тестирование входных данных
8. Тестирование на обработку ошибок
9. Тестирование на слабую криптографию
10. Тестирование бизнес-логики
11. Тестирование на стороне клиента

В моей работе будет рассмотрена часть методов OWASP Web Security Testing Guide, выбранных с учетом одного из наиболее популярных справочных документов, фактически став стандартом безопасности веб-приложений, OWASP Top-10 [10].

OWASP Top-10 представляет собой список из 10 самых распространенных уязвимостей веб-приложений, составленный на основе данных, полученных от организаций, специализирующихся на безопасности веб-приложений, отраслевых исследованиях, а также данных, обнаруженных в 100000 реальных приложениях.

Эти уязвимости включают в себя:

1. Инъекции– уязвимости, возникающие в результате отправки непроверенных данных интерпретатору в составе запросов или команд (SQL, NoSQL, OS и LDAP).
2. Недостатки аутентификации – уязвимости, вызванные неправильной реализацией функций аутентификации и управления сессиями, приводящие к компрометации ключей, токенов сессий, паролей.
3. Разглашение конфиденциальных данных – уязвимости, вызванные плохой защитой конфиденциальной информации (например, отсутствие шифрования), которую может получить злоумышленник и использовать в своих целях.
4. Внешние сущности XML (XXE) – уязвимости, вызванные устаревшими или плохо настроенными XML-процессорами, которые обрабатывают ссылки на внешний сущности внутри документов.
5. Недостатки контроля доступа – уязвимости, вызванные недостаточным или некорректным контролем действий, разрешенных аутентифицированным пользователям.
6. Некорректная настройка параметров безопасности – уязвимости, возникающие при использовании стандартных параметров безопасности, неполной конфигурации, некорректных заголовков HTTP запросов и подробных сообщениях об ошибках.
7. Межсайтовый скриптинг – уязвимости, которая появляются при добавлении приложением на страницу непроверенных, непреобразованных данных, позволяя злоумышленнику украсть сессии пользователей, перенаправить их на вредоносный сайт, подменить страницы.
8. Небезопасная десериализация - уязвимости, возникающие в результате небезопасной или ошибочной десериализации, которые могут привести к удаленному выполнению кода.
9. Использование компонентов с известными уязвимостями – библиотеки, фреймворки и программные модули запускаются с привилегиями приложением, в следствие чего использование таких компонентов с известными уязвимости нарушает безопасность приложения и может привести к плачевным последствиям.
10. Недостатки журналирования и мониторинга – отсутствие или неэффективное использование системы реагирования на инциденты, позволяет злоумышленникам совершить ряд вредоносных действий, после чего скрыть свое присутствие в системе.

При выборе методов учитывались следующие критерии OWASP Top-10:

* Сложность эксплуатации
* Распространенность уязвимости
* Сложность обнаружения
* Технические последствия

В связи с этим, такие угрозы, как небезопасная десериализация и недостатки журналирования и мониторинга в работе не рассмотрены.

## 5. Методы тестирования на проникновение веб-приложений

### 5.1.Сбор информации

#### 5.1.1. Определение веб-сервера

Одной из важных задач тестирования безопасности веб-приложения является идентификация веб-сервера, а именно определение типа и версии сервера, на котором работает приложение. Информация о типе и версии сервера может позволить тестировщикам определить известные уязвимости для данной версии и использовать соответствующие эксплойты при тестировании.

Задача тестирования состоит в определении версии и типа веб-сервера приложения при помощи таких методов, как захват баннера, отправка на сервер неверных запросов и использование автоматических средств сканирования. Цель всех методов заключается в получении некоторого ответа от веб-сервера, который можно сравнить с базой данных известных ответов и поведения, и сопоставить с известными типами серверов.

* Захват баннера

Метод реализуется путем отправки HTTP-запросов на сервера и изучение заголовков ответов от него. Реализуется посредством различных инструментов, включая telnet для HTTP-запросов или openssl для запросов через SSL.

* Отправка искаженных запросов

Метод реализуется путем отправки на веб-сервер искаженных запросов и изучения ответов от сервера на ошибки, а в случае если такие ответы не настроены, то изучения страниц ошибок по умолчанию.

* Использование инструментов автоматического сканирования

Автоматические средства сканирования - инструменты, позволяющие отправлять запросы на сервер, описанные выше, а также более специфичные для серверов, и сравнивать ответы с огромными базами известных ответов, выполняя эту работу намного быстрее и точно, чем при ручном тестировании.

Часто используемые инструменты сканирования:

* Netcraft - онлайн-инструмент, который сканирует веб-сайты на предмет информации, включая веб-сервер.
* Nikto - инструмент сканирования командной строки с открытым исходным кодом.
* Nmap - инструмент командной строки с открытым исходным кодом, который также имеет графический интерфейс Zenmap[11].

#### 5.1.2. Проверка содержимого веб-страницы на предмет утечки информации

Зачастую программистам во время разработки следует включать подробные комментарии и метаданные в свой код, однако эти комментарии и данные могут раскрыть внутреннюю информацию, которая не должна быть доступна потенциальным злоумышленникам.

Для современных веб-приложений использование клиентского Javascript (ReactJs, AngularJs, Vue) для построения интерфейса становится все более популярным. Нередко программисты прямо кодируют конфиденциальную информацию в переменных JavaScript во внешнем интерфейсе. Эта информация может включать: закрытые ключи API, внутренние IP-адреса, конфиденциальные маршруты (маршрут к скрытым страницам администратора) и учетные данные.

* Просмотр комментариев и метаданных;

Комментарии HTML часто используются разработчиками для включения отладочной информации приложения и нередко программисты забывают их удалить и оставляют в производственной среде. Для поиска таких комментариев в коде HTML страницы следует искать строки, начинающиеся с «*<! --*».

* Определение кода и файлов JavaScript;

В данном методе следует проверять исходный код HTML и искать код между тегами <script> и </script>, а также определить внешние файлы JavaScript для проверки кода на наличия утечки конфиденциальной информации.

#### 5.1.3. Составление карты веб-приложения

Для тщательного тестирования безопасности веб-приложения тестировщику первоначально необходимо понять структуру и процессы этого приложения.

Прежде чем тестировать безопасность веб-приложения тестировщику необходимо первоначально понять структуру, процессы и принципы функционирования этого приложения, за счет отслеживания распространения данных по программе, полученных от внешнего источника, поиска каждого возможного исполняемого пути программы и анализ поведения приложения при изменении одних и тех же параметров при параллельном запуске приложения.

При тестировании методом черного ящика достаточно проблематично протестировать всю кодовую базу приложения по ряду причин:

* Тестировщик не имеет доступа к коду;
* Проверка всего кода требует огромного количества времени;

Существует несколько подходов к тестированию и измерению покрытия кода:

* Path – проверка всех путей приложения с использованием комбинаторных и граничных тестов значений;
* Data-flow – тестирование переменных, посредством внешнего взаимодействия (от лица пользователя), нацелен на передачи, преобразовании и использовании данных при работе приложения;
* Race – тестирование нескольких экземпляров приложения, манипулирующих одним и теме же данными.

«Паук» - инструмент, используемый для автоматического обнаружения ресурсов на определенном веб-сайте.

### 5.2. Тестирование конфигурации и управления развертывания

#### 5.2.1. Тестирование HTTP методов

HTTP предоставляет ряд методов работы с веб-сервером, наиболее распространёнными методами на сегодня являются GET и POST, однако существуют и другие методы, которые могут использоваться как разработчиками, так и злоумышленниками в гнусных целях при неправильной настройке веб-сервера. Целью данного теста является определение поддерживаемых методов HTTP, проверка возможности обхода контроля доступа, проверка XST-уязвимости, а также методов переопределения запросов HTTP.

Для выполнения этого теста предварительно необходимо выяснить, какие методы HTTP поддерживаются исследуемым веб-сервером. Сделать это можно посредством отправки запроса с методом OPTIONS, а также при помощи Nmap скрипта по типу: *nmap -p 443 --script http-methods --script-args http-methods.url-path='/index.php' localhost*

Тестирование метода PUT:

1. При помощи прокси захватить базовый запрос цели;
2. Изменить метод запроса на PUT, добавить к запросу файл и отправить на сервер;

Если ответ от сервера будет с кодами 2хх или 3хх и запрос GET на получение отправленного файла будет успешен, то приложение уязвимо.

Обход контроля доступа:

Для теста необходимо найти страницу, которая имеет ограничение доступа и при GET запросе возвращает код 302 (перенаправление), а затем выполнять запросы к этой странице, используя различные методы, такие как HEAD, POST, PUT, а также произвольные методы как BILBAO, FOOBAR, CATS. Если приложение отвечает кодом 200, то злоумышленник может обойти аутентификацию и авторизацию.

Тестирование потенциальной межсайтовой трассировки:

Метод TRACE, предназначенный для тестирования и отладки, дает указание серверу отразить полученное сообщение обратно клиенту. Метод может быть использован в сценариях с кражей учетных данных.

Тестирование замены методов HTTP

Некоторые веб-фреймворки предоставляют возможность переопределить фактические HTTP-методы, запрещенные на веб-сервере, в запросе, эмулируя отсутствующие глаголы, передавая специальный заголовок в запросе. Следующие заголовки могут использоваться для выполнения этой цели:

* X-HTTP-метод;
* X-HTTP-метод-переопределение;
* X-метод-переопределение;

Для проверки следует воспроизводить запросы с добавлением этих альтернативных заголовков для переопределения методов.

### 5.3. Тестирование управления идентификацией

#### 5.3.1.Определение ролей пользователей

Веб-приложения имеют ряд различных функций и служб, доступ к которым зависит от типа пользователя, например, администратор, аудитор, инженер службы поддержки. Для правильного предоставления доступа к этим службам и функциям осуществляется настройка определения ролей, на основе которых пользователь может выполнить требуемую задачу.

Задачей теста является определение и документирование ролей, используемых в приложении, попытка переключение роли и получения доступа к другой роли, а также оценки степени детализации ролей и функциональность, обеспечиваемая предоставленными разрешениями.

Определение ролей может осуществляться любым из предложенных методов:

* Документация по приложению;
* Рекомендации разработчиков или администраторов приложения;
* Комментарии к приложению;
* Расставьте возможные роли:
* переменная cookie (например, role = admin, isAdmin = True);
* переменная аккаунта (например, роль: менеджер);
* скрытые каталоги или файлы (например, / admin, / mod, / backups)

После определения роли следует понять, можно ли получить доступ к конкретной роли.

#### 5.3.2.Тестирование процесса регистрации пользователей

На некоторых веб-сайтах процесс регистрации пользователей осуществляется с автоматизированным предоставления пользователям доступа к системе, поскольку объем базы пользователей не дает возможности управлять доступами вручную. Задачей теста является проверка соответствия требований к регистрации пользователей бизнес-требованиями и требованиям безопасности, а также проверка самой регистрации.

Требования:

1. Можно ли зарегистрироваться в системе?
2. Регистрационные данные проверяются автоматически или вручную администраторами?
3. Может ли один пользователь зарегистрироваться несколько раз?
4. Могут ли пользователи регистрироваться с разным уровнем привилегий?
5. Какое удостоверение личности необходимо для регистрации?
6. Проверяются ли зарегистрированные личности?

### 5.4. Тестирование аутентификации

#### 5.4.1.Тестирование учетных данных, передаваемых по зашифрованному каналу

В веб-приложениях все данные аутентификации пользователей (учетные данные пользователя, токен сеанса) при передаче должны быть зашифрованы. Шифрование не позволяет злоумышленникам получить учетные записи путем перехвата трафика, однако это не означает, что трафик полностью безопасен, также на безопасность влияет алгоритм шифрования, надежность ключей. В задачу теста входит: оценка возможности передачи между клиентов и сервером веб-приложения передачи незашифрованных учетных данных пользователей, при помощи захвата трафика.

#### 5.4.2.Проверка слабого механизма блокировки

Для предотвращения грубых атак по подбору пароля или имени пользователя в веб-приложениях используются механизмы блокировки учетных записей. Учетные данные обычно блокируются через 3-5 неудачных попыток входа и могут быть разблокированы только через определенный период времени с помощью автоматизированного механизма или вмешательства администратора. Задача теста: оценить способность механизма блокировки учетных записей снизить вероятность подбора пароля методом перебора, а также оценить устойчивость механизма разблокировки к действиям несанкционированного пользователя.

Для проверки надежности механизмов блокировки необходим доступ к учетной записи, которую можно заблокировать. Для оценки механизма блокировки необходимо несколько раз выполнить вход с неправильным паролем, прежде чем использовать правильный и убедиться, что учетная запись не заблокирована.

#### 5.4.3. Тестирование ненадежной политики генерации паролей

Одним из наиболее распространенных и простых методов аутентификации является использование статического пароля. Пароль представляет собой хорошую меру безопасности, однако пользователи сами подрывают эту безопасность, делая своими паролями «12345678» или «qwerty» для упрощения использования. Цель тестирования: определение устойчивости приложения к подбору пароля брутфорсом с использованием доступных словарей паролей, путем оценки требований к длине, сложности, повторному использованию и устареванию паролей.

### 5.5. Тестирование авторизации

#### 5.5.1.Тестирование повышения привилегий

На этом этапе тестирования рассматривается проблема повышения уровня привилегий пользователя, и задача тестировщика состоит в том, чтоб убедиться, что пользователь не может изменять свои привилегии или роли. Повышение уровня привилегий позволяет пользователю получить доступ к большему количеству ресурсов и функций, чем они обычно разрешены и такое повышение или изменение должно быть предотвращено приложением, так как злоумышленник, получив такой доступ, сможет использовать функции администратора (в случае повышения привилегий вертикального доступа) или же иметь возможность получить информацию других пользователей (в случае повышения привилегий горизонтального доступа).Задача теста: определение точек внедрения, связанных с манипулированием уровней привилегий.

### 5.6. Тестирование управления сессиями(сеансом)

#### 5.6.1. Тестирование тайм-аутов сессии

Все веб-приложения должны реализовывать тайм-ауты простоя или бездействия для сессий. Тайм-аут определяет время, в течение которого сессия будет активна в случае отсутствия активности пользователя. Закрытие и аннулирование сессии происходит после определенного периода простоя с момента последнего HTTP запроса от пользователя.

Наиболее распространенный сценарий атаки такого типа - общедоступный компьютер, который используется для доступа к некоторой частной информации.

На этом этапе тестировании проверяется, что приложение выполняет автоматический выход пользователя из системы, если пользователь бездействует в течение определенного времени, гарантируя, что нет возможности повторного использования этой сессии и что конфиденциальные данные не сохраняются в кеше браузера.

Задача теста: проверка, что в приложении реализован жесткий тайм-аут сессий.

#### 5.6.2. Тестирование функции выхода из системы

Завершении сессии - важная часть жизненного цикла сессии, поскольку сведение к минимуму времени жизни токенов сессии снижает вероятность успешных атаки перехвата сессии, межсайтовых сценариев и подделки межсайтовых запросов. Для безопасного завершения сессии необходимы следующие компоненты: доступность элементов управления пользовательского интерфейса, позволяющих пользователю выйти из системы, тайм-ауты сессии, правильное аннулирование сессии на стороне сервера.

Существует ряд проблем, связанных с эффективным завершением сессии:

* неясные и неоднозначные функции выхода из системы пользовательского интерфейса;
* После выхода из системы, для токенов сессии на стороне клиента устанавливается нового значение, в то время, как состояние на сервере остаётся активным, и его можно использовать, вернув cookie сессии к предыдущему значению;

Задачи теста: оценка пользовательского интерфейса выхода, анализ тайм-аутов сессии и корректность завершения сессии после выхода из системы.

Тестирование пользовательского интерфейса выхода из системы:

Необходимо проверить внешний вид и видимость функции выхода из системы, для этого требуется просмотреть каждую страницу приложения с точки зрения пользователя, который хочет выйти из системы.

Признаки хорошего интерфейса выхода из системы:

* Кнопка выхода присутствует на всех страницах веб-приложения;
* Кнопка выхода должна быть легко идентифицирована пользователем;
* Кнопка выхода размещается в области страницы, которая не зависит от прокрутки содержимого;

Тестирование завершения сессии на стороне сервера:

Для проверки завершения сессия на стороне сервера необходимо выполнить следующие шаги:

* Сохранить значение файлов cookie, которые используются для идентификации сессии;
* Вызвать функцию выхода из системы и проанализировать поведение cookie файлов сессии;
* Перейти на страницу, которая видна в аутентифицированной сессии;
* Если функция выхода приводит к установке нового значения файлов cookie сессии, восстановить старое значение и перезагрузить страницу.

Если тест не обнаружил уязвимостей попробовать на различных страницах веб-приложения.

### 5.7. Тестирование проверки входных данных

#### 5.7.1. Тестирование отраженных межсайтовых сценариев

Reflected XSS возникают, когда злоумышленник вводит исполняемый код браузера в параметры HTTP запроса или в форму HTML. Эти данные не хранятся в приложении и воздействуют только на пользователей, которые открыли созданную злоумышленником стороннюю веб-страницы.

В случае если веб-приложение уязвимо для такого рода атака оно возвращается клиенту непроверенный данные, отправленные через запросы.

Общий способ такой атаки состоит из этапов:

* Проектирование – злоумышленник создает и тестирует злонамеренную ссылку;
* Социальная инженерия – злоумышленник убеждает жертву перейти по вредоносной ссылке;

Как правило, эта уязвимость используется с целью кражи cookie, буфера обмена и изменения содержимого страницы жертвы.

Трудность предотвращение XSS уязвимостей заключается в правильном экранировании символов, например, веб-приложение может отфильтровать *<script>*, но не может отфильтровать *%3cscript%3e* .

Тестирование черного ящика включает три этапа:

* Обнаружение ввода – для каждой страницы веб-приложения необходимо определить все пользовательские переменные приложения (параметры HTTP запросов, скрытые значения полей форм) и способ их ввода;
* Анализ векторов входных данных - необходимо проанализировать каждый вектор входных данных для обнаружения потенциальной уязвимости посредством ввода специальных безвредных тестовых данных, которые вызывают ответа от браузера, демонстрируя уязвимость;
* Проверка воздействия – для каждого ввода тестовых данных требуется проанализировать результат и определить, представляет ли он уязвимость, влияющую на безопасность приложения. После обнаружения необходимо идентифицировать специальные символы, которые не были закодированы, заменены или отфильтрованы должным образом.

