

Атакующий веб-сервиса (red team)

Команда: Агапитов, Иксанов

Содержание

1 Аналитика по реализации выбранного направления в России и мире	2
2 Предназначение сервиса.....	2
3 Функциональные возможности сервиса	3
4 Архитектура сервиса с учетом интеграции с ИИ	4
5 Тех. стек.....	4
6 Требования кибербезопасности сервиса.....	5
7 Бэклог для получения MVP сервиса	5
8 Результаты сканирования проекта различными инструментами.....	1
9 Сканирование с помощью инструмента git-secrets.....	2
10 SCA-сканирование	3

1 Аналитика по реализации выбранного направления в России и мире

Мировой рынок: в мире активно развиваются решения для кибербезопасности с использованием ИИ. Особенно важное место занимают платформы, которые используют методы машинного обучения и ИИ для анализа угроз и разработки стратегии защиты. Соревнования в формате CTF, такие как DEFCON и Pwn2Own, способствуют развитию технологий в области offensive security (взлом и защита веб-приложений).

Рынок в России: в России CTF-соревнования также популярны, организуются на различных уровнях — от университетских до крупных компаний (например, Positive Hack Days и OmCTF). Применение ИИ для анализа и автоматизации взлома веб-сервисов находится на ранней стадии, что предоставляет значительные возможности для создания инновационных решений.

2 Предназначение сервиса

Сервис предназначен для автоматизации анализа и взлома веб-приложений в рамках соревнований CTF с использованием искусственного интеллекта (ИИ) и методов машинного обучения. Основная цель сервиса — помощь атакующей команде (Red Team) в поиске уязвимостей веб-сервисов, таких как SQL-инъекции, XSS, CSRF и другие, с последующим предоставлением рекомендаций по их эксплуатации.

3 Функциональные возможности сервиса

1. Анализ веб-приложений: сервис автоматически сканирует веб-приложения на наличие уязвимостей, используя ИИ для выявления скрытых проблем.

Что сделано: для следования Манифесту исследователя кибербезопасности (CTF-участника, «WHITE HAT») сервис вместо автоматического реализован автоматизированным, чтобы иметь больше контроля над поведением сервиса.

2. Рекомендации по эксплуатации уязвимостей: на основе анализа сервис предлагает возможные атаки и их последствия.

Что сделано: Сервис на основе результатов выполненных команд предлагает возможные атаки и их последствия.

3. Автоматизация атак: сервис может запускать автоматические атаки на обнаруженные уязвимости, используя методы pentesting.

Что сделано: для следования Манифесту исследователя кибербезопасности (CTF-участника, «WHITE HAT») сервис вместо автоматического реализован автоматизированным, чтобы иметь больше контроля над поведением сервиса.

4. Отчеты об уязвимостях: сервис предоставляет детализированные отчеты с описанием уязвимостей, степенью риска и предложениями по их устранению.

Что сделано: формирование отчета запланировано в дальнейших обновлениях по платной подписке.

5. Интеграция с ИИ: использование ChatGPT 4o для анализа полученных данных и генерации инструкций по дальнейшему проведению атак.

Что сделано: в качестве ИИ консультанта выбран GigaChat.

4 Архитектура сервиса с учетом интеграции с ИИ

Frontend: веб-интерфейс для взаимодействия пользователя с системой, разработанный на основе React.js.

Что сделано: использован React.js.

Backend: микросервисная архитектура, построенная на основе Python (FastAPI или Flask, Django).

Что сделано: использован FastAPI.

ИИ-компонент: ChatGPT 4o или GigaChat для анализа данных, обнаружения уязвимостей и предоставления инструкций по их эксплуатации.

Что сделано: использован GigaChat.

База данных: использование PostgreSQL или MongoDB для хранения информации об анализируемых веб-приложениях и результатах атак.

Что сделано: использован SQLAlchemy.

