Найдёнов А.А. ИД22-3

Разработка контейнера для обеспечения безопасности рекомендательной системы в финансовом секторе при угрозе неправомерных действий в каналах связи

Рекомендательные системы в финансовом секторе играют ключевую роль в предоставлении персонализированных услуг: кредитование, инвестиции, страхование, планирование бюджета. Однако их работа с персональными данными и высокой доступностью делает их уязвимыми к сетевым атакам, таким как DDoS, SQL-инъекции, XSS. Для обеспечения безопасности и соответствия современным стандартам DevOps была разработана система на основе контейнеризации, где каждый компонент изолирован и выполняет свою строго определенную функцию.

Проект реализован с использованием Docker и Docker Compose, что позволило создать гибкую и масштабируемую архитектуру. Основные контейнеры:

* Nginx — прокси-сервер с логированием в JSON.
* Flask — бэкенд рекомендаций, реализующий безопасный поиск через ORM.
* MySQL — хранение данных рекомендаций.
* Fail2ban — планируется для блокировки IP при частых запросах (не настроен).

Система демонстрирует работу через HTML-интерфейс, поддерживает HTTPS, логирует трафик и защищена от SQL-инъекций через SQLAlchemy. Документация оформлена в соответствии с ГОСТ

**Анализ проблемы и обоснование выбора технологий**

Финансовый сектор стал одной из главных целей кибератак из-за высокой ценности обрабатываемых данных. Наиболее распространенные угрозы:

* SQL-инъекции: Злоумышленник внедряет вредоносные SQL-запросы через параметры поиска (**?q=**).
* XSS-атаки: Вставка JavaScript-кода в поля ввода для исполнения в браузере жертвы.
* DDoS-атаки: Частые запросы, направленные на перегрузку сервера.

Для противодействия этим угрозам выбраны технологии, которые не только обеспечивают функциональность, но и имеют встроенные механизмы защиты:

* Nginx — как прокси и логирующий модуль.
* Flask с SQLAlchemy ORM — для безопасного поиска и экранирования вводимых данных.
* MySQL — надежное хранение данных.
* Fail2ban — для автоматической блокировки подозрительных IP.

Nginx выбран как прокси-сервер благодаря своей производительности и гибкости. Он позволяет:

* Перенаправлять HTTP-запросы на Flask.
* Логировать данные в JSON-формате для последующего анализа.
* Обеспечивать шифрование через TLS 1.2+, что соответствует требованиям к защите данных в финансовых системах.

Nginx принимает все внешние запросы, шифрует их через TLS 1.2+ и направляет на Flask. Логирование в JSON позволяет:

* Хранить метаданные: время, IP, тип запроса, статус.
* Интегрироваться с системами анализа, такими как ELK или Grafana.
* Обеспечить аудит действий пользователей.

Конфигурация в nginx.conf включает:

* Редирект с HTTP на HTTPS для обеспечения целостности данных.
* Передачу заголовков Host, X-Real-IP, X-Forwarded-For для анализа и отслеживания трафика.

Flask был выбран как легковесный фреймворк, позволяющий быстро реализовать бизнес-логику. Для защиты от SQL-инъекций использован SQLAlchemy — ORM, которая автоматически экранирует входные данные. Например, поиск рекомендаций осуществляется через:

results = Recommendation.query.filter(Recommendation.product.ilike(f"%{query}%")).all()

Это исключает возможность выполнения вредоносных SQL-запросов, так как ORM преобразует входные данные в параметризованные запросы.

HTML-генерация выполняется через шаблоны, что снижает риски XSS. Например, карточки рекомендаций строятся динамически, но без исполнения JavaScript-кода.

Flask реализует бизнес-логику:

* Прием запросов от Nginx.
* Поиск рекомендаций через ORM.
* Генерация HTML-страницы с Bootstrap.

Особое внимание уделено безопасности:

* SQL-инъекции: Защита через параметризованные запросы (ORM).
* XSS: Экранирование входных данных (например, **?q=<script>alert(1)</script>** не исполняется).
* DDoS: Хотя Fail2ban не настроен, структура системы позволяет добавить защиту в будущем.

MySQL выбрана как надежная и широко используемая СУБД с поддержкой ACID-транзакций. Инициализация данных через **init.sql** гарантирует, что таблица **recommendations** создается автоматически:

CREATE TABLE recommendations (

id INT AUTO\_INCREMENT PRIMARY KEY,

product VARCHAR(100),

reason TEXT

);

INSERT INTO recommendations (product, reason) VALUES

('Credit Card', 'High rewards for spending'),

('Investment Fund', 'Low risk, steady growth');

Fail2ban предназначен для блокировки IP-адресов при подозрительном поведении, например, 50+ запросов за 60 секунд. Однако текущая реализация столкнулась с ограничением: JSON-логи Nginx не могут быть напрямую проанализированы Fail2ban. Для корректной работы потребовалось бы перейти на строковый формат логов или использовать парсер

Fail2ban настроен в **docker-compose.yml**, но не активен из-за сложностей с JSON-логами Nginx. В будущем можно реализовать:

* Переход на строковый формат логов Nginx.
* Настройку **failregex** для извлечения IP.
* Интеграцию с **iptables** для блокировки.