В противовес атакам межсайтовых сценариев используется брандмауеры веб-приложений, блокирующие вредоносный ввод, или специальные механизмы современных браузеров. Однако тестировщику необходимо проводить поиск уязвимостей с предположением о том, что встроенные функции безопасности отключены. В следствие этого безопасность приложения от этого вида атак в большинстве своем должна осуществляться благодаря механизмам очистки и фильтрации пользовательского ввода, в свою очередь, которые, тоже можно обойти рядом методов:

* Использование других символов или кодировки;
* Обход не рекурсивной фильтрации;
* Включение внешнего скрипта;

#### 5.7.2.Тестирование хранимых межсайтовых сценариев

Stored XSS (хранимые межсайтовые сценарий) – наиболее опасный тип межсайтового скрипптинга. Веб-приложения, позволяющие пользователям хранить данные потенциально подвержены этому типу атак. Этот вид уязвимости возникает, когда веб-приложение получает от пользователя потенциально вредоносные данные и сохраняет их в хранилище, для последующего использования. В случае, если в веб-приложении не настроена фильтрация входной информации, вредоносные данные будут отображаться, как часть веб-сайта при каждом запуске приложения.

Уязвимость может быть использована с целью:

* Захват конфиденциально информации;
* Порча приложения;
* Направленная отправка эксплойтов;

Для Stored XSS не требуется вредоносная ссылка, код отрабатывает, когда пользователь посещает страницу с сохраненным XSS.

Атака состоит из следующих этапов:

* Злоумышленник сохраняет вредоносный код на уязвимой странице;
* Пользователь аутентифицируется и посещает уязвимую страницу;
* Вредоносный код отрабатывает в браузере пользователя;

Для тестирования необходимо определить точки приложения, в которых пользовательский ввод хранится в фоновом режиме и отображается приложением.

Можно использовать XSS stored с расширенными фреймворками Javascript, такими как Beef, XSS Proxy, посредством внедрения в веб-приложения хука JavaScript, связывающего приложение с платформой злоумышленника.

Также следует проверить, возможна ли в веб-приложении загрузка HTML содержимого.

#### 5.7.3.Тестирование SQL-инъекций

Тестирование SQL-инъекций используется с целью проверки, возможно ли ввести данные в приложение таким образом, чтоб оно выполняло управляемый пользователем SQL-запрос в базе данных. Уязвимости SQL-инъекций обнаруживаются в тех приложениях, которые используют вводимые пользователем данных для формирований SQL запросов без надлежащей проверки и фильтрации. Успешная эксплуатация такой уязвимости позволяет злоумышленнику получить доступ к базе данных и манипулировать ее данными.

Атака sql –инъекций состоит из вставки частичного или полного SQL-запроса в поле ввода пользовательских данных.

Атаки с использованием SQL-инъекций можно разделить на три класса:

* Inband – данные извлекаются с использованием того же канала, через который вводится код SQL, полученные данные отображаются непосредственно на странице веб-приложения.
* Out-of-band – данные извлекаются с использованием другого канала, например посредством создания электронного письма с результатами запроса.
* Inferential or Blind – фактической передачи данных нет, тестировщик восстанавливает информацию посредством наблюдения за поведением сервера БД в ответ на определенные запросы.

Задачи теста заключаются в определении точек внедрения SQL, а также оценки серьезности инъекций и уровня доступа, который может быть достигнут в результате успешной атаки.

Методы обнаружения:

Первоначально в тесте необходимо понять, как приложение взаимодействует с сервером базы для доступа к данным. Как правило, приложению необходимо взаимодействовать с базой данных в следующих случаях:

* Формы аутентификации;
* Строки поиска;
* Страницы электронной коммерции (продукты и их характеристики);

После определения точек входа используются конкретные методы sql-инъекций:

* Классические sql-инъекции;
* Оператор Select;
* Составные запросы;

Для лучшего составления sql-инъекций необходимо определить какой тип СУБД использует веб-приложение. Несмотря на то, что SQL является стандартом, каждая СУБД имеет свои отличительные особенности, такие как: специальные команды, функции для извлечения данных. Одним из способов выяснить, какая база данных используется - изучение ошибок, возвращаемых приложением.

Методы эксплуатации SQL-инъекций:

* Использование оператора UNION;
* Эксплуатации на основе ошибок;
* Внедрение хранимой процедуры;

#### 5.7.4. Тестирование XML-инъекций

XML-инъекции – уязвимость веб-приложения, позволяющая злоумышленнику вмешаться в обработку XML-данных приложением, что дает возможность злоумышленнику просматривать файлы приложения и взаимодействовать с серверными системами.

Задачи теста: определение точек внедрения XML и оценка типов уязвимостей, которые могут быть достигнуты, а также серьезность последствий.

Поиск уязвимостей осуществляется посредством попытки вставки метасимволов XML:

* Одиночная «’» или двойная кавычка «”». - при неправильной обработке символы могут вызвать исключение во время синтаксического анализа XML, в результате чего XML файл станет недействительным;
* Угловые скобки «<>»;
* Тег комментария <!-- / --> - последовательность символов интерпретируется как начало или конец комментария. Приводит к неправильной последовательности XML;
* Разделители разделов CDATA: <! \ [CDATA \ [/]]> - разделы CDATA используются для экранирования блоков текста, содержащих символы, которые в противном случае были бы распознаны как разметка, то есть символы, заключенные в раздел CDATA, не анализируются анализатором XML;

После получения некоторой информации о структуре XML-документа злоумышленник имеет возможность внедрить в этот файл данные или теги XML, что может привести к различному роду атак, например, повышения привилегий.

Помимо этого, специалист по тестированию также может изучить исходный код на предмет наличия правил DTD (компьютерный язык, который используется для записи фактических синтаксических правил метаязыка SGML и расширяемого языка разметки XML).

#### 5.7.5. Тестирование инъекций заголовка Host

Как правило веб-серверы размещают несколько веб-приложений на одном IP-адресе, обращаясь к каждому приложению через виртуальный хост. Во входящем запросе серверы отправляют запрос на целевой виртуальный хост на основе значения, указанного в заголовке Host. Если для этого заголовка не установлена надлежащая проверка значения, злоумышленник может заменить это значение с целью:

* Отправлять запросы на первый виртуальный хост в списке
* Вызывать перенаправление в домен, контролируемый злоумышленником
* Выполнять отправку кеша
* Управлять функциями сброса пароля

Цель тестирования: оценить, выполняется ли в приложение динамический анализ заголовка Host.

Для тестирования необходимо просто указать другой домен в поле заголовка Host. Воздействие определяется тем, как сервер обработает значение заголовка, атака считается действительной, если веб-сервер обрабатывает входные данные для отправки на хост злоумышленника, а не на свой собственный. В случае если предусмотрена проверка заголовка на недопустимый ввод, можно передать значение в заголовок X-Forwarded-Host.

### 5.8. Тестирование на обработку ошибок

#### 5.8.1.Тестирование на неправильную обработку ошибок

В любом приложении (веб-приложение, сервер, база данных) при определенных обстоятельствах генерируются ошибки. Эти ошибки отображаются в виде трассировки стека, тайм-аутов сети, несоответствия ввода. Разработчики часто игнорируют обработку этих ошибок, что может позволить злоумышленнику получить информацию об внутренних используемых системах, фреймворках, API, версиях и типах приложений, а также совершить Dos-атаку путем ввода системы в тупик.

В задачи теста входит: определение существующего вывода ошибок и анализ возвращаемых ошибкой данных.

Для получения сообщений об ошибках тестировщику требуется выполнить следующие действия:

* Осуществить поиск случайных папок и фалов, которые не будут найдены (404);
* Выполнить запросы к существующим папкам и проанализировать поведение сервера;
* Выполнить запросы, нарушающие HTTP RFC;

Приложение в большей степени совершают отправку сообщений об ошибках, поскольку разработчикам необходимо отслеживать и обрабатывать все возможные дефекты. Чтоб приложение выдало эти ошибки тестировщик должен определить возможные точки ввода данных, произвести анализ ожидаемого типа входных данных (строки, целые числа, JSON, XML).

### 5.9. Тестирование слабой криптографии

#### 5.9.1.Тестирование отправки конфиденциальной информации по незашифрованному каналу

При передаче, конфиденциальные данные такие как: учетные данные, токены, идентификаторы сессии, файлы cookie – обязательно должны быть защищены, посредством HTTPS или других методов шифрования. При передаче приложением конфиденциальной информации по незашифрованным каналам (HTTP), то в приложении автоматически появляется угроза безопасности.

В качестве конфиденциальной информации может выступать:

* Номера социального страхования;
* Номера банковских счетов;
* Информация о паспорте;
* Информация, связанная со здравоохранением;
* Информация о медицинском страховании;
* Информация для студентов;
* Номера кредитных и дебетовых карт;
* Информация о водительских правах и государственном идентификаторе;

Задачи теста:

* Определить конфиденциальную информацию, передаваемую по различным каналам;
* Оценить безопасность используемых каналов;

Для тестирования необходимо выяснить, какую проверку подлинности использует базовая аутентификации или аутентификации на основе форм, а также выяснить установлен ли для cookie файлов сессии флаг безопасности Secure.

### 5.10. Тестирование бизнес-логики

#### 5.10.1. Тестирование возможности подделки запросов

Формирование запросов - метод, которым злоумышленники пользуются для обхода внешнего графического интерфейса пользователя, с целью прямой отправки HTTP-запросов через перехватывающий прокси-сервер со значениями данных, которые не поддерживаются, не защищены или не поддерживаются бизнес-логикой приложения.

Посредством таких атак злоумышленник может раскрыть скрытые функции, таких как включения отладки или отображение специальных окон, полезных на этапе разработки, однако которые могут привести к утечке информации.

Уязвимость возможности подделывания запросов уникальна для каждого приложения и направлена на нарушение рабочего процесса бизнес-логики, поэтому в таких системах должна быть предусмотрена проверка логики.

Как протестировать:

Определение предполагаемых значений:

Посредством использования перехватывающего прокси сервера проанализировать и HTTP-запросы и определить легко предсказуемые значения и в случае, если такое значение есть, попробовать его изменить для получения неожиданных результатов.

Определение скрытых параметров

С использованием перехватывающего прокси сервера исследовать HTTP-запросы на наличие использования скрытых функций, таких как отладка, и попробовать изменить параметры с целью получения другого поведения приложения.

### 5.11.Тестирование на стороне клиента

#### 5.11.1.Тестирование межсайтового скриптинга на основе DOM

DOM (объектная модель документа) - это структурный формат, используемый для представления документов в браузере, позволяющий динамическим сценариям (например, JavaScript), ссылаться на компоненты страницы, такие как поля форм или cookie файлы сессии.

Межсайтовый скриптинг на основе DOM - уязвимость, возникающая в результате модификации активного содержимого страниц, посредством специального запроса, созданным злоумышленником. В отличие от более распространенных типов межсайтовых сценариев (отраженных и сохраненных, в которых код отправляется на сервер, а затем обратно в браузер) он выполняется непосредственно в браузере пользователя без контакта с сервером. Результатом такой атаки может быть кража cookie файлов, внедрение вредоносных сценариев и т.д.

Приложения, написанные на JavaScript, существенно отличаются от приложений других типов, так как они часто генерируются сервером. Для проведения атаки необходимо просканировать веб-приложение, определить точки выполнения JavаScript, места, где код динамически записывается на страницу и прочие места, где модифицируется DOM.

#### 5.11.2.Тестирование HTML инъекций

HTML-инъекция – тип уязвимости, которая возникает, когда пользователь может управлять точкой ввода данных и внедрить произвольный HTML-код на уязвимую страниц веб-приложения. Уязвимость возникает, когда вводимые пользователем данные не обрабатываются требуемым для безопасности образом

В результате такой уязвимости злоумышленник может получить cookie файлы сессии, а также изменить содержимое страницы, видимое жертвами.

В задачи теста входит определение точек внедрения пользователем HTML.

# ГЛАВА 3. ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ МЕТОДОВ OWASP WEB SECURITY TESTING GUIDE V4.2

## 1. Тестовый стенд

Для практического применения выбранных методов тестирования на проникновение веб-приложений был использовано тестовое приложение Owasp Juice Shop. Owasp Juice Shop представляет собой приложения Интернет магазина для тестирования различных методов по обнаружению уязвимостей.

Для использования тестового стенда предварительно на рабочем ПК (Windows 10) был установлен Docker.

Docker - программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.

Затем приложение Owasp Juice Shop развернуто посредством следующих команд, указанных в документации:

* Run docker pull bkimminich/juice-shop
* Run docker run --rm -p 3000:3000 bkimminich/juice-shop

Открыть приложение можно в браузере, прописав в адресной строке требуемый адрес (рис.1):

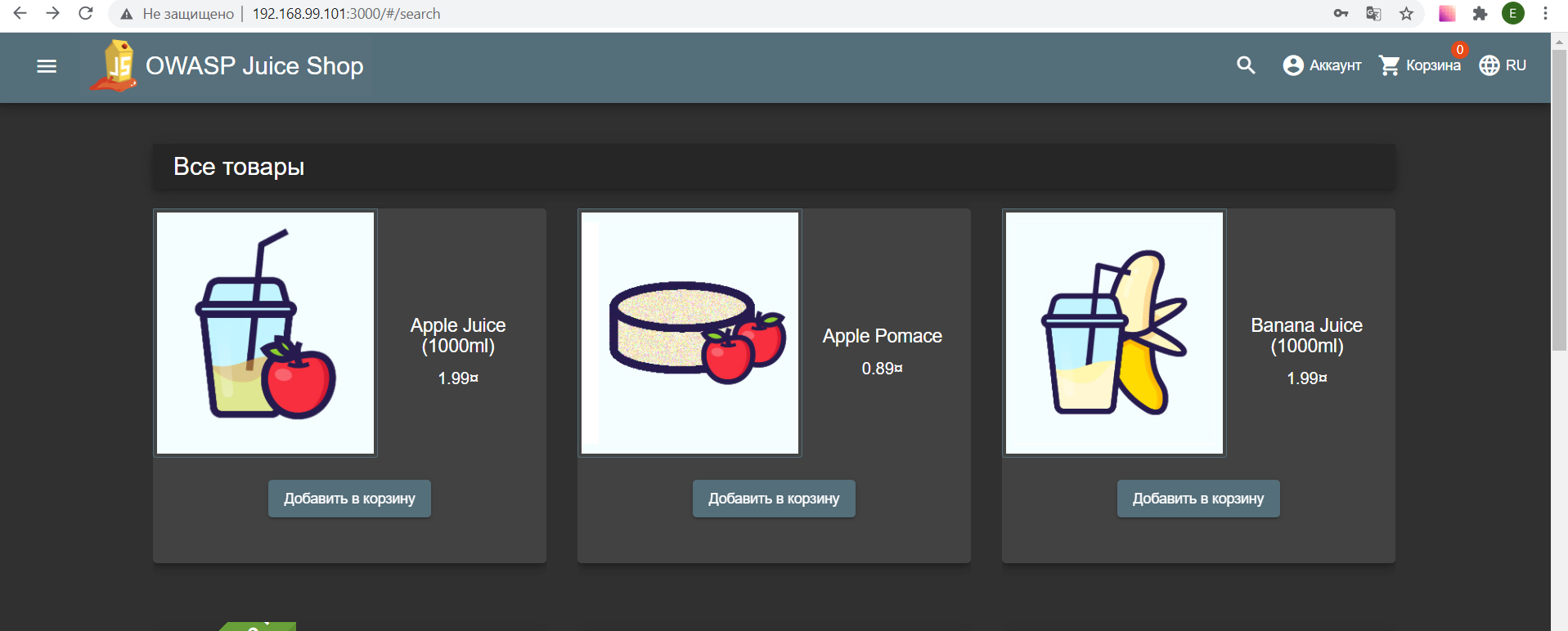


Рис. 1. Приложение owasp juice\_shop

## 2. Сбор информации

### 2.1. Определение веб-сервера

Применение: для определения веб-сервера выбранного приложения был выбран инструмент Netcraft (рис.2). Netcraft представляет собой бесплатное расширение для браузеров таких как Chrome, Firefox и Opera, которое позволяет легко узнавать информацию о посещаемых веб-сайтах и обеспечивает защиту от фишинга.

