**Описание приложения Juice Shop**

OWASP Juice Shop — это веб-приложение с открытым исходным кодом для демонстрации самых распространенных уязвимостей. Поддерживается некоммерческой организацией Open Web Application Security Project (OWASP) и волонтерами. Это уязвимое веб-приложение, разработанное для того, чтобы помочь специалистам по безопасности, разработчикам и студентам изучать и применять на практике концепции безопасности приложений.

**Основные характеристики**

* **Тип приложения**: Веб-приложение (SPA на Angular + REST API на Node.js/Express).
* **Цель**: Демонстрация **реальных уязвимостей** в безопасной учебной среде.
* **Уровень сложности**: От новичка до эксперта (включает >100 уязвимостей).
* **Лицензия**: MIT (открытый исходный код).

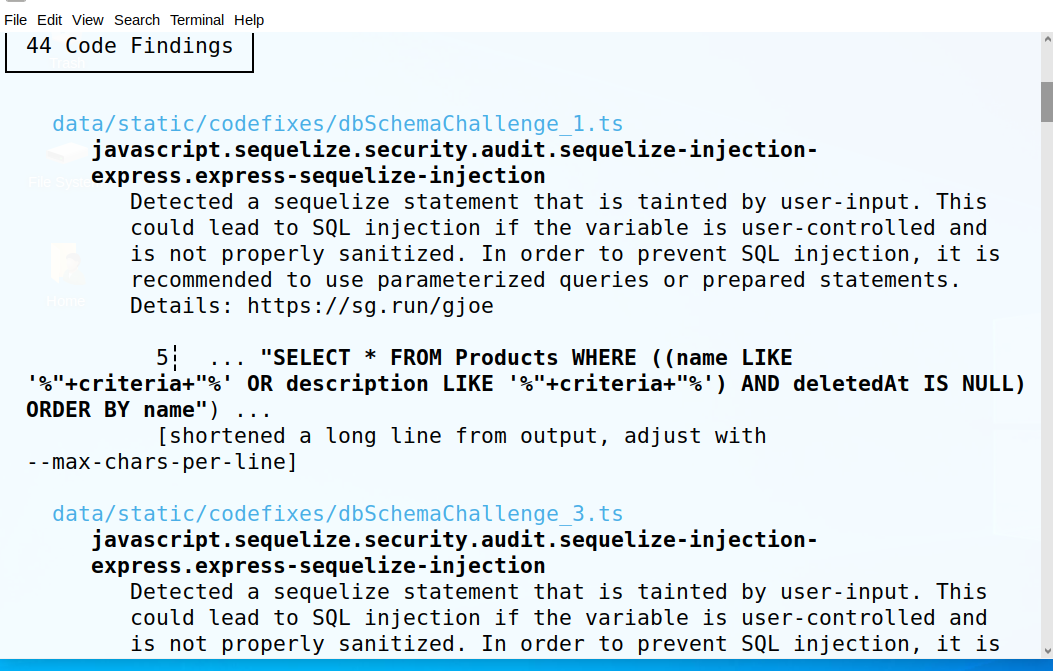
**Ключевые особенности**

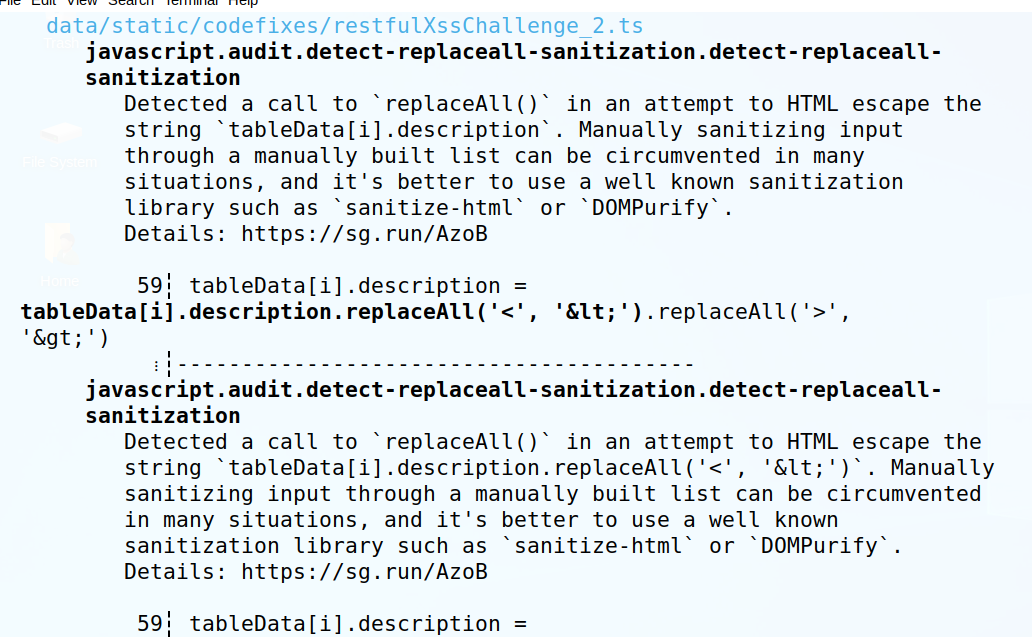
1. **Реалистичный функционал**
   * Интернет-магазин соков (каталог, корзина, отзывы, аккаунты).
   * Админ-панель, платежи, восстановление пароля.
2. **Разнообразие уязвимостей**
   * **OWASP Top 10**: SQL-инъекции, XSS, CSRF, небезопасные API, JWT-уязвимости.
   * **Сложные сценарии**: RCE, SSRF, десериализация, логические баги.
3. **Интерактивное обучение**
   * **Score Board**: Доска заданий с подсказками (например: "Логинься как администратор").
   * **Hacking Instructor**: Встроенный гид для новичков.
4. **Интеграция с инструментами**
   * Поддержка **Burp Suite**, **Semgrep**, **ZAP**, **Nmap**.
   * REST API для автоматизированного тестирования.

