**Раздел:** Анализ защищённости ПО

**Практическое задание:** Анализ защищённости веб-приложений

**Выполнил:** Василий Костюков

**Задание:** Проведите анализ защищённости уже знакомого вам приложения Juice Shop. В отчёте укажите только уязвимости из OWASP Top-10; CWE, которые их вызывают; а также рекомендации по их устранению.

Отчёт должен иметь следующую структуру:

1. Введение — описание приложения.
2. Результаты статического анализа — общие, можно без детализации.
3. Уязвимости из OWASP Top-10, обнаруженные в результате статического анализа, — минимум пять штук.
4. Демонстрация эксплуатации трёх уязвимостей из OWASP Top-10 — скриншоты эксплуатации, проведённой с помощью инструмента Burp Suite. Выбор этих уязвимостей остаётся на ваше усмотрение.
5. Рекомендации по устранению к трём продемонстрированным уязвимостям — можно взять основу [с сайта MITRE ATT&CK](https://attack.mitre.org/) под найденные CWE.

В отчёте обязательно должны присутствовать:

1. Скриншоты (минимум два) или выгрузка результатов статического сканирования в пункте 2.
2. Список уязвимостей из OWASP Top-10 с доказательствами в пункте 3 (в виде скриншотов, найденных анализатором уязвимостей; отдельно от остальных уязвимостей).
3. Скриншоты из Burp Suite в пункте 4, а также обязательно текстовое описание эксплуатации: что необходимо сделать, чтобы воспроизвести уязвимость.

Не обязательно выбирать три разные уязвимости. Вы можете выбрать три вариации одной и той же уязвимости.

Итоговый отчёт необходимо загрузить в формате DOCX или PDF на свой гит. Приложите ссылку на отчёт в форму для ответа.

**Выполнение задания:**

1. Введение

Для начала приведем список топ-10 уязвимостей с сайта OWASP:

**Top 10 Web Application Security Risks**

There are three new categories, four categories with naming and scoping changes, and some consolidation in the Top 10 for 2021.