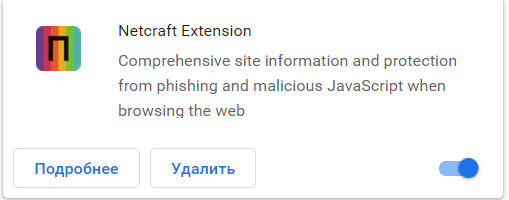


Рис.2. Расширение Netcraft Extension

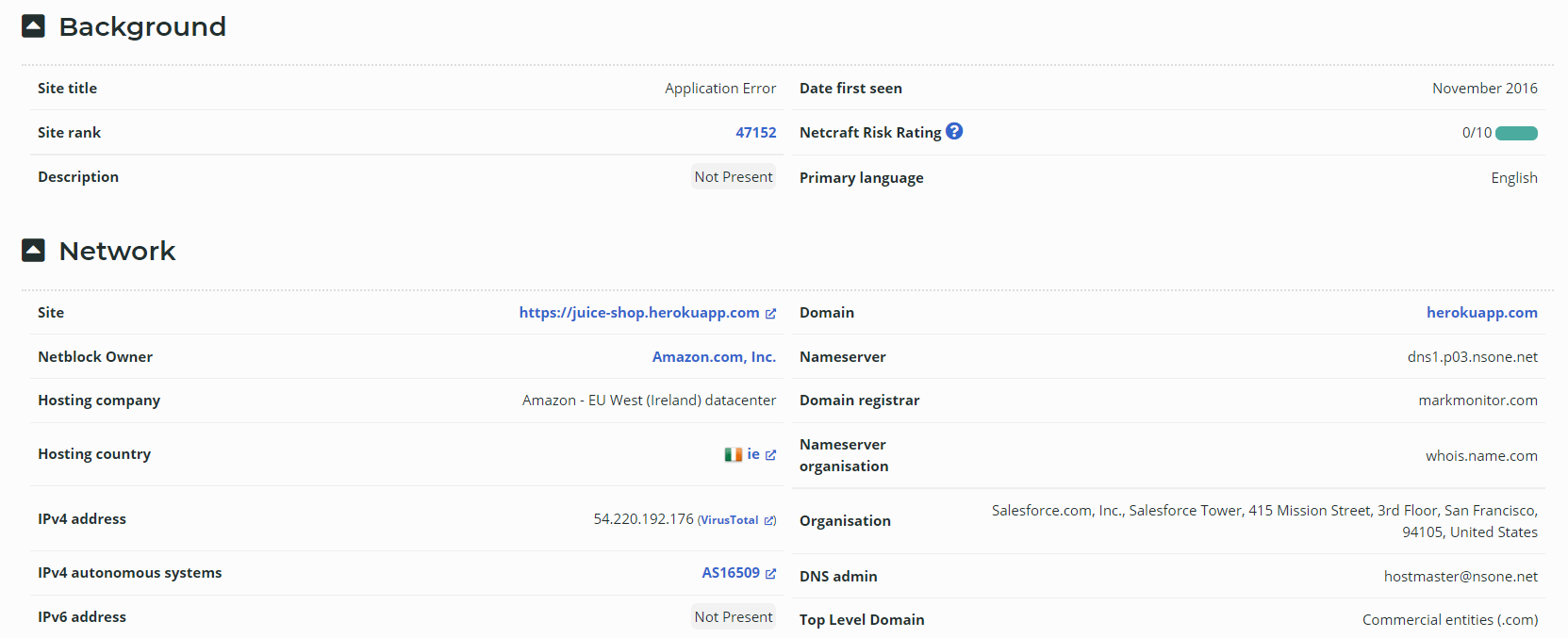
В ходе проведения теста на Chrome браузере были получены следующие результаты, представленные на рисунках 3 и 4:

Рис. 3. Полученная информация благодаря расширению

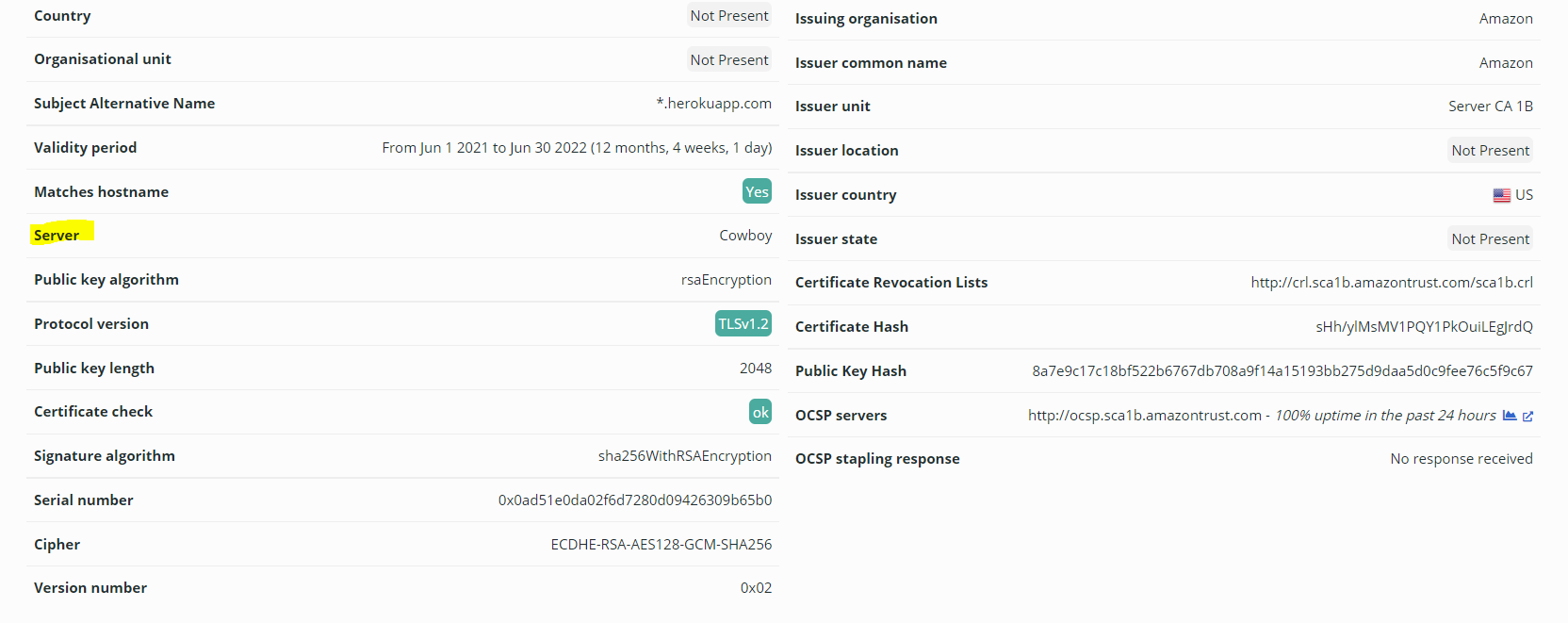


Рис.4. Полученная информация благодаря расширению

Исходя из которых можно сделать вывод, что приложение работает на веб-сервере Cowboy. Помимо типа сервера вывод Netcraft предоставляет много другой информации, которая может быть полезна при выборе атак на приложение.

### 2.2. Проверка содержимого веб-страницы на предмет утечки информации

Применение: для поиска утечек конфиденциальной информации в содержимом веб-страниц использовались Инструменты разработчика Chrome, рис.5:



Рис.5. Html код главной страницы приложения

В ходе проведения теста в коде HTML не было обнаружено конфиденциальной информации, одна при анализе клиентских файлов JavaScript была найдена следующая информация, полезная для дальнейшего тестирования на проникновение, (рис.6.):

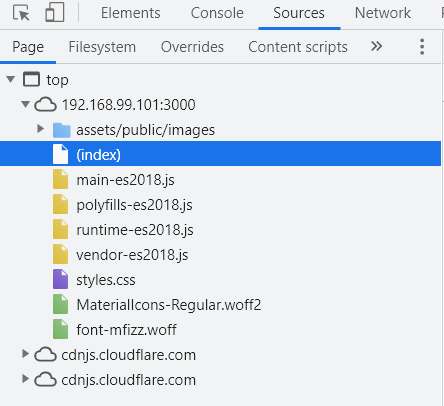


Рис.6. Набор клиентских файлов

Скрытые пути к функциям администратора и конфигурации приложения, (рис.7):



Рис.7. Скрытые пути в клиентском js файле

Наличие путей для запросов к API веб-приложения, (рис.8):

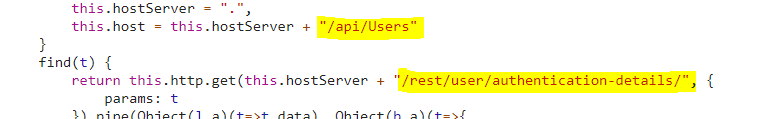


Рис.8. Скрытые пути в клиентском js файле

Работу методов по смене и сбросу пароля пользователя, (рис.9):



Рис.9. Скрытые пути в клиентском js файле

А также другую информацию, помогающую определить карту веб-приложения.

### 2.3. Составление карты веб-приложения

**Применение:** для проведения теста был использован паук «OWASP ZAP»[12], с предварительной настройкой прокси сервера браузера при помощи расширения «FoxyProxy», (рис.10).

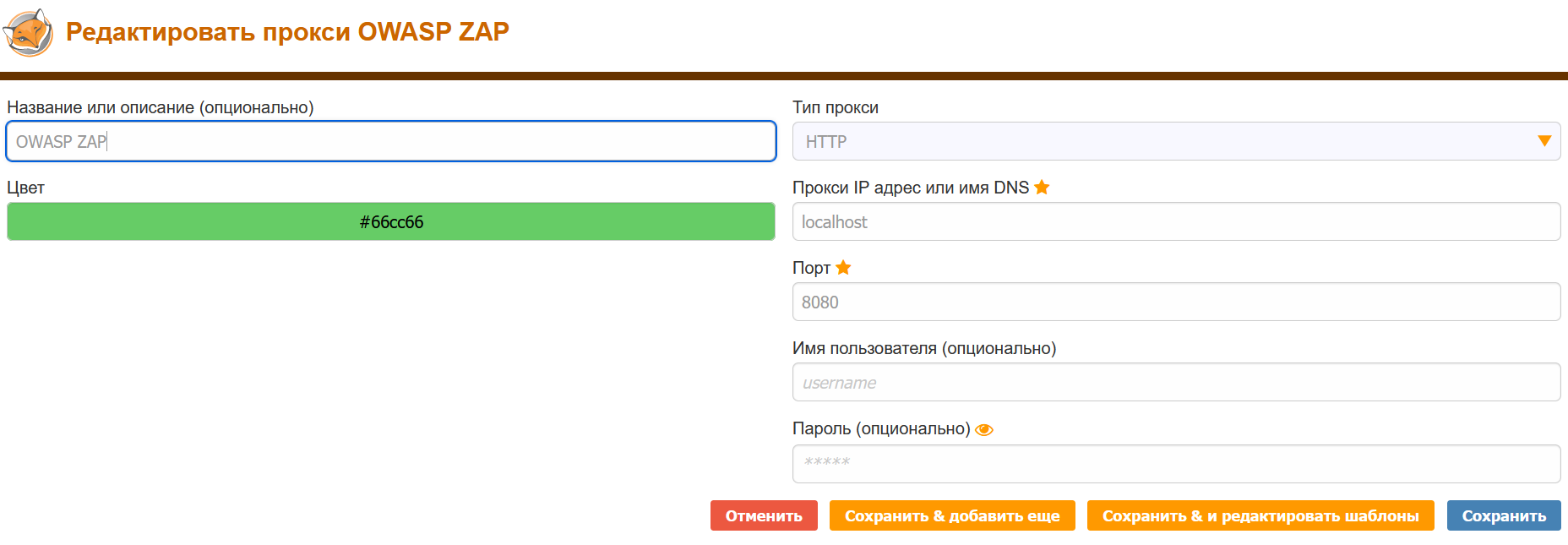


Рис. 10. Прокси сервер FoxyProxy

После обновления страницы веб-приложения, была получена информация о запросах, (рис.11.)

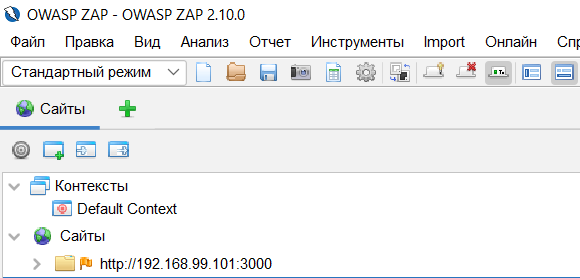


Рис.11. Перехваченный запрос в OWASP ZAP

После использовалась атака пауком,( рис.12, рис.13):

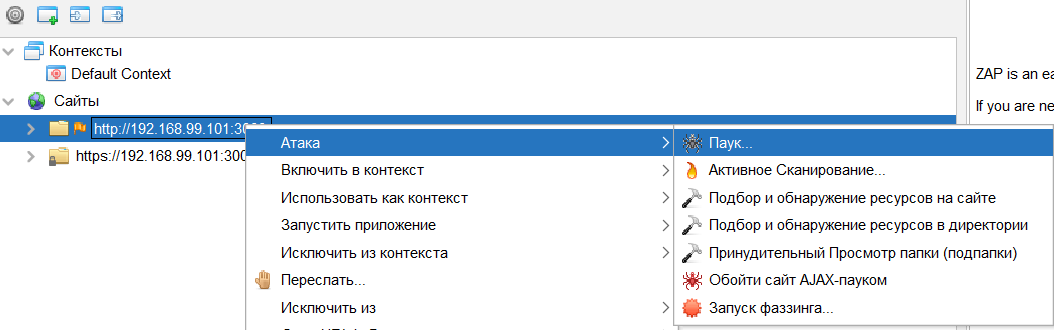


Рис.12. Использование OWASP ZAP для поиска путей приложения

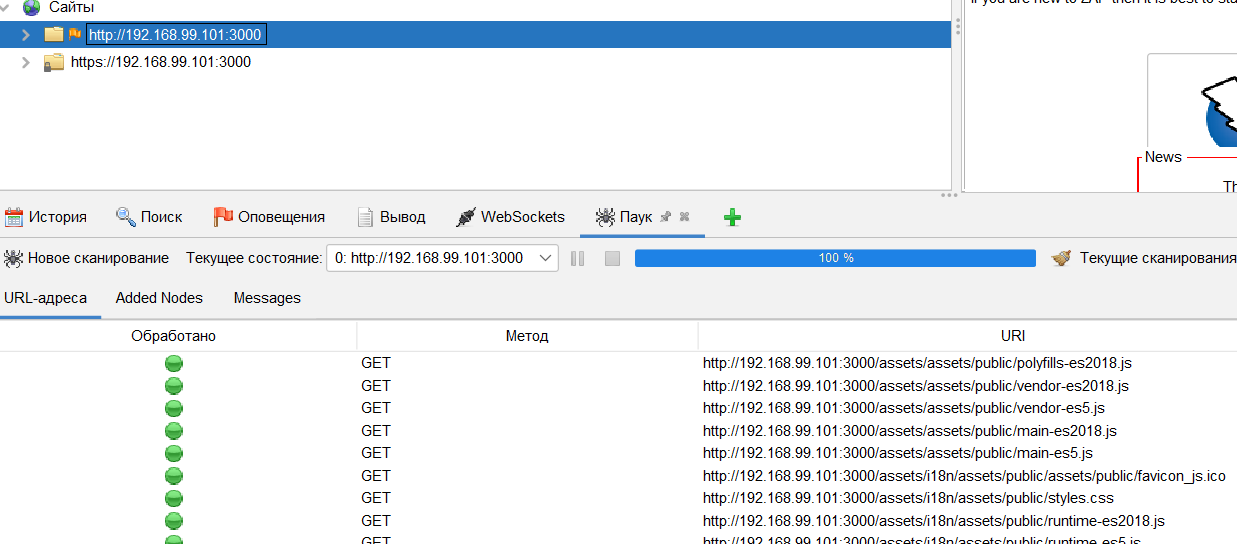


Рис.13. Использование OWASP ZAP для поиска путей приложения

В результате атаки были получены результаты о структуре веб-приложения, (рис.14):

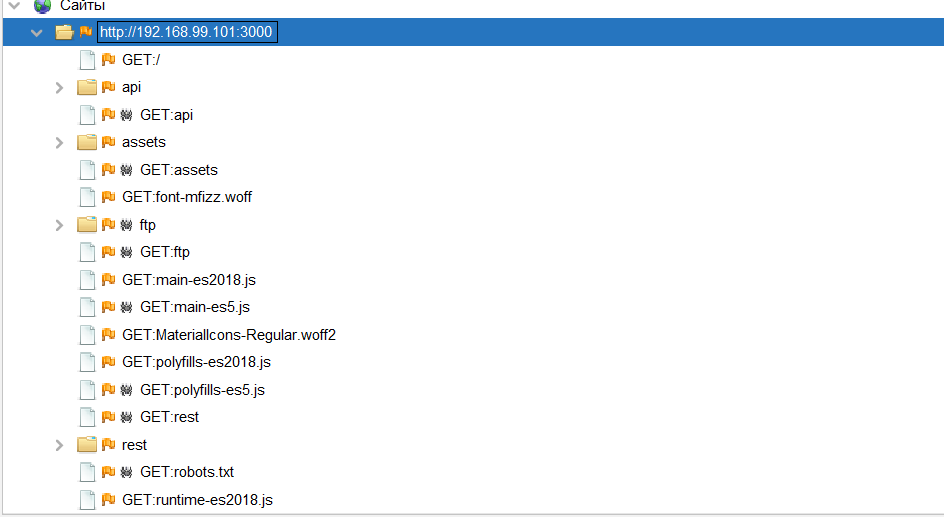


Рис.14.Результаты работы OWASP ZAP

После чего информация о всех URL веб-приложения, обнаруженных пауком, была экспортирована в текстовый файл, (рис. 15):

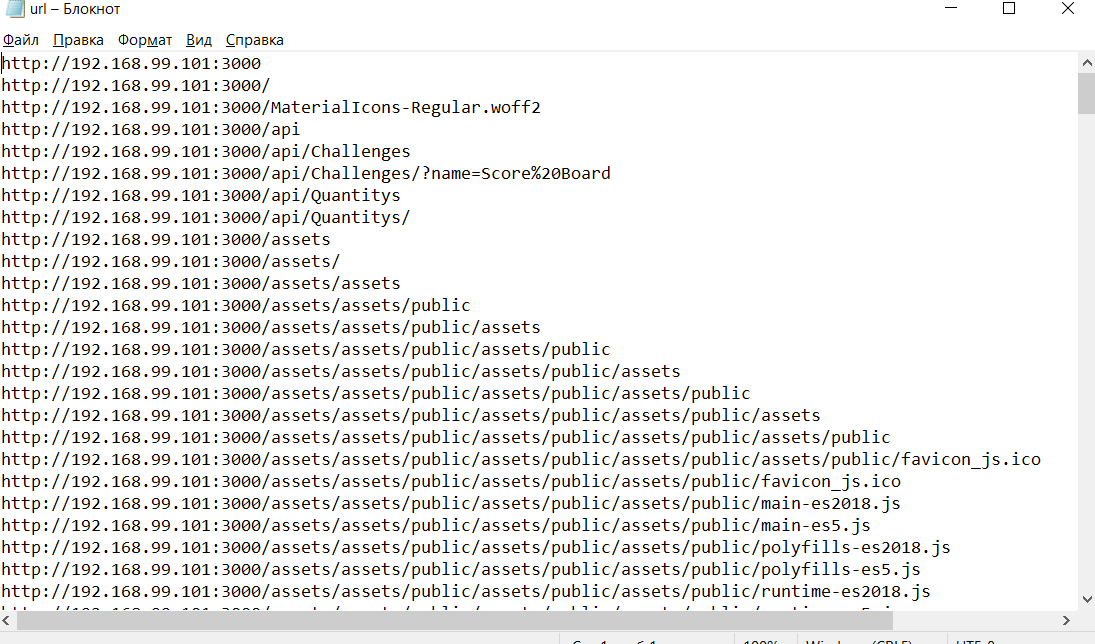


Рис.15. Результаты работы, экспортированные в файл

## 3. Тестирование конфигурации и управления развертывания

### 3.1. Тестирование HTTP методов

Первоначально для выполнения были определены поддерживаемые HTTP методы, посредством инструмента отправки запросов POSTMAN. На адрес веб-приложения был отправлен OPTIONS запрос, который вернул следующий результат, (рис.16):

