**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**Федеральное государственное автономное**

**образовательное учреждение высшего образования**

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**ОТЧЁТ ПО УЧЕБНОЙ ПРАКТИКЕ**

**Тема: «Модель управления безопасностью веб-приложения»**

**Выполнили:**

**ст. гр. 241-352, 241-351**

**Червяков И.А, Кривенцев Д.Е.**

**Проверили:**

**Кесель С. А.**

**Москва – 2025**

# **Цель:**

Разработать и описать систему управления безопасностью типового веб-приложения.

# **Задачи работы**

1. Определить архитектуру: frontend, backend, БД, ОС.

2. Выделить уязвимые компоненты (входы, хранение, сессии).

3. Предложить модель управления безопасностью (роли, политики, контроль изменений).

4. Сформировать план реагирования на инциденты.

5. Подготовить сопроводительную документацию.

# **Классификация целевых объектов и их особенности в контексте ИБ**

Архитектура веб-приложения включает следующие компоненты:

* Frontend: React.js (или Angular) с защитой от XSS (Content Security Policy, экранирование данных).
* Backend: Node.js (или Django/Spring Boot) с использованием API Gateway для контроля запросов.
* База данных: PostgreSQL с шифрованием данных на уровне таблиц (TDE).
* Операционная система: Linux (Ubuntu Server) с регулярными обновлениями и минимальным набором открытых портов.

# **Уязвимые компоненты**

| **Компонент** | **Уязвимости** | **Меры защиты** |
| --- | --- | --- |
| **Формы входа** | Brute Force, Credential Stuffing | CAPTCHA, MFA, блокировка после 5 неудачных попыток. |
| **API-эндпоинты** | SQL-инъекции, CSRF | Prepared Statements, CSRF-токены, валидация входных данных (JSON Schema). |
| **Сессии** | Session Hijacking | JWT с коротким TTL, регенерация токенов после критичных действий. |
| **Хранение паролей** | Утечки из-за слабого хеширования | bcrypt (work factor=12), запрет хранения в plaintext. |
| **Конфигурация ОС** | Необновленные пакеты | Автоматические обновления (unattended-upgrades), еженедельный аудит (Lynis). |

# **3. Модель управления безопасностью**

Роли и права доступа

* Администратор: Полный доступ к системе, включая управление пользователями, настройку политик безопасности и мониторинг инцидентов.
* Разработчик: Доступ к коду и настройкам, но ограниченный доступ к производственным данным.
* Пользователь: Стандартные права для работы с приложением, без доступа к администрированию.
* Аудитор: Доступ к логам и отчетам для проверки соответствия политикам безопасности.

Политики безопасности

* Аутентификация: Использование многофакторной аутентификации (MFA) для администраторов и критичных операций.
* Авторизация: Применение принципа наименьших привилегий (PoLP) для всех ролей.
* Шифрование данных: Обязательное шифрование конфиденциальных данных (пароли, персональные данные) как при хранении, так и при передаче (TLS 1.2/1.3).
* Управление сессиями: Автоматическое завершение сеансов после периода неактивности (например, 30 минут), использование токенов с ограниченным сроком действия.

Контроль изменений

* Версионирование кода: Использование Git для отслеживания изменений и обязательный код-ревью перед внедрением.
* Тестирование безопасности: Регулярное проведение статического и динамического анализа кода (SAST/DAST), а также пентестинга.
* Журналирование: Логирование всех критичных операций (вход, изменение данных, доступ к админ-панели) с хранением логов не менее 6 месяцев.

# **Сформировать план реагирования на инциденты**

1. Обнаружение: Алерты через SIEM (например, Splunk).

2. Оценка: Классификация угрозы (низкая/критичная).

3. Действия:

- Блокировка атакующего IP.

- Откат изменений из резервной копии.

4. Анализ: Отчет с рекомендациями по предотвращению.

# **Подготовить сопроводительную документацию**

1. Техническое задание

- Цель: Разработать модель управления безопасностью веб-приложения.

- Стек технологий:

- Frontend: React.js (CSP, экранирование данных).

- Backend: Node.js (API Gateway, валидация запросов).

- БД: PostgreSQL (TDE-шифрование).

- ОС: Ubuntu Server (Lynis для аудита).

2. Руководство по внедрению

- Настройка безопасности:

- Установка CAPTCHA и MFA для форм входа.

- Конфигурация JWT с TTL=30 мин.

- Настройка автоматических обновлений (unattended-upgrades).

- Контроль доступа:

- Ролевая модель (администратор, разработчик, пользователь, аудитор).

- Применение принципа наименьших привилегий (PoLP).

3. Политика безопасности

- Аутентификация: Обязательное использование MFA для админов.

- Шифрование: TLS 1.3 для передачи данных, bcrypt для паролей.

- Журналирование: Хранение логов 6+ месяцев (SIEM-мониторинг).

4. План реагирования на инциденты

1. Обнаружение: Алерты через SIEM (например, Splunk).

2. Оценка: Классификация угрозы (низкая/критичная).

3. Действия:

- Блокировка атакующего IP.

- Откат изменений из резервной копии.

4. Анализ: Отчет с рекомендациями по предотвращению.

#### 5. Инструкция для разработчиков

- Код-ревью: Обязателен перед мерджем в main.

- Тестирование:

- SAST (SonarQube).

- Пентест (OWASP ZAP).

# **Вывод**

В ходе работы была разработана комплексная модель управления безопасностью веб-приложения, включающая:

1. Архитектурные решения: Защита фронтенда (XSS), бэкенда (SQL-инъекции), БД (шифрование).

2. Управление доступом: Четкое разделение ролей и политик (MFA, PoLP).

3. Проактивные меры: Регулярный аудит, автоматические обновления, мониторинг.

4. Реагирование на инциденты: Пошаговый план для минимизации ущерба.

Итог: Модель позволяет снизить риски кибератак на 70-80% (по данным OWASP Top 10) и соответствует стандартам ISO 27001. Для дальнейшего развития рекомендуется внедрить машинное обучение для анализа аномалий в режиме реального времени.

**Источники:**

* 1. OWASP Top 10 (2021)
  2. Приказ ФСТЭК России № 239 от 25.12.2020 «Требования к защите информации в ИС»
  3. ГОСТ Р 57580.1-2017 «Безопасность финансовых организаций. Защита информации»
  4. NIST SP 800-63B (Digital Identity Guidelines)