**Для кого предназначен?**

* **Пентестеры**: Тренировка перед реальными аудитами.
* **Разработчики**: Понимание, как **нельзя** писать код.
* **Студенты**: Основы кибербезопасности в игровой форме.
* **CTF-команды**: Подготовка к соревнованиям.

**Результаты статического анализа**





data/static/codefixes/dbSchemaChallenge\_1.ts

**javascript.sequelize.security.audit.sequelize-injection-**

**express.express-sequelize-injection**

data/static/codefixes/dbSchemaChallenge\_3.ts

**javascript.sequelize.security.audit.sequelize-injection-**

**express.express-sequelize-injection**

data/static/codefixes/restfulXssChallenge\_2.ts

**javascript.audit.detect-replaceall-sanitization.detect-replaceall-**

**sanitization**

**javascript.audit.detect-replaceall-sanitization.detect-replaceall-**

**sanitization**

data/static/codefixes/unionSqlInjectionChallenge\_1.ts

**javascript.sequelize.security.audit.sequelize-injection-**

**express.express-sequelize-injection**

data/static/codefixes/unionSqlInjectionChallenge\_3.ts

**javascript.sequelize.security.audit.sequelize-injection-**

**express.express-sequelize-injection**

data/static/users.yml

**generic.secrets.security.detected-generic-secret.detected-generic-**

**secret**

data/staticData.ts

**javascript.lang.security.audit.path-traversal.path-join-resolve-**

**traversal.path-join-resolve-traversal**

frontend/src/app/app.guard.spec.ts

