МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Корпоративные информационные системы
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка средства автоматизированного поиска уязвимостей для веб-приложений»

Обучающийся Н.Д. Никифоров
(Инициалы Фамилия) (личная подпись)

Руководитель к.э.н., доцент, Т.А. Раченко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Тема бакалаврской работы: «Разработка средства автоматизированного поиска уязвимостей для веб-приложений».

Бакалаврская работа посвящена разработке средства автоматизированного поиска уязвимостей для Веб-приложений.

В ходе выполнения исследований по бакалаврской работе была поставлена задача, осуществлены проектирование и разработка автоматизированной системы поиска уязвимостей.

Во введении прописывается актуальность темы, написаны цель и задачи.

В первом разделе ставится задача на исследование, рассматриваются инструменты тестирования безопасности Веб-приложений и методы и фреймворки автоматизированного тестирования уязвимостей.

Во втором разделе рассматриваются алгоритмы и методы обнаружения уязвимостей, архитектура системы и описываются проблемы и решения в процессе разработки.

Третий раздел содержит реализацию автоматизированной системы поиска уязвимостей.

В заключении представлены результаты выполнения выпускной квалификационной работы.

Бакалаврская работа состоит из введения, трёх разделов, заключения и списка использованной литературы.

Бакалаврская работа состоит из 51 страниц, 16 рисунков, 4 таблиц и 25 источников.

Abstract

The title of the bachelor’s thesis is “Development of an automated vulnerability search tool for Webapplications”.

The research is devoted to developing an automated vulnerability search tool for Webapplications.

When doing a research, the task was set, the design and development of an automated vulnerability search system were carried out.

The introduction reveals the relevance of the research and gives a brief description of the work done.

The first section a research task is set, Webapplication security testing tools and methods and frameworks for automated vulnerability testing are considered.

The second section algorithms and methods of vulnerability detection, system architecture is considered and problems and solutions in the development process are described.

The third section contains the implementation of an automated vulnerability search system.

In conclusion, the conclusions of the entire work are drawn.

The bachelor’s thesis consists of an introduction, three sections, a conclusion and list of used literature.

The volume of the bachelor's thesis is 51 pages, it also contains 16 figures, 4 tables and a list of 25 references.

Содержание

Введение

Кибербезопасность стала серьезной проблемой в сегодняшнюю цифровую эпоху. Поскольку все больше предприятий и организаций полагаются на Web-приложения для выполнения своих повседневных операций, риски кибератак и утечек данных значительно возросли. Для решения этой проблемы разработка автоматизированного поиска уязвимостей в Веб-приложениях стала важнейшим аспектом индустрии кибербезопасности. Этот диплом направлен на изучение различных методов и методологий, используемых при разработке автоматизированного поиска уязвимостей в Веб-приложениях. Цель состоит в том, чтобы вооружить студентов необходимыми знаниями и навыками для разработки и внедрения эффективных решений для обеспечения безопасности Веб-приложений.

Эта работа актуальна так как в современном мире, так как интернет стал неотъемлемой частью нашей жизни. С ростом использования приложений также возросла потребность в мерах безопасности для защиты конфиденциальных данных. Уязвимости Веб-приложений могут привести к нарушениям безопасности, что может привести к потере личной и финансовой информации пользователей, ущербу для репутации организаций и юридическим последствиям.

Поэтому разработка автоматизированного поиска уязвимостей для приложений необходима для обнаружения и устранения проблем безопасности до того, как ими смогут воспользоваться хакеры. Этот диплом обеспечит всестороннее понимание методов, инструментов и методологий, используемых для разработки механизма автоматического поиска уязвимостей в приложениях.

Целью данной дипломной работы является разработка и внедрение инструмента автоматизированного поиска для выявления уязвимостей в Веб-приложениях.

Задачи:

Провести всесторонний анализ различных инструментов сканера уязвимостей, доступных на рынке, и оценить их эффективность и результативность. Также определить наиболее распространенные типы уязвимостей, обнаруженных в Веб-приложениях, а также инструменты и методологии, используемые для их обнаружения.

Предметом исследования будет процесс создания и внедрения автоматизированных средств обнаружения и выявления уязвимостей в Веб-приложениях.

Объектом исследования является процесс обнаружения уязвимостей в Веб-приложениях путем разработки автоматизированного инструмента.

1. Анализ предметной области

1.1. Обзор существующих методов и инструментов для поиска уязвимостей в Веб-приложениях

Уязвимости Веб-приложений – это слабые места в системе безопасности, которые злоумышленники могут использовать для получения несанкционированного доступа к конфиденциальным данным или нарушения целостности и доступности приложения. Эти уязвимости могут появиться во время проектирования и разработки приложения или из-за неадекватных методов обеспечения безопасности во время развертывания и обслуживания.

Некоторые наиболее распространенные уязвимости Веб-приложений и их влияние на безопасность приложения.

SQL-инъекция. Внедрение SQL – это тип атаки, при котором злоумышленник внедряет вредоносный код SQL в поля ввода приложения, чтобы получить несанкционированный доступ к базе данных [21]. Эта уязвимость позволяет злоумышленнику просматривать, изменять или удалять конфиденциальные данные из базы данных. Последствия успешной атаки путем внедрения SQL-кода могут быть серьезными: от финансовых потерь до компрометации конфиденциальной информации.

Чтобы предотвратить внедрение SQL, разработчики должны использовать параметризованные запросы и проверку ввода, чтобы убедиться, что пользовательский ввод очищается перед обработкой приложением.

Межсайтовый скриптинг (XSS) – это тип атаки, при котором злоумышленник внедряет вредоносные скрипты или код на Web-страницу, просматриваемую другими пользователями [18]. Эти скрипты могут использоваться для кражи конфиденциальной информации или выполнения несанкционированных действий от имени пользователя. XSS-атаки можно разделить на две категории: сохраненные и отраженные. Сохраненные XSS-

атаки позволяют злоумышленнику внедрить вредоносный код, который постоянно хранится на сервере, а отраженные XSS-атаки внедряют код, который немедленно отражается обратно пользователю.

Чтобы предотвратить XSS, разработчики должны дезинфицировать пользовательский ввод и кодировать вывод, чтобы предотвратить внедрение скриптов. Кроме того, использование Content Security Policy и файлов cookie только для HTTP может еще больше снизить риск XSS.

Подделка межсайтовых запросов (CSRF) – это тип атаки, при котором злоумышленник заставляет жертву выполнить действие в Web-приложении без ее ведома или согласия [7]. Атаки CSRF могут использоваться для выполнения несанкционированных действий, таких как изменение пароля жертвы или совершение несанкционированных покупок. Последствия успешной атаки CSRF могут варьироваться от незначительных неудобств до серьезных финансовых потерь.

Чтобы предотвратить CSRF, разработчики должны внедрить токены CSRF и управление сеансами, чтобы гарантировать, что запросы могут выполняться только авторизованными пользователями.

Сломанная аутентификация и управление сессиями. Нарушенная проверка подлинности и управление сеансом возникают, когда приложение не выполняет должным образом проверку подлинности пользователей или управление их сеансами. Это может позволить злоумышленнику обойти аутентификацию или перехватить сеанс пользователя, предоставив ему доступ к конфиденциальной информации или несанкционированным действиям. Последствия успешной атаки могут варьироваться от незначительных неудобств до серьезных финансовых потерь или репутационного ущерба.

Чтобы предотвратить нарушение аутентификации и управления сеансами, разработчики должны использовать безопасные методы аутентификации, такие как многофакторная аутентификация, и обеспечивать безопасное хранение и надлежащее аннулирование токенов сеанса.

Небезопасные прямые ссылки на объекты возникают, когда злоумышленник может получить доступ к конфиденциальным данным или функциям напрямую, манипулируя параметрами приложения. Это может позволить злоумышленнику просматривать, изменять или удалять конфиденциальные данные без надлежащей авторизации. Последствия успешной атаки могут варьироваться от незначительных неудобств до серьезных финансовых потерь или репутационного ущерба.

Чтобы предотвратить небезопасные прямые ссылки на объекты, разработчики должны использовать косвенные ссылки на объекты и должным образом проверять пользовательский ввод, чтобы гарантировать, что только авторизованные пользователи могут получить доступ к конфиденциальным данным или функциям.

Далее на рисунке 1 представлена классификация уязвимостей и атак

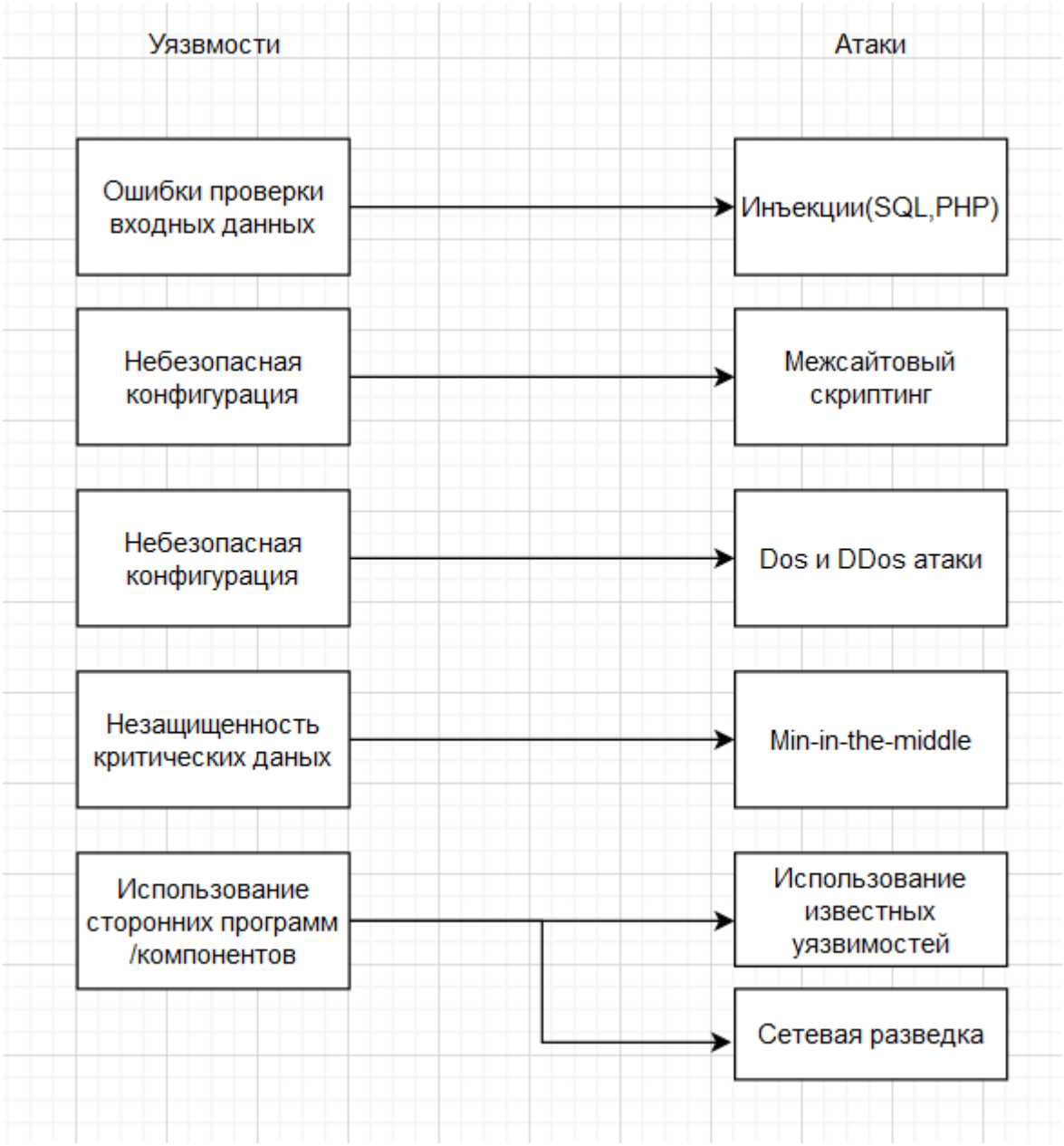

Рисунок 1 – Классификация уязвимостей и атак

Следует отметить, что уязвимости Веб-приложений представляют значительный риск для безопасности приложений и конфиденциальных данных, с которыми они работают. Разработчики и специалисты по безопасности должны работать вместе, чтобы выявить и устранить эти уязвимости, чтобы обеспечить безопасность приложения и его пользователей. Внедряя передовые методы безопасной разработки и

тестирования, организации могут снизить риск успешных атак и защитить свои активы от вреда.

Далее в таблице 1 представлены наиболее распространенные уязвимости

В данном рейтинге представлены только вредоносные программы, исключены из него рекламные программы, которые действуют весьма назойливо и доставляют неприятности пользователю, но не наносят вреда компьютеру.