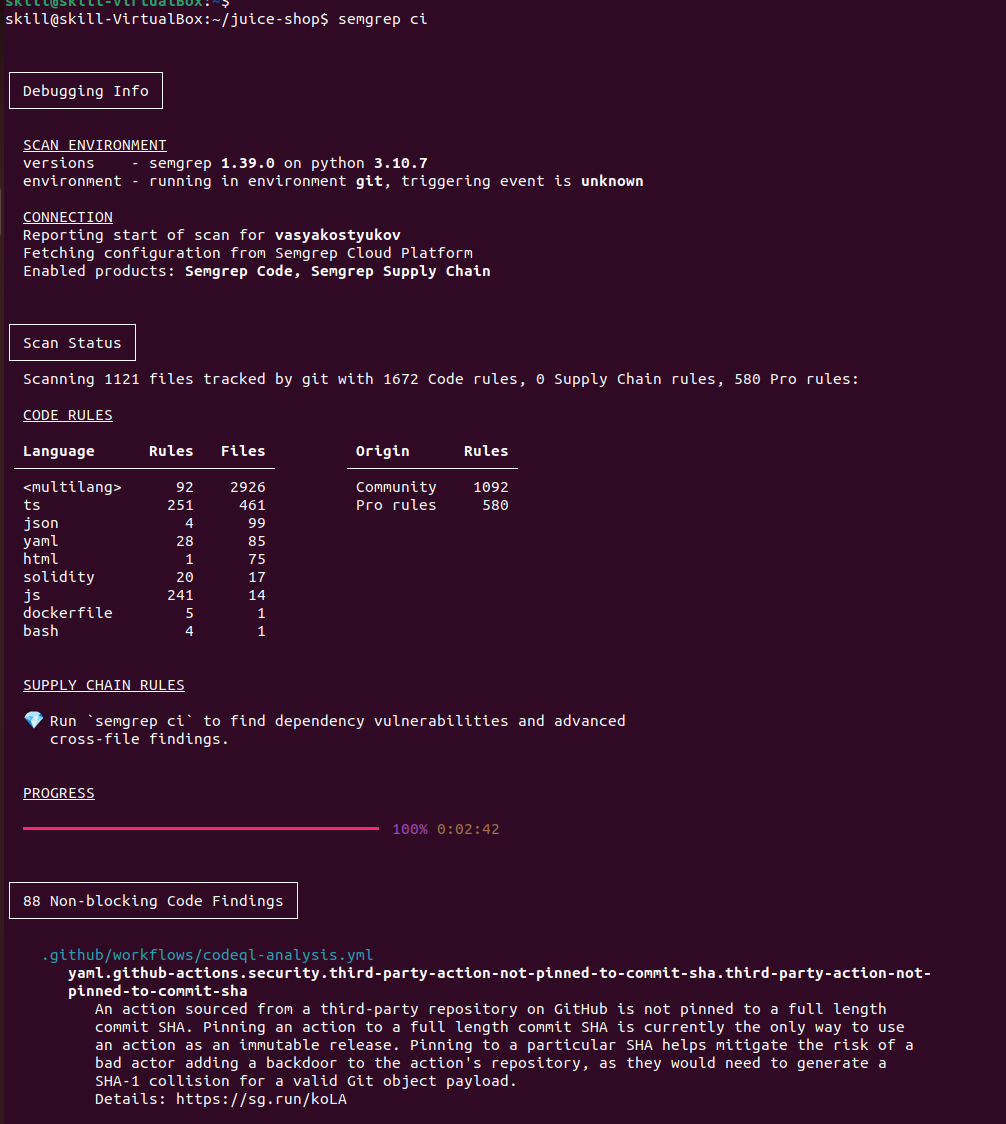


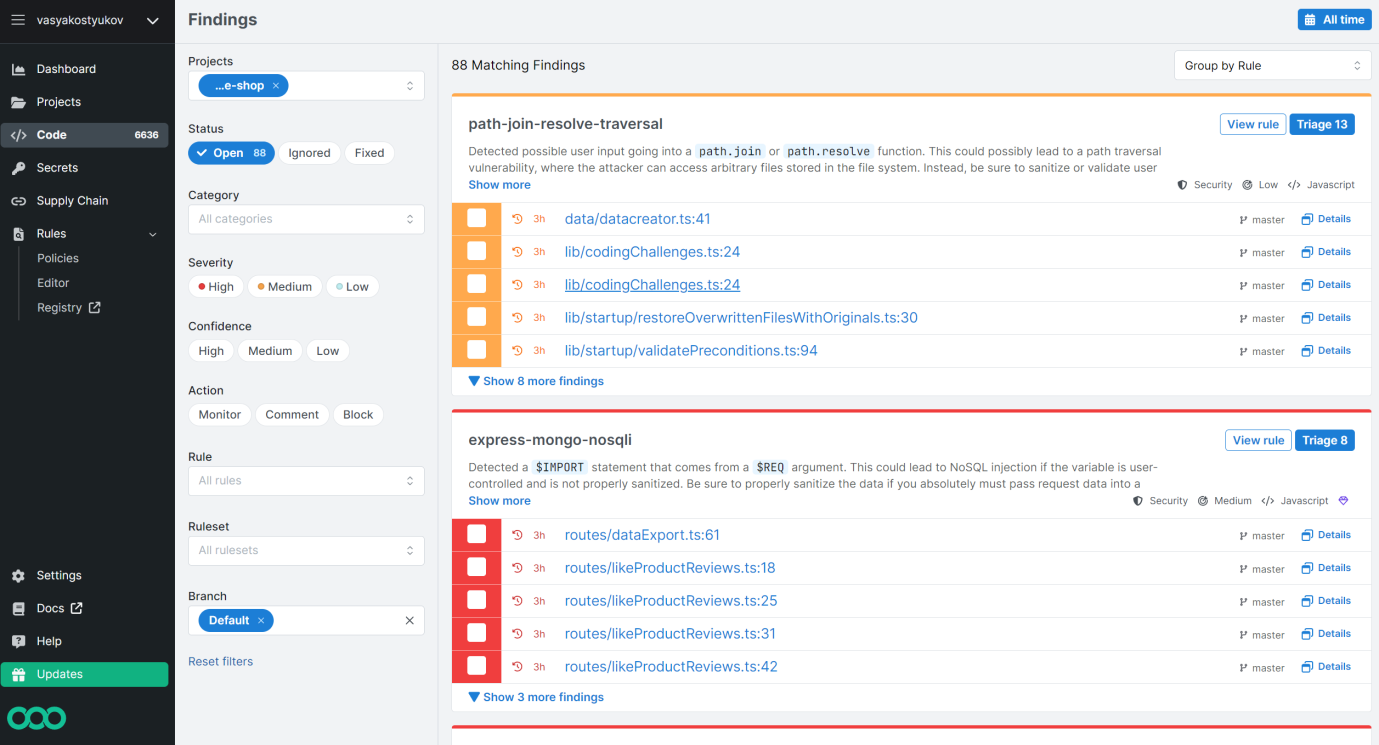
* [**A01:2021-Broken Access Control**](https://owasp.org/Top10/A01_2021-Broken_Access_Control/) moves up from the fifth position; 94% of applications were tested for some form of broken access control. The 34 Common Weakness Enumerations (CWEs) mapped to Broken Access Control had more occurrences in applications than any other category.
* [**A02:2021-Cryptographic Failures**](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/) shifts up one position to #2, previously known as Sensitive Data Exposure, which was broad symptom rather than a root cause. The renewed focus here is on failures related to cryptography which often leads to sensitive data exposure or system compromise.
* [**A03:2021-Injection**](https://owasp.org/Top10/A03_2021-Injection/) slides down to the third position. 94% of the applications were tested for some form of injection, and the 33 CWEs mapped into this category have the second most occurrences in applications. Cross-site Scripting is now part of this category in this edition.
* [**A04:2021-Insecure Design**](https://owasp.org/Top10/A04_2021-Insecure_Design/) is a new category for 2021, with a focus on risks related to design flaws. If we genuinely want to “move left” as an industry, it calls for more use of threat modeling, secure design patterns and principles, and reference architectures.
* [**A05:2021-Security Misconfiguration**](https://owasp.org/Top10/A05_2021-Security_Misconfiguration/) moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it’s not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category.
* [**A06:2021-Vulnerable and Outdated Components**](https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/) was previously titled Using Components with Known Vulnerabilities and is #2 in the Top 10 community survey, but also had enough data to make the Top 10 via data analysis. This category moves up from #9 in 2017 and is a known issue that we struggle to test and assess risk. It is the only category not to have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploit and impact weights of 5.0 are factored into their scores.
* [**A07:2021-Identification and Authentication Failures**](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/) was previously Broken Authentication and is sliding down from the second position, and now includes CWEs that are more related to identification failures. This category is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping.
* [**A08:2021-Software and Data Integrity Failures**](https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/) is a new category for 2021, focusing on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity. One of the highest weighted impacts from Common Vulnerability and Exposures/Common Vulnerability Scoring System (CVE/CVSS) data mapped to the 10 CWEs in this category. Insecure Deserialization from 2017 is now a part of this larger category.
* [**A09:2021-Security Logging and Monitoring Failures**](https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/) was previously Insufficient Logging & Monitoring and is added from the industry survey (#3), moving up from #10 previously. This category is expanded to include more types of failures, is challenging to test for, and isn’t well represented in the CVE/CVSS data. However, failures in this category can directly impact visibility, incident alerting, and forensics.
* [**A10:2021-Server-Side Request Forgery**](https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/) is added from the Top 10 community survey (#1). The data shows a relatively low incidence rate with above average testing coverage, along with above-average ratings for Exploit and Impact potential. This category represents the scenario where the security community members are telling us this is important, even though it’s not illustrated in the data at this time.