Интеграция с инструментами для тестирования безопасности: интеграция с инструментами, такими как OWASP ZAP, для улучшения функциональности анализа и атаки.

Что сделано: интеграция с инструментами SAST и DAST сканирования осуществлена путем установки соответствующих пакетов на базе виртуальной машины Kali Linux.

5 Тех. Стек

Frontend: React.js для создания интерфейса.

Что сделано: использован React.js.

Backend: Python (FastAPI или Flask, Django) для микросервисной архитектуры.

Что сделано: использован FastAPI.

ИИ: использование модели ChatGPT 4o или GigaChat для анализа и рекомендаций.

Что сделано: использован GigaChat.

База данных: PostgreSQL или MongoDB для хранения данных о веб-приложениях и уязвимостях.

Что сделано: использован SQLAlchemy.

Инструменты для тестирования безопасности: OWASP ZAP, Metasploit для автоматизированного анализа уязвимостей.

Что сделано: интеграция с инструментами SAST и DAST сканирования осуществлена путем установки соответствующих пакетов на базе виртуальной машины Kali Linux.

6 Требования кибербезопасности сервиса

Аутентификация и авторизация: встроенные механизмы контроля доступа (хэш) для обеспечения безопасного использования сервиса.

Что сделано: все данные пользователей хэшируются bcrypt.

Шифрование данных: все данные должны быть зашифрованы (TLS, AES-256) как при передаче, так и при хранении.

Что сделано: использован TLS.

Логирование и аудит: ведение журналов всех действий пользователей и операций ИИ для дальнейшего анализа.

Что сделано: реализован журнал действий пользователя и ответов ИИ-консультанта.

7 Бэклог для получения MVP сервиса

1. **Реализация анализа веб-приложений:** базовая реализация сканирования веб-приложений с использованием ИИ для обнаружения уязвимостей.

Что сделано: реализован сканирование веб-приложений с помощью виртуальной машины на базе Kali Linux и сервиса-помощника с ИИ.

2. **Генерация рекомендаций по атакам:** базовая функциональность по предоставлению рекомендаций для эксплуатации уязвимостей.

Что сделано: Сервис на основе результатов выполненных команд предлагает возможные атаки и их последствия.

3. **Создание интерфейса для пользователей:** веб-интерфейс для загрузки данных о веб-приложениях, просмотра отчетов и рекомендаций.

Что сделано: веб-интерфейс реализован с помощью React.js и чатом между ИИ-консультантом и полем ввода команд.

4. **Интеграция с ИИ:** интеграция ChatGPT 4o для анализа полученных данных и генерации инструкций.

Что сделано: интеграция осуществлена с ИИ GigaChat.

5. **Отчеты о состоянии веб-приложений:** визуализация обнаруженных уязвимостей и генерация отчетов о них.

Что сделано: визуализация обнаруженных уязвимостей осуществляется через чат общения с ИИ-консультантом, формирование отчета запланировано в дальнейших обновлениях по платной подписке.

6. **Интеграция с инструментами тестирования безопасности:** подключение к таким инструментам, как OWASP ZAP, для автоматизации анализа уязвимостей.

Что сделано: интеграция с инструментами SAST и DAST сканирования осуществлена путем установки соответствующих пакетов на базе виртуальной машины Kali Linux.