Таблица 1 – Наиболее распространенные уязвимости

Название	% атакованных пользователей
Malicious	57,0
Trojan.Script.Generic	24,7
Trojan.Script.Iframer	16,0
Exploit.Script.Blocker	4,1
Trojan-Downloader.Win32.Generic	2,5
Trojan.Win32.Generic	2,3
Trojan-Downloader.JS.Iframe.diq	2,0

Все представители рейтинга – это, в основном, само-распространяющиеся программы и их компоненты.

Данные показывают, что инструментарий атак на бизнес отличается от того, что применяют в атаках на домашних пользователей. В атаках на корпоративных пользователей значительно чаще используются эксплойты к офисным приложениям, вредоносные файлы часто оказываются подписанными валидными цифровыми сертификатами, плюс к этому злоумышленники стараются использовать в своих целях доступные легальные программы, чтобы дольше оставаться незамеченными.

Тестирование безопасности Веб-приложений – важный процесс, который помогает обеспечить безопасность и надежность Веб-приложений. Он включает в себя выявление уязвимостей или слабых мест, которые могут быть использованы злоумышленниками, и принятие соответствующих мер для их устранения [11]. Этот процесс требует использования специализированных инструментов, предназначенных для выполнения

различных типов тестов Веб-приложений. В этой статье мы обсудим некоторые из самых популярных инструментов тестирования безопасности Веб-приложений:

- Burp Suite – это мощный инструмент для тестирования безопасности Веб-приложений, который позволяет вам перехватывать, изменять и анализировать трафик HTTP(S) между вашим браузером и целевым Web-приложением. Он включает в себя ряд инструментов для выполнения различных типов тестов, включая активное и пассивное сканирование, фаззинг и обнаружение уязвимостей. Burp Suite широко используется профессионалами в области безопасности и доступен как в бесплатной, так и в платной версиях;
- OWASP ZAP (Zed Attack Proxy) – это инструмент тестирования безопасности Веб-приложений с открытым исходным кодом, предназначенный для поиска уязвимостей в Веб-приложениях. Он включает в себя ряд функций, таких как инструменты сканирования, поиска и фаззинга, а также мощный механизм сценариев. OWASP ZAP прост в использовании и может быть интегрирован с другими инструментами тестирования безопасности;
- Nmap – популярный инструмент сетевого сканирования, который также можно использовать для тестирования безопасности Веб-приложений. Он включает в себя ряд функций, таких как сканирование портов, определение версии и снятие отпечатков ОС, которые можно использовать для выявления уязвимостей в Веб-приложениях. Nmap доступен бесплатно и широко используется профессионалами в области безопасности;
- Metasploit: Metasploit – это широко используемый инструмент тестирования на проникновение, который включает ряд модулей для тестирования безопасности Веб-приложений. Он включает в себя такие функции, как разработка эксплойтов, генерация полезной нагрузки и модули постэксплуатации, которые можно использовать

для проверки безопасности Веб-приложений. Metasploit доступен как в бесплатной, так и в платной версиях и широко используется профессионалами в области безопасности;

- Nikto – это сканер Web-серверов с открытым исходным кодом, который можно использовать для выявления уязвимостей в Веб-приложениях. Он включает в себя ряд функций, таких как сканирование CGI, поддержка SSL и сканирование портов, которые можно использовать для выявления уязвимостей в Веб-приложениях. Nikto бесплатен и прост в использовании, что делает его популярным среди профессионалов в области безопасности;
- SQLMap – популярный инструмент с открытым исходным кодом для тестирования безопасности Веб-приложений, использующих базы данных SQL. Он включает в себя ряд функций, таких как автоматическое обнаружение уязвимостей SQL-инъекций, автоматическое использование уязвимостей SQL-инъекций и поддержку ряда различных систем баз данных. SQLMap широко используется профессионалами в области безопасности и доступен бесплатно;
- Vega – это инструмент для тестирования безопасности Веб-приложений с открытым исходным кодом, который включает в себя ряд функций, таких как автоматическое сканирование, пассивный сканер и обнаружение уязвимостей. Он включает в себя мощный механизм сценариев, который можно использовать для автоматизации повторяющихся задач и выполнения пользовательских тестов Веб-приложений.

Тестирование безопасности Веб-приложений – важный процесс, который помогает обеспечить безопасность и надежность Веб-приложений. Существует широкий спектр инструментов для выполнения различных типов тестов Веб-приложений, и важно выбрать правильный инструмент для работы. Перечисленные выше инструменты являются одними из самых

популярных доступных инструментов тестирования безопасности Веб-приложений и широко используются профессионалами в области безопасности.

1.2. Анализ типичных уязвимостей и их последствий для Веб-приложений

Анализ распространенных уязвимостей в Веб-приложениях необходим для разработки автоматического средства поиска уязвимостей для Веб-приложений. Приложения широко используются для доступа к различным службам и данным в Интернете и уязвимы для различных типов угроз. Эти уязвимости могут привести к утечке данных, потере конфиденциальной информации и несанкционированному доступу к Web-приложению.

Одной из самых распространенных уязвимостей в Веб-приложениях является SQL-инъекция. Атаки с внедрением SQL используют слабые места в коде SQL для доступа или изменения конфиденциальных данных в базе данных. Их можно использовать для кражи учетных данных пользователя, удаления данных или получения доступа к конфиденциальной информации.

Другой распространенной уязвимостью являются атаки с использованием межсайтовых сценариев (XSS). В этих атаках злоумышленник внедряет вредоносный код в просматриваемую пользователем Web-страницу, которая может украсть конфиденциальную информацию или выполнить другие вредоносные действия.

Небезопасная связь между Web-приложением и сервером также является распространенной уязвимостью. Злоумышленники могут перехватывать данные, передаваемые между Web-приложением и сервером, чтобы получить доступ к конфиденциальным данным, таким как учетные данные для входа.

Неправильные настройки безопасности – еще одна распространенная уязвимость в Веб-приложениях. Неправильная настройка параметров

безопасности, таких как слабые пароли или отсутствие шифрования, может сделать Web-приложение уязвимым для атак.

Последствия этих уязвимостей могут быть серьезными. Нарушение данных может привести к финансовым потерям, ущербу для репутации бренда и юридическим последствиям. Кроме того, уязвимости могут привести к потере конфиденциальной информации, что может повлиять на конфиденциальность и безопасность пользователей.

Автоматический поиск уязвимостей может помочь быстро и эффективно идентифицировать эти уязвимости, снижая риск успешной атаки. Используя автоматизированные инструменты, разработчики могут выявлять уязвимости и предпринимать шаги по их устранению до того, как их можно будет использовать.

Необходимо проанализировать распространенные уязвимости в Веб-приложениях для разработки автоматического средства поиска уязвимостей. Web-приложения уязвимы для различных типов угроз, включая SQL-инъекцию, XSS-атаки, небезопасную связь и неверные настройки безопасности. Последствия этих уязвимостей могут быть серьезными, приводя к утечке данных и потере конфиденциальной информации. Автоматический поиск уязвимостей может помочь снизить эти риски, повысить безопасность Веб-приложений и защитить данные пользователей.

1.3. Обзор существующих методов защиты Веб-приложений от уязвимостей

Тестирование уязвимостей – важный процесс обеспечения безопасности Веб-приложений. Он включает в себя выявление уязвимостей в Web-приложении, которые могут быть использованы злоумышленниками [4]. Существует два основных метода тестирования уязвимостей: ручной и автоматический. Ручное тестирование требует вмешательства человека, в то время как автоматизированное тестирование использует программные

средства для сканирования Веб-приложений на наличие уязвимостей. В этой статье делается попытка сравнить два метода тестирования уязвимостей и установить преимущества и недостатки каждого подхода.

Защита Веб-приложений от уязвимостей становится все более важной с ростом числа кибератак. Существуют различные методы защиты Веб-приложений, некоторые из которых являются ручными, а другие автоматизированными [14]. В этом обзоре мы обсудим некоторые из существующих методов защиты Веб-приложений от уязвимостей.

Ручные методы:

– тестирование на проникновение: это ручной метод, в котором участвует группа специалистов по безопасности, которые имитируют атаку на Web-приложение для выявления уязвимостей. Затем команда сообщает разработчикам, которые могут исправить уязвимости;
– проверка кода. Этот метод включает ручную проверку исходного кода Web-приложения для выявления уязвимостей. Это трудоемкий процесс, но он может быть эффективным, если все сделано правильно;
– обучение по вопросам безопасности. Этот метод включает в себя обучение разработчиков передовым методам написания безопасного кода. Это можно сделать с помощью тренингов, семинаров или онлайн-курсов.

Автоматизированные методы:

– брандмауэры Веб-приложений (WAF). Это автоматизированный метод, при котором между Web-приложением и пользователем размещается брандмауэр для фильтрации вредоносного трафика. WAF могут выявлять и блокировать распространенные атаки на Web-приложения, такие как внедрение SQL и межсайтовые сценарии [16];

– сканирование уязвимостей. Это автоматизированный метод, который включает использование инструмента для сканирования Web-приложения на наличие уязвимостей. Инструмент может выявлять известные уязвимости и сообщать о них разработчикам;
– анализ кода. Этот метод предполагает использование инструмента для анализа исходного кода на наличие уязвимостей. Инструмент может обнаруживать ошибки кодирования, которые могут привести к уязвимостям, до того, как приложение будет выпущено.

Существуют различные методы защиты Веб-приложений от уязвимостей. Ручные методы могут быть эффективными, но они отнимают много времени и требуют команды специалистов по безопасности. С другой стороны, автоматизированные методы быстрее и эффективнее, но они могут быть не такими точными, как ручные. Сочетание ручных и автоматических методов может быть лучшим подходом к обеспечению безопасности Веб-приложений. Разработка автоматизированного поиска уязвимостей для Веб-приложений может стать перспективным подходом к повышению эффективности автоматизированных методов.

Ручное тестирование уязвимостей – это ручной процесс, в котором группа тестировщиков выполняет исчерпывающую проверку кода, архитектуры и дизайна Web-приложения. Тестировщики используют различные методы, такие как тестирование на проникновение, проверку кода и сканирование сети, для выявления потенциальных уязвимостей. Ручное тестирование – это трудоемкий процесс, который может занять несколько недель или даже месяцев. Тем не менее, это тщательный и подробный процесс, который может выявить уязвимости, которые могут пропустить автоматизированные инструменты [8].

Одним из существенных преимуществ ручного тестирования уязвимостей является то, что это тщательный процесс. Тестировщики могут использовать свой опыт, креативность и интуицию для выявления потенциальных уязвимостей, которые могут пропустить автоматизированные

инструменты. Ручное тестирование также позволяет тестировщикам имитировать реальные атаки и понимать, как злоумышленники могут использовать уязвимости. Кроме того, ручное тестирование может обнаружить сложные уязвимости, требующие глубокого понимания архитектуры приложения и стека технологий.

Ручное тестирование уязвимостей – это трудоемкий процесс, для которого требуется команда опытных тестировщиков. Он также подвержен человеческим ошибкам, и если тестировщики пропустят уязвимость, это может привести к серьезному нарушению безопасности. Кроме того, ручное тестирование не масштабируется, что затрудняет тестирование больших Веб-приложений со значительным количеством страниц.

Автоматическое тестирование уязвимостей включает использование программных инструментов для выявления потенциальных уязвимостей в Web-приложении. Инструменты используют различные методы, такие как сканирование Web-страниц, проверка полей ввода и анализ ответов сервера для выявления уязвимостей. Автоматическое тестирование быстрее и эффективнее, чем ручное, и позволяет просканировать тысячи Web-страниц за считанные минуты.

Одним из существенных преимуществ автоматизированного тестирования уязвимостей является его быстрота и эффективность. Автоматизированные инструменты могут сканировать большие Web-приложения с тысячами страниц за считанные минуты. Кроме того, автоматизированное тестирование является более точным, чем ручное тестирование, поскольку оно не подвержено человеческим ошибкам. Автоматизированное тестирование также является масштабируемым, что позволяет тестировать большие Web-приложения со сложной архитектурой.

Автоматическое тестирование уязвимостей имеет некоторые ограничения. Автоматизированные инструменты могут пропустить некоторые уязвимости, для обнаружения которых требуется вмешательство человека. Кроме того, автоматизированное тестирование не может

имитировать реальные атаки, что затрудняет понимание того, как злоумышленники могут использовать уязвимости. Автоматизированное тестирование также ограничено уязвимостями, которые может обнаружить инструмент, что затрудняет обнаружение сложных уязвимостей, требующих глубокого понимания архитектуры приложения и стека технологий.

Как ручное, так и автоматизированное тестирование уязвимостей имеют свои преимущества и недостатки. Ручное тестирование – это тщательный процесс, который может обнаружить сложные уязвимости, которые могут пропустить автоматизированные инструменты. Однако это занимает много времени и подвержено человеческим ошибкам. Автоматизированное тестирование является быстрым, эффективным и масштабируемым, что делает его идеальным для тестирования крупных Веб-приложений. Однако он ограничен уязвимостями, которые инструмент может обнаружить, и не может имитировать реальные атаки. В заключение следует отметить, что сочетание ручного и автоматизированного тестирования является наилучшим подходом к комплексному тестированию уязвимостей в Веб-приложениях [19], [25].