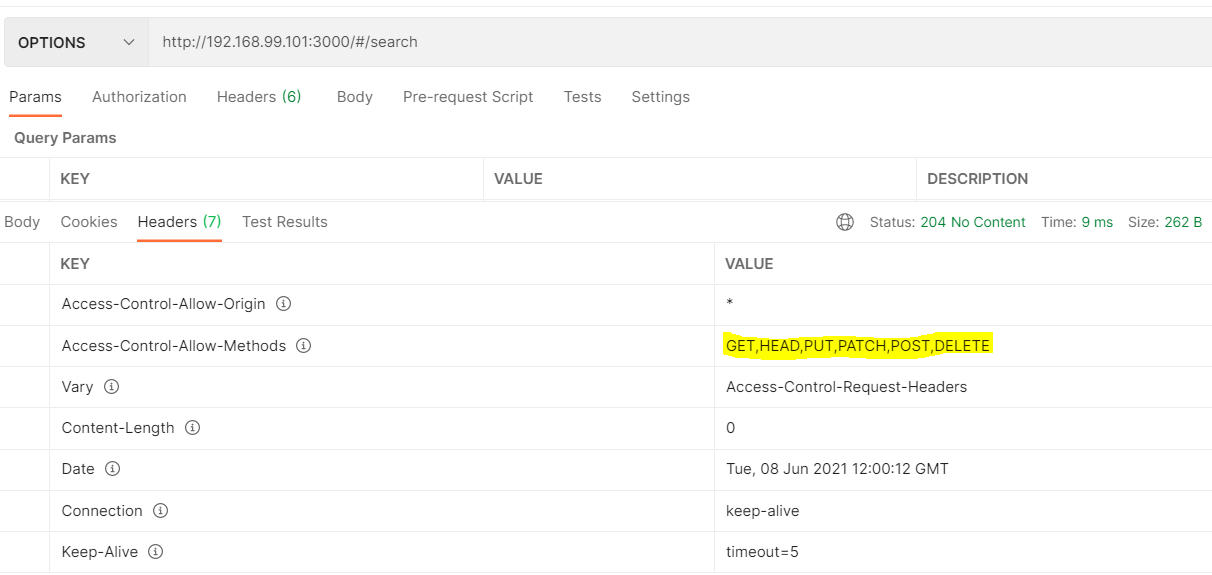


Рис. 16. Отправка запроса Options

По которому мы можем сказать, что на сервере доступны методы:

GET, POST, HEAD, PUT, PATCH, DELETE.

### 3.2. Тестирование метода PUT

При помощи прокси были перехвачены запросы при обновлении страницы веб-приложения, один из которых был изменен на PUT с добавлением HTML файла, (рис.17). Ответ на запрос был успешный, однако отправленный файл отсутствовал.

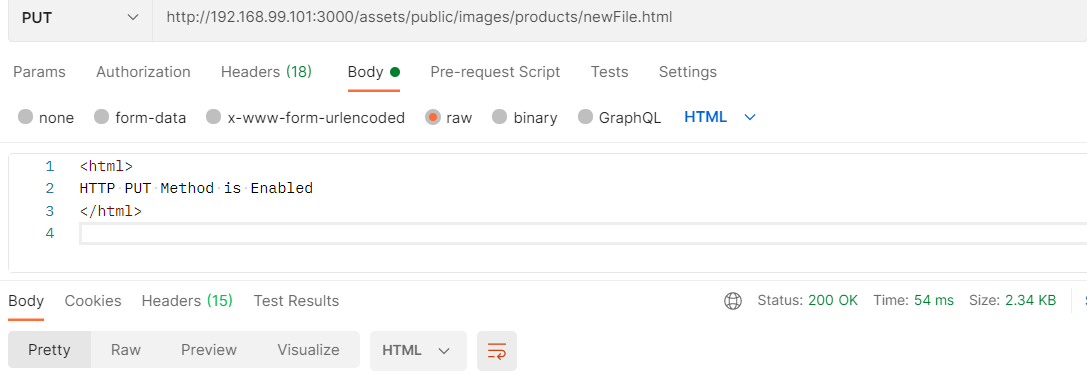


Рис.17. Замена метода HTTP запроса

### 3.3. Тестирование потенциальной межсайтовой трассировки

При помощи инструмента Ncat был отправлен запрос на сервер, (рис.18):

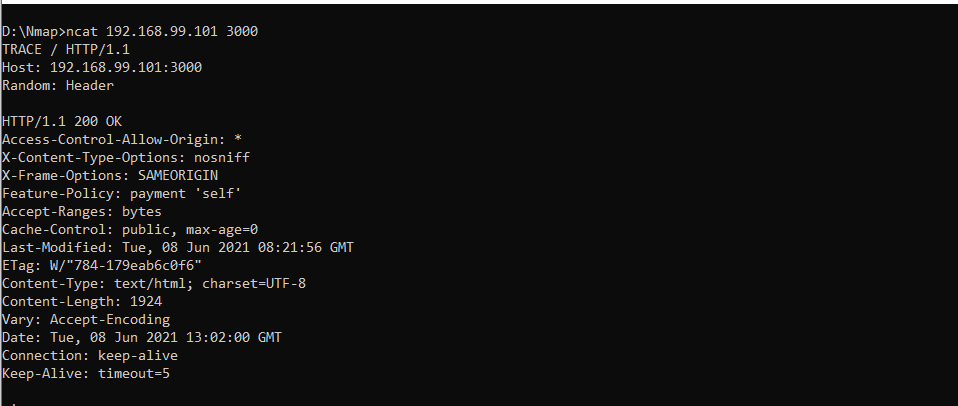


Рис 18. Отправка запроса TRACE приложению

Однако сервер не вернул отправляемое нами сообщения, из чего можно сделать, что к этой атаке веб-приложению устойчиво.

Тестирование замены методов HTTP запросов показало, что приложение устойчиво к этому виду уязвимостей.

## 4. Тестирование управления учетными записями

### 4.1. Тестирование ролей пользователей

Отталкиваясь от информации, обнаруженной в тесте «Составление карты приложения», а именно о том, что в приложение существует директория «/api/Users», можно сделать предположение о том, что если отправить Get запрос на этот URL, то можно получить список всех пользователей системы.

Отправив get запрос по выбранному URL был получен следующий ответ (рис.19), из которого можно сделать вывод, что запрос неудачен по причине отсутствия прав доступа.

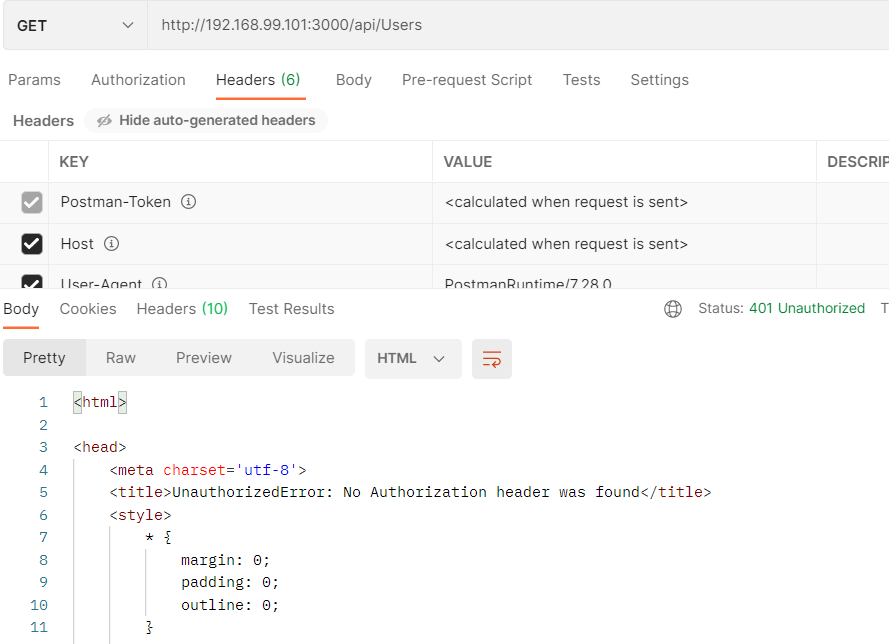


Рис.19. Запрос к api приложения

Использовав эту информацию, было решено перехватить при помощи прокси запрос авторизированного пользователя, заменить метод и попробовать отправить запрос еще раз.

Среди перехваченных запросов был выбран запрос к api веб-приложения, (рис. 20 и рис.21):

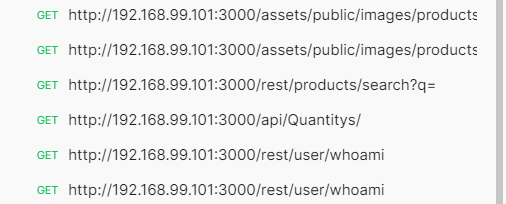


Рис. 20. Список перехваченных запросов

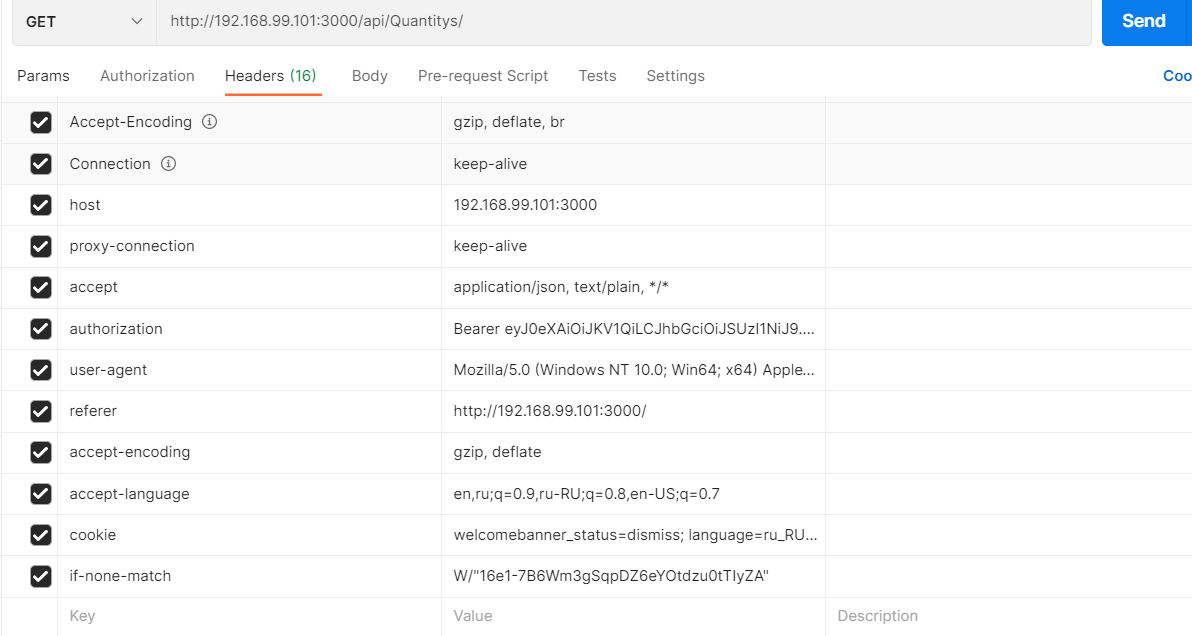


Рис 21. Выбор перехваченного запроса

И после отправки измененного запроса на api/Users был получен следующий результат, а именно список всех пользователей приложения, их роли, почта и прочая полезная информация для проведения атак, (рис.22):

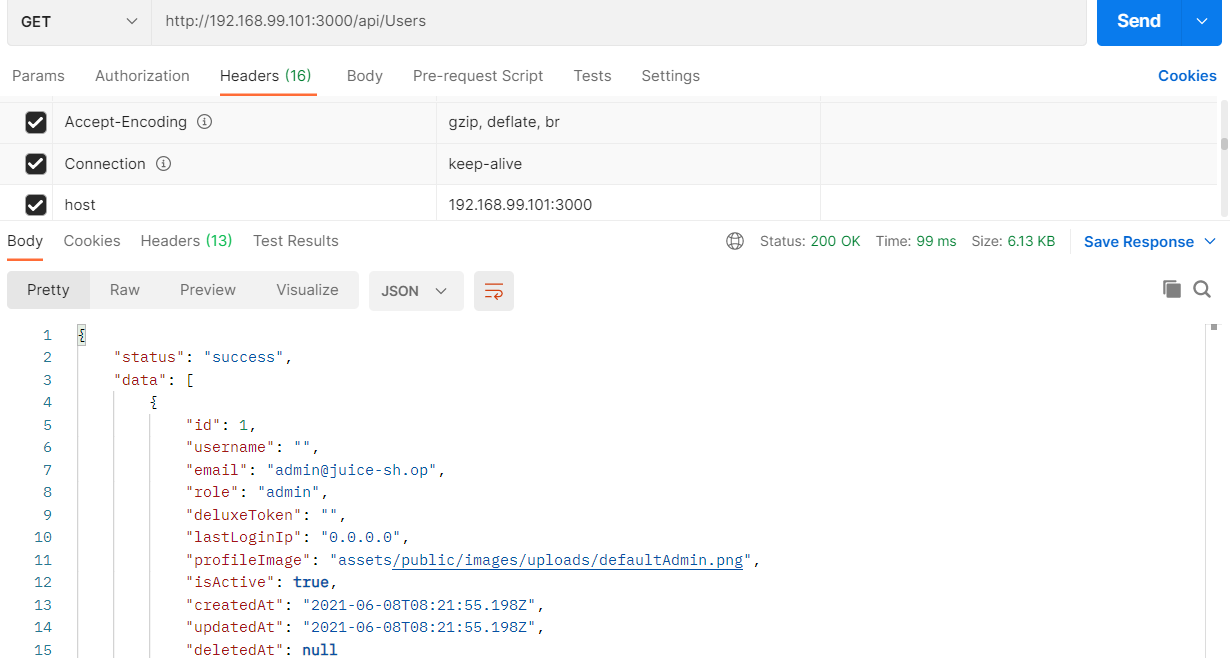


Рис.22. Результат отправки изменённого запроса

По полученному списку можно сделать вывод, что в веб-приложении существует три роли:

* Admin
* Customer
* Deluxe
* Accounting

### 4.2. Тестирование процесса регистрации пользователей

Страница регистрации выбранного веб-приложения выглядит следующим образом, (рис.23):

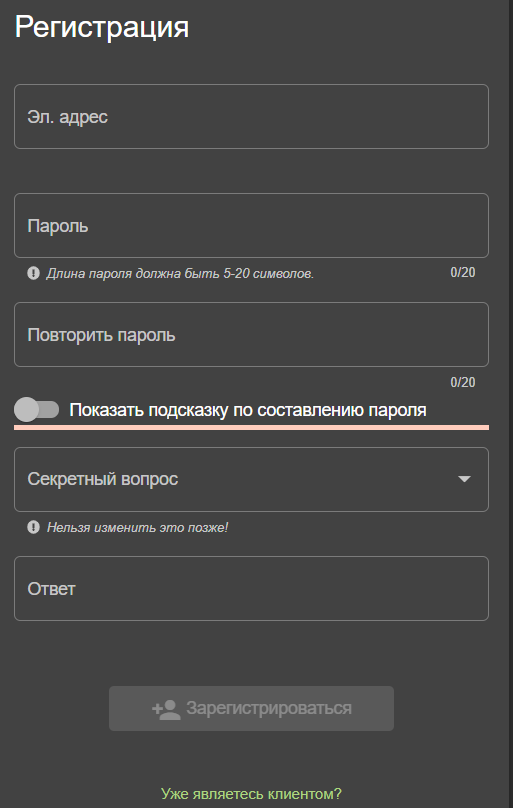


Рис. 23. Форма регистрации пользователей

Соответствие требованиям:

1. Пользователь веб-приложения имеет возможность зарегистрироваться
2. Регистрационные данные пользователя проверяются автоматически
3. Пользователь имеет возможность зарегистрироваться в системе несколько раз, при этом должен указывать новую почту, однако проверка поля Email чувствительна к регистру, поэтому почта aKotov@mail.ru и почта akotov@mail.ru будут считаться разными почтами.
4. Всем пользователям при регистрации выдается роль Customer
5. Для регистрации не требуется удостоверение личности.
6. Зарегистрированные личности не проверяются.

Данные регистрации пользователя можно перехватить, использовав прокси, (рис.24):

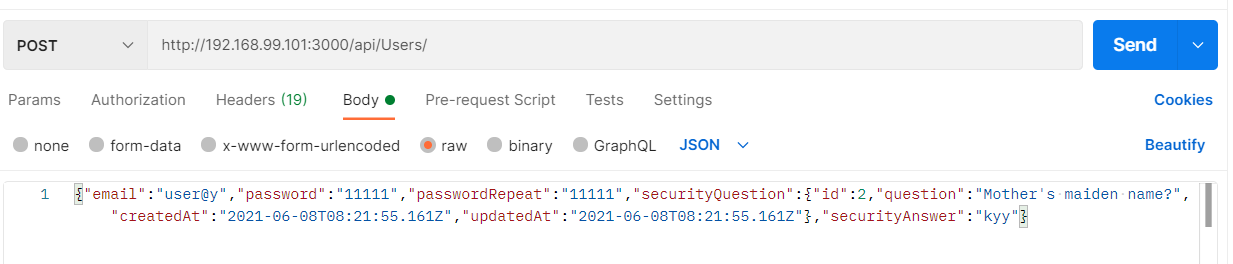


Рис.24. Перехваченный запрос регистрации пользователя

Также во время проверки формы регистрации была обнаружена уязвимость: при вводе и успешном подтверждении пароля, пользователь может изменить пароль и регистрации произойдет успешно, (рис.25-рис.27.)