**generic.secrets.security.detected-jwt-token.detected-jwt-token**

frontend/src/app/last-login-ip/last-login-ip.component.spec.ts

**generic.secrets.security.detected-jwt-token.detected-jwt-token**

frontend/src/hacking-instructor/helpers/helpers.ts

**javascript.lang.security.audit.prototype-pollution.prototype-pollution- loop.prototype-pollution-loop**

frontend/src/hacking-instructor/index.ts

**javascript.browser.security.insecure-document-method.insecure-document- method**

frontend/src/index.html

**html.security.audit.missing-integrity.missing-integrity**

routes/b2bOrder.ts

**javascript.express.security.audit.express-detect-notevil-usage.express- detect-notevil-usage**

routes/captcha.ts

**javascript.browser.security.eval-detected.eval-detected**

routes/chatbot.ts

**javascript.express.security.injection.raw-html-format.raw-html-format**

routes/dataErasure.ts

**javascript.express.security.audit.express-path-join-resolve-**

**traversal.express-path-join-resolve-traversal**

**javascript.lang.security.audit.path-traversal.path-join-resolve-**

**traversal.path-join-resolve-traversal**

routes/fileServer.ts

**javascript.express.security.audit.express-res-sendfile.express-res- sendfile**

**javascript.lang.security.audit.path-traversal.path-join-resolve-**

**traversal.path-join-resolve-traversal**

routes/fileUpload.ts

**javascript.lang.security.audit.path-traversal.path-join-resolve-**

**traversal.path-join-resolve-traversal**

**javascript.express.security.audit.express-libxml-vm-noent.express-libxml-vm-noent**

routes/keyServer.ts