В таблице 2 наглядно видно все приемущества и недостатки ручного и автоматического тестирования, также прописано для чего каждое подходит и примеры.

Таблица 2 – Сравнение ручного и автоматического тестирования

	Ручное тестирование уязвимостей	Автоматическое тестирование уязвимостей
Определение	Процесс выявления уязвимостей и слабых мест в системе или приложении посредством ручной проверки и тестирования.	Процесс выявления уязвимостей и слабых мест в системе или приложении с помощью инструментов тестирования.
Плюсы	Позволяет применять более персонализированный подход и может обнаруживать сложные уязвимости, которые могут пропустить инструменты.	Может быстро и эффективно обрабатывать большой объем кода, экономя время и ресурсы.

Продолжение таблицы 2

	Ручное тестирование уязвимостей	Автоматическое тестирование уязвимостей
Минусы	Может отнимать много времени и требует квалифицированного персонала для эффективной работы.	Может привести к ложным срабатываниям или пропустить более тонкие уязвимости, требующие анализа человеком.
Подходит для	небольших или специализированных приложений или ситуаций, когда необходим более индивидуальный подход.	крупномасштабных приложений или ситуаций, когда время и ресурсы ограничены.
Примеры	Тестирование на проникновение, проверка кода, моделирование угроз.	Статический анализ, динамический анализ, нечеткое тестирование.

1.4. Описание основных требований к средству автоматизированного поиска уязвимостей для Веб-приложений

Автоматическое тестирование уязвимостей становится все более важным, поскольку Web-приложения становятся все более сложными, а количество потенциальных уязвимостей растет. Тестирование уязвимостей – это процесс выявления уязвимостей в программном обеспечении и принятия мер по их уменьшению или устранению. Он включает в себя выявление слабых мест в коде, которые могут быть использованы злоумышленниками для получения несанкционированного доступа к системе или данным. Инструменты автоматизированного тестирования уязвимостей используются для автоматизации процесса обнаружения уязвимостей и могут использоваться для выявления широкого круга проблем, включая SQL-инъекции, межсайтовые сценарии и другие уязвимости [15].

Было проведено несколько исследований по автоматизированному тестированию уязвимостей для Веб-приложений. Одно из таких исследований было проведено исследователями из Калифорнийского университета в Санта-Барбаре, которые разработали инструмент под названием «Peach Fuzz» для автоматизированного тестирования уязвимостей.

Инструмент был разработан для обнаружения уязвимостей в Веб-приложениях путем создания набора тестовых случаев, которые можно использовать для выявления слабых мест в приложении. Исследователи обнаружили, что Peach Fuzz смог обнаружить несколько уязвимостей в Веб-приложениях, включая SQL-инъекции и межсайтовые сценарии [2].

Другое исследование было проведено исследователями из Университета Мэриленда, которые разработали инструмент под названием «WebSSARI» для автоматизированного тестирования уязвимостей. Инструмент был разработан для обнаружения уязвимостей в Веб-приложениях путем анализа кода и выявления потенциальных уязвимостей. Исследователи обнаружили, что WebSSARI смог обнаружить несколько уязвимостей в Веб-приложениях, включая SQL-инъекции и межсайтовые сценарии [3].

В исследовании, проведенном исследователями из Вашингтонского университета, они разработали инструмент под названием «AppSealer» для автоматизированного тестирования уязвимостей. Инструмент был разработан для обнаружения уязвимостей в мобильных приложениях путем анализа кода и выявления потенциальных уязвимостей. Исследователи обнаружили, что AppSealer смог обнаружить несколько уязвимостей в мобильных приложениях, в том числе небезопасное хранение данных, небезопасную связь и небезопасную авторизацию [6].

Еще одно исследование, проведенное исследователями из Техасского университета в Остине, они разработали инструмент под названием «Sage» для автоматизированного тестирования уязвимостей. Инструмент был разработан для обнаружения уязвимостей в Веб-приложениях путем создания набора тестовых случаев, которые можно использовать для выявления слабых мест в приложении. Исследователи обнаружили, что Sage смогла обнаружить несколько уязвимостей в Веб-приложениях, включая SQL-инъекцию, межсайтовый скриптинг и другие уязвимости [12].

Автоматизированное тестирование уязвимостей становится все более важным, поскольку Web-приложения становятся более сложными, а количество потенциальных уязвимостей растет. Было проведено несколько исследований по автоматизированному тестированию уязвимостей для Веб-приложений, и было разработано множество инструментов для автоматизации процесса обнаружения уязвимостей. Эти инструменты могут помочь разработчикам выявлять и устранять уязвимости в Веб-приложениях, делая их более безопасными и менее уязвимыми для атак.

Инструмент автоматического поиска уязвимостей для Веб-приложений – это программа, предназначенная для выявления и эксплуатации уязвимостей в Веб-приложениях. Инструмент должен уметь быстро и точно сканировать Web-приложения и предоставлять подробные отчеты о любых найденных уязвимостях.

Основными требованиями к такому инструменту будут следующие:

– точное и всестороннее сканирование. Инструмент должен быть способен сканировать все части Web-приложения, включая внешний интерфейс, серверную часть и базу данных. Он должен уметь обнаруживать уязвимости, такие как SQL-инъекция, межсайтовый скриптинг и переполнение буфера;
– простота использования. Инструмент должен быть простым в установке и использовании, с удобным интерфейсом, позволяющим пользователям настраивать параметры сканирования и просматривать результаты;
– скорость. Инструмент должен иметь возможность быстро сканировать Web-приложения с минимальным влиянием на производительность сканируемого приложения;
– интеграция с другими инструментами. Инструмент должен быть способен интегрироваться с другими инструментами и платформами безопасности, такими как системы управления уязвимостями и брандмауэры Веб-приложений;

– отчетность. Инструмент должен предоставлять подробные отчеты о любых обнаруженных уязвимостях, включая серьезность уязвимости, местоположение уязвимости и рекомендации по устранению;
– непрерывное сканирование. Инструмент должен выполнять непрерывное сканирование Веб-приложений либо по регулярному расписанию, либо в режиме реального времени, чтобы гарантировать скорейшее обнаружение любых новых уязвимостей.

В целом, автоматизированный инструмент поиска уязвимостей для Веб-приложений должен предоставлять комплексное и надежное решение для выявления и снижения рисков безопасности в Веб-приложениях.

Выводы по разделу 1

В первом разделе ВКР представлен обзор постановки проблемы, основных понятий и определений, обзора уязвимостей Веб-приложений, инструментов тестирования безопасности Веб-приложений, сравнения ручного и автоматического тестирования уязвимостей, методов и сред для автоматизированного тестирования уязвимостей, и соответствующие исследования по автоматизированному тестированию уязвимостей. Эта информация обеспечивает основу для понимания важности автоматического тестирования уязвимостей в безопасности Веб-приложений. В нем подчеркивается необходимость эффективных и действенных инструментов и методов тестирования на уязвимости для обеспечения безопасности Веб-приложений.

2 Проектирование средства автоматизированного поиска уязвимостей для Веб-приложений

2.1 Описание архитектуры средства

Далее приведем алгоритмы и методы, которые помогут нам для разработки приложения.

Существует несколько алгоритмов и методов обнаружения уязвимостей в Веб-приложениях. Вот некоторые из наиболее используемых:

- статический анализ. Этот метод включает сканирование исходного кода Web-приложения для выявления потенциальных уязвимостей. Это популярный метод из-за его эффективности и надежности. Однако, поскольку он только анализирует код, он может пропустить некоторые уязвимости, возникающие во время выполнения;
- динамический анализ. Этот метод включает отправку входных данных в Web-приложение и анализ выходных данных для выявления потенциальных уязвимостей. Он более всеобъемлющий, чем статический анализ, поскольку тестирует приложение в реальной среде. Однако это может занять много времени и может привести к ложным срабатываниям;
- fuzz-тестирование. Этот метод включает отправку большого количества случайных данных в Web-приложение для выявления потенциальных уязвимостей. Он эффективен при поиске уязвимостей, которые не могут быть обнаружены другими методами, но также может привести к большому количеству ложных срабатываний [10];
- тестирование на проникновение. Этот метод включает попытку использовать уязвимости в Web-приложении для определения их серьезности. Это комплексный метод, который проверяет

приложение с точки зрения хакера. Однако это отнимает много времени и требует наличия опытных тестировщиков.

Выбор оптимального алгоритма и метода разработки автоматизированного инструмента поиска уязвимостей для Веб-приложений зависит от конкретных потребностей и целей проекта (таблица 3). Однако комбинация статического анализа и динамического анализа часто эффективна при выявлении уязвимостей. Кроме того, машинное обучение можно использовать для повышения точности обнаружения уязвимостей [23].

Таблица 3 – Алгоритмы и методы обнаружения уязвимостей

Алгоритм/метод	Описание	Примеры
Статическое тестирование безопасности приложений (SAST)	Анализ исходного кода для выявления уязвимостей	SQL-инъекция, XSS, переполнение буфера
Динамическое тестирование безопасности приложений (DAST)	Тестирование приложения в работающем состоянии на выявление уязвимостей	Инъекционные атаки, нарушенная аутентификация и управление сессиями, небезопасная связь
Пушистое тестирование	Отправка больших объемов случайных данных в приложение для обнаружения уязвимостей	Переполнение буфера, уязвимости строки формата, утечки памяти
Ручное тестирование	Тестировщик вручную тестирует приложение для выявления уязвимостей	Логические ошибки, ошибки бизнес-логики, атаки социальной инженерии
Гибридное тестирование	Объединение нескольких методов тестирования для повышения точности и полноты	SAST, DAST, фазз-тестирование

Для разработки инструментов автоматизированного поиска уязвимостей для Веб-приложений требуется подходящая системная архитектура и компоненты. Вот схема рекомендуемой системной архитектуры и компонентов:

– внешний интерфейс: компонент внешнего интерфейса будет отвечать за предоставление пользователям пользовательского

интерфейса. Он предоставит пользователям интерфейс для ввода информации о Web-приложении. Интерфейс внешнего интерфейса должен быть простым для навигации и понимания, а также должен иметь возможность отображать результаты сканирования уязвимостей в четкой и лаконичной форме;

- бэкэнд-сервер будет отвечать за хранение отсканированных данных Веб-приложений, запуск сканеров уязвимостей и создание отчетов. Бэкенд-сервер также должен иметь возможность управлять рабочей нагрузкой процесса сканирования уязвимостей;
- сканеры уязвимостей будут сканировать Web-приложение на наличие уязвимостей. Сканеры могут быть как сторонними, так и изготовленными по индивидуальному заказу. Рекомендуется использовать несколько сканеров для повышения точности результатов сканирования;
- база данных потребуется для хранения данных Web-приложения, результатов сканирования уязвимостей и другой соответствующей информации;
- отчетность, компонент отчетности будет отвечать за создание отчетов о сканирование уязвимостей. Отчеты должны быть четкими и понятными для пользователей;
- безопасность должна быть ключевым фактором при разработке архитектуры системы для инструментов автоматического поиска уязвимостей. Система должна быть разработана с учетом требований безопасности, чтобы предотвратить несанкционированный доступ к данным Веб-приложений и результатам сканирования уязвимостей;
- интеграция. Архитектура системы должна быть разработана для интеграции с другими инструментами и системами безопасности, такими как SIEM, чтобы обеспечить комплексное решение для обеспечения безопасности.

В целом, системная архитектура и компоненты должны быть разработаны таким образом, чтобы обеспечить надежное и всестороннее автоматизированное средство поиска уязвимостей для Веб-приложений, как это показано на рисунке 2.