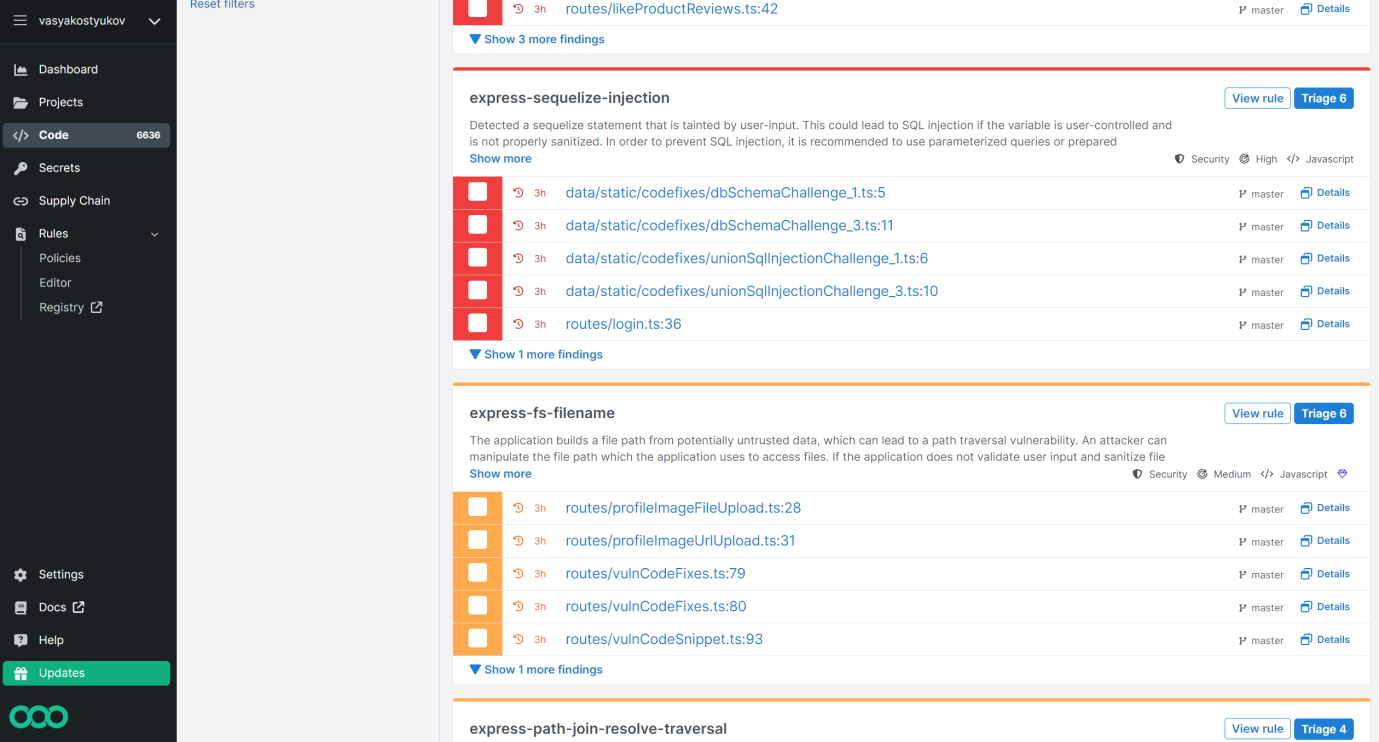
Теперь, когда мы знаем основные уязвимости, перейдем к объекту исследования. Мы будет исследовать приложение OWASP Joise Shop. OWASP Juice Shop - это, пожалуй, самое современное и изощренное небезопасное веб-приложение. Его можно использовать на тренингах по безопасности, ознакомительных демонстрациях, CTF и в качестве подопытного кролика для инструментов безопасности. Juice Shop включает в себя уязвимости из всей первой десятки OWASP наряду со многими другими недостатками безопасности, обнаруженными в реальных приложениях.