**javascript.express.security.audit.express-res-sendfile.express-res-sendfile**

**javascript.express.security.audit.express-path-join-resolve-traversal.express-path-join-resolve-traversal**

**javascript.lang.security.audit.path-traversal.path-join-resolve-traversal.path-join-resolve-traversal**

routes/logfileServer.ts

**javascript.express.security.audit.express-res-sendfile.express-res- sendfile**

**javascript.express.security.audit.express-path-join-resolve-traversal.express-path-join-resolve-traversal**

**javascript.lang.security.audit.path-traversal.path-join-resolve-traversal.path-join-resolve-traversal**

routes/login.ts

**javascript.sequelize.security.audit.sequelize-injection-express.express-sequelize-injection**

routes/order.ts

**javascript.lang.security.audit.path-traversal.path-join-resolve-traversal.path-join-resolve-traversal**

routes/profileImageUrlUpload.ts

**javascript.express.security.audit.express-ssrf.express-ssrf**

routes/quarantineServer.ts

**javascript.express.security.audit.express-res-sendfile.express-res- sendfile**

**javascript.express.security.audit.express-path-join-resolve-traversal.express-path-join-resolve-traversal**

**javascript.lang.security.audit.path-traversal.path-join-resolve-traversal.path-join-resolve-traversal**

routes/redirect.ts

**javascript.express.security.audit.express-open-redirect.express-open-redirect**

routes/search.ts