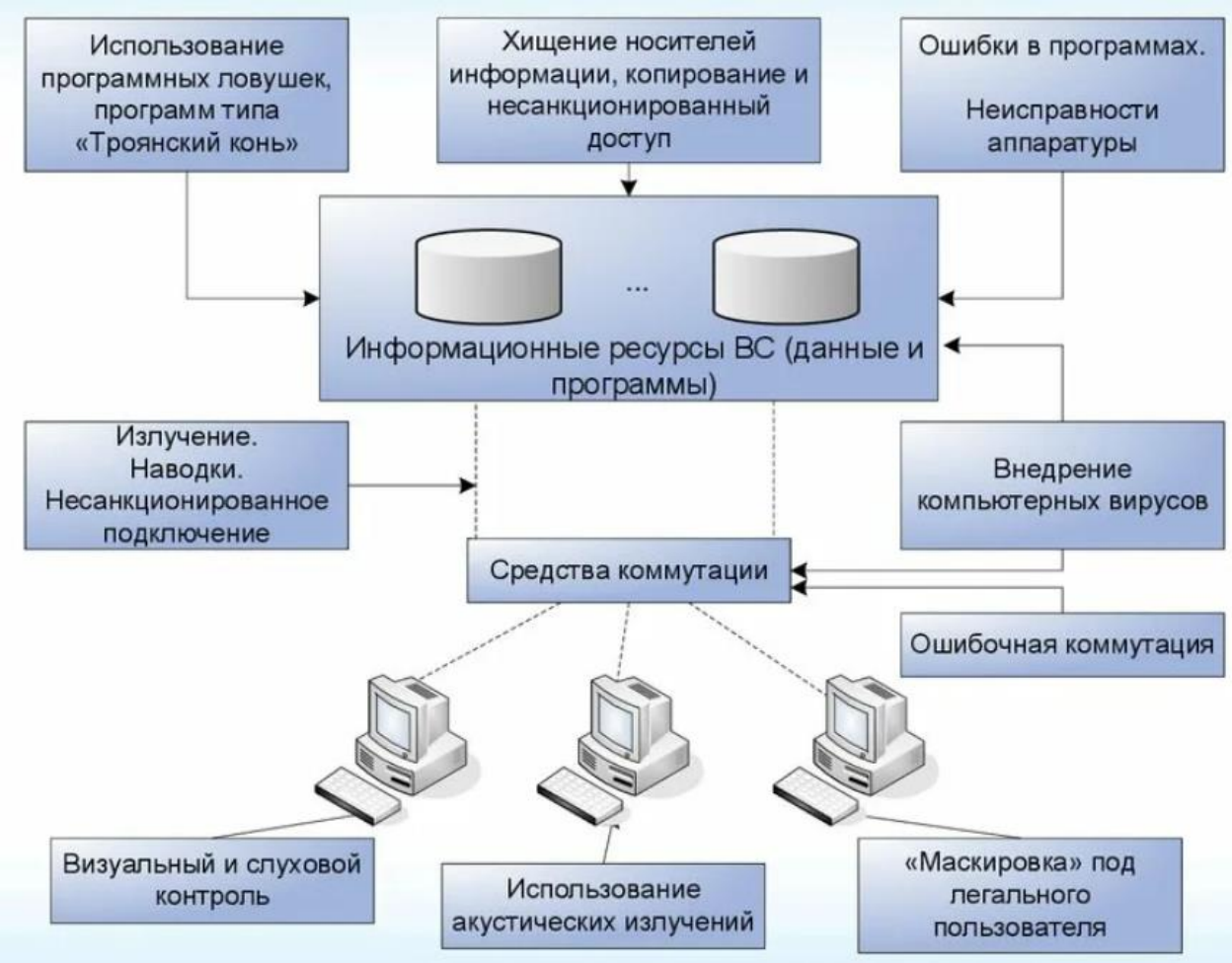

Рисунок 2 – Архитектура системы и компоненты

Для перехвата сетевого трафика обычно используются сетевые карты, переведенные в режим прослушивания. Прослушивание сети Интернет требует подключения компьютера с запущенным перехватчиком к сегменту сети, после чего хакеру становится доступным весь сетевой трафик, отправляемый и получаемый компьютерами в данном сетевом сегменте. Еще проще выполнить перехват трафика радиосетей, использующих беспроводные сетевые посредники, в этом случае не требуется даже искать

место для подключения к кабелю. Или же злоумышленник может подключиться к телефонной линии, связывающей компьютер с сервером Интернета, найдя для этого удобное место. При прохождении авторизации на сайте без шифрования личных данных, злоумышленник при успешном подключении к сети предприятия перехватывает личные данные пользователя.

Сетевая модель OSI (systems interconnection basic reference model) – это модель взаимодействия сетевых протоколов. А протоколы в свою очередь, это стандарты, которые определяют каким образом, будут обмениваться данными различные программы [13].

Программа, проанализировав пакет и найдя в нем пароль пользователя, выводит уведомление об угрозе и добавляет сайт в список опасных сайтов. Вид пакета с открытыми данными в базе данных изображен на рисунке 3 , в приложении изображен на рисунке 4.

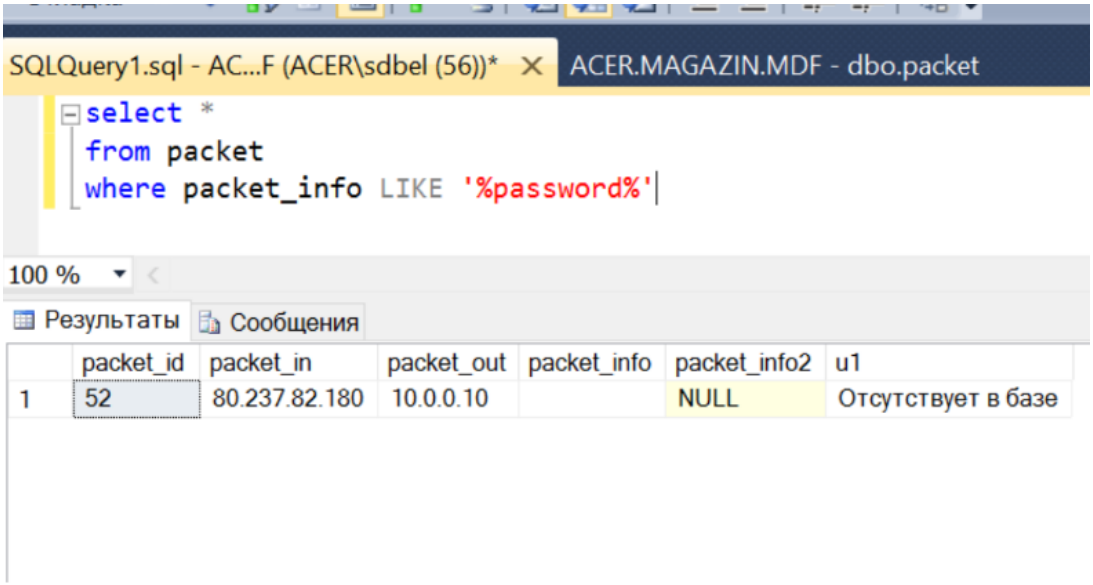

Рисунок 3 – Отображение пакета с данными пользователя в базе данных

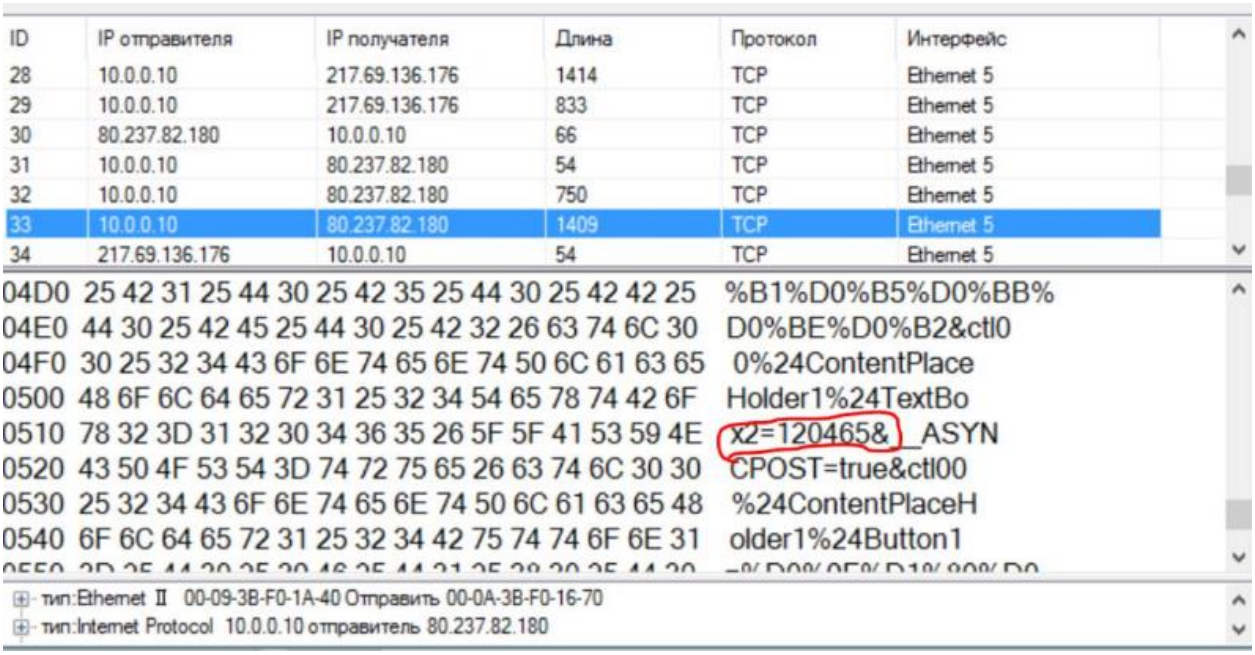

Рисунок 4 – Отображение пакета с данными пользователя в приложении

Просмотр отчета об опасных сайтах изображен в таблице 4.

Таблица 4 – Отчет о найденных сайтах с уязвимостями

IP адрес	Дата обнаружения	Тип уязвимости	Выводы
80.237.82.180	14.06.2023	Передача данных авторизации в открытом виде	Сайт является опасным, возможна кража данных

2.2 Разработка модулей и функциональности средства

Уязвимость – это недостаток или слабость приложения, которое может быть недостатком дизайна или ошибкой реализации, что позволяет злоумышленнику наносить вред заинтересованным сторонам приложения. Заинтересованные стороны включают владельца приложения, пользователей приложений и другие объекты [22].

Для тестирования каждого сканера уязвимостей Веб-приложений в отношении Web-приложения используется определенный метод,

включающий набор процедур, таких как инициализация, выполнение, классификация и анализ.

Использование как безопасных, так и небезопасных версий пользовательского Web-приложения может выявить ложные срабатывания и ложноотрицательные результаты сканеров уязвимостей, которые могут быть связаны с используемыми ими методами. Корреляция между используемыми методами и возникновением ложных положительных или ложных отрицательных результатов может помочь в определении способов улучшения методов сканирования для Веб-приложений.

На основе общего алгоритма мы привели конкретные алгоритмы для обнаружения определенных уязвимостей, таких как SQL Инъекция и XSS.

В первую очередь инициализируем SQL-символы в массиве.

Алгоритм для инъекций SQL на рисунке 5.

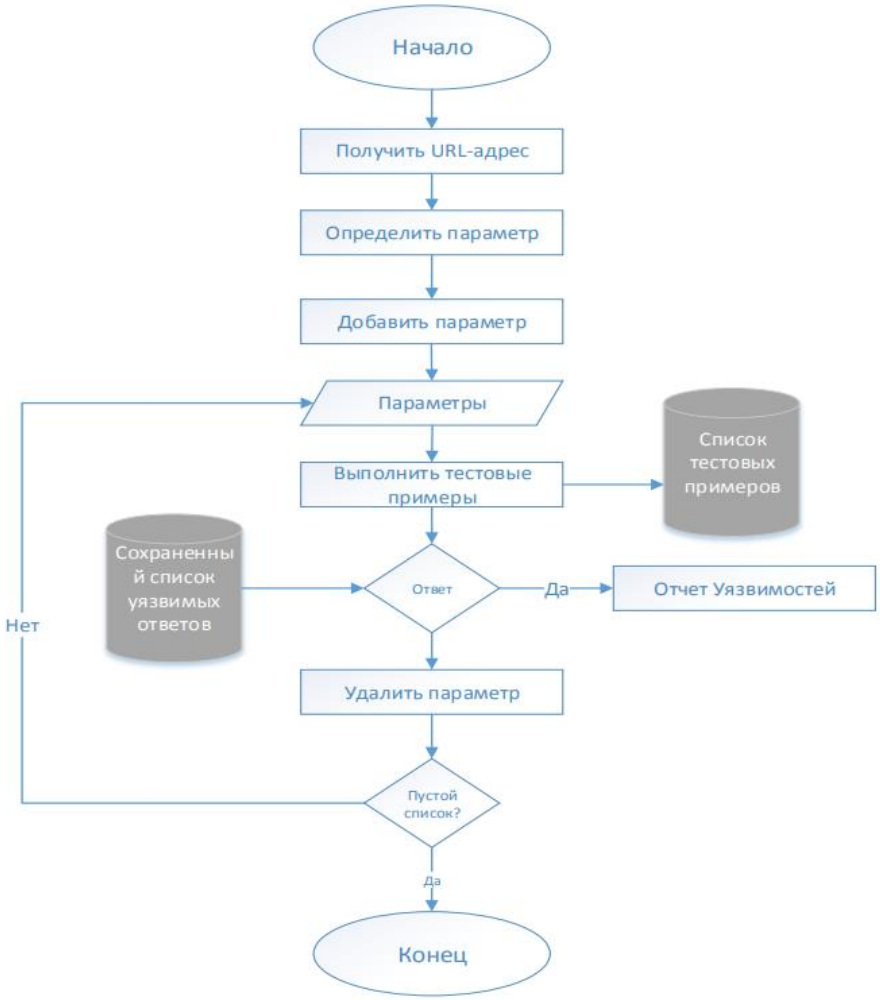

Рисунок 5 – Блок-схема алгоритма инъекций SQL

Создадим два списка для хранения сообщений об ошибках SQL:

– один для хранения сообщений об ошибках конкретной базы данных, таких как сообщения об ошибках SQL и т. д;

– другое для хранения общих сообщений об ошибках базы данных.