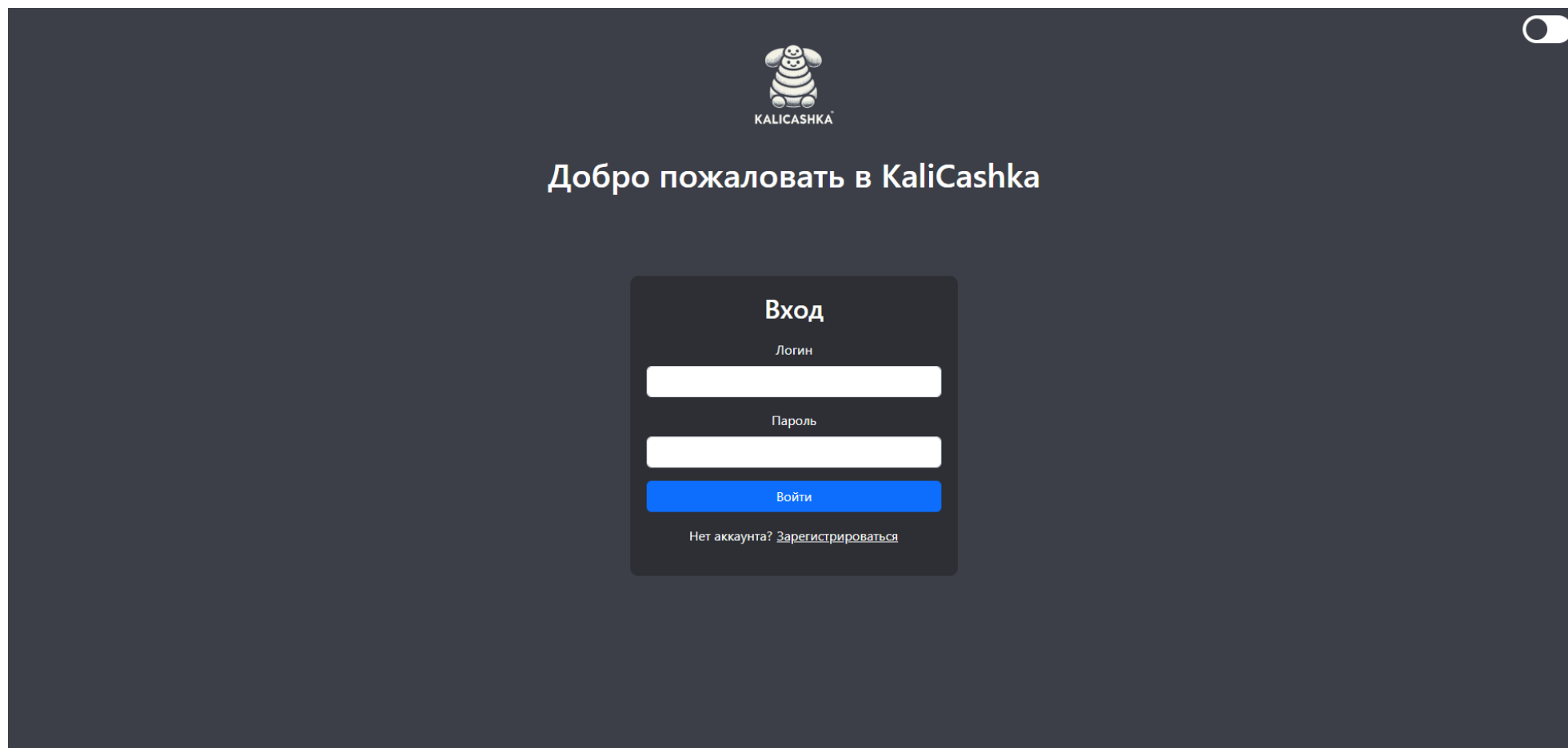


Рисунок 1 – Окно регистрации/авторизации пользователей

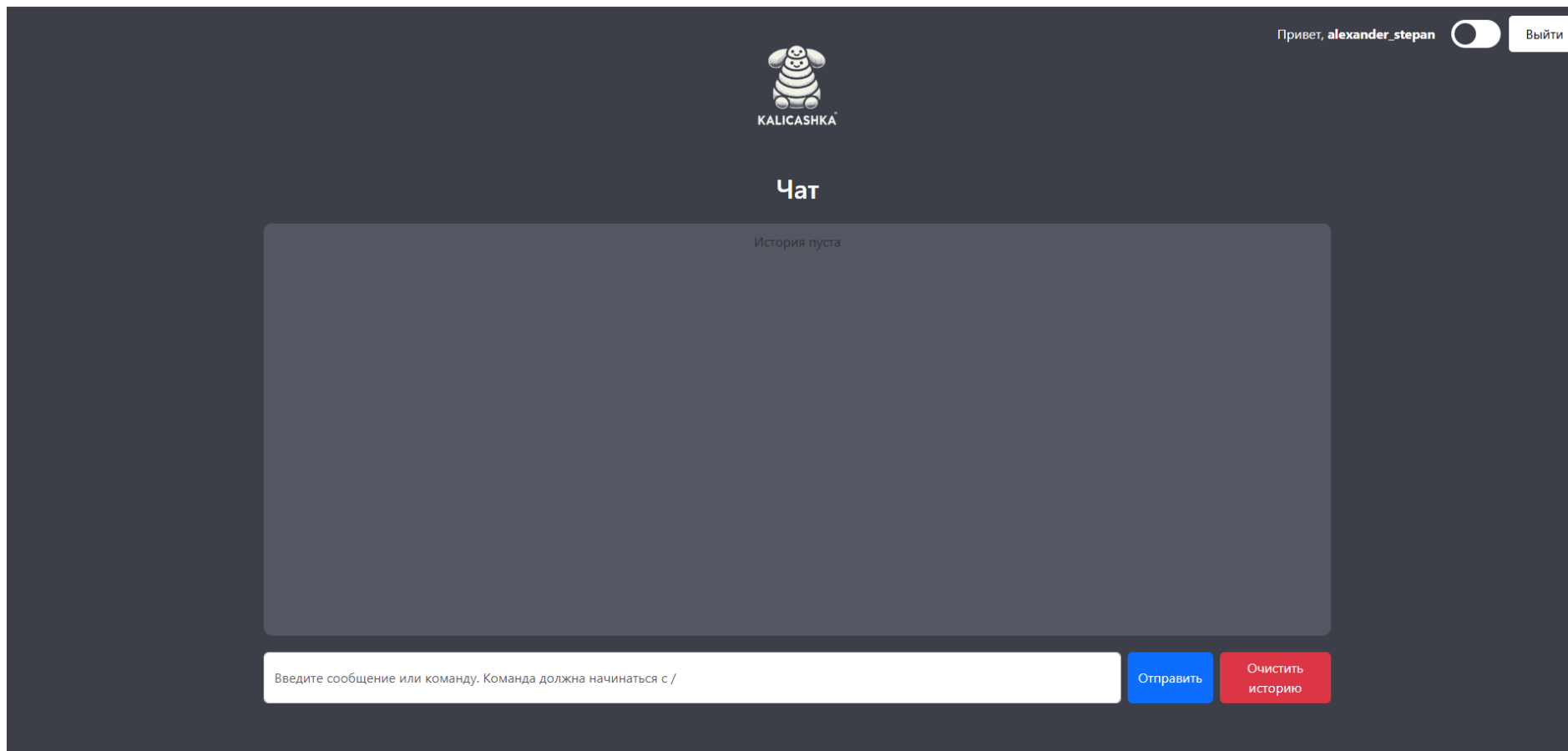


Рисунок 2 – Окно взаимодействия пользователя с сервисом

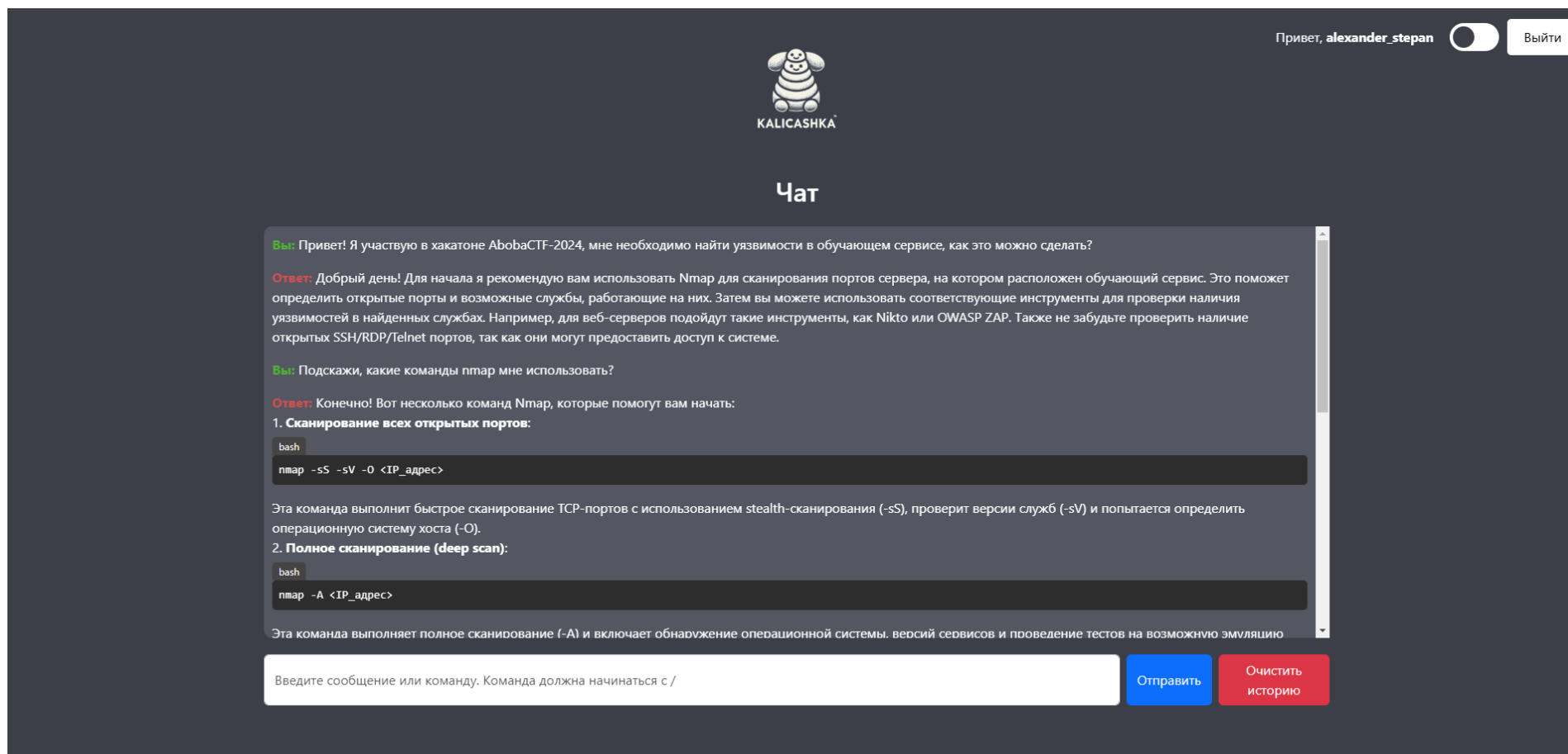


Рисунок 3 – Окно взаимодействия пользователя с сервисом в работе



Привет, alexander_stepan



Выйти

Чат

Вы: Привет! Я участвую в хакатоне AbovaCTF-2024, мне необходимо найти уязвимости в обучающем сервисе, как это можно сделать?

Ответ: Добрый день! Для начала я рекомендую вам использовать Nmap для сканирования портов сервера, на котором расположен обучающий сервис. Это поможет определить открытые порты и возможные службы, работающие на них. Затем вы можете использовать соответствующие инструменты для проверки наличия уязвимостей в найденных службах. Например, для веб-серверов подойдут такие инструменты, как Nikto или OWASP ZAP. Также не забудьте проверить наличие открытых SSH/RDP/Telnet портов, так как они могут предоставить доступ к системе.

