**SafeThisSpace - Research Document**

**1 Анализ конкурентов**

На рынке существует ряд платформ и инструментов для поиска уязвимостей в программном обеспечении, многие из которых используют технологии искусственного интеллекта и машинного обучения. Мы проанализировали наиболее успешные, на наш взгляд, компании, чтобы выделить их преимущества и возможные недостатки и использовать эти знания в своем проекте.

**1.1 Wapiti**

Сканер уязвимостей веб-приложений. Предоставляется отчёт в формате HTML-страницы. Относительно большая вероятность ложных срабатываний. Бесплатный инструмент с открытым исходным кодом. Об инструменте достаточно мало информации на русском языке. Ресурс достаточно сложен в использовании для неопытного пользователя

**1.2 Acunetix**

Сканер веб-уязвимостей, автоматизирующий процесс проверки безопасности веб-приложений. Тестирует приложение на наличие SQL-инъекций, XSS, XXE, SSRF и многих других веб-уязвимостей. Проводит глубокий анализ и выводит подробный отчет об уязвимостях. Делает большое количество проверок, однако это может усложнить бизнес-процессы исследуемого сервиса.

**1.3 OWASP ZAP**

Сканер веб-приложений, основанный на методике DAST. В качестве преимуществ можно выделить кроссплатформенность, открытый исходный код, возможность работы как через графический интерфейс, так и интерфейс через командной строки, поддержка плагинов для расширения функциональности.

**1.4 Заключение и рекомендации**

Исходя из анализа перечисленных конкурентов, платформа ИИ для поиска уязвимостей может занять свою нишу на рынке, если предложит:

* более простую интеграцию с существующими инструментами разработчиков,
* упрощенный интерфейс для работы, ориентированный даже на неопытного пользователя,
* отчёт об уязвимостях и их возможное решение, что поможет объяснить пользователю проблемы его кода,
* бесплатное использование или низкую стоимость, что особенно важно для малых компаний,
* улучшенный функционал AI для минимизации ложных срабатываний и повышения точности анализа.

Также для улучшения нашего проекта мы планируем использовать WebGPU. WebGPU — это один из современных API, предназначенных для работы с компьютерной графикой. На данный момент WebGPU — это относительно новая технология, и её применение в веб-приложениях для анализа кода на уязвимости только начинает развиваться. Существуют инструменты для проверки безопасности веб-сайтов и анализа исходного кода, однако большинство из них не используют WebGPU для выполнения нейросетевых моделей непосредственно в браузере.

**2 Целевая аудитория и ее потребности**

**2.1 Целевая аудитория проекта**

Целевая аудитория проекта включает:

* разработчиков программного обеспечения, которым нужны инструменты, которые помогают обнаруживать уязвимости на стадии разработки, чтобы снизить затраты на исправление ошибок в дальнейшем,
* команды DevOps, интегрирующие безопасность в процессы разработки (DevSecOps),
* команды по безопасности заинтересованы в более глубоком анализе приложений, системной безопасности и интеграции инструментов в CI/CD;
* стартапы и малый бизнес, которым сложно нанять специализированных специалистов,
* крупные технологические корпорации, заинтересованные в оптимизации процесса тестирования на проникновение.

**2.2 Потребности**

* хороший инструмент для поиска уязвимостей кода,
* быстрая и точная обратная связь о состоянии безопасности,
* интуитивно понятный интерфейс и интеграция с существующими рабочими процессами,
* отчёт и рекомендации по исправлению обнаруженных уязвимостей.

**3 Описание технологий, используемых в проекте**

Проект применяет следующие технологии:

**3.1 Frontend**

* *React.js* для создания динамичного, отзывчивого пользовательского интерфейса,
* *Redux* для управления состоянием клиентской части приложения, что обеспечит синхронизацию данных на клиенте,
* *Material-UI / Tailwind CSS* для быстрой стилизации компонентов и обеспечения современного, удобного интерфейса,
* *Axios* для работы с HTTP-запросами, взаимодействия с API бэкенда и обработки данных с сервера.

**3.2 Backend**

* *FastAPI (Python)* для разработки высокопроизводительных API, оптимизированных для обработки больших объемов данных и работы с нейронными сетями,
* *Spring Boot (Java)* для разработки микросервисов, которые будут обрабатывать функциональность системы, не связанную с машинным обучением,
* *PyTorch (Python)* для разработки, обучения и оптимизации моделей машинного обучения, которые будут анализировать код и выявлять уязвимости,
* *SQLAlchemy (Python)* для взаимодействия с реляционными базами данных, хранения результатов анализа и данных пользователей,
* *PostgreSQL* для хранения данных о пользователях, проектах, уязвимостях и отчетах,
* *Redis* для кэширования и повышения производительности системы.

**3.3 Machine Learning**

* *Custom Dataset* для создания и использования кастомных датасетов, обученных на парах «уязвимость/решение», что обеспечит точность и актуальность анализа,
* *NLP* для анализа текста в коде, комментариях и конфигурационных файлах, что позволит выявлять потенциальные проблемы и уязвимости (CodeBERT или LLaMA и тд).

**3.4 DevOps и оркестрация**

* *Docker* для контейнеризации всех компонентов системы, что позволит легко развертывать и изолировать сервисы в различных окружениях,
* *Kubernetes* для управления оркестрацией контейнеров, автоматического масштабирования и обеспечения высокой доступности приложения в продакшн-среде,
* *Prometheus* для мониторинга.