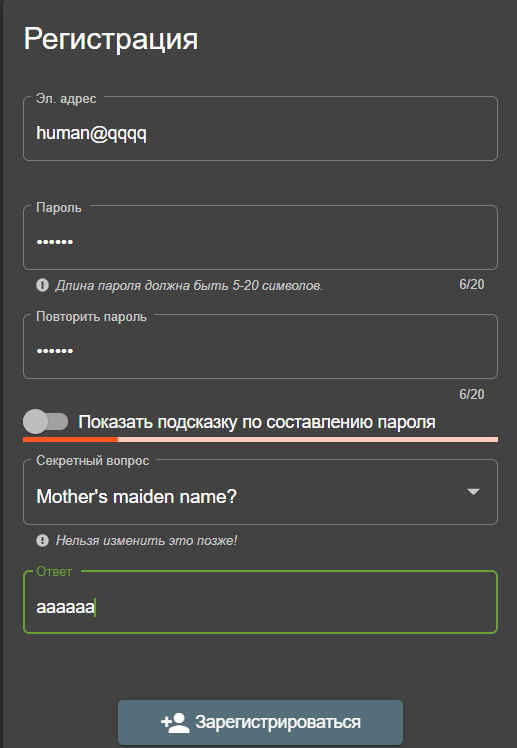


Рис.25. Ввод корректных данных при регистрации

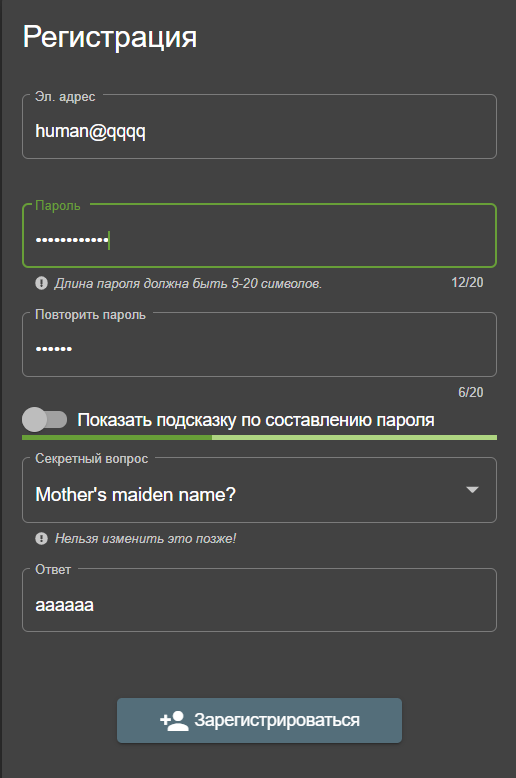


Рис.26. Изменение поля подтверждения пароля

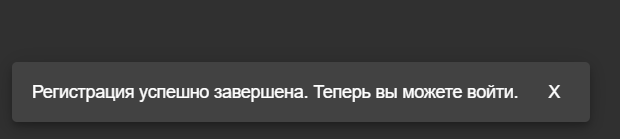


Рис.27. Результат найденной уязвимости

## 5. Тестирование аутентификации

### 5.1. Тестирование учетных данных, передаваемых по зашифрованному каналу

Веб-приложение работает изначально по протоколу HTTP, что не подразумевает под собой шифрование данных при передаче, что демонстрируют следующие действия:

Для проверки шифрования данных был захвачен трафик веб-приложения при помощи инструмента WireShark, (рис.28.):

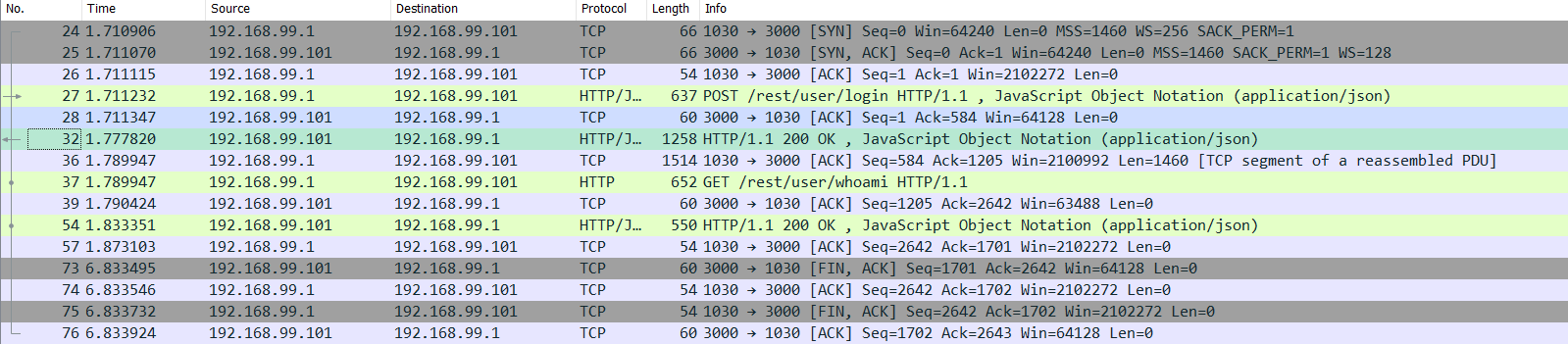


Рис.28. Перехваченный трафик запроса посредством Wireshark

После того, как пользователь прошел аутентификацию, появляется возможность обнаружить POST запрос к странице login. Далее при помощи команды «Следовать HTTP» открывается окно с исходным кодом, на котором можно обнаружить введенный логин и пароль пользователя, (рис.29):

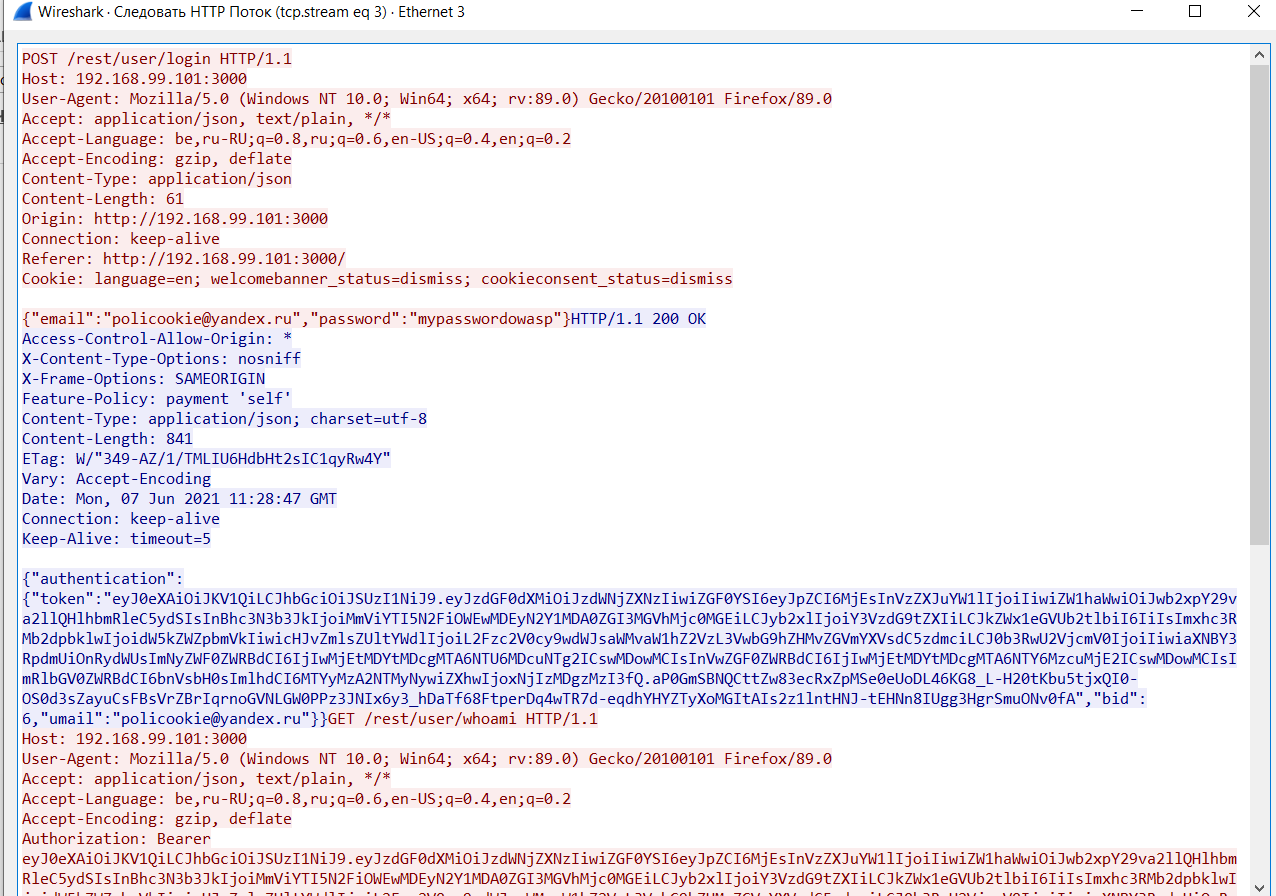


Рис.29. Перехваченный пакет

### 5.2. Проверка слабого механизма блокировки

Посредством ручных проверок формы аутентификации пользователя, было обнаружено, что алгоритма блокировки нет, из чего следует, что, зная почту пользователя при помощи брутфорса можно подобрать его пароль.

В предыдущих тестах была обнаружена информация о всех пользователях системы с указанием их почты, (рис.30):



Рис.30. Данные об администраторе приложения

Для брутфорса использовался инструмент Burp Suite [13]. Атака включала в себя следующие действия, (рис.31):

1. Настройка прокси для Burp Suite
2. Настройка прокси браузера, в котором запущено приложение (Firefox)
3. Аутентификация на странице логина с корректной учетной записью
4. Поиск в Burp Suite требуемого запроса для аутентификации

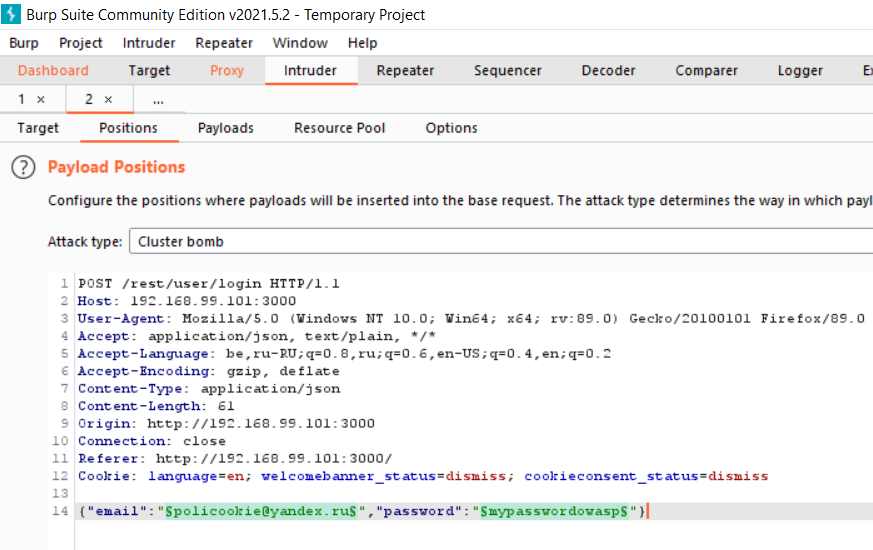


Рис.31. Перехваченные запрос аутентификации при помощи инструмента Burp suite

Брутфорс осуществляется посредством автоматической отправки перехваченного запроса веб-приложению с заданными логином и паролем

Далее инструмент отправляет запросы веб-приложения с заданными параметрами логина и пароля, которые можно загрузить из файлов.

1. Посредством поиска был выбран и скачан набор наиболее распространённых паролей.
2. В ходе атаки было обнаружено, что ответ на запрос с логином [admin@juice-sh.op](mailto:admin@juice-sh.op) и паролем admin123 вернул ответ 200, что говорит о том, что запрос был произведен успешно и пароль подошел. (рис. 32.)

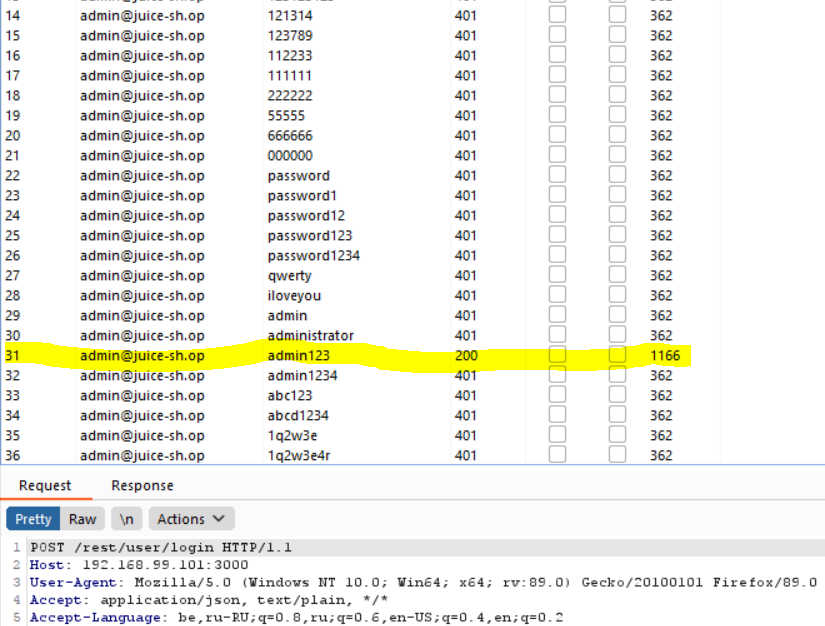


Рис.32. Найденный пароль в результате брутфорса

А также если рассмотреть ответ от сервера можно обнаружить токен администратора (рис.33.):

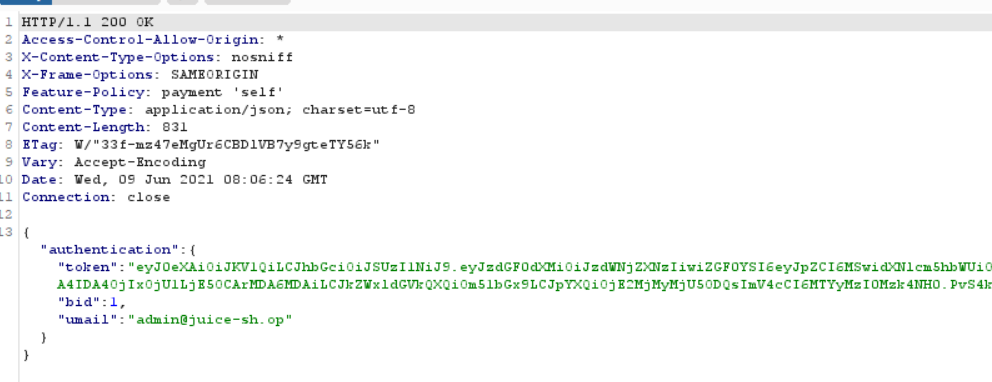


Рис.33. Перехваченный пакет после успешной аутентификации

Используя эти логин и пароль был успешно произведен вход в систему в роли администратора (рис. 34.)



Рис 34. Успешный вход в систему в качестве администратора

### Тестирование ненадежной политики генерации паролей

**Применение:**

Для проведения этого теста необходимо ответить на следующие вопросы:

1. Какие символы разрешено и запрещено использовать в пароле? Требуется ли от пользователя использовать символы из разных наборов символов, такие как строчные и прописные буквы, цифры и специальные символы?

Ответ: для пароля можно использовать любые символы из латинского и русского алфавита, цифры, а также спецсимволы и пробелы. Ограничений и условий для пароля, кроме его длины от 5 до 20 символов, не предусматривается.

2. Как часто пользователь может менять свой пароль? Как быстро пользователь может изменить свой пароль после предыдущего изменения? Есть ли возможность использовать предыдущий пароль?

Ответ: ограничений на смену пароля нет, пароль можно изменить мгновенно, также существует возможность изменить пароль на ранее используемый.

3. Когда пользователь должен изменить свой пароль?

Ответ: срок действия пароля не ограничивается.

1. Как часто пользователь может повторно использовать пароль? Сохраняет ли приложение историю ранее использованных пользователем паролей?

Ответ: приложение не сохраняет историю паролей пользователей, пользователь может повторно использовать пароль.

1. Насколько следующий пароль должен отличаться от последнего пароля?

Ответ: пароль может не отличаться от предыдущего.

6. Запрещено ли пользователю использовать свое имя пользователя или другую информацию учетной записи (например, имя или фамилию) в пароле?

Ответ: Ограничения к паролю не определены.

1. Какую минимальную и максимальную длину пароля можно установить, и подходят ли они для конфиденциальности учетной записи и приложения?

Ответ: Минимальная длина пароля составляет 5 символов, максимальная длина пароля 20 символов, однако при определенных манипуляциях длина пароля может составлять 1 символ.

8. Можно ли установить общие пароли, такие как Password1 или 123456?

Ответ: в приложении не существует ограничений на распространенные пароли.

Изучив политику генерации паролей приложения, можно сделать вывод, о том, что она абсолютно не соответствует требованиям безопасности, предоставляя злоумышленнику с легкостью подобрать пароль или обойти систему аутентификации.

## Тестирование авторизации

### Тестирование повышения привилегий

Несмотря на то, что по умолчанию при регистрации пользователя ему выдается роль customer, существует возможность задать эту роль отличной от customer, например, admin или deluxe, тем самым повысив его привилегии. Для присвоения прав администратора обычному пользователю на этапе регистрации были выполнены следующие шаги:

1. При помощи прокси получить запросы, которые выполняются при регистрации и входе в систему этим пользователем.
2. Изменить запрос аутентификации пользователя с добавлением роли (рис. 36.)