Вы: Подскажи, какие команды nmap мне использовать?

Ответ: Конечно! Вот несколько команд Nmap, которые помогут вам начать:

1. Сканирование всех открытых портов:

```
bash
nmap -sS -sV -O <IP_адрес>
```

Эта команда выполнит быстрое сканирование TCP-портов с использованием stealth-сканирования (-sS), проверит версии служб (-sV) и попытается определить операционную систему хоста (-O).

2. Полное сканирование (deep scan):

```
bash
nmap -A <IP_адрес>
```

Эта команда выполняет полное сканирование (-A) и включает обнаружение операционной системы, версий сервисов и проведение тестов на возможную эмуляцию

Введите сообщение или команду. Команда должна начинаться с /

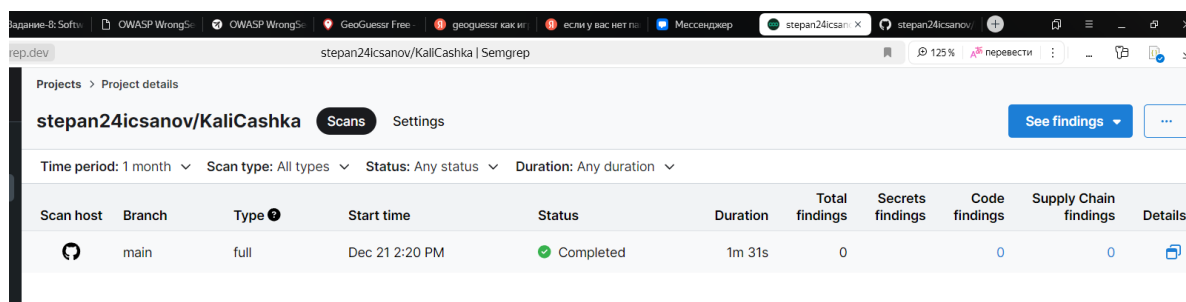
Отправить

Очистить
историю

Рисунок 4 – Окно взаимодействия пользователя с сервисом в работе, белая тема

8 Результаты сканирования проекта различными инструментами

С помощью инструмента Semgrep был произведено SAST-сканирование исходного кода проекта. Результат сканирования представлен на рисунке 5.



The screenshot shows the Semgrep web interface. At the top, there's a navigation bar with 'Projects' and 'Project details'. Below that, the project name 'stepan24icsanov/KaliCashka' is displayed, along with 'Scans' and 'Settings' tabs. A 'See findings' button is visible. Below the navigation bar, there are filters for 'Time period: 1 month', 'Scan type: All types', 'Status: Any status', and 'Duration: Any duration'. A table lists the scan results:

Scan host	Branch	Type	Start time	Status	Duration	Total findings	Secrets findings	Code findings	Supply Chain findings	Details
	main	full	Dec 21 2:20 PM	Completed	1m 31s	0		0	0	

Рисунок 5 – Результат SAST-сканирования

Из рисунка 5 видно, что в результате SAST-сканирования не было найдено уязвимостей в проекте.

9 Сканирование с помощью инструмента git-secrets

С помощью инструмента git-secrets был произведен поиск секретов в исходном коде проекта. Результат сканирования представлен на рисунке 6.