Система спроектирована в микросервисном стиле, где каждый контейнер отвечает за свою часть функционала и изолирован от других через сеть **secure-network**

**Пользователь**

**│**

**▼**

**[ proxy:443 (Nginx) ] → [ fail2ban: (IP-блокировка) ]**

**│**

**▼**

**[ recommendation\_system:5000 (Flask) ] ← [ db:3306 (MySQL) ]**

Все контейнеры работают в изолированной сети secure-network, созданной через docker-compose.yml. Это позволяет им безопасно обмениваться данными, не exposing внутренние порты наружу.

Nginx — первый уровень взаимодействия с пользователем. Он принимает HTTPS-запросы и перенаправляет их на Flask , который работает внутри своего контейнера. Nginx также записывает логи в формате JSON, включая метаданные: @timestamp, remote\_addr, request, status. Эти данные могут быть использованы для анализа трафика, но в текущей версии Fail2ban не настроен, так как не удалось корректно считать JSON-логи.

Flask — бэкенд, который отвечает за бизнес-логику. Получив запрос от Nginx, он обращается к MySQL для извлечения рекомендаций. Поиск осуществляется через SQLAlchemy, что исключает SQL-инъекции. Ответ отправляется обратно через Nginx, который отдает его пользователю в виде HTML-страницы.

MySQL — хранит рекомендации в таблице recommendations. На старте контейнер запускает init.sql, который создает таблицу и заполняет её тестовыми записями: кредитные карты, инвестиционные фонды, ипотека, страхование и личные займы.

Fail2ban — планируемый контейнер для блокировки IP-адресов, отправляющих частые запросы. Он подключается к access.log Nginx и анализирует логи. Если количество запросов от одного IP превышает лимит (maxretry), Fail2ban блокирует этот IP через iptables. На данный момент контейнер не настроен.

Проект хранится в репозитории, где все компоненты организованы в четкой иерархии. Основные файлы и папки:

docker-compose.yml — основной файл, который определяет все контейнеры: Nginx, Flask, MySQL, Fail2ban. Здесь прописаны порты, сети, переменные окружения и пути к логам. Он управляет оркестрацией системы и позволяет запускать все контейнеры одной командой docker-compose up -d

Dockerfile — содержит инструкции для сборки образа Flask-приложения. В нем указаны зависимости, установка библиотек (Flask, SQLAlchemy, pymysql), копирование app.py и запуск приложения. Этот файл гарантирует, что контейнер будет собираться одинаково на любой машине.

app.py — бэкенд рекомендательной системы. Через Flask реализованы маршруты / и /search, где происходит поиск рекомендаций по запросу пользователя. Используется SQLAlchemy для безопасного обращения к БД, предотвращающего SQL-инъекции. HTML-генерация ведётся с помощью шаблонов, чтобы избежать XSS.

nginx.conf — конфигурация Nginx. Указан JSON-формат логов, настроены прокси-параметры, SSL-шифрование, редирект с HTTP на HTTPS, а также безопасная передача заголовков. Логирование в JSON позволяет легко интегрировать систему с инструментами анализа (например, ELK, Grafana, Prometheus).

init.sql — инициализирует MySQL. Создаёт базу данных recommendation\_db, таблицу recommendations с полями id, product, reason. Вставляет тестовые данные для демонстрации рекомендаций.

README.md — основная инструкция по запуску и тестированию системы. Здесь описаны команды для сборки системы.

gost/ — папка с документами по ГОСТ: пояснительная записка, руководство программиста

docs/ — дополнительная техническая документация описание программы, текст программы, программа и методы испытаний.

Система проверена через:

# Сборка и запуск

docker-compose up -d --build

# Проверка контейнеров

docker ps

# Тестирование интерфейса

curl -k https://localhost

Результат:

* Все контейнеры запущены.
* Данные из MySQL доступны через docker exec.

Безопасность

SQL-инъекции - проверка через: curl -k "https://localhost/search?q=1' OR '1'='1"

Результат:

* SQL-инъекция не влияет на БД благодаря ORM.

XSS-атаки – проверка через curl -k "https://localhost/search?q=<script>alert(1)</script>"

Результат:

* XSS-запросы логируются, но не исполняются.

DDoS Проверка через: for i in {1..100}; do curl -k "https://localhost"; done

Результат:

* Все запросы проходят, система не готова к блокировке через Fail2ban

Выводы и результаты

Достигнутые цели

* Работа системы: Nginx → Flask → MySQL.
* Безопасность: Защита через ORM и экранирование данных.
* Логирование: JSON-формат для анализа и аудита.
* Масштабируемость: Структура позволяет добавить WAF (ModSecurity), кэширование (Redis), мониторинг (Prometheus, Grafana).

Неудачи

* Fail2ban не настроен: Проблема с JSON-логами Nginx.
* XSS-атаки: Требуется более глубокая обработка вводимых данных.

План доработки

* Переход на строковый формат логов Nginx для совместимости с Fail2ban.
* Добавление ModSecurity для глубокой проверки трафика и блокировки XSS.
* Интеграция Redis для кэширования популярных запросов.
* Мониторинг через Prometheus + Grafana для отслеживания атак.
* Улучшение HTML-генерации для исключения возможности вставки вредоносного кода.