**javascript.sequelize.security.audit.sequelize-injection-express.express-sequelize-injection**

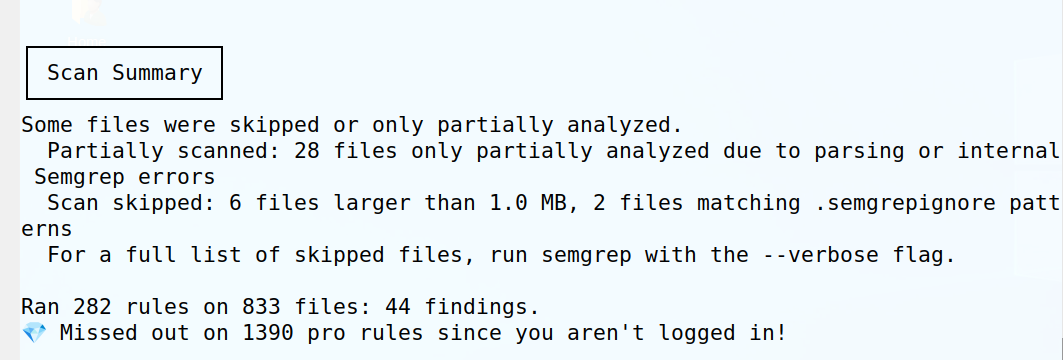
routes/userProfile.ts

**javascript.browser.security.eval-detected.eval-detected**

**javascript.express.security.express-insecure-template-usage.express-insecure-template-usage**

routes/videoHandler.ts

**javascript.lang.security.audit.unknown-value-with-script-tag.unknown-value-with-script-tag**

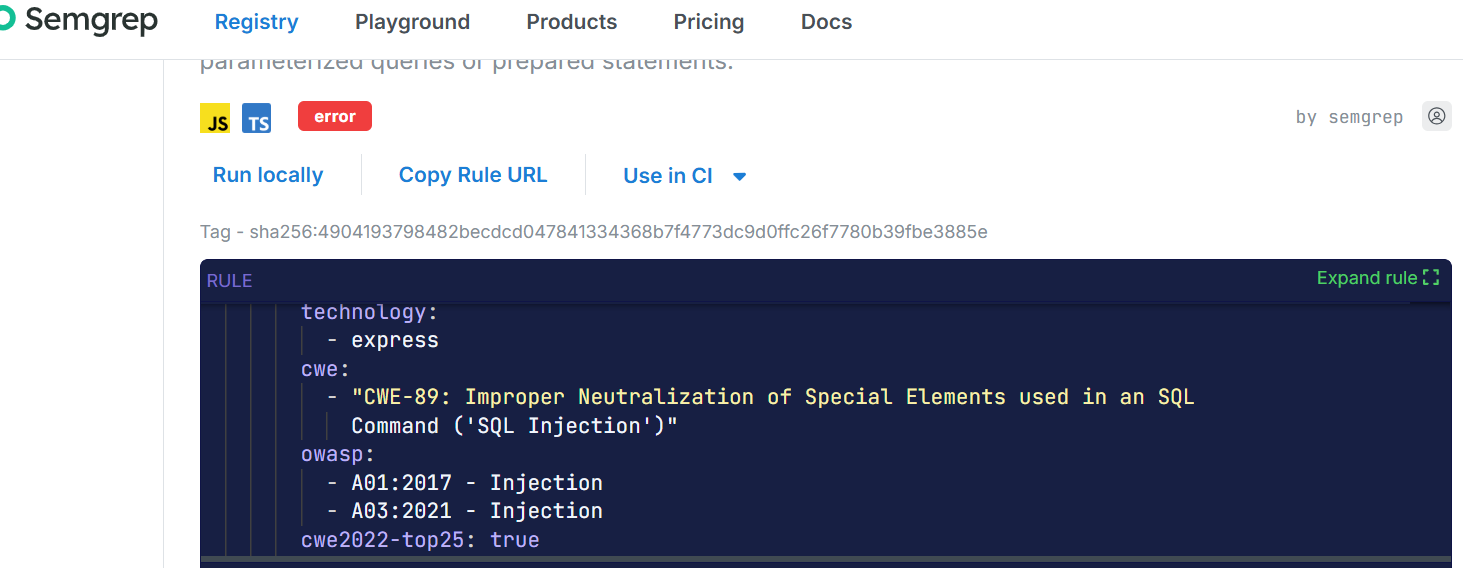


**Уязвимости из OWASP Top-10 со скриншотами.**

data/static/codefixes/dbSchemaChallenge\_1.ts

**javascript.sequelize.security.audit.sequelize-injection-**

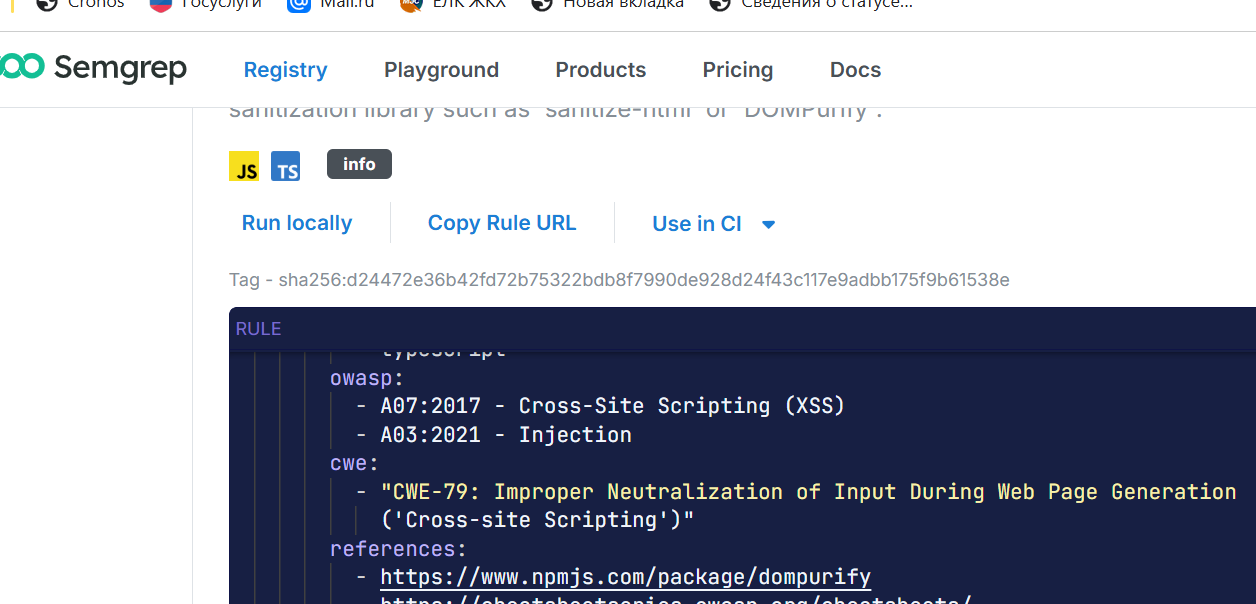
**express.express-sequelize-injection**

****

data/static/codefixes/restfulXssChallenge\_2.ts

**javascript.audit.detect-replaceall-sanitization.detect-replaceall-**

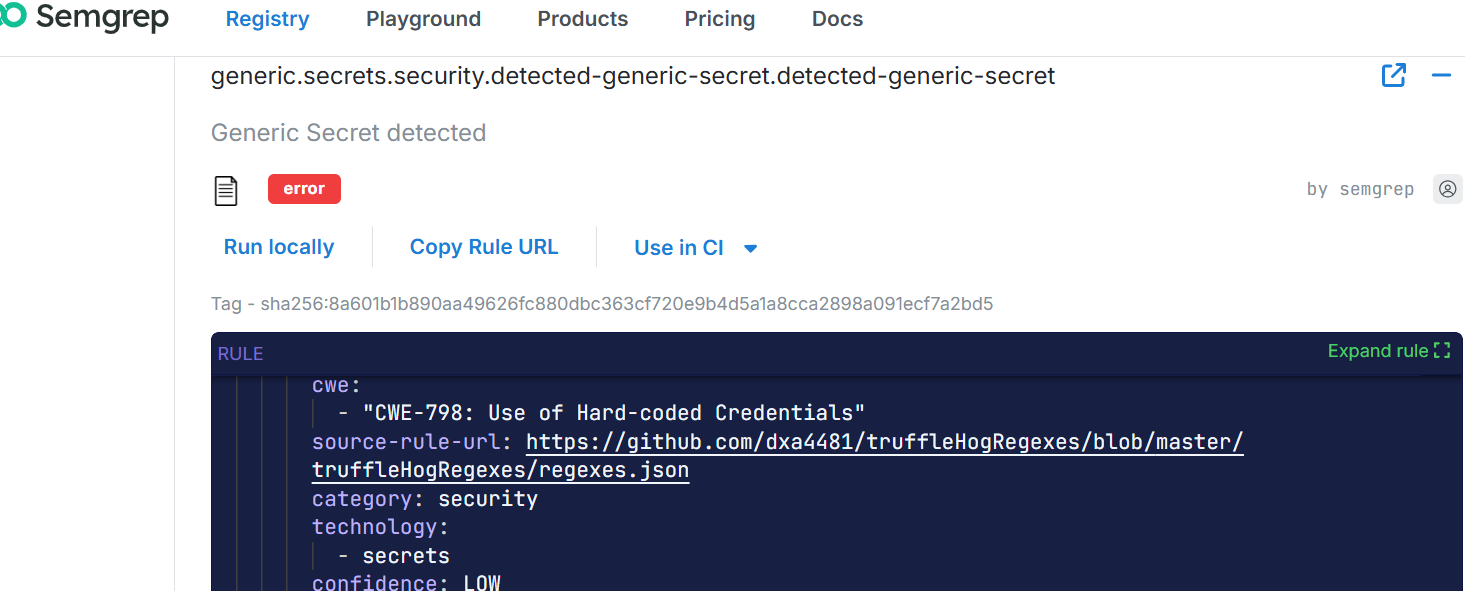
**sanitization**



data/static/users.yml

**generic.secrets.security.detected-generic-secret.detected-generic-**

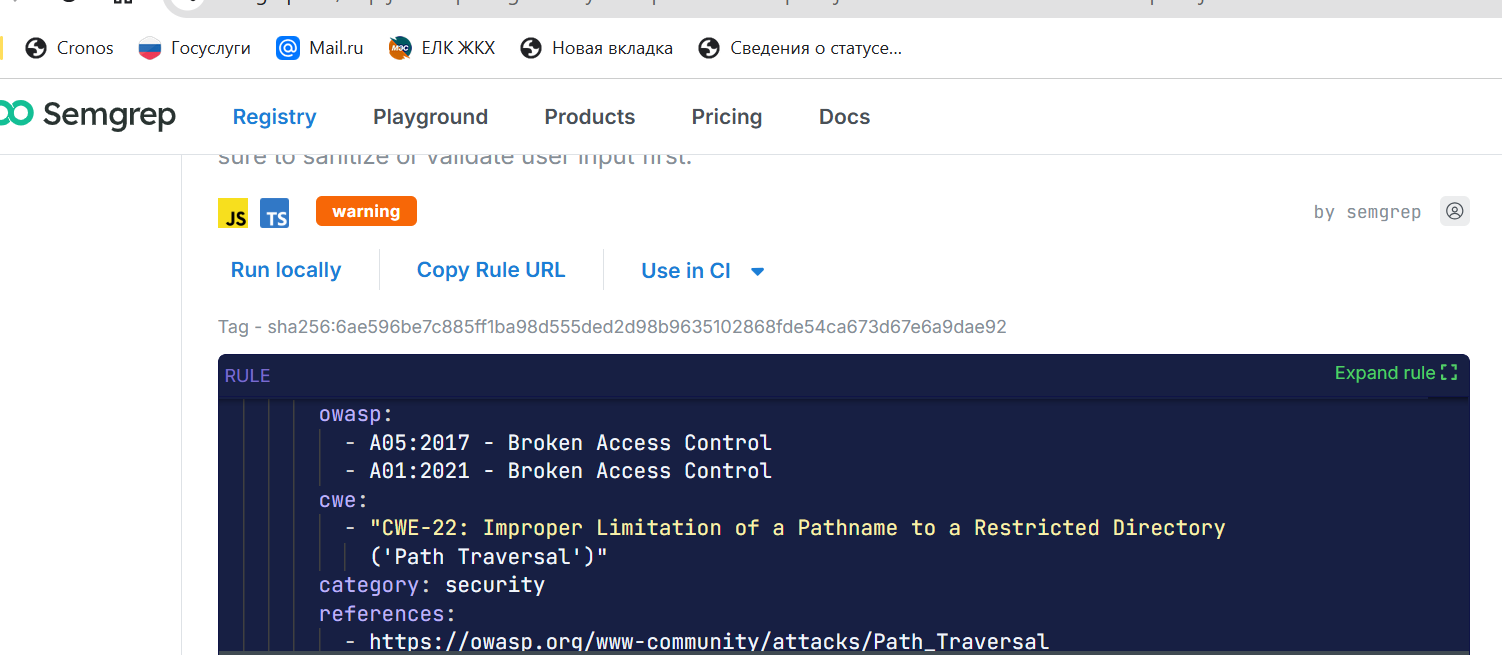
**secret**

****

data/staticData.ts

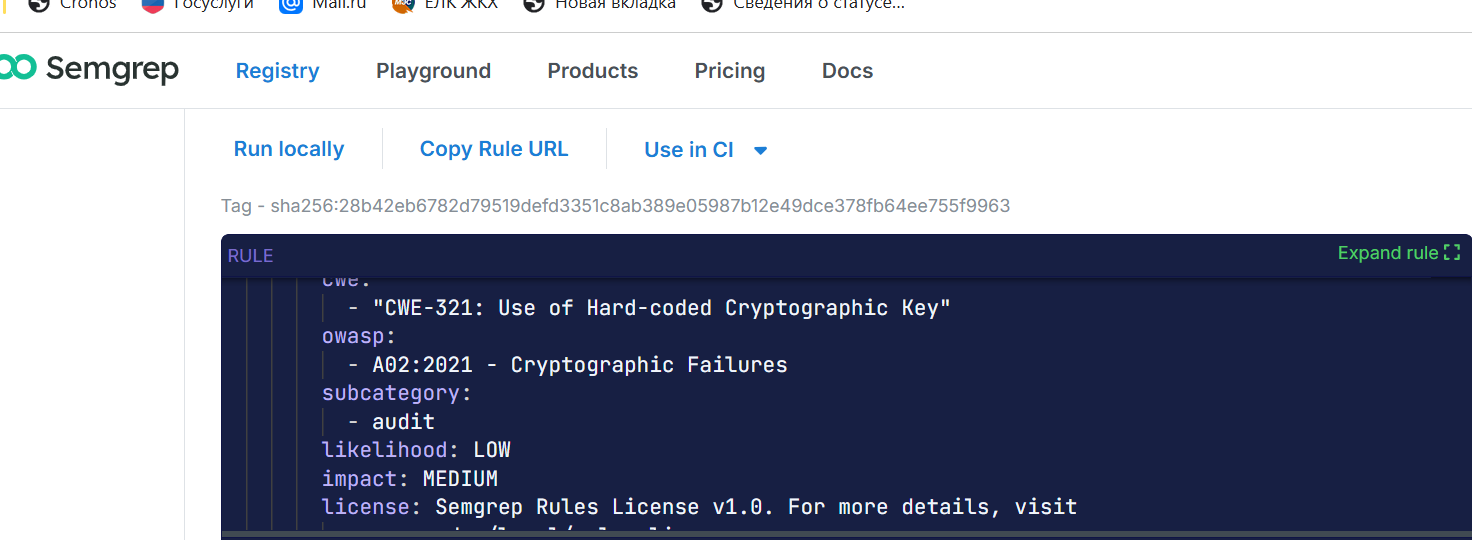
**javascript.lang.security.audit.path-traversal.path-join-resolve-**

**traversal.path-join-resolve-traversal**



frontend/src/app/app.guard.spec.ts

**generic.secrets.security.detected-jwt-token.detected-jwt-token**

****

**Демонстрация эксплуатации трёх уязвимостей из OWASP Top-10.**

CWE-89 (SQL Injection)

Цель: обойти аутентификацию SQL-инъекцией.