**3.5 Защита**

* *JWT && LemonID* для аутентификации и авторизации пользователей, защиты данных и взаимодействия с API,
* *SSL/TLS* для обеспечения безопасной передачи данных между клиентом и сервером,
* Открытые утилиты для проверки кода (запуск на хосте).

**4 Анализ существующих технических решений**

При анализе технических решений для обнаружения уязвимостей важно учитывать разнообразие подходов, методов и технологий, применяемых различными инструментами и платформами. Ниже представлена более подробная информация о текущих решениях на рынке.

**4.1 Статический анализ кода (SAST)**

Статический анализ кода — это анализ исходного кода приложения без его реального исполнения. Также этот метод называют методом «белого ящика».  Основной фокус в этом методе — на коде. Как правило, статические анализаторы ограничиваются диагностикой простых случаев. Основной смысл в том, чтобы как можно раньше находить ошибки в исходном коде программ. Однако статический анализ, как правило, слаб в диагностике утечек памяти и параллельных ошибок. Пример инструмента – *SonarQube* - открытый анализатор, который предоставляет информацию о качестве кода и уязвимостях.

**4.2 Динамический анализ кода (DAST)**

Динамический анализ осуществляется путём тестирования приложения в работающем состоянии. DAST позволяет выявить уязвимости, которые могут проявиться только при выполнении кода.Динамический анализ выполняется с помощью набора данных, которые подаются на вход исследуемой программе. Поэтому эффективность анализа напрямую зависит от качества и количества входных данных для тестирования. Пример инструмента - *OWASP ZAP (Zed Attack Proxy)*.

**4.3 Машинное обучение в анализе уязвимостей**

Современные инструменты начали включать алгоритмы машинного обучения для повышения точности анализа. Машинное обучение часто является способ улучшения статического или динамического анализа кода. Например, при использовании ML SAST, снижается количество ложных срабатываний. Пример инструмента **–** *Darktrace* - использует ИИ для обнаружения аномалий в сетевом трафике, что может указывать на потенциальные уязвимости.

Существует множество различных методов и решений анализа кода на уязвимости. Каждый из них имеет свои преимущества и недостатки. Какой или какие использовать, зависит от задачи, которая встает перед инструментом или платформой по поиску уязвимостей. Хорошим решением является использование сразу нескольких методов.

**5 Описание и источники датасета**

Для обучения модели qwen2.5-coder используются специально подготовленные датасеты: CVEfixes, MoreFixes.

CVEfixes представляет собой датасет, содержащий пары уязвимого и исправленного кода, связанные с уязвимостями безопасности, зарегистрированными в базе CVE (Common Vulnerabilities and Exposures). Датасет включает в себя изменения кода (патчи), которые устраняют известные уязвимости, а также метаданные, такие как идентификаторы CVE и описание уязвимостей. Основная цель CVEfixes — предоставить исследователям и разработчикам материалы для изучения методов автоматического исправления и анализа уязвимостей программного обеспечения.

MoreFixes представляет собой обширный датасет, который содержит более широкий спектр исправлений кода, не ограничиваясь только уязвимостями безопасности. Датасет включает разнообразные изменения кода из популярных проектов с открытым исходным кодом, охватывающие улучшения функциональности, исправления ошибок и оптимизации.

**Список использованных источников**

1 Обзор OWASP ZAP. Сканер для поиска уязвимостей в веб-приложениях [Электронный ресурс] // URL: <https://habr.com/ru/companies/first/articles/709586/> (дата обращения: 26.11.2024).

2 ML SAST. Часть 1: как работают инструменты SAST и какие проблемы может решить применение машинного обучения? [Электронный ресурс] // URL: <https://habr.com/ru/companies/ussc/articles/783744/> (дата обращения: 13.12.2024).

3 Wapiti: a Free and Open-Source the web-application vulnerability scanner. [Электронный ресурс] // URL: <https://wapiti-scanner.github.io/> (дата обращения: 27.11.2024).

4 MoreFixes Dataset Github [Электронный ресурс] // URL: <https://github.com/JafarAkhondali/Morefixes> (дата обращения: 14.12.2024).

5 MoreFixes: A Large-Scale Dataset of CVE Fix Commits Mined through Enhanced Repository Discovery [Электронный ресурс] // URL: <https://dl.acm.org/doi/pdf/10.1145/3663533.3664036> (дата обращения: 14.12.2024).

6 CVEfixes Github [Электронный ресурс] // URL: <https://github.com/secureIT-project/CVEfixes> (дата обращения: 15.12.2024).

7 CVEfixes: Automated Collection of Vulnerabilities and Their Fixes from Open-Source Software [Электронный ресурс] // URL: <https://arxiv.org/pdf/2107.08760v1> (дата обращения: 15.12.2024).

8 Code-Switching Metrics Using Intonation Units [Электронный ресурс] // URL: <https://aclanthology.org/2023.emnlp-main.1047.pdf> (дата обращения: 16.12.2024).

9 Test-based and metric-based evaluation of code generation models for practical question answering[Электронный ресурс] // URL: <https://ieeexplore.ieee.org/document/10114665> (дата обращения: 16.12.2024).

10 NLPMetrics [Электронный ресурс] // URL: <https://github.com/gcunhase/NLPMetrics> (дата обращения: 16.12.2024).