Автотестирование веб-безопасности

Система автоматизированного тестирования веб-приложений предназначен для автоматического поиска, подтверждения и документирования уязвимостей в веб-приложениях (XSS, SQLi, CSRF, открытые конфиги и т.д.).

Цель: сократить время обнаружения уязвимостей и дать разработчикам воспроизводимый процесс демонстрации уязвимостей, и рекомендации для исправления.

Пользовательские роли:

1. Инженер по информационной безопасности – запускает глубокие проверки, анализирует результаты;
2. разработчик – получает отчеты и PоС, исправляет уязвимости;
3. менеджер проекта – смотрит сводные отчеты и прогресс по устранению;
4. CI/CD система (автоматический пользователь) – при каждом развертывании кода или обновлении проекта автоматически инициируется процесс сканирования.

Пользовательские истории:

1. Как инженер по информационной безопасности, хочу запустить сканирование по URL/репозиторию, чтобы быстро получить список уязвимостей;
2. как разработчик, хочу получить PoC (запрос/ответ, скриншот), чтобы быстрее воспроизвести и исправить баг;
3. как менеджер, хочу видеть агрегированные отчеты (количество уязвимостей по статусу и уровню серьезности), чтобы отслеживать прогресс;
4. как CI, хочу автоматически запускать скан после процесса объединения изменений из одной ветки в основную ветку репозитория, чтобы не допускать новых уязвимостей в рабочей среде;
5. как инженер по информационной безопасности, хочу настроить scope (область сканирования), whitelist (список разрешенных объектов), blacklist (список объектов, которые сканер должен игнорировать).

Пользовательские сценарии:

1. Запуск ручного скана.

Пользователь: инженер по информационной безопасности.

Шаги: выводит URL/адрес стенда -> задает область сканирования и, при необходимости, предоставляет учетные данные -> нажимает «Run» -> видит прогресс -> получает отчет.

Система: автоматически обходит и исследует все доступные точки входа на указанном адресе, запускает ZAP/Nuclei, запускает Semgrep, пытается воспроизвести РоС через Playwright, сохраняет результаты.

1. Ручная верификация найденной уязвимости.

Пользователь: разработчик.

Шаги: открывает карточку уязвимостей -> видит запрос/ответ, РоС-скриншот, рекомендацию -> отмечает статус «в работе» -> загружает исправление или обновление кода.

Система: ведет историю статусов, логирует время изменений.

1. Автоматический скан в CI.

Пользователь: CI.

Шаги: пуш в ветку main -> webhook запускает задачу в оркестраторе -> по завершении отправляется отчет в канал и в багтрекер.

Система: сканирование выполняется без UI в фоновом режиме, возвращает exit code >0 при найденных критических уязвимостях.

1. Агрегация и дашборд для менеджера.

Пользователь: менеджер.

Шаги: открывает дашборд -> видит сводку, распределение по уровням серьезности.

Система: строит графики, позволяет экспорт в PDF.

Таблица №1 – функциональные требования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название функции | Входные параметры | Входные параметры | Что делает |
| 1 | Запуск сканирования | URL, scope (ограничения в области сканирования), creds (учетные данные), параметры сканирования. | Task\_id, статус запуска. | Запускает сканирование. |
| 2 | Получение статуса задачи | Task\_id. | Прогресс, текущий шаг, логи | Возвращает статус и логи выполнения задач. |
| 3 | Получение результатов/отчета | Task\_id. | JSON-отчет, HTML/PDF, список уязвимостей | Сводный отчет с уязвимостями, распределение по уровням серьезности, РоС (запрос/ответ/скриншот), рекомендации. |
| 4 | Воспроизведение РоС | Vuin\_id. | РоС (запрос/ответ/скриншот). | Автоматическое воспроизведение РоС через headless-браузер и сохраняет результат. |
| 5 | Интеграция с CI | Webhook/API trigger. | Task\_id, exit code. | API для запуска скана из CI и возврат статуса |
| 6 | Управление областью/белым списком | Правила, определяющие, какие ресурсы включены или исключены из сканирования. | Подтверждение. | Позволяет задать допустимые хосты, пути и исключения. |
| 7 | Управление пользователями | Учетные данные пользователя. | JWT и session. | Управление доступом для ролей. |
| 8 | Дашборд | Период времени и фильтры. | Визуализация и экспорт. | Группировка уязвимостей, экспорт в PDF. |
| 9 | Хранилище данных | Файлы РоС, логи. | Ссылка. | Хранение артефактов и логов. |
| 10 | Планирование сканирования | Расписание и политика для автоматического запуска. | Task\_id. | Планирование периодических сканов. |

Аналоги:

1. OWASP ZAP.

Open-Source DAST-сканер, который ищет SQLi, XSS, уязвимости конфигурации.

Достоинства:

* Бесплатный и популярный;
* поддержка активного и пассивного сканирования;
* большое сообщество и плагины.

Недостатки:

* Работает только как DAST, без SAST;
* результат часто содержит много ложных срабатываний;
* нет удобнлй интеграции и сырые отчеты;

1. Nuclei.

Это инструмент для шаблонного тестирования.

Достоинства:

* Быстрый и легкий;
* гибкие YAML-шаблоны;
* хорошо автоматизируется в CI.

Недостатки:

* Только DAST;
* нет анализа кода;
* не строит РоС сам – только факт уязвимости;
* нет дашборда.

1. Burp Suite Professional.

Коммерческий инструмент для пантестеров. Имеет прокси, сканер, фреймворк для ручного и автоматического тестирования.

Достоинства:

* Отличный сканер, поддержка множества протоколов;
* гибкость, плагины, расширения.

Недостатки:

* Платный;
* требует ручной работы;
* нет легкой интеграции в CI/CD без платных надстроек.

Разрабатываемая система автоматизированного тестирования веб-безопасности будет обладать рядом преимуществ по сравнению с существующими аналогами. Система будет объединять DAST, SAST и PoC с помощью headless-браузера. Такой подход позволит находить потенциальные проблемы, подтверждать их воспроизводимость и снижать уровень ложных срабатываний.

Система обеспечит приоритизацию и дедупликацию найденных уязвимостей, формируя удобные отчёты, и будет легко интегрироваться в процессы CI/CD.