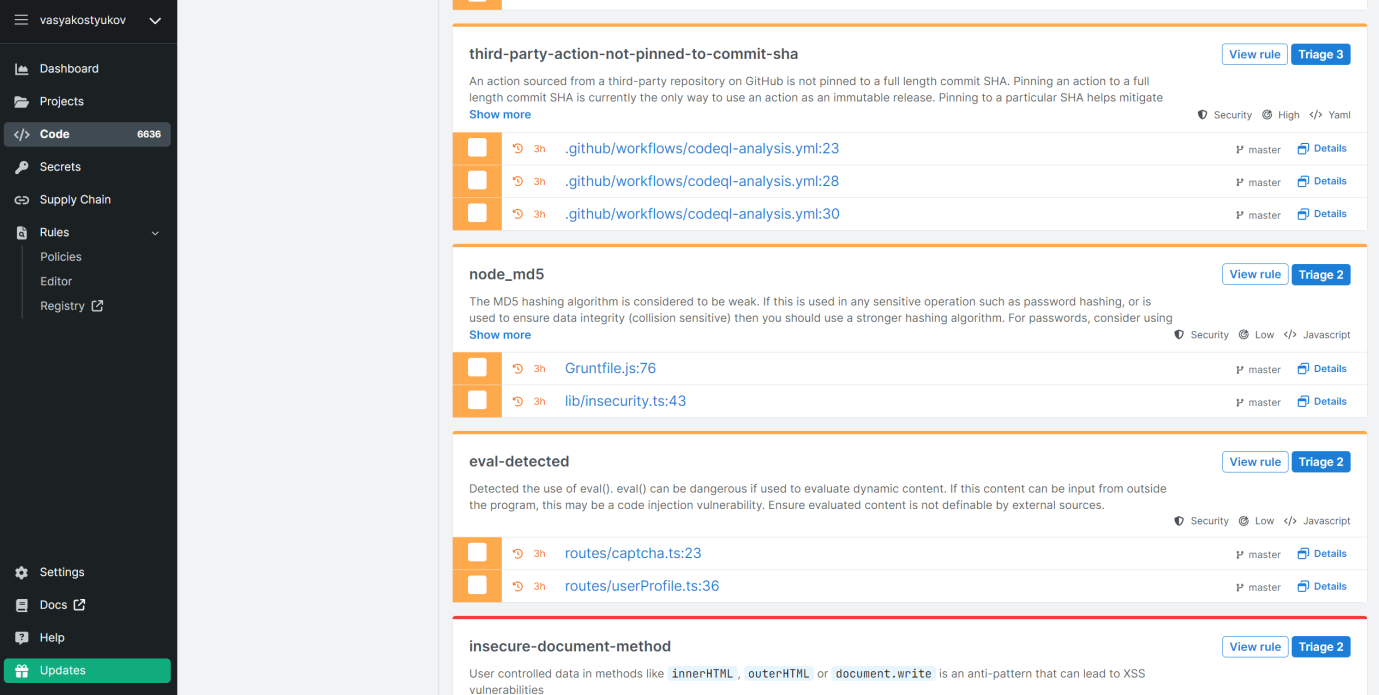
1. **Результаты статического анализа**

С помощью статического анализатора Semgrep было обнаружено 88 уязвимостей. Ниже приведены скриншоты с некоторыми из них.









1. Уязвимости из OWASP Top-10, обнаруженные в результате статического анализа Juicy Shop.