Шаги:

1. Переходим на http://localhost:3000/#/login

2. Вводим:

Email: ' OR 1=1--

Password: vasja

3. В Burp Proxy перехватываем запрос POST /rest/user/login

4. Отправляем в Repeater:

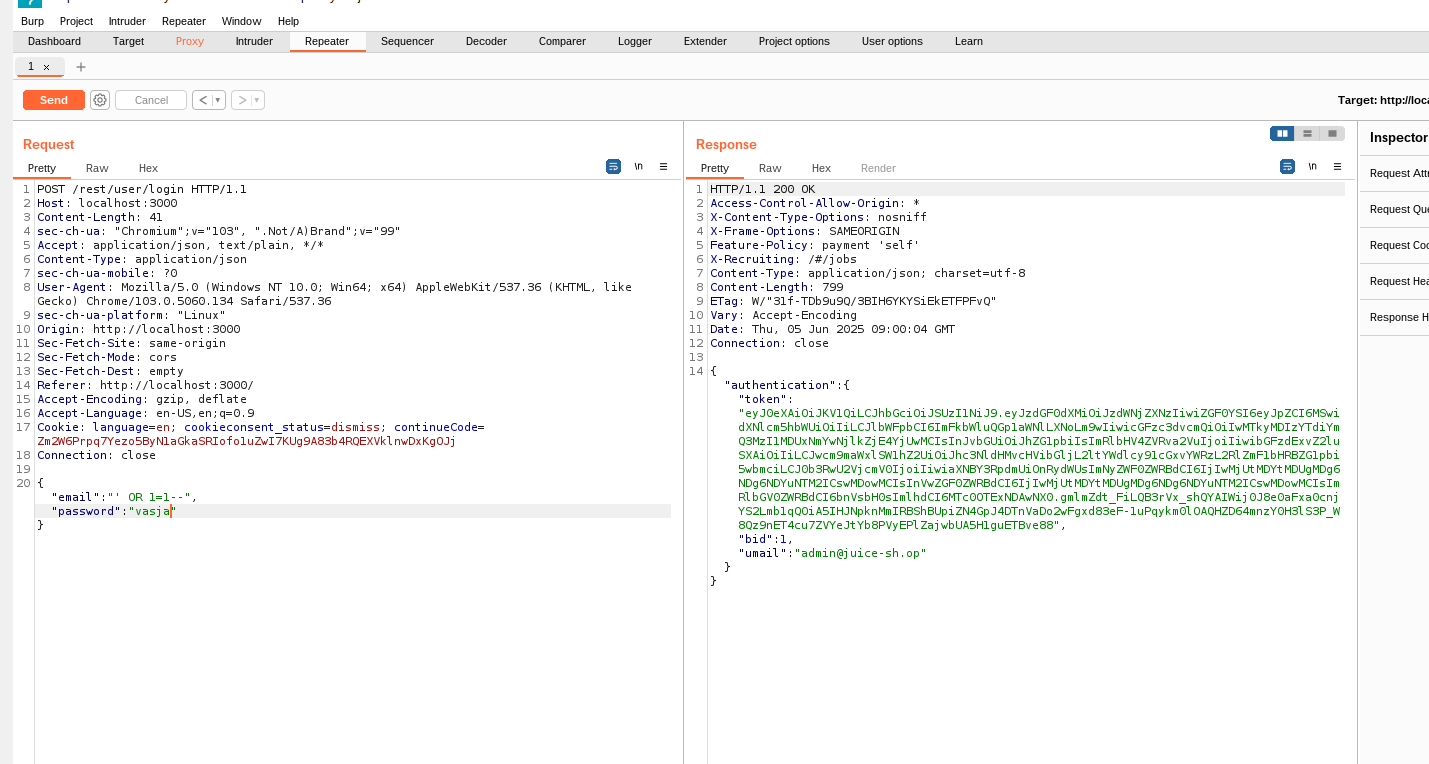
{

"email": "' OR 1=1--",

"password": "vasja"

}

5. Нажимаем Send → получаем токен без корректного пароля. Уязвимость подтверждена.