Далее инициализируем значения ошибок в картах/списке, указанных выше. После инициализируем метод сканера - сканер принимает сообщение http в качестве ввода от искателя. HTTP-сообщение содержит сведения о каждом запросе или URL-адресе с списком параметров.

Для каждого параметра в сообщении HTTP:

– введем SQL-символы из массива SQL-символов;

– проверим ответ, чтобы проверить соответствует ли сообщениям об ошибках из двух карт или списков;
– если происходит совпадение Flag as SQL-уязвимость;
– else - повторите шаг до тех пор, пока не будет достигнут конец списка параметров.

Рассмотрим на рисунке 6 алгоритм для XSS.

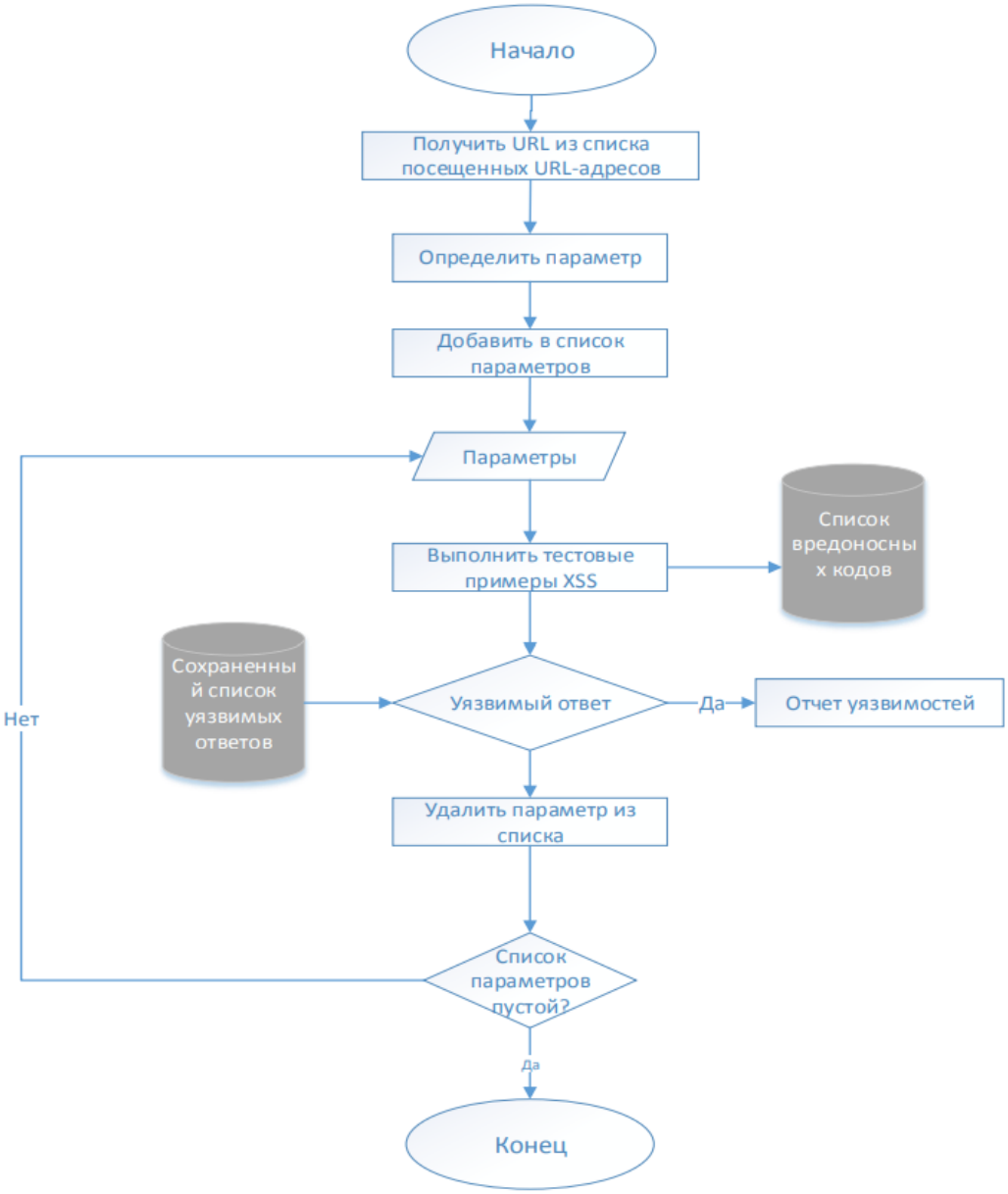

Рисунок 6 – Блок-схема алгоритма XSS

Для каждого URL-адреса в списке посещаемых URL-адресов:

– определяем все параметры;
– вводим параметры в список параметров;
– для каждого параметра в очереди параметров;
– поставим сценарий или тестовый пример XSS в качестве входного параметра и передайте запрос;
– проверим ответ, чтобы определить предоставленный сценарий или тестовый случай, отраженный назад.

Далее сообщаем об этой уязвимости, если в ответе есть скрипт.

2.3 Разработка основных алгоритмов работы средства

Чтобы создать сканер уязвимостей Web-сайта, мы использовали Python 3 (интегрированный с пакетом Anaconda), среда программирования – Pycharm; Web-сервер Apache, SQL-сервер MySQL (все компоненты включены в установочный пакет XAMPP).

Интерактивная модель представлена на рисунке 7.

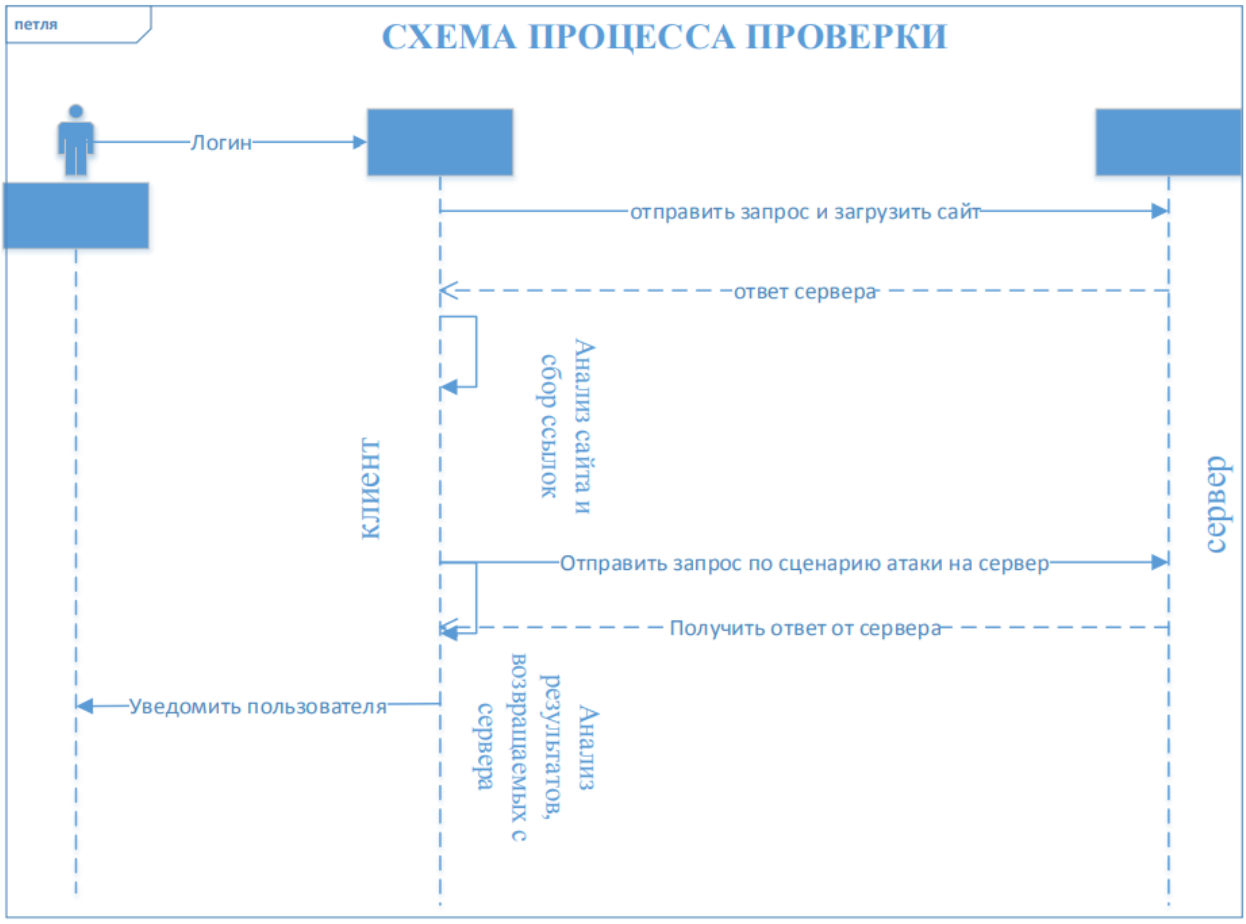

Рисунок 7 – Схема процесса проверки

Описание схемы процесса проверки:

– пользователь: используйте клиентскую программу (написанную на Python), чтобы проверить наличие уязвимостей на Web-сайте, связав сайт с тестовым сайтом;

– клиентская программа подключается к серверу, загружает соответствующую Web-страницу, анализирует, чтобы получить все связанные ссылки; Затем выполните соответствующую атаку на Web-сайте и получите результат. Анализирует, содержит ли Web-сайт дефект и уведомляет пользователя;

– серверная программа: получать соединение с клиентом, отвечать соответствующим требованиям. С соответствующими настройками

для тестирования на сервере, который содержит любой Web-сайт, который может, содержать недостатки или нет.

Согласно анализу интерактивной модели, мы моделируем производительность программы на основе этих взаимодействий.

На рисунке 8 представлена операционная модель программы.

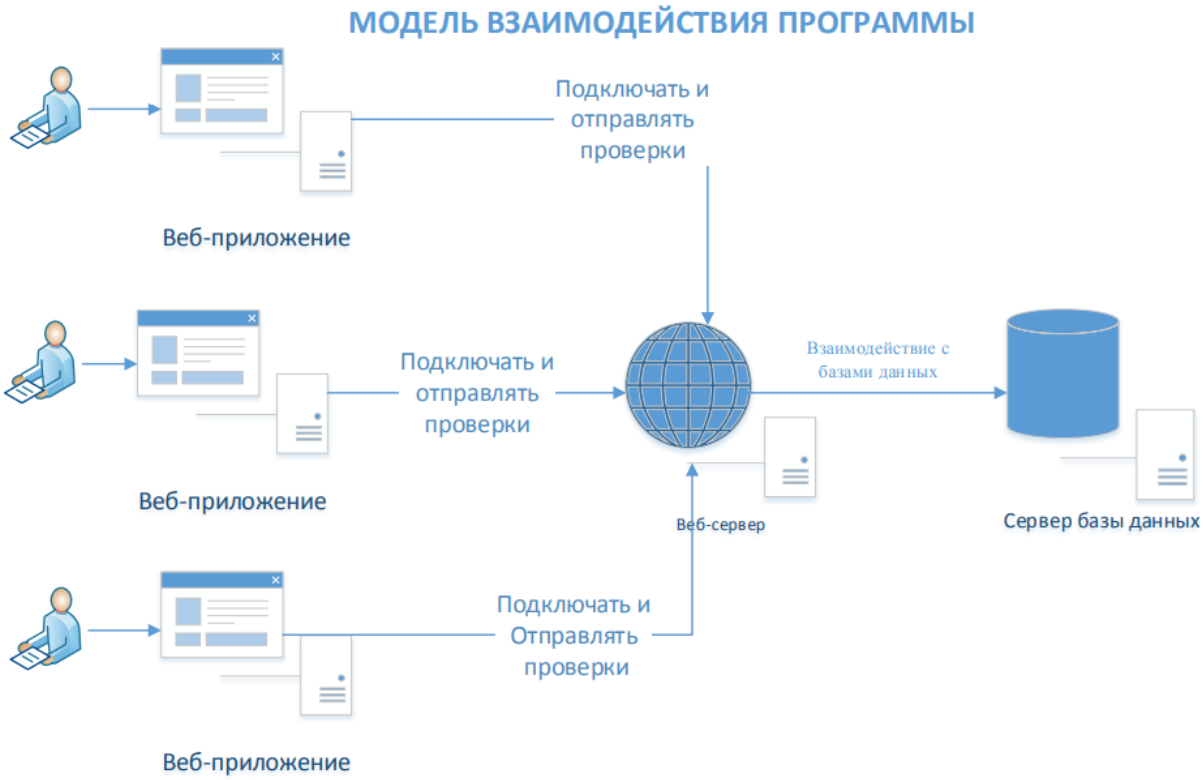

Рисунок 8 – Модель взаимодействия программы

Механизм сканирования устанавливает соединение с сервером, используя ссылку, предоставленную пользователем через параметр. Затем он анализирует входные параметры, чтобы идентифицировать соответствующий механизм сканирования или злоумышленника. При отсутствии каких-либо параметров движок выполняет сканирование всех компонентов [20].