3.1. A01\_2021-Broken\_Access\_Control (Forged Feedback).

3.2. A01\_2021-Broken\_Access\_Control (Forged Review).

3.3. A01\_2021-Broken\_Access\_Control (Manipulate Basket).

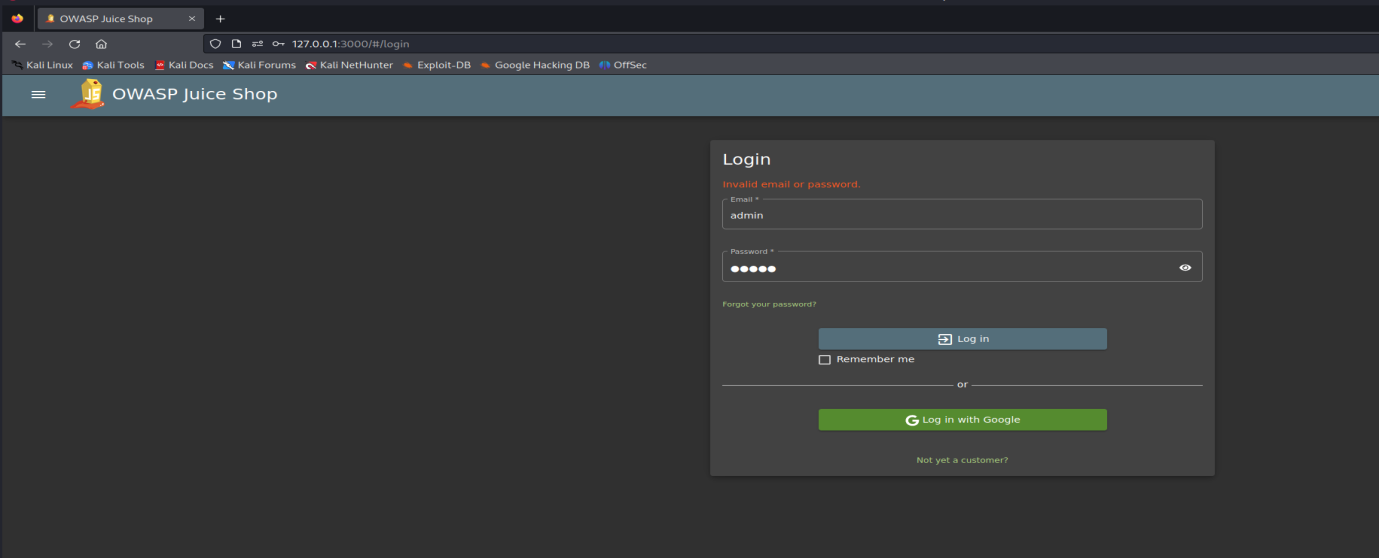
3.4. A03:2021-Injection (Database Schema).

