УДК 004.056

ФАЗЗИНГ В ФОРМАТЕ СЕРВИСА: МАСШТАБИРУЕМАЯ ОБЛАЧНАЯ АРХИТЕКТУРА ДЛЯ АВТОМАТИЗИРОВАННОГО ФАЗЗИНГА

Дудкин А.С. (ВКА), **Низамидинов М.Ф.** (ВКА), **Шаповалов Д. Г.** (ВКА), **Яроцкий Г.Д.** (ВКА)
Научный руководитель – кандидат технических наук, доцент Дудкин А.С.
(Военно-космическая академия имени А.Ф.Можайского)

Введение. В современных реалиях фаззинг является одним из ключевых методов поиска уязвимостей в программном обеспечении. Он позволяет автоматически генерировать входные данные и выявлять неожиданные ошибки, приводящие к сбоям и потенциальным угрозам безопасности. В условиях стремительного роста цифровизации и увеличения числа кибератак важно минимизировать риски, связанные с эксплуатацией уязвимостей. Однако полноценное фаззинг-тестирование требует значительных вычислительных мощностей, а также наличия специалистов, способных правильно настраивать и интерпретировать результаты тестирования. Это создает барьер для многих организаций, затрудняя выявление уязвимостей до выпуска программных продуктов. Таким образом множество программных решений остаются непроверенными и становятся простой целью для предполагаемого злоумышленника [1].

Основная часть. Разработка программного решения основана на облачной архитектуре, включающей следующие компоненты: клиентский интерфейс (веб-приложение), серверная часть (бэкенд-логика), уровень хранения данных (СУБД), CI/CD-конвейер на основе GitLab CI и контейнеризированную среду выполнения фаззеров. Пользователь предлагаемого сервиса взаимодействует с простым веб-интерфейсом, позволяющим добавить программный код и обозначить «цель» для фаззинга. После передачи данных серверная часть (API) обрабатывает запрос и интегрирует его с GitLab, обеспечивая дальнейшее выполнение операций [2]. Далее определяется язык программирования и выбирается соответствующий фаззер, после чего создается изолированный контейнер с необходимым инструментарием. Компиляция выполняется с использованием специализированных компиляторов **afl-clang-fast**, **afl-clang-fast++** [3], интегрированных в систему для преобразования исходного кода. Тестирование происходит с помощью фаззеров **afl++** и **libfuzz**, для которого автоматически выбирается самая большая функция и изменяется код исследуемого продукта для тестирования через данные генерируемые для фаззинга [4]. В режиме реального времени сервис визуализирует ключевые показатели: покрытие кода, скорость выполнения операций, частоту запусков (запросов/сек). Все зафиксированные аварийные завершения (crashes) визуализируются с помощью Grafana и доступны тестировщику. Результаты анализа спустя определённое время, указанное пользователем, приходят тестировщику на почту.

Применение контейнеризации обеспечивает гибкость, масштабируемость и безопасность. Контейнерный подход позволяет запускать несколько параллельных сессий фаззинга эффективно распределяя нагрузку, а изолированная среда снижает риск выполнения потенциально вредоносного кода [5].

Выводы. Таким образом, представленный подход предлагает сервис для поиска уязвимостей, значительно снижая затраты на тестирование и повышая безопасность программных продуктов как с открытым исходным кодом, так и без него. В дальнейшем планируется данного решения в процессы разработки программных решений, что позволит компаниям быстрее находить и устранять уязвимости, улучшая качество и безопасность создаваемого программного обеспечения. Масштабируемость и достаточное количество ресурсов позволят проводить более качественную проверку кода, обнаруживая большее количество уязвимостей за минимальное время.

Список использованных источников:

1. Книга: "The Fuzzing Book" [Электронный ресурс]. – URL: https://www.fuzzingbook.org/ – (дата обращения 01.01.2025).
2. Документация GitLab CI/CD [Электронный ресурс]. – URL: https://docs.gitlab.com/ee/ci/ – (дата обращения 01.01.2025).
3. AFL++ (American Fuzzy Lop++) – Инструмент фаззинга [Электронный ресурс]. – URL: https://aflplus.plus/ – (дата обращения 01.01.2025).
4. libFuzzer – Встроенный фаззер от LLVM [Электронный ресурс]. – URL: https://llvm.org/docs/LibFuzzer.html – (дата обращения 01.01.2025).
5. Документация по Docker [Электронный ресурс]. – URL: https://docs.docker.com/ – (дата обращения 01.01.2025)

Яроцкий Г.Д. (автор) Подпись

Дудкин А.С. (автор) Подпись

Шаповалов Д.Г. (автор) Подпись

Низамидинов М.Ф. (автор) Подпись