На сервере Веб-приложений могут быть уязвимости в сети, базе данных или связанные уязвимости, которые делают его восприимчивым к проблемам проверки входных данных. Если сканер способен обнаружить

такие уязвимости и выдать предупреждение, программисту необходимо обязательно выполнить проверку безопасности на Web-сайте [5].

Архитектура программы представлена на рисунке 9.

Программа встроена в разные модули, выполняя конкретные задачи:

– нижняя часть - это модули языка программирования Python с соответствующими библиотеками;
– модуль Core содержит более мелкие модули, которые выполняют разные задачи: модуль собирает ссылки на Web-сайт, преобразует их в разные форматы в соответствии с обычной структурой Web-сайта, такими как: форма, текст, ссылки, текстовое поле, модули также предоставляют инструменты для ведения журнала, параметры анализа и некоторые другие вспомогательные функции;
– модуль содержит атаки для выполнения проверок на сайте, включая XSS, CSRF, Breach, Clickjack.

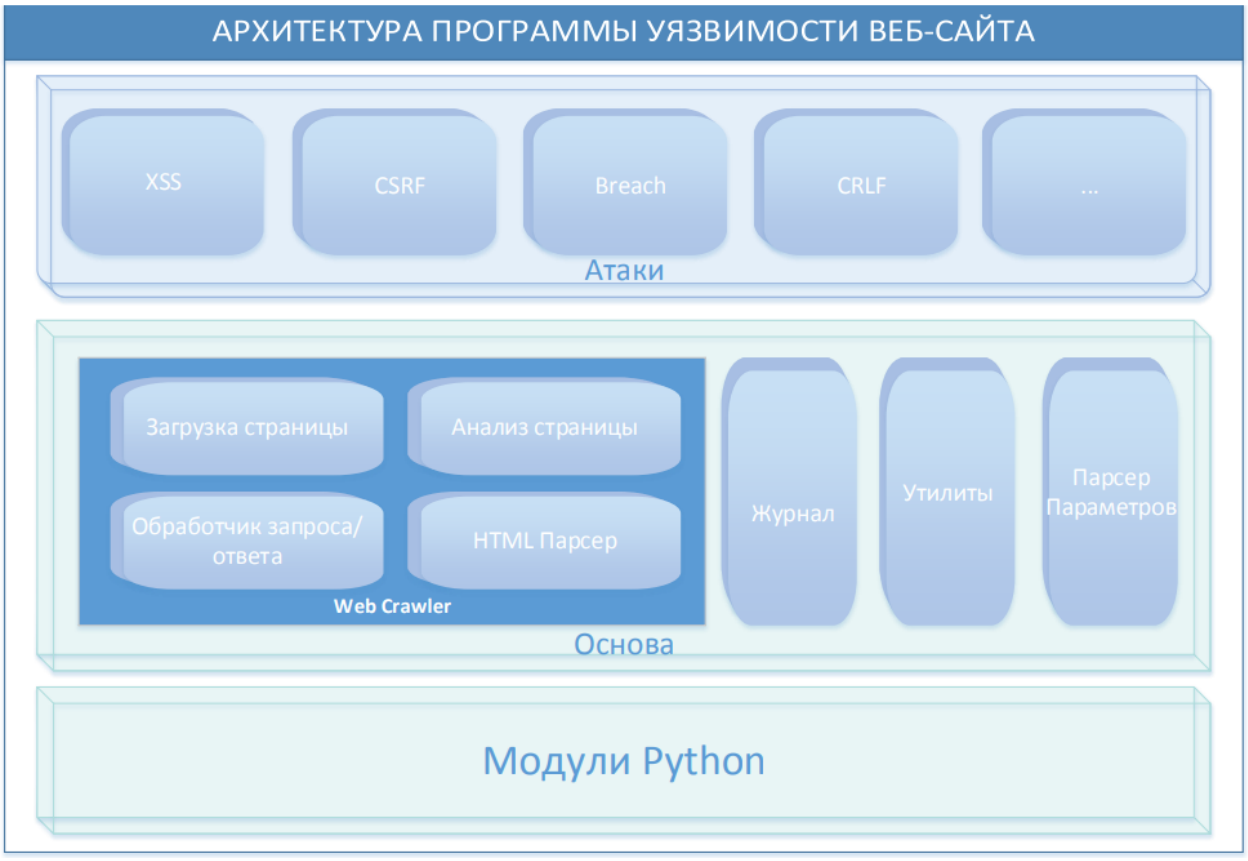

Рисунок 9 – Архитектура программы уязвимости Web-сайта

2.4 Описание методов тестирования и проверки работоспособности средства

Разработка инструмента автоматизированного поиска уязвимостей для Веб-приложений требует тщательного тестирования для обеспечения его работоспособности.

Несколько методов тестирования и проверки работоспособности инструмента:

– функциональное тестирование включает в себя тестирование функций инструмента, чтобы убедиться, что он соответствует требованиям, изложенным в диссертации. Инструмент следует протестировать, чтобы убедиться, что он может автоматизировать поиск уязвимостей в Веб-приложениях [1];
– тестирование производительности. Сюда входит тестирование производительности инструмента для проверки его скорости и эффективности. Инструмент следует протестировать, чтобы убедиться, что он может быстро и эффективно сканировать Web-приложения [24];
– тестирование совместимости. Сюда входит проверка совместимости инструмента с различными Web-приложениями и операционными системами. Инструмент следует протестировать, чтобы убедиться, что он может работать с различными типами Веб-приложений и операционных систем [17];
– тестирование безопасности. Это включает в себя тестирование безопасности инструмента, чтобы убедиться, что он не содержит уязвимостей, которые могут быть использованы злоумышленниками. Инструмент следует протестировать, чтобы убедиться, что он не создает новых уязвимостей в сканируемых Веб-приложениях [9];
– юзабилити-тестирование. Сюда входит проверка удобства использования инструмента, чтобы убедиться, что он прост в

использовании и не требует обширных технических знаний. Инструмент следует протестировать, чтобы убедиться, что он удобен для пользователя и может использоваться нетехническими пользователями.

Выводы по разделу 2

Разработка автоматизированного инструмента поиска уязвимостей для Веб-приложений имеет решающее значение для обеспечения безопасности online-систем. Архитектура инструмента обеспечивает эффективное и действенное сканирование Веб-приложений на наличие уязвимостей. Разработка модулей и функциональных возможностей инструмента приводит к созданию комплексного инструмента, который охватывает широкий спектр потенциальных уязвимостей. Базовые алгоритмы инструмента обеспечивают прочную основу для обнаружения уязвимостей. Наконец, тестирование и проверка функциональности инструмента гарантируют, что он работает должным образом и дает надежные результаты. В целом разработка автоматизированного инструмента поиска уязвимостей является важным шагом на пути повышения безопасности Веб-приложений.

3 Реализация и тестирование средства автоматизированного поиска уязвимостей для Веб-приложений

3.1 Выбор языка программирования и фреймворка для реализации средства

Язык программирования C# предлагает несколько преимуществ благодаря своей относительно недавней разработке:

- был разработан специально для использования с усовершенствованной платформой Microsoft .NET Framework, которая предоставляет комплексную платформу для разработки, развертывания и запуска распределенных приложений;
- основан на широко признанной методологии объектно-ориентированного проектирования, которая особенно подходит для разработки корпоративных информационных систем;
- воспользовавшись опытом предыдущих языков программирования, включил успешные подходы и исключил те, которые были сочтены неудачными.

Microsoft руководила разработкой C#, который впервые был представлен в виде альфа-версии в середине 2000 года, а Андерс Хейлсберг, известный специалист в этой области, возглавлял группу разработчиков [**Ошибка! Источник ссылки не найден.**].

Версия .NET, используемая при разработке – 3.5. В этой версии в язык программирования C# добавлены следующие возможности:

- обеспечивает поддержку строго типизированных запросов для эффективного взаимодействия с различными структурами данных;
- анонимные типы можно использовать для представления структуры типа, а не только его поведения;

- методы расширения можно использовать для расширения функциональности существующего типа, устраняя необходимость создания подклассов;
- возможность использования лямбда-операций, упрощающих работу с типами делегатов;
- доступен новый синтаксис инициализации объекта, упрощающий установку значений свойств во время создания объекта.

Причинами выбора этого языка являются:

- наличие средств, позволяющих относительно просто и быстро создавать надежные и производительные сетевые службы;
- наличие большого опыта разработки на этом языке;
- возможность интеграции с другими системами организации.

В качестве среды для разработки программного средства для обнаружения и устранения уязвимостей сетевых ресурсов была использована программа Microsoft Visual Studio 2015. Причины выбора:

- поддержка языка программирования C#;
- возможность бесплатного использования.

3.2 Система управления базой данных

Для хранения данных о Web-сайтах, результатах проверки, списках IP-адресов и других необходимых данных для выявления и устранения сетевых уязвимостей в нашем программном обеспечении реализована встроенная реляционная СУБД SQL SERVER.

Этот тип СУБД отличается от более крупных систем тем, что не использует структуру клиент-сервер. Вместо этого SQL SERVER работает как библиотека, интегрированная в клиентское приложение, что упрощает программу и повышает производительность.

Еще одним из преимуществ этой СУБД является то, что все данные хранятся в одном файле, расположенном на том устройстве, где установлено

клиентское приложение. В результате файловая система контролирует все операции чтения и записи в базу данных.

Существуют еще несколько отличительных особенностей:

- наличие в каждой таблице виртуального столбца rowid, определяющего первичный ключ, даже если таковой не создан;
- возможность построения базы данных в оперативной памяти;
- переносимость и кроссплатформенность;
- тип данных не определяет тип хранимого значения и служит только для сравнения типов.

SQL SERVER не имеет как таковых традиционных типов, вместо этого используются пять классов данных:

- null – определяет отсутствие значения;
- integer – определяет целочисленное значение, где размер поля соответствует размеру числа;
- real – определяет значение с плавающей точкой;
- text – определяет текстовую строку, хранимую в кодировке, указанной для базы данных;
- blob – определяет данные в исходном состоянии, включая изображения или другие файлы.

Факторы, лежащие в основе выбора именно этой СУБД:

- находится в свободном доступе для распространения;
- позволяет быстро развертывать системы обновлений благодаря своей бессерверной архитектуре;
- высокая производительность;
- имеет низкий уровень сложности;
- позволяет разрабатывать клиентское приложение на C#.

3.3 Тестирование средства на тестовых Веб-приложениях с известными уязвимостями

Обнаружение сайтов, не шифрующих данные авторизации пользователя.

Запускаем программу, открывается главное окно (рисунок 10).