3.5. A03:2021-Injection (Login Admin).

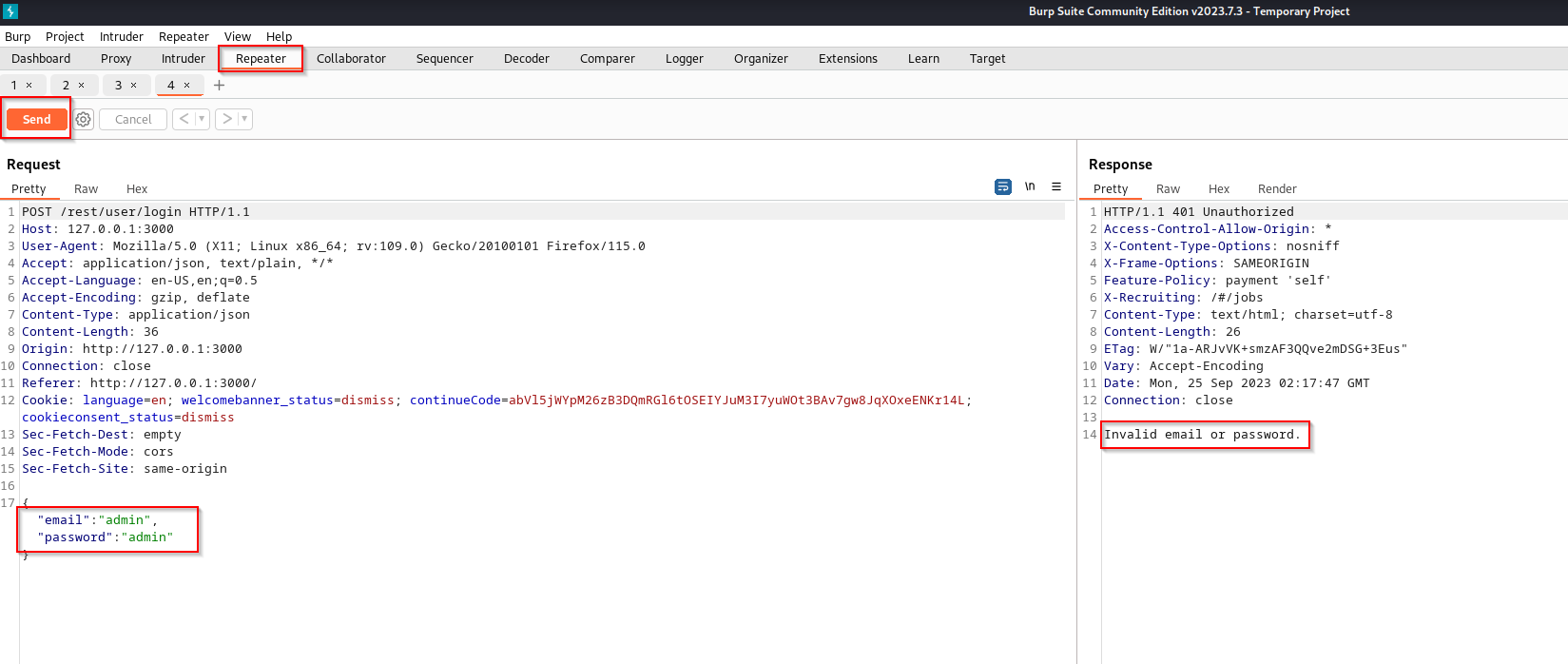
**4. Демонстрация эксплуатации трёх уязвимостей из OWASP Top-10 на установленной Juice Shop.**

**4.1 A03:2021-Injection, Login Admin**

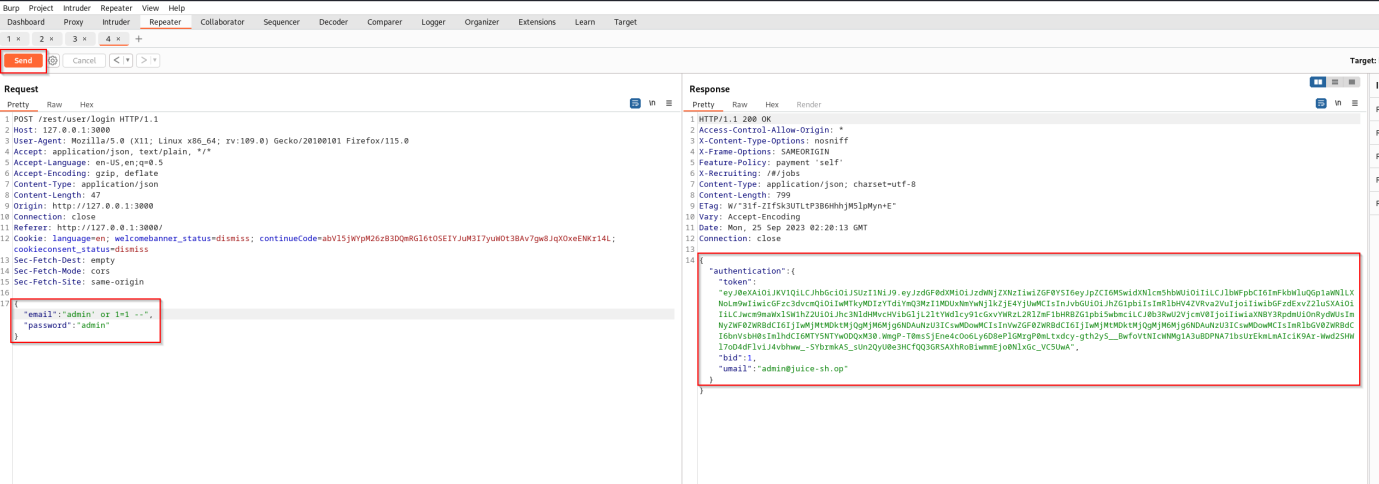
Для начала пробуем вбить самый стандартный вариант плохого логина/пароля для администратора : admin/admin.