CWE-321: Use of Hard-coded Cryptographic Key

Цель: декодировать JWT с предсказуемым/жестко заданным ключом

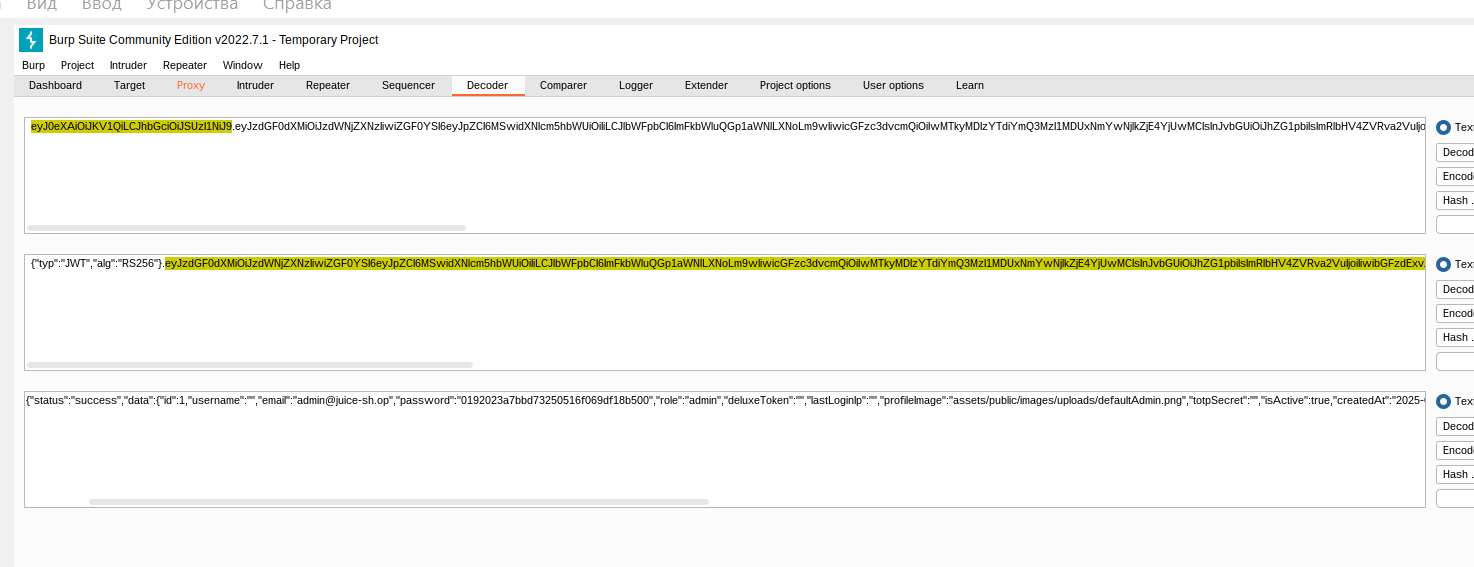
Шаги:

1. Используем токе, полученный в предыдущей эксплуатации.

2. Копируем JWT (из authentication.token)

3. Отправим его в Burp Decoder.

4. После декодирования получаем логин и пароль. Уязвимость подтверждена.

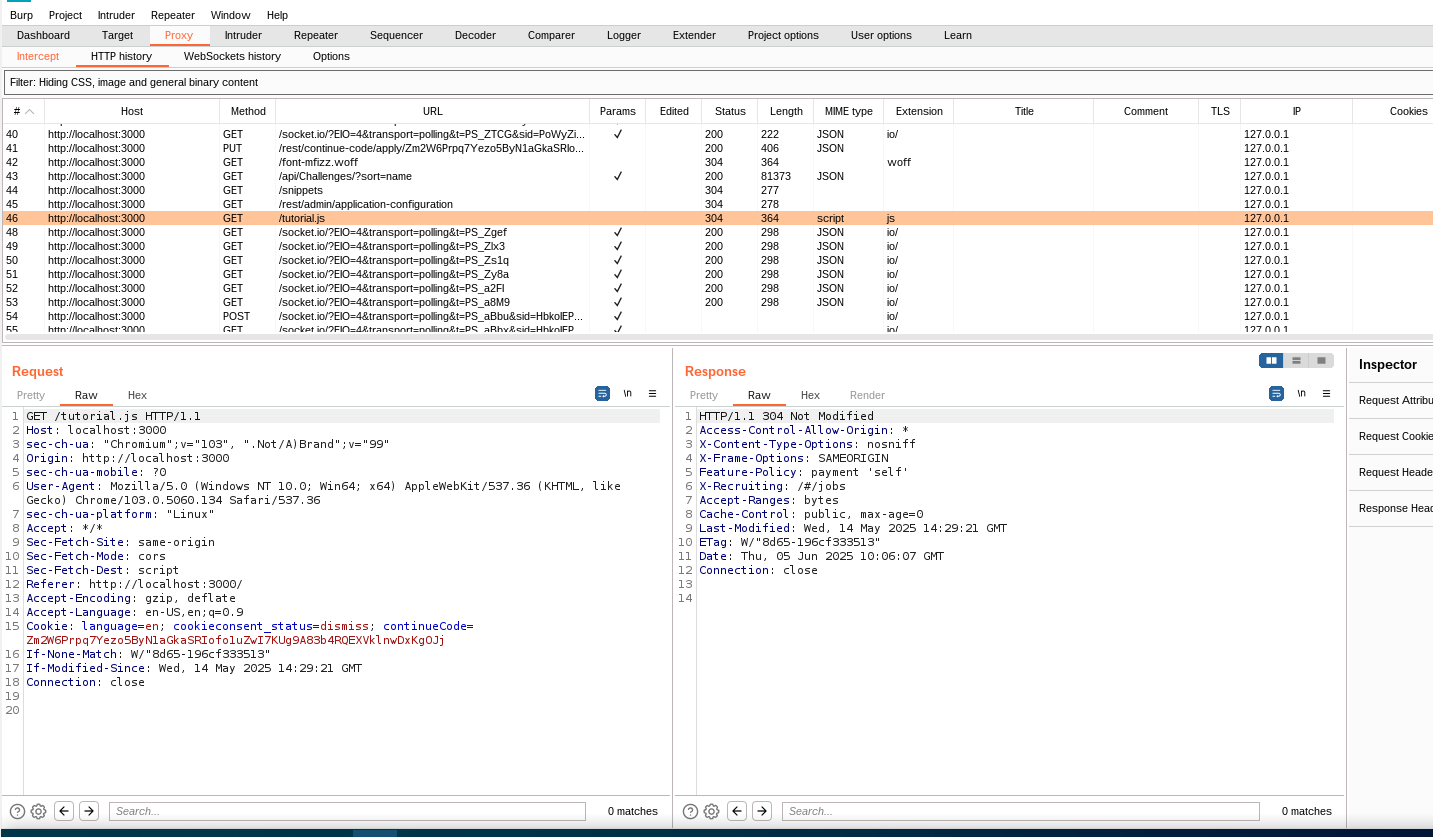


CWE-22: Path Traversal

Цель: кража конфиденциальных данных.