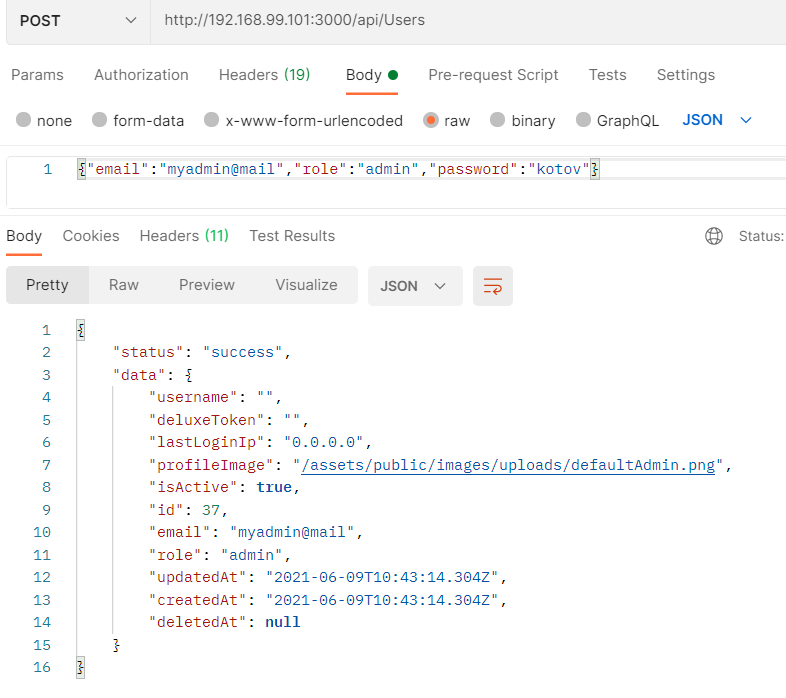


Рис. 36 результат измененного запроса с целью смены роли пользователя

После чего мы получаем нового пользователя с ролью администратора. Аналогичным образом можно зарегистрировать пользователя с ролью accounting, получив доступ к управлению заказами пользователей.

## Тестирование управления сессиями(сеансом)

### Тестирование тайм-аутов сессии

Эмпирическим методом было выяснено, что тайм-аут сессии не установлен.

### Тестирование функции выхода из системы

**Применение:**

**Тестирование пользовательского интерфейса:**

В ходе проведения теста было обнаружено:

* что кнопка выхода из системы присутствует не на всех страницах веб-приложения (рис.37.):

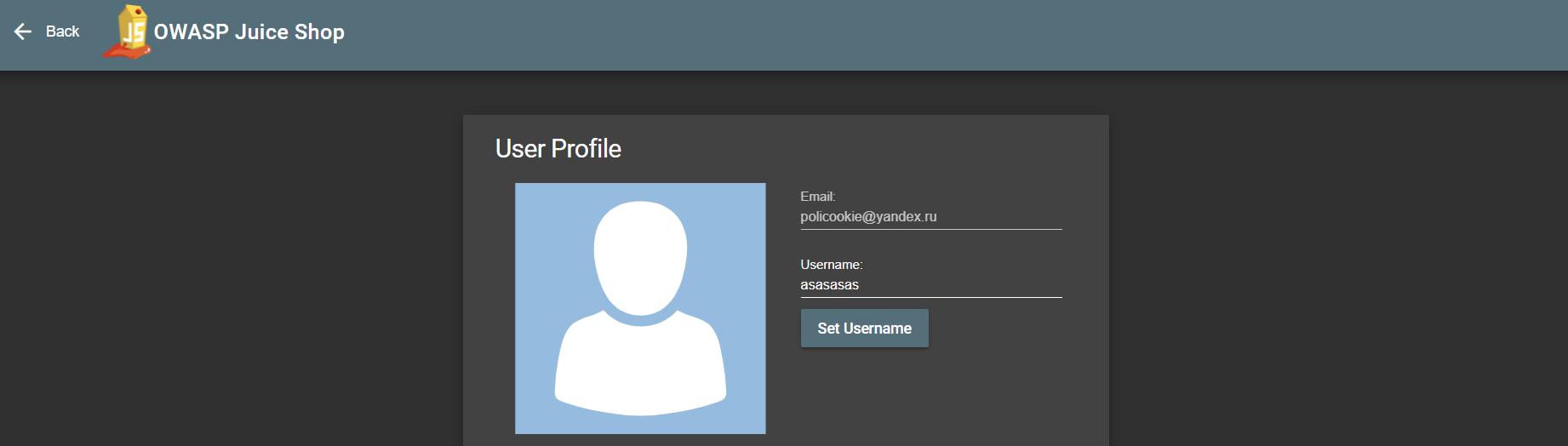


Рис. 37. Форма изменения данных о пользователе

* кнопка выхода всегда расположена в верхнем меню сайта, при прокрутке контента исчезает из видимости
  1. **Тестирование завершения сессии на стороне сервера:**

Тестирование осуществлялось на странице профиля пользователя. Первоначально были получены cookie (рис.38.):

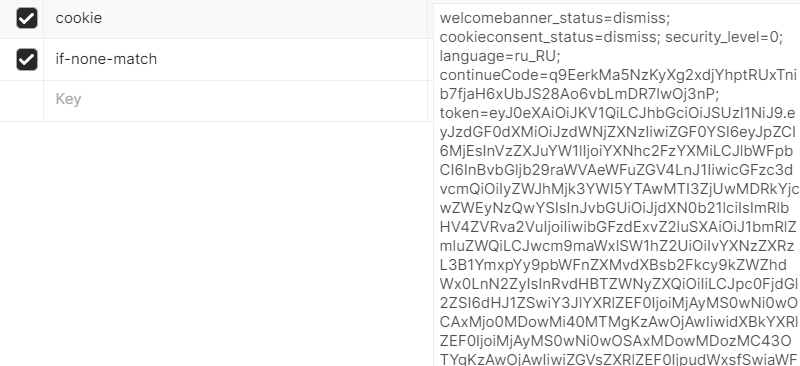


Рис. 38 Cookie файлы перехваченного запроса авторизированного пользователя

После чего был произведен выход из системы, и попытка зайти на страницу профиля в ходе которой был получен следующий результат (рис.39.)

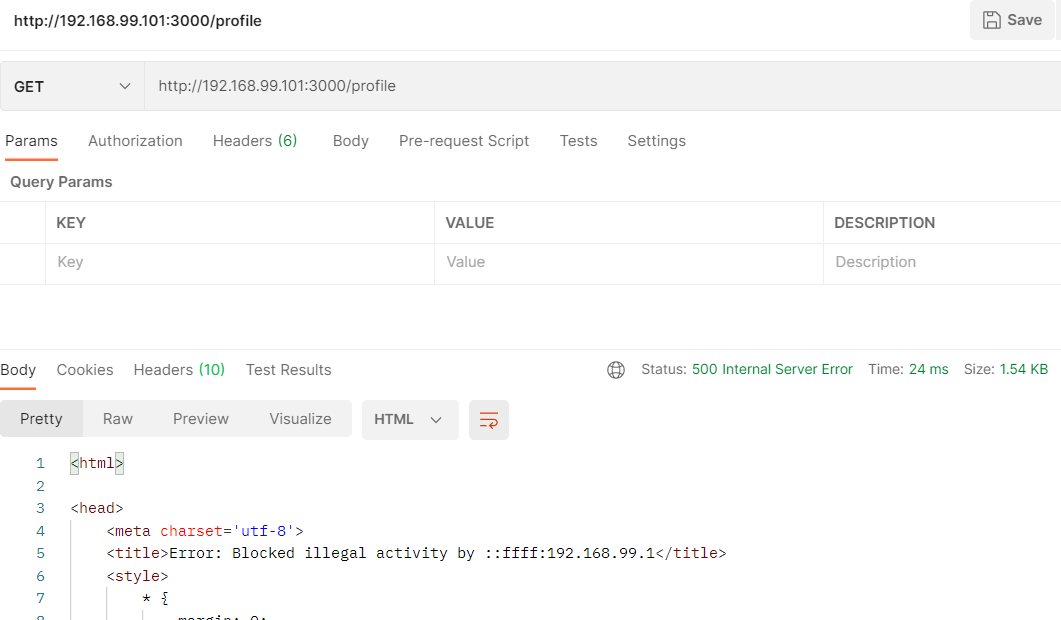


Рис 39. Запрос к странице, разрешенной к просмотру только аутентифицированному пользователю

Проанализировав файлы cookie, аутентифицированного и не аутентифицированного пользователя, можно сделать вывод, что отличают они только наличием токена. Все остальные данные совпадают (рис.40, рис.41.).

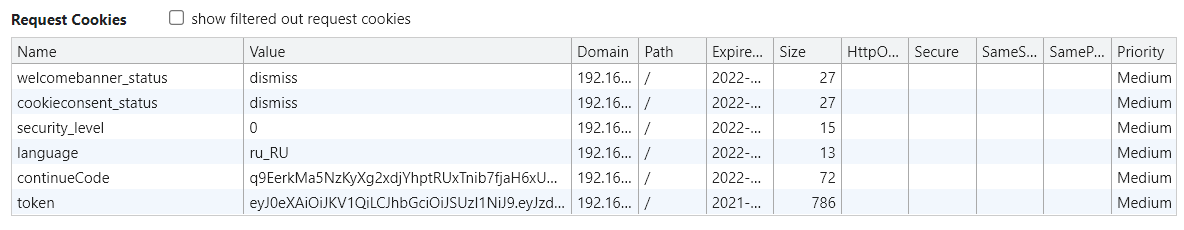


Рис. 40 Header запроса страницы аутентифицированного пользователя

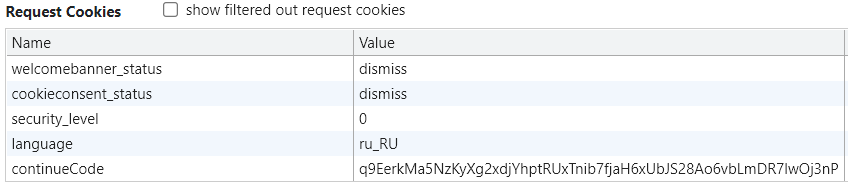


Рис. 41 Header запроса страницы не аутентифицированного пользователя

После замены текущего cookie на cookie аутентифицированного пользователя и отправки запроса на сервер страницы профиля был получен следующий результат (рис.42.):

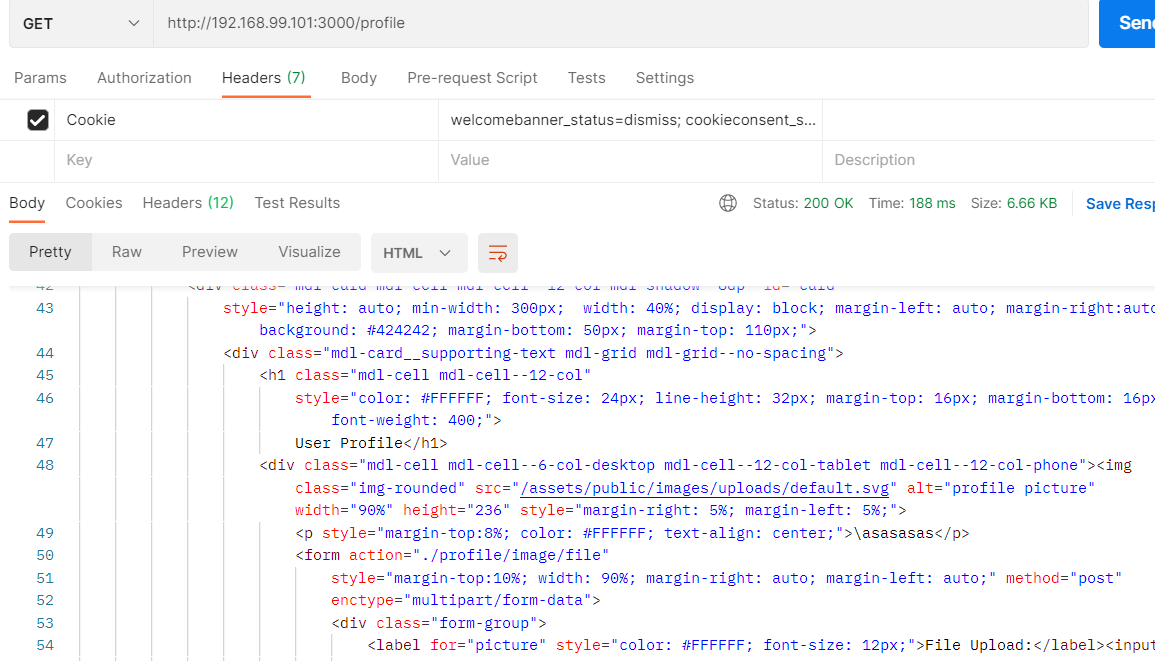


Рис 42. Результат запроса с измененным Header

Из которого можно сделать вывод, что токен сессии на сервере не обновляется после выхода из системы.

## Тестирование валидации входных данных

### Тестирование отраженных межсайтовых сценариев

**Применение:**

В приложении реализуется функциональность оформления заказов на покупку товаров, после которой пользователь имеет возможность отследить свой заказ (рис. 43.).

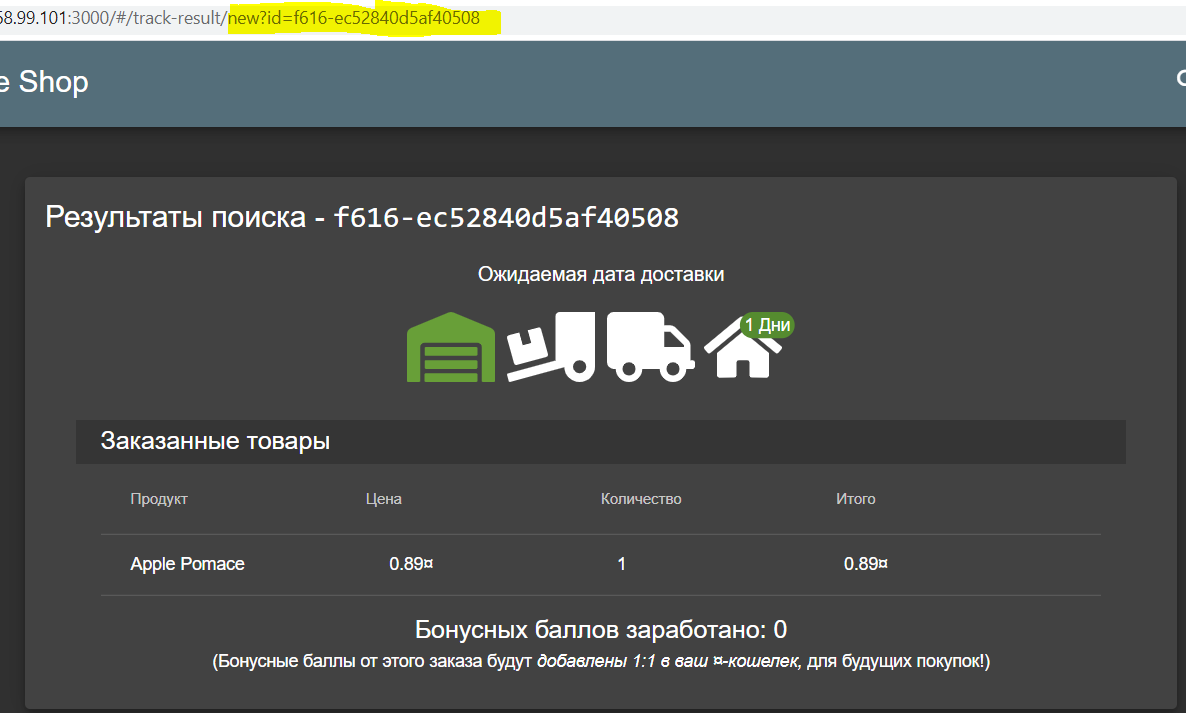


Рис 43. Страница отслеживания заказа

После анализа текущей страницы веб-приложение были сделаны выводы о том, что потенциальным вектором атаки может послужить url страницы. Для тестирования этого вектора атаки была сформирована следующая строка [14] *<iframe src="javascript:alert('attack!');">* , в результате ввода которой в url страницы атака была успешно произведена (рис. 44.).

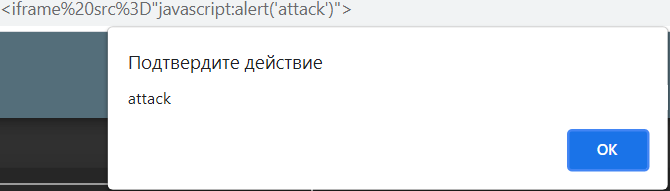


Рис. 44. Результат применения reflected xss

### Тестирование хранимых межсайтовых сценариев

**Применение:**

Для проведения XSS Stored на сайте было обнаружено поле ввода, хранимое в фоновом режиме, а именно имя пользователя на странице профиля. Для тестирования были сформированы различные скрипты JavaScript кода:

*<</>script>alert('xss');<</>/script>*

*<a><</>script><</>/script></a>*

*<a><</>script>alert('xss');<</>/script></a>*

Вывод: приложение устойчиво к атаке на URL по причине кодирования символов «<>» на мнемоники «lt» и «jt».

### Тестирование SQL-инъекций

**Применение:**

Отталкиваясь от предположения о том, что при входе в систему веб-приложение взаимодействует с базой данных, для тестирования SQL-инъекций была выбрана форма аутентификации (рис.45.):

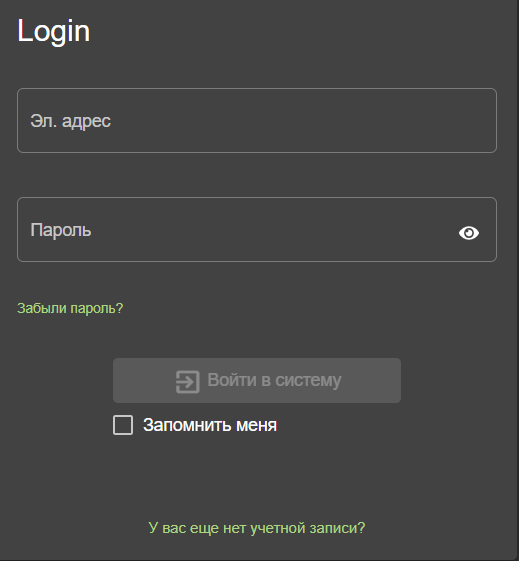


Рис. 45. Форма аутентификации пользователя

В результате проведения предыдущих этапов тестирования был определен список всех пользователей системы, в том числе с повышенными привилегиями.