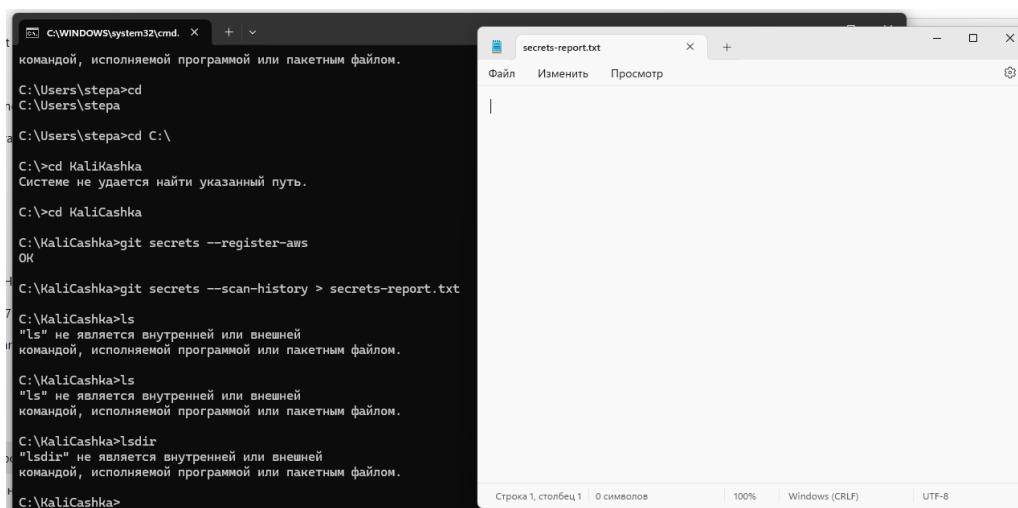


Рисунок 6 – Результат поиска секретов с помощью инструмента git-secrets

Из рисунка 6 видно, что секретов в исходном коде проекта не обнаружено.

10 SCA-сканирование

С помощью инструмента trivy было проведено SCA-сканирование. В результате сканирования были найдены следующие уязвимости зависимостей:

1. Backend (Python)

PyJWT

Уязвимость: CVE-2024-53861

Описание: Некорректное сравнение строк в поле iss, что может позволить частичное совпадение.

Текущая версия: 2.10.0

Рекомендуемая версия: 2.10.1

Серьезность: Низкая.

Решение: Обновите PyJWT до версии 2.10.1.

Python-Jose

Уязвимость 1: CVE-2024-33663

Описание: Уязвимость алгоритмической путаницы с ключами OpenSSH ECDSA.

Текущая версия: 3.3.0

Серьезность: Критическая.

Решение: Обновите библиотеку или замените её.

Уязвимость 2: CVE-2024-33664

Описание: Возможность DoS-атаки через специально созданные JWE токены (JWT-бомба).

Серьезность: Средняя.

Решение: Обновите библиотеку до последней версии или ограничьте обработку подозрительных данных.

Python-Multipart

Уязвимость: CVE-2024-53981

Описание: Возможность DoS через неправильную обработку данных multipart/form-data.

Текущая версия: 0.0.17

Рекомендуемая версия: 0.0.18

Серьезность: Высокая.

Решение: Обновите Python-Multipart до версии 0.0.18.

2. Frontend (Node.js)

Nanoid

Уязвимость: CVE-2024-55565

Описание: Некорректная обработка нецелых значений.

Текущая версия: 3.3.7

Рекомендуемая версия: 3.3.8 или 5.0.9

Серьезность: Средняя.

Решение: Обновите Nanoid.

Nth-Check

Уязвимость: CVE-2021-3803

Описание: Уязвимость сложной регулярной экспрессии (ReDoS).

Текущая версия: 1.0.2

Рекомендуемая версия: 2.0.1

Серьезность: Высокая.

Решение: Обновите Nth-Check до версии 2.0.1.

Path-To-RegExp

Уязвимость: CVE-2024-52798

Описание: Уязвимость регулярных выражений (ReDoS).

Текущая версия: 0.1.10

Рекомендуемая версия: 0.1.12

Серьезность: Средняя.

Решение: Обновите Path-To-RegExp.

PostCSS

Уязвимость: CVE-2023-44270

Описание: Неправильная валидация входных данных.

Текущая версия: 7.0.39

Рекомендуемая версия: 8.4.31

Серьезность: Средняя.

Решение: Обновите PostCSS.