ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на разработку системы анализа защищённости веб-приложений на основе моделирования сетевых атак (без DoS и DDoS)

Задача:

Разработка автоматизированной системы для выявления уязвимостей в веб-приложениях путём безопасного моделирования типовых сетевых атак в изолированной тестовой среде. Система должна проводить многоуровневый анализ (кода, логов, трафика), формировать структурированные отчёты с рекомендациями и предоставлять удобную веб-панель управления с ролевым доступом. Все проверки должны выполняться без воздействия на продуктивные сервисы.

Разработка системы анализа защищённости веб-приложений

Цель разработки

Создание инструмента, позволяющего выявлять уязвимости в веб-приложениях **до их выхода в эксплуатацию**, тем самым снижая риски компрометации, утечек данных и несанкционированного доступа. Система должна быть полностью изолирована от боевой инфраструктуры и

соответствовать принципам безопасного тестирования.

Основные функции системы

Система должна уметь:

- **Моделировать типовые сетевые атаки**, включая:
  - SQL-инъекции (все основные векторы: классические, слепые, time-based);
  - Межсайтовый скриптинг (XSS): reflected, stored, DOM-based;
  - Подделка межсайтовых запросов (CSRF);
  - Brute-force атаки на формы аутентификации (с учётом лимитов попыток, CAPTCHA и т.п.);
  - Перехват/угадывание сессионных токенов;
  - Ошибки конфигурации: открытые директории, отладочные режимы, незащищённые API-эндпоинты, утечки версий ПО.

Важно: моделирование DoS/DDoS-атак **запрещено**. Система не должна генерировать нагрузку, способную нарушить работу тестируемого приложения.

- **Выполнять все проверки в изолированной тестовой среде**, которая:
  - Полностью отделена от продуктивной инфраструктуры;
  - Имеет ограниченный сетевой стек (только внутренние IP-адреса);
  - Не имеет выхода в интернет без явного

разрешения;
    - Поддерживает развёртывание копии тестируемого приложения (через Docker, виртуальные машины или иные средства).
- **Проводить многоуровневый анализ**:
    - **SAST** (статический анализ исходного кода): поддержка основных языков (Python, JavaScript, PHP, Java, C#);
    - **Анализ логов приложения**: выявление подозрительных паттернов (например, повторяющиеся ошибки 500, попытки SQL-инъекций в логах);
    - **Анализ сетевого трафика**: перехват и анализ HTTP/HTTPS-запросов между компонентами приложения в тестовой среде.
- **Формировать структурированные отчёты**, содержащие:
    - Тип уязвимости (с привязкой к CWE/OWASP Top 10);
    - Уровень критичности (низкий / средний / высокий / критический);
    - Локацию (URL, путь к файлу, строка кода, параметр запроса);
    - Рекомендации по устранению (с примерами безопасного кода);
    - Возможность добавления комментариев аналитиком.
- **Поддерживать экспорт отчётов в форматах**:
    - PDF — для внутреннего использования и аудита;
    - HTML — для удобного просмотра в браузере;

- JSON — для интеграции с системами управления уязвимостями (например, DefectDojo, Jira, SIEM).

Веб-интерфейс и управление доступом

- Единая веб-панель управления с поддержкой ролевой модели:
  - **Администратор**: управление пользователями, ролями, настройками системы, обновлением базы уязвимостей;
  - **Аналитик безопасности**: запуск, остановка и мониторинг сканирований, просмотр и экспорт отчётов, добавление комментариев;
  - **Разработчик**: просмотр только тех результатов, которые относятся к его проектам, с возможностью фильтрации по статусу («не исправлено», «в работе», «исправлено»).
- Интерфейс должен быть **интуитивно понятным**, даже для пользователей без глубоких технических знаний.
- Поддержка **многоязычности** (минимум: русский и английский).

База уязвимостей и сценариев атак

- Встроенная база данных известных уязвимостей и сценариев атак с возможностью:
  - Поиска и фильтрации по типу, уровню риска, CWE ID, OWASP ID, языку программирования;
  - Регулярного обновления (вручную или

автоматически через API);
- Добавления и редактирования пользовательских сценариев (в том числе кастомных payload’ов);
- Просмотра истории изменений базы.

Технологические и эксплуатационные требования

Платформенная совместимость

- Поддержка операционных систем: **Windows 10/11, Windows Server 2016+**, **Ubuntu 20.04/22.04 LTS**, **CentOS 7/8**.

Технологический стек

- Серверная часть: **Python 3.9+** с использованием асинхронных фреймворков (FastAPI, Celery);
- Клиентская часть (веб-интерфейс): **React 18+** с TypeScript;
- СУБД: **PostgreSQL 14+** (предпочтительно) или **SQLite** (для лёгких развёртываний);
- Контейнеризация: **Docker** и **Docker Compose** для упрощения развёртывания.

Интеграция

- Поддержка интеграции с **GitLab CI/CD**:
  - Автоматический запуск сканирования при пуше кода или создании merge request;
  - Отображение результатов прямо в интерфейсе GitLab (через pipeline artifacts или comments).

- Наличие **REST API** для:
  - Запуска и остановки сканирований;
  - Получения статуса и результатов;
  - Управления политиками и сценариями;
  - Интеграции с внешними системами (SIEM, SOAR, Jira, DefectDojo и др.).

Надёжность и безопасность

- Система должна сохранять данные при сбоях (journaling, транзакции в БД);
- Обеспечивать ведение **журнала действий (audit log)** со следующими полями: пользователь, действие, дата/время, IP-адрес;
- Поддерживать **резервное копирование** конфигураций, результатов сканирований и базы уязвимостей (ежедневно по расписанию);
- Продолжать работу при ошибках в отдельных модулях моделирования (изоляция компонентов);
- Все данные передаются по защищённому соединению (**HTTPS/TLS 1.3**);
- Аутентификация пользователей — через **встроенную учётную запись** или **интеграцию с LDAP/AD** (опционально).

Ограничения

- **Запрещено моделирование DoS/DDoS-атак**;
- Система **не должна генерировать трафик за пределы тестовой среды**;
- **Не допускается автоматическое сканирование продуктивных (боевых) систем** без явного

согласования и изоляции;
- Все сетевые запросы ограничены внутренней инфраструктурой тестовой зоны.

Формат передачи результата

- Исходный код системы с полной документацией по установке и эксплуатации;
- Docker-образы и docker-compose.yml для быстрого развёртывания;
- Примеры конфигураций для интеграции с GitLab CI и REST API;
- Пользовательская документация (в формате PDF и веб-справка в интерфейсе);
- Руководство администратора с описанием всех настроек, политик и сценариев;
- Тестовый набор данных для демонстрации работы системы.

Срок выполнения работ

6 (шесть) недель с момента подписания договора.