С целью получить данные этих пользователей, а конкретно пользователя ciso@juice-sh.op, и предполагая, что sql-запрос [15] для проверки учетных данных пользователя имеет вид:

*SELECT \* FROM Users WHERE Username='$username' AND Password='$password'*

входные данные в форму логина и пароля были модифицированы, а именно к логину пользователя была добавлена строка «'--», используемая в SQL для комментирования. Таким образом, наш запрос стал иметь вид:

*SELECT \* FROM Users WHERE Username='$usernam'--' AND Password='$password'*

, тем самым игнорируя проверку пароля (рис.46.).

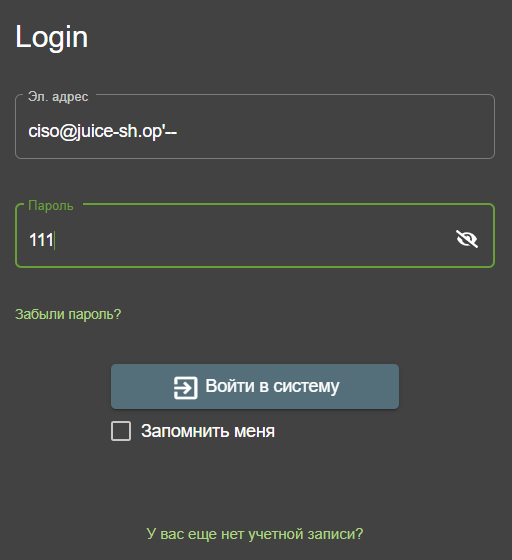


Рис. 46. Форма аутентификации пользователя с sql-инъекцией

Аналогичным образом входные данные логина можно было изменить следующим образом (рис. 47.), что также привело к успешному входу в систему без корректного пароля (рис.48.):

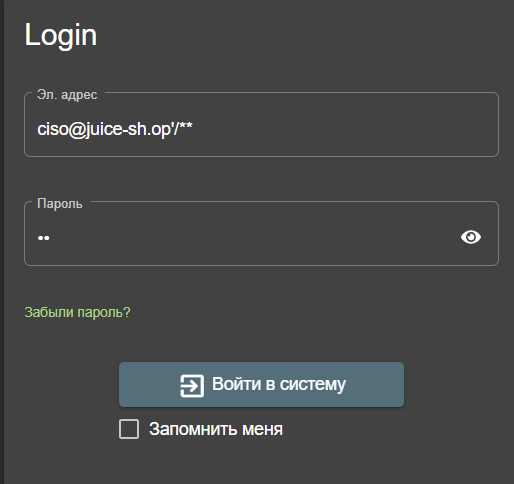


Рис. 47. Форма аутентификации пользователя с sql-инъекцией

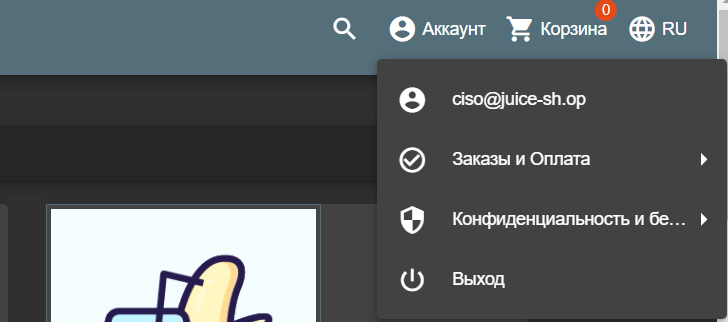


Рис. 48. Результат использования sql-инъекции

Также посредством ввода различных запросов sql в поле логина, была обнаружена ошибка sql-сервера, демонстрирующая нам, каким действительно образом сформирован запрос аутентификации пользователя, а также какая СУБД используется в веб-приложении (рис.49.):

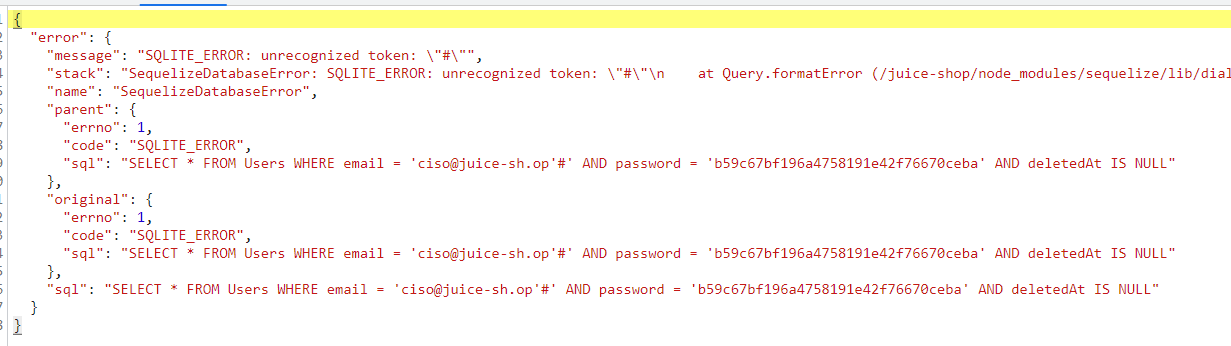


Рис 49. Ошибки sql-сервера

Эта информация может быть полезна при манипулировании данными базы данных.

### Тестирование инъекций заголовка Host

**Применение:**

В ходе тестирования была выбрана страница веб-приложения, на которой происходило перенаправление и использованы оба предложенных метода инъекции заголовка (рис.50, рис.51, рис.52).

Вывод: результаты тестирования показали, что в веб-приложении существует проверка заголовка Host.

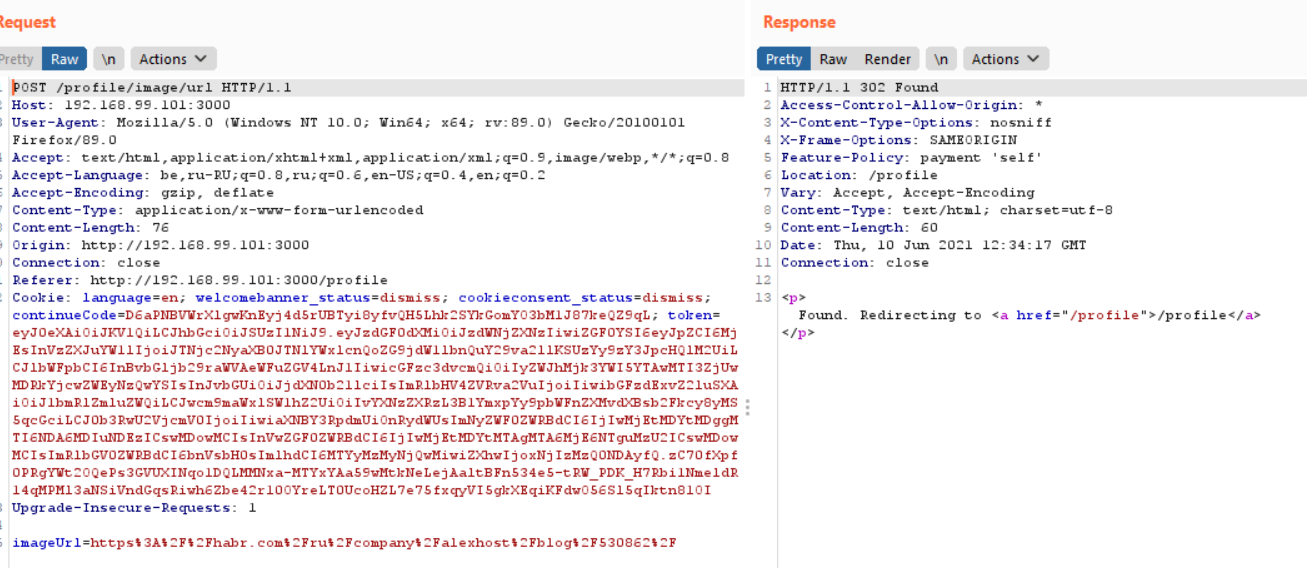


Рис 50. Перехваченный запрос перенаправления на другую страницу

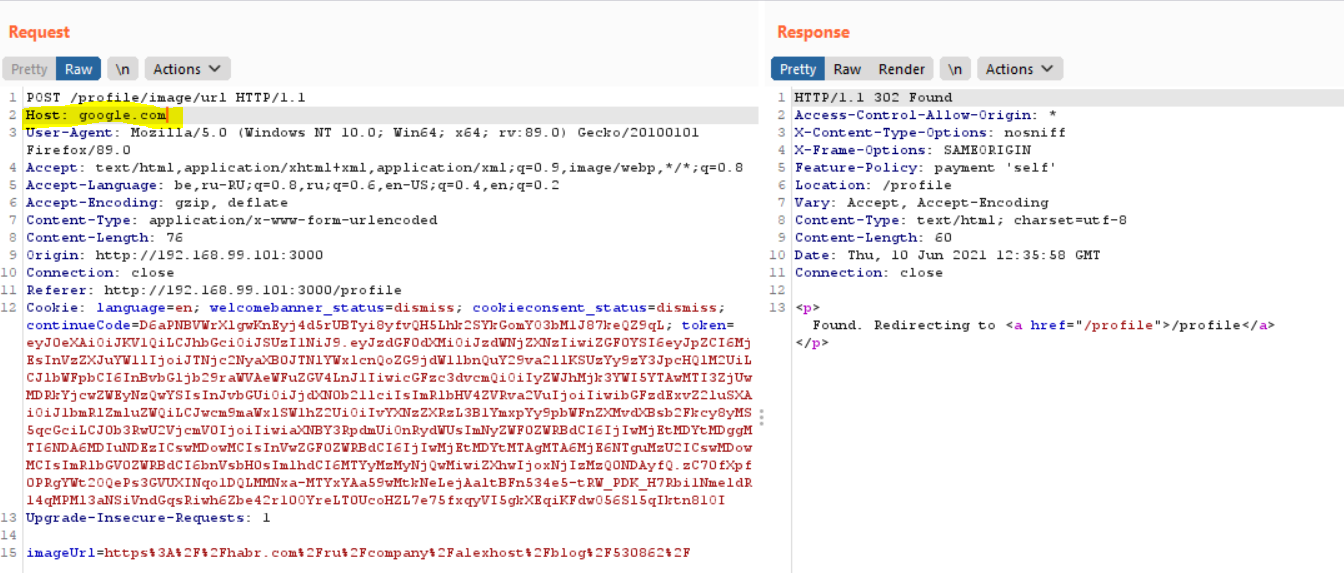


Рис 51. Измененное заголовок Host перехваченного запроса

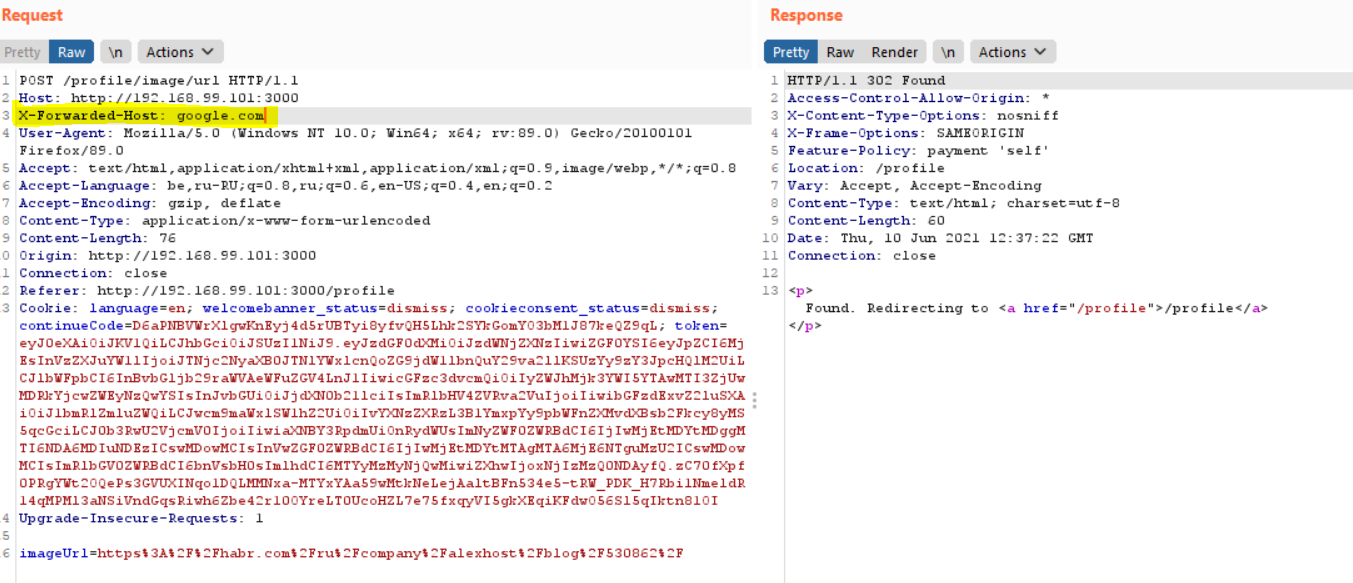


Рис 52. Добавленный заголовок X-Forward-Host перехваченного запроса

Выводы: в ходе проведения тестирования приложения на проверку входных данных было выяснено, что приложение уязвимо к атакам межсайтового скриптинга, sql-инъекций, однако защищено от инъекций заголовка Host. Использование XML-инъекций не было протестировано, поскольку в приложении XML-файлы не используются.

## Тестирование на обработку ошибок

### Тестирование на неправильную обработку ошибок

**Применение:**

Для тестирования приложения на неправильную обработку ошибок были проделаны следующие действия:

1. Перехвачен трафик при помощи инструмента Burp Suite(рис.53)

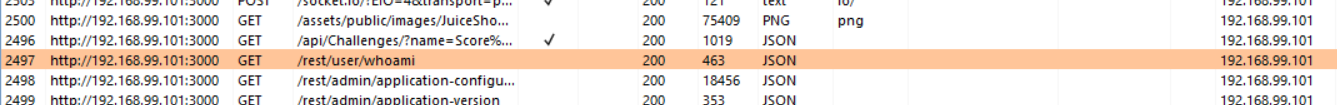


Рис. 53. Перехваченные запросы посредством инструмента Burp Suite

1. В запросе изменен путь на несуществующую директорию (рис.54.)



Рис. 54. Измененный перехваченный запрос к приложению

1. Повторно отправлен запрос на несуществующий путь и получен следующий результат (рис.55):

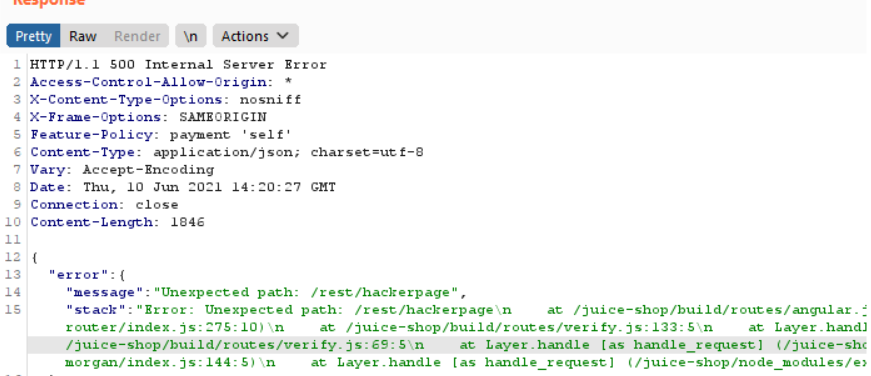


Рис. 55. Результаты отправки неверного запроса

, содержащий чрезмерно подробную информацию о полученной ошибке, позволяющей злоумышленнику лучше изучить структуру и методы приложения, а также в дальнейшем использовать эти данных в целях обхода защиты.

## Тестирование слабой криптографии

### Тестирование отправки конфиденциальной информации по незашифрованному каналу

В ходе проведения тестирования посредством захвата трафика с помощью инструмента Burp Suite было определено, что для cookie файлов флаг безопасности secure не устанавливается (рис.56).



Рис. 56. Тело перехваченного запроса к приложению

Данные пользователя, передаваемые при аутентификации не зашифрованы (рис. 57).

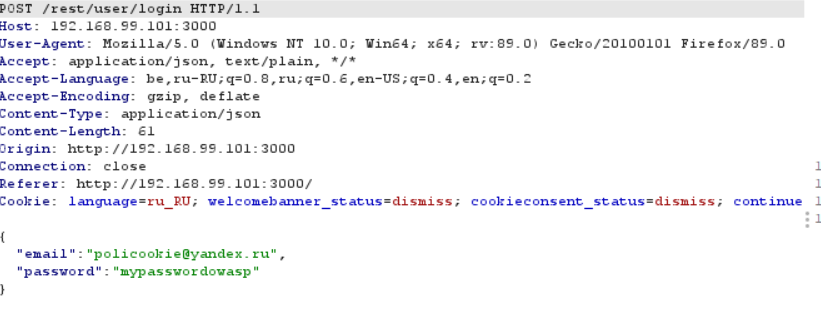


Рис. 57. Тело перехваченного запроса аутентификации пользователя

Конфиденциальная информация, как номер кредитной карты, при передаче также не шифруется (рис.58).



Рис. 58. Тело перехваченного запроса добавления кредитной карты пользователя

Выводы: в ходе проведения тестирования приложения на слабую криптографию было выяснено, что приложение не предусматривает шифрования передаваемой конфиденциальной информации, что может привести к серьезным проблемам безопасности.

## Тестирование бизнес-логики

### Тестирование возможности подделки запросов

**Применение:**

В тестируемом приложении есть система пополнения электронного кошелька условной валютой. Количество валюты, на которое можно пополнить свой кошелек единовременно, находится в пределах от 10 до 1000 условных единиц (рис.59).

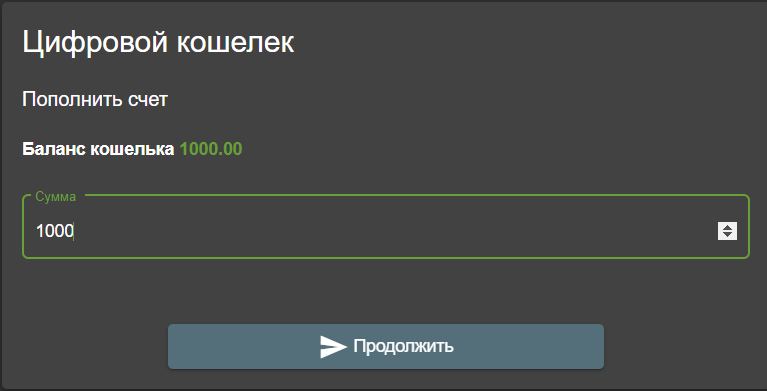


Рис. 59. Цифровой кошелек пользователя

После определения количества валюты, пользователю требуется выбрать карту, с которой можно пополнить свой кошелек (рис.60, рис.61.).

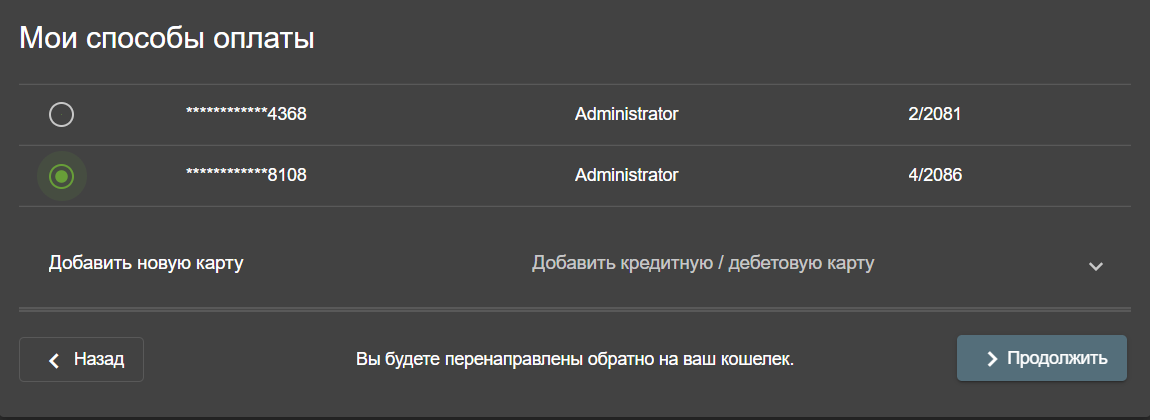


Рис. 60. Выбор карты пополнения цифрового кошелька

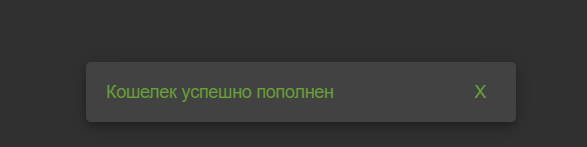


Рис. 61. Сообщение об успешном пополнении электронного кошелька

Уже на данном этапе прослеживается нарушение бизнес логики:

* Нет проверки существования карты
* Не требуется подтверждение перевода денег с карты
* Не существует лимита на количество операций пополнения кошелька с карты
* Не предусмотрена логика отсутствия финансов на карты, с которой производится пополнение.

Кроме того, была обнаружена следующая ошибка приложения, которая воспроизводиться следующими действиями:

1. Осуществить пополнение кошелька с карты
2. Нажать кнопку назад в панели браузера, тем самым вернувшись на страницу выбора способа оплаты
3. На этой странице повторить пополнение кошелька

В итоге, был получен результат, сообщающий нам, о том, что баланс нашего кошелька составляет «NaN» (рис.62.) (Специальный тип JavaScript – Not a Number), из чего можно сделать вывод, что приложение имеет серьезный дефект, а вся валюта с кошелька пропала.

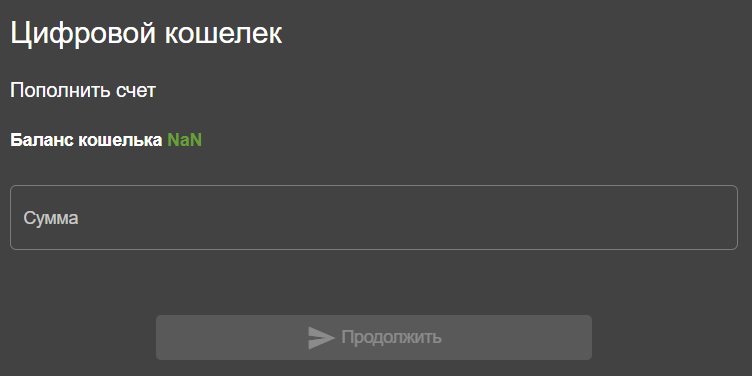


Рис. 62. Обнаруженный дефект приложения при пополнения электронного кошелька

Далее процесс пополнения кошелька с кредитной/дебетовой карты был рассмотрен с точки зрения HTTP-запросов с использованием инструмента Burp Suite.

Запрос пополнения кошелька с карты выглядит следующим образом (рис.63.):

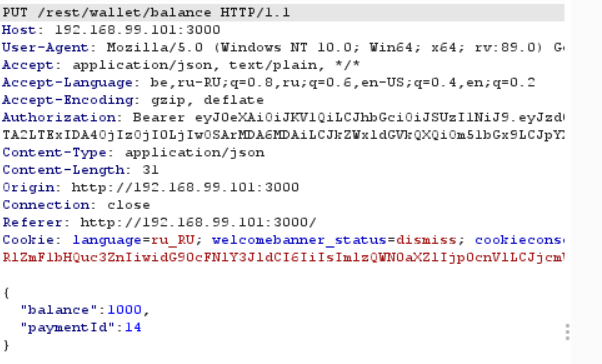


Рис. 63. Тело перехваченного запроса пополнения электронного кошелька

Изучив структуру запроса появился ряд вопросов:

1. Изменив параметр balance можно ли обойти ограничение в 1000 условных единиц?
2. Как будет вести себя приложение при подстановке в параметр balance неожидаемого значения, например, строки.
3. Можно ли, изменив параметр paymentId, отправить деньги на другой электронный кошелек

Ответы:

1. Перехватив запрос пополнения кошелька и изменив параметр balance на 10000 (рис. 64), кошелек успешно пополнился на 10000 условных единиц, тем самым обойдя существующее ограничение (рис.65).

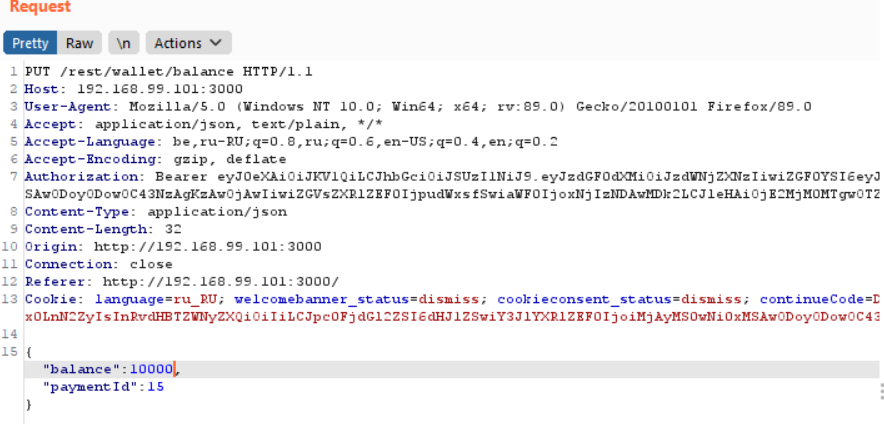


Рис. 64. Тело запроса пополнения электронного кошелька с измененными полем balance

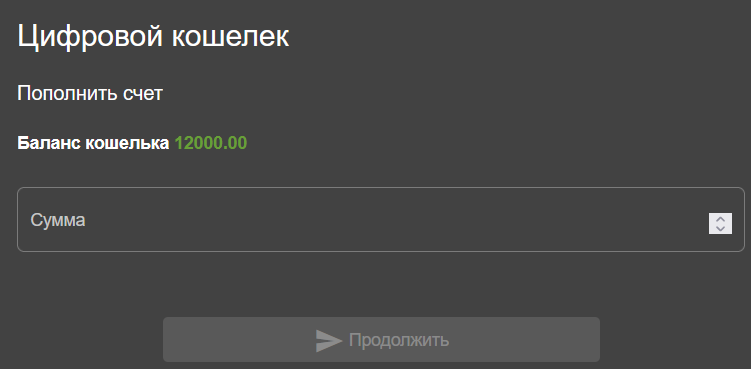


Рис. 65. Результаты отправки измененного запроса пополнения электронного кошелька

2. Изменяя параметр balance на неожидаемые данные, было определено, что приложение успешно принимает запрос, однако не изменяет значения кошелька.

3. Для ответа на этот вопрос был создан еще один тестовый аккаунт, его электронный кошелек пополнен на 1000 единиц, а также посредством изучения запросов, получен id электронного кошель paymentId.

Изменив параметр paymentId запроса первого тестового аккаунта на paymentId нового тестового аккаунта, было выяснено, что изменение запроса отправить деньги на другой кошелек не получится (рис.66.):

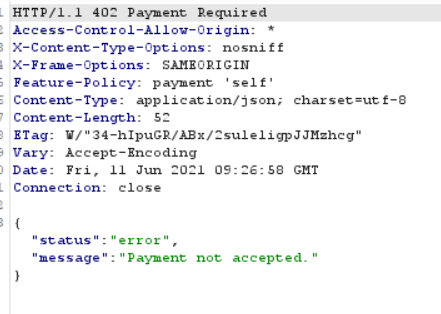


Рис. 66. Результат отправки запроса пополнения электронного кошелька с измененным полем paymantId

Выводы: тестирование веб-приложения вышеперечисленными методами показало, что ряд процессов не отвечают требованиям бизнес-логики приложения.

## Тестирование на стороне клиента

### Тестирование межсайтового скриптинга на основе DOM

**Применение:**

**Обнаружение ввода:**

На странице веб-приложения есть поле ввода для фильтрации продуктов. При отслеживании запросов веб-приложения посредством прокси-сервера, было обнаружено, что при вводе значения фильтра товаров, запроса к серверу не происходит, из чего можно сделать вывод, что пользовательские данные динамически отображаются на странице (рис. 67.).



Рис. 67. Поле ввода фильтра товаров

Для анализа этого вектора входных данных была сформирована следующая строка: *<iframe src="javascript:alert('attack!');">* , в результате ввода которой атака была успешно произведена (рис. 68).

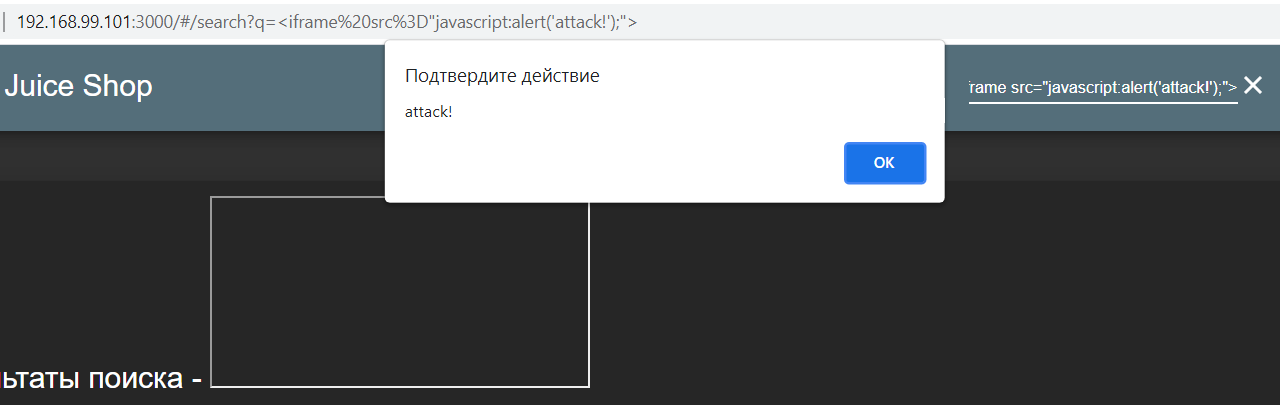


Рис. 68. Результат использования XSS dom

### 12.2. Тестирование HTML инъекций

**Применение**:

Для тестирования HTML инъекции [16] была сформирована строка и введена в форму поиска по сайту:

*<a href="https://habr.com/ru/company/owasp/blog/326362/»> Результат можно просмотреть, перейдя по ссылке</a>*

В результате на странице отобразилась ссылка, нажав на которую пользователь перейдет на сайт habr.com.(рис.69.)

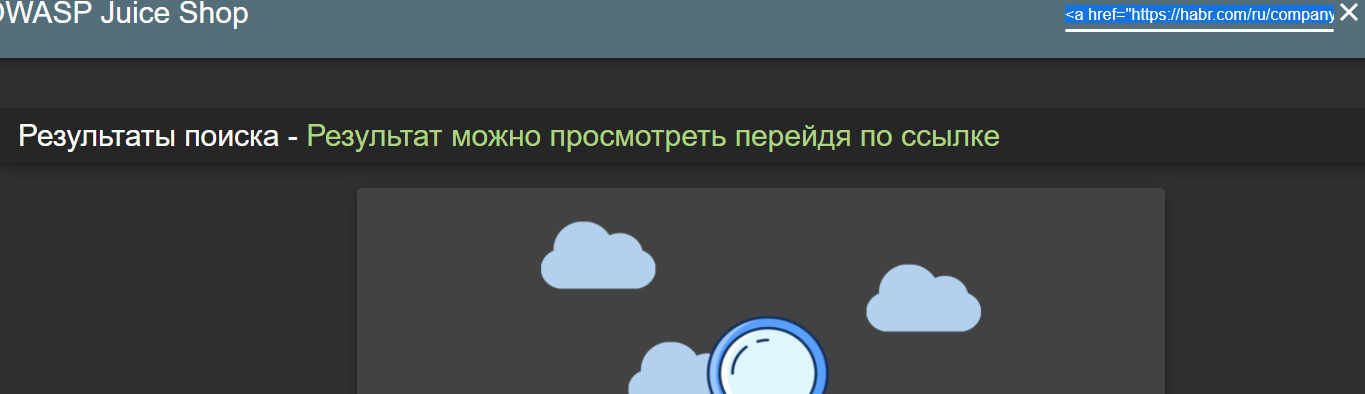


Рис. 69. Результат использования HTML-инъекции

Аналогичным образом, злоумышленник может полностью изменить содержание страницы веб-приложения с целью кражи данных жертвы.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы были выполнены следующие задачи:

1. Рассмотрены существующие методологии тестирования на проникновение такие как OSSTMM, PTES, OWASP Testing Guide.
2. Изучены методы тестирования на проникновение веб-приложений методологии OWASP Testing Guide в рамках OWASP Top-10.
3. Продемонстрирована работа методов тестирования на проникновение веб-приложений, описанных в OWASP Testing Guide, на примере тестового стенда.

Результаты проведенного исследования позволяют сделать вывод о том, что среди рассмотренных базовых методологий тестирования на проникновение, таких как:

* OSSTMM;
* PTES;
* OWASP Testing Guide

веб-приложений, изученная методология OWASP Web Security Testing Guide, за счет структурного подхода к тестированию и широкого спектра методов, показала свою эффективность в достижении цели обеспечения информационной безопасности веб-приложений.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Макаренко С. И., Смирнов Г. Е. Анализ стандартов и методик тестирования на проникновение 2020 28с.
2. Whay is Penetration testing. Электронный ресурс. URL: https://www.coresecurity.com/penetration-testing
3. Приказ об утверждении требований по обеспечению безопасности значимых объектов критической информационной инфраструктуры российской федерации. Электронный ресурс. URL:

https://fstec.ru/normotvorcheskaya/akty/53-prikazy/1592-prikaz-fstek-rossii-ot-25-dekabrya-2017-g-n-239

1. Приказ об утверждении состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных. Электронный ресурс. URL: https://fstec.ru/normotvorcheskaya/akty/53-prikazy/691
2. Положение Банка России от 09.06.2012 N 382-П (ред. от 07.05.2018) "О требованиях к обеспечению защиты информации при осуществлении переводов денежных средств и о порядке осуществления Банком России контроля за соблюдением требований к обеспечению защиты информации при осуществлении переводов денежных средств". Электронный ресурс. URL: http://www.consultant.ru/document/cons\_doc\_LAW\_131473/34bc7aedea02bb3a7586002ebe8a2a9a83350a03/
3. Тестирование на проникновение – ответственность. Электронный ресурс. URL: https://defcon.ru/legislation/6426/
4. OSSTMM 3. Электронный ресурс. URL: https://www.isecom.org/research.html#content5-9d
5. The Penetration testing Execution Standart: электронный ресурс. URL: http://www.pentest-standard.org/index.php/Main\_Page
6. OWASP Web Security Testing Guide. Электронный ресурс. URL:https://owasp.org/www-project-web-security-testing-guide/
7. Owasp Top Ten. Электронный ресурс. URL: https://owasp.org/www-project-top-ten/
8. Nmap: The network Mapper – Free Security Scanner. Электронный ресурс. URL: https://nmap.org/
9. OWASP ZAP. Электронный ресурс. URL: https://www.zaproxy.org/
10. Burp Suite. Электронный ресурс. URL: https://portswigger.net/burp
11. Руководство по осуществлению XSS. Электронный ресурс. URL: https://cisoclub.ru/rukovodstvo-po-osushhestvleniyu-cross-site-scripting-xss/
12. Уязвимость SQL-инъекция (ч. 1): Основы SQLi, простая инъекция с UNION. Электронный ресурс. URL: https://hackware.ru/?p=3362
13. Подробное руководство по HTML-инъекциям https://habr.com/ru/company/alexhost/blog/530862/