Получаем сообщение, что логин или пароль не верны. Далее заходим в программу Burpsuite, Proxy, HTTP History и находим там наш POST запрос с введенными логином и паролем. Пересылаем этот POST запрос в Repeater. Нажимаем кнопку Send, чтобы посмотреть код возвращаемого ответа.

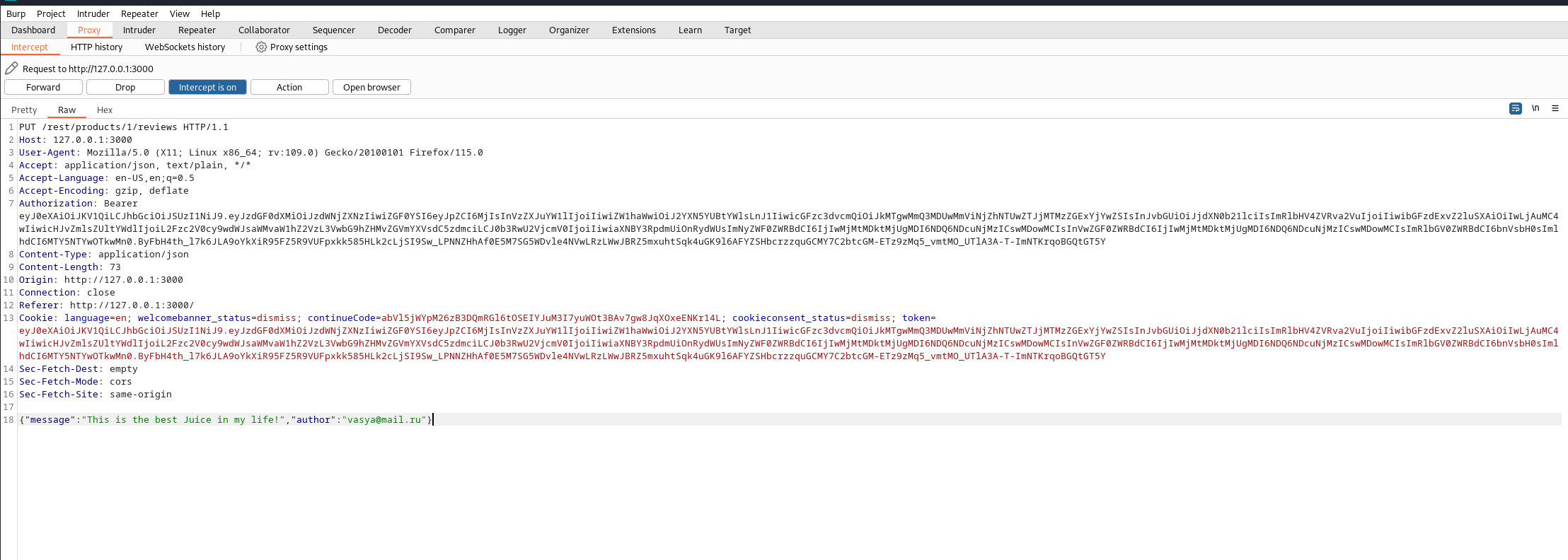


Пробуем провести инъекцию, изменяя поле “email”:”admin” на “email”:”admin’ or 1=1 --“. Нажимаем Send и мы получаем доступ к аккаунту администратора.