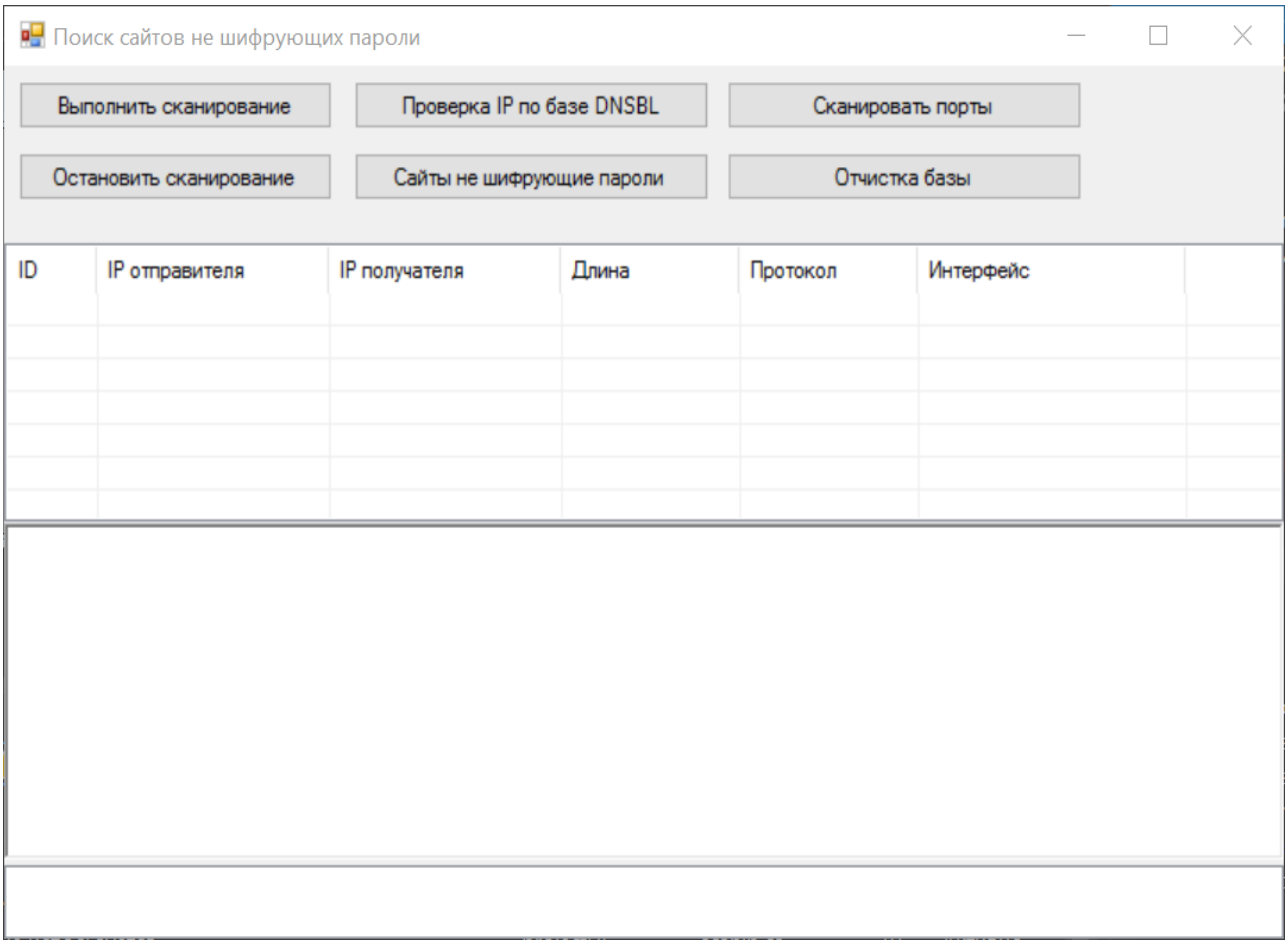

Рисунок 10 – форма «Главное окно»

Чтобы начать сканирование, в главном меню нажимаем кнопку «Выполнить сканирование». Появится окно, позволяющее выбрать интерфейс, который нужно сканировать. Нажимаем кнопку «Начать».

Далее в окне сканирования (рисунок 11) нажимаем на кнопку запуска поиска сайтов. Появится новое окно, в котором выбираем интерфейс. После того, как сделали свой выбор, нажимаем кнопку «Начать».

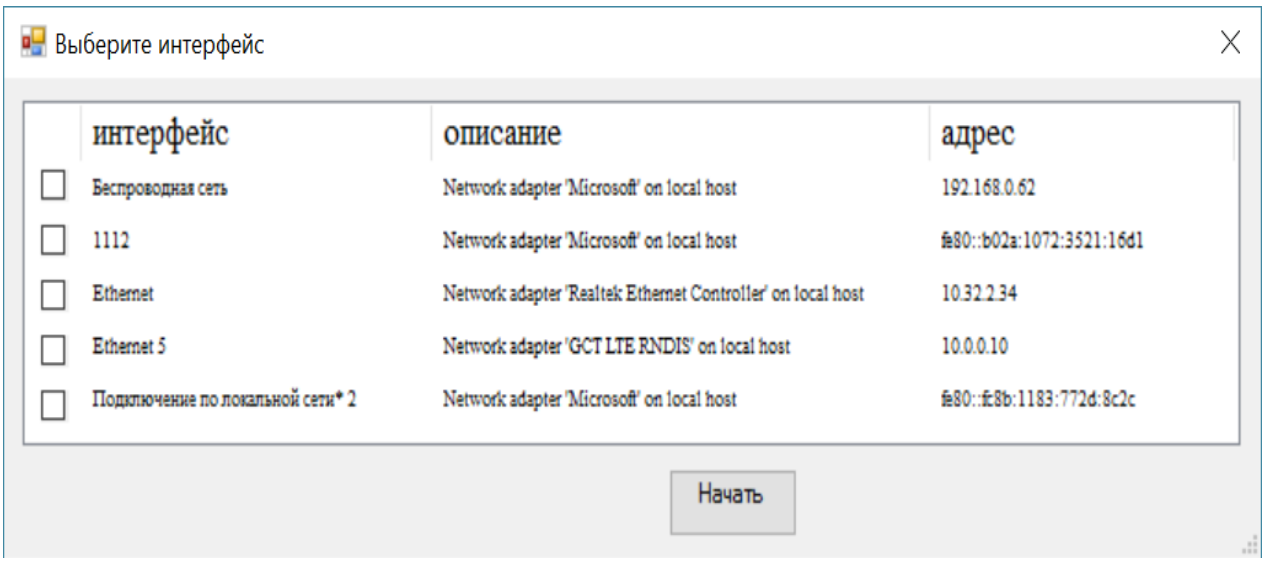

Рисунок 11 – форма «Выберите интерфейс»

В окне поиска сайтов начинают отображаться пакеты, которые отправляются и посылаются через выбранный ранее интерфейс, как это показано на рисунке 12.

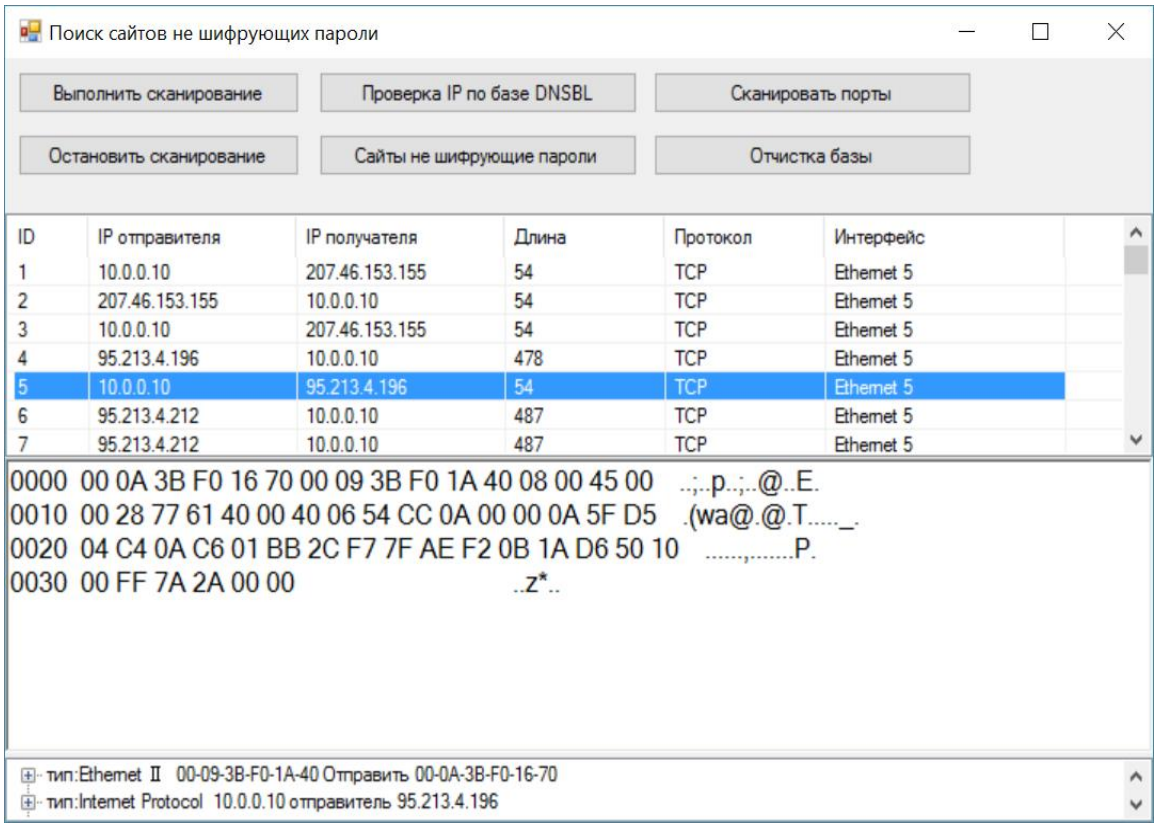

Рисунок 12 – форма «Поиск сайтов, не шифрующих пароли»

Чтобы начать процесс, мы запускаем браузер и вводим учетные данные для входа на выбранный сайт. При обнаружении каких – либо уязвимостей в системе безопасности в строке поиска сайта появится уведомление «Найдены угрозы». При нажатии на вкладку «Посмотреть отчет» откроется новое окно, в котором будут показаны URL-адрес и домен сайта, на котором была обнаружена уязвимость (рисунок 13).

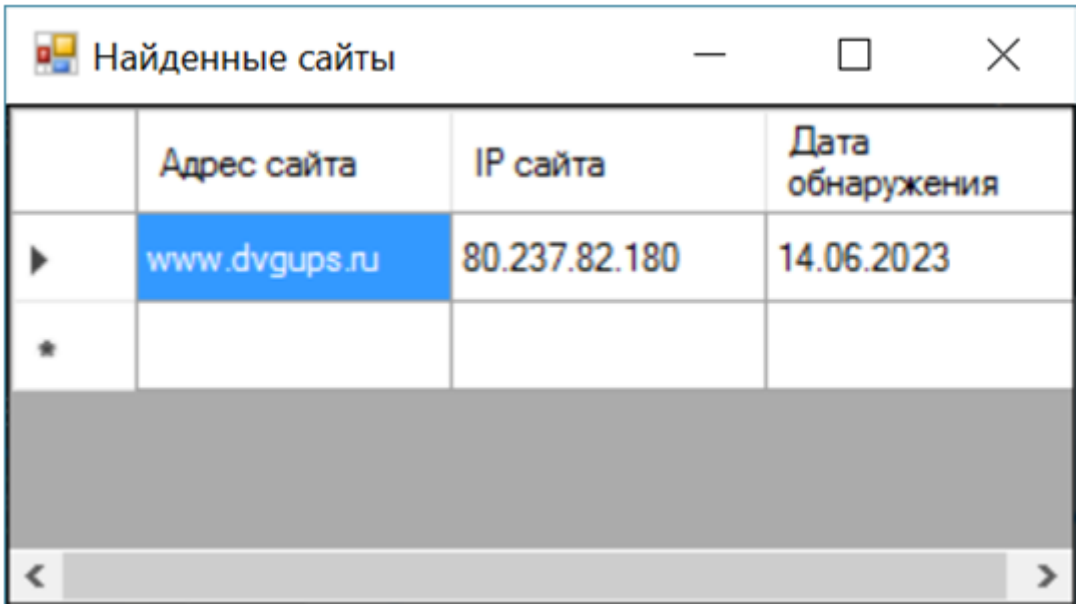

Рисунок 13 – форма «Найденные сайты»

Чтобы начать процесс проверки IP адресов на наличие их в черном списке DNSBL, открываем главное окно приложения и выбираем «Проверка IP по базе DNSBL». Появится новое окно, где должны ввести адрес сайта или доменное имя, которое требует проверки (рисунок 14).

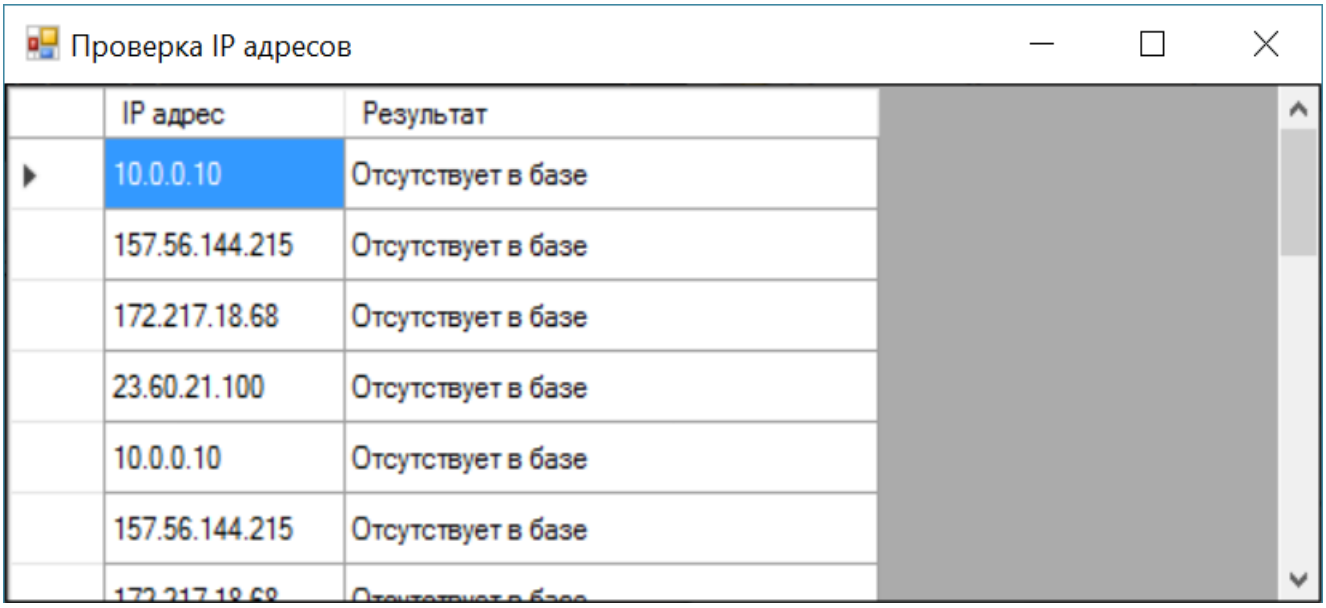

Рисунок 14 – форма «Проверка сайтов на уязвимости»

Запускается новая форма для отображения списка IP-адресов, которые обменивались пакетами. Во второй колонке показаны результаты теста. В случаях, когда IP-адрес отсутствует в базе данных, в столбце отображается сообщение «Отсутствует в базе».