Шаги:

1. Перехватываем запрос файла:

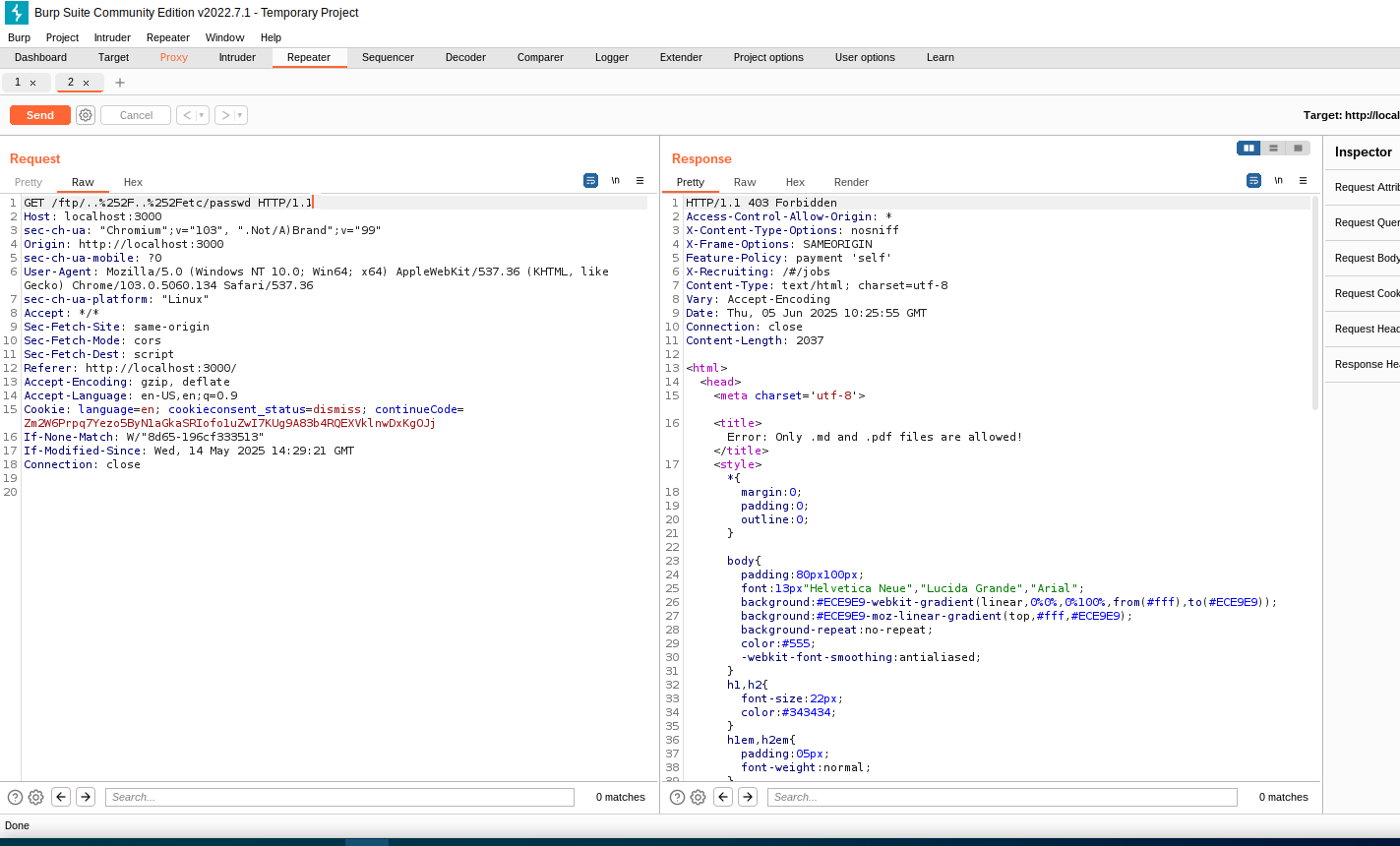


2. Отправляем его в Repitor.

3. Изменяем запрос, добавив . . / с двойным кодированием:

GET /ftp/..%252F..%252Fetc/passwd HTTP/1.1

4. Отправляем измененный запрос, получаем содержимое файла. Уязвимость подтверждена.



**Рекомендации по устранению к трём продемонстрированным уязвимостям.**

1. Injection - CWE-89

MITRE ATT&CK: T1190

- Использовать параметризованные запросы (prepared statements)

- Валидация входных данных

Ввести запрет на спецсимволы (', ", ;, --).

Применять белые списки (разрешённые символы).

- Отключить вывод SQL-ошибок

- Минимизация прав БД

Запускать СУБД от пользователя с ограниченными правами (не root).

- WAF (Web Application Firewall)

Настроить правила для блокировки UNION SELECT, OR 1=1 и т.д.

2. Cryptographic Failures - CWE-321

MITRE ATT&CK: T1552.001

- Исключить использование жёстко заданных ключей

- Использовать секреты из защищённого хранилища

- Регулярно менять ключи

Автоматическая смена каждые 30-90 дней.

3. Path Traversal - CWE-22:

MITRE ATT&CK: T1083

- Нормализация путей

- Белые списки файлов

- Запрет символов ../

location ~\* \.(\.\./|%2e%2e%2f) { deny all; }

- Хранить файлы вне корня веб-сервера