Для сканирования портов вводим желаемый адрес интерфейса и диапазон портов (рисунок 15). Система сгенерирует список открытых и закрытых портов.

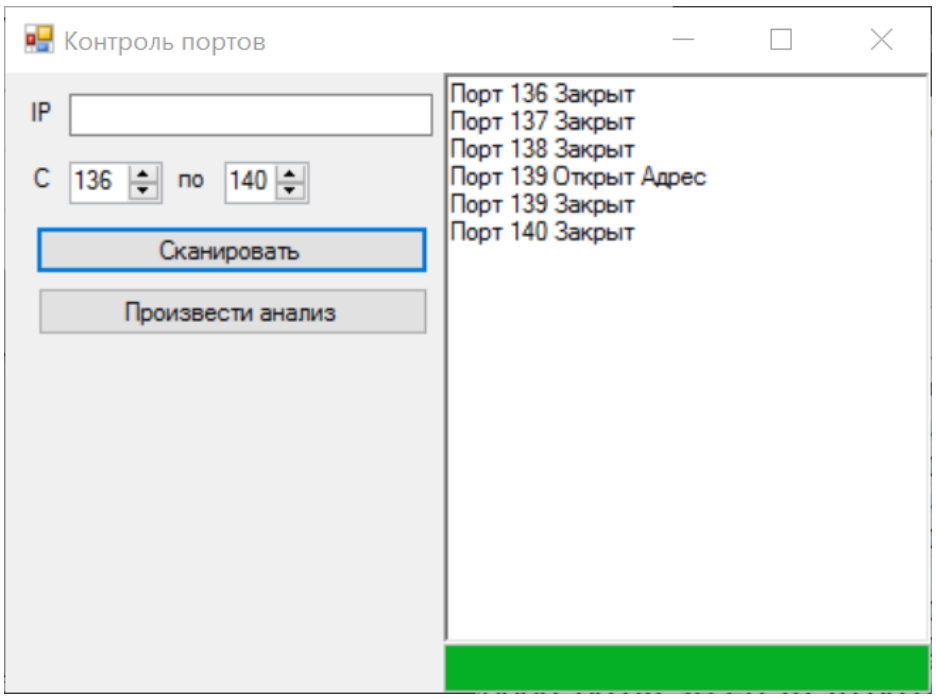

Рисунок 15 – форма «Контроль портов»

После нажатия кнопки анализа становится доступной форма «Работа с портами» (рисунок 16), в которой отображается полный список доступных открытых портов. В этой форме у нас есть возможность запретить входящий и исходящий трафик для интересующего порта, а также разрешить его.

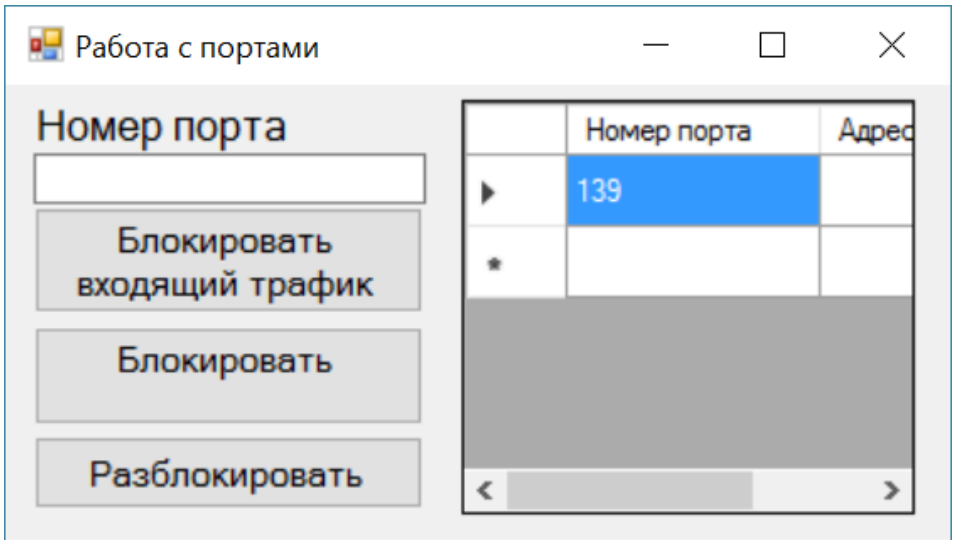

Рисунок 16 – форма «Работа с портами»

Выводы по разделу 3

В разделе 3 обсуждается процесс разработки автоматизированного поиска уязвимостей. Первая часть этого раздела посвящена выбору программного обеспечения, которое необходимо протестировать на наличие уязвимостей. Критерии выбора программного обеспечения основаны на различных факторах, таких как популярность программного обеспечения, важность системы, в которой используется программное обеспечение, и потенциальное влияние уязвимости на безопасность системы. Вторая часть раздела посвящена важности руководства пользователя программного обеспечения в процессе автоматизированного поиска уязвимостей. Руководство пользователя является важным источником информации о программном обеспечении, которое может помочь в выявлении потенциальных уязвимостей.

Заключение

Бакалаврская работа посвящена разработке средств автоматизированного поиска уязвимостей для Веб-приложений.

В первом разделе ВКР представлен обзор постановки проблемы, основных понятий и определений, обзора уязвимостей Веб-приложений, инструментов тестирования безопасности Веб-приложений, сравнения ручного и автоматического тестирования уязвимостей, методов и сред для автоматизированного тестирования уязвимостей, и соответствующие исследования по автоматизированному тестированию уязвимостей.

Системы по сбору и предварительной обработке данных обеспечили прочную основу для обнаружения потенциальных уязвимостей, а используемые алгоритмы и методы оказались эффективными для выявления и устранения этих уязвимостей.

Архитектура и компоненты системы были хорошо спроектированы с акцентом на эффективность и точность. В частности, определение широковещательных IP-адресов и предотвращение внешнего сканирования.

Обсуждается процесс разработки автоматизированного поиска уязвимостей. Сделали выбор программного обеспечения, которое необходимо протестировать на наличие уязвимостей. Критерии выбора программного обеспечения основаны на различных факторах, таких как популярность программного обеспечения, важность системы, в которой используется программное обеспечение, и потенциальное влияние уязвимости на безопасность системы. Посвятили важности руководства пользователя программного обеспечения в процессе автоматизированного поиска уязвимостей. Руководство пользователя является важным источником информации о программном обеспечении, которое может помочь в выявлении потенциальных уязвимостей.

Задачи, определённые для достижения цели работы, выполнены в полном объёме.

Список используемой литературы

1. Актуально ли на сегодня моделирование в IDEF0? // Время успешных проектов [Электронный ресурс]. URL: http://projectimo.ru/biznes-processy/idef0.html (дата обращения: 21.04.2023).
2. Аллен, М. E-Learning: Как сделать электронное обучение понятным, качественным и доступным / М. Аллен - Альпина Паблишер, 2016 - 200 с.
3. Буренин, С. Н. Web-программирование и базы данных [Электронный ресурс] : учеб. практикум / С. Н. Буренин. - Москва: Моск. гуманит. ун-т, 2014. – 120 с. - ISBN 978-5-906768-17-9.
4. Бьюли, А. Изучаем SQL. Учебное пособие [Текст] / А. Бьюли – Символ-Плюс, 2007. – 312 с.: ил. – 2000 экз. – ISBN 978-5-93286-051-9.
5. Вичугова, А.А. Методы и средства концептуального проектирования информационных систем: сравнительный анализ структурного и объектно-ориентированного подходов систем [Электронный ресурс]: учеб. пособие / А.А. Вичугова ; Томский политехнический университет. - Томск: Эль Учебное пособие, 2014. – 56 с. - ISBN 978-5-4162-0323-1
6. Данные о предприятии // РаботкиАК [Электронный ресурс]. URL: https://rak-nn.ru/osnovnye-svedenija/ (дата обращения: 21.04.2023).
7. Золотов, С.Ю. Проектирование информационных систем [Электронный ресурс]: учеб. пособие / С. Ю. Золотов ; Томский гос. ун-т систем управления и радиоэлектроники. - Томск: Эль Учебное пособие Контент, 2013. - 86 с. - ISBN 978-5-4332-0083-8.
8. Инюттткина. О. Г. Проектирование информационных систем: учебное пособие / О. Г. Инюттткина. - Екатеринбург - Форт-Диалог Исеть, 2014 – 240 с.
9. Исаев, Г. Н. Проектирование информационных систем: учебное пособие / Г. Н. Исаев - Омега-Л, 2015 - 424 с.

10. Калинкина, Т.И. Телекоммуникационные и вычислительные сети. Архитектура, стандарты и технологии. Учебное пособие [Текст] / Т.И. Калинкина. – BHV-СПБ, 2010. – 288 с.: ил. – 1000 экз. – ISBN: 978-5-9775-0573-4.

11. Тихомирова, Е. В. Живое обучение. Что такое e-learning и как заставить его работать / Е. В. Тихомирова - Альпина Паблишер, 2017 - 238 с.

12. Хетагуров, Я. А. Проектирование автоматизированных информационных систем обработки информации и управления (АСОИУ): учебник / Я.А. Хетагуров - М.: Бином. Лаборатория знаний, 2015 - 240 с.

13. HR-систем управления персоналом, сотрудниками и кадрами// Creator [Электронный ресурс]. URL: https://creator-arseny.site/programmnoe-obespechenie/sistemy-upravleniya-personalom (дата обращения: 21.04.2023).

14. Моделирование бизнес-процессов // Блог Enterchain [Электронный ресурс]. URL:https://www.enterchain.ru/experience/mbp/modelirovanie-biznes-protsessov-tseli-metody-i-rezultaty/ (дата обращения: 21.04.2023).

15. Олифер, В. А. Компьютерные сети. Принципы, технологии, протоколы. Учебное пособие для вузов [Текст] / В.А. Олифер, Н.А. Олифер. – Питер, 2011. – 944 с.: ил. – 3000 экз. – ISBN: 978-5-459-00920-0.

16. Проектирование информационных систем // [Электронный ресурс]. URL: https://moodle.kstu.ru/course/view.php?id=4638 (дата обращения: 21.04.2023).

17. Структура и органы управления образовательной организации // РаботкиАК [Электронный ресурс]. URL: https://rak-nn.ru/struktura-i-organy-upravlenija-obrazovatelnoj-organizacii/ (дата обращения: 21.04.2023).

18. Суэринг, С. PHP и MySQL. Библия программиста. Учебное пособие [Текст] / С. Суэринг, Т.Конверс, Д.Парк. – Диалектика, 2010. – 912 с.: ил. – 1000 экз. – ISBN: 978-5-8459-1640-2.

19. Шаньгин, В. Ф. Информационная безопасность компьютерных систем и сетей. Учебное пособие [Текст] / В. Ф. Шаньгин – М.: «ФОРУМ»:

ИНФРА-М, 2008. – 416 с.: ил. 3000 экз. – ISBN: 978-5-8199-0331-5.

20. Brok J. Business Process Management Cases: Digital Innovation and Business Transformation in Practice - Springer, 2018.

21. Jacobson, A. Automated Learning Support System to Provide Sustainable Cooperation between Adult Education Institutions and Enterprises / A. Jacobson, S. Cakila - Procedia Computer Science - 2015 - P. 90 - 98.

22. Huoing, M. Truong Integrating learning styles and adaptive e-learning system: Current developments, opportunities and problems - Computers in Human Behavior - 2016 - P. 1175 – 1192

23. Marcella La Rosa, Pnina Soffer Business Process Management Workshops - Springer, 2012.

24. Philipp J. Pratt, Mari Z. Last A Guide to SQL - Course Technology, 2014.

25. PostgreSQL vs MySQL // habr [Электронный ресурс]. URL: https://habr.com/company/mailru/blog/248845/ (дата обращения: 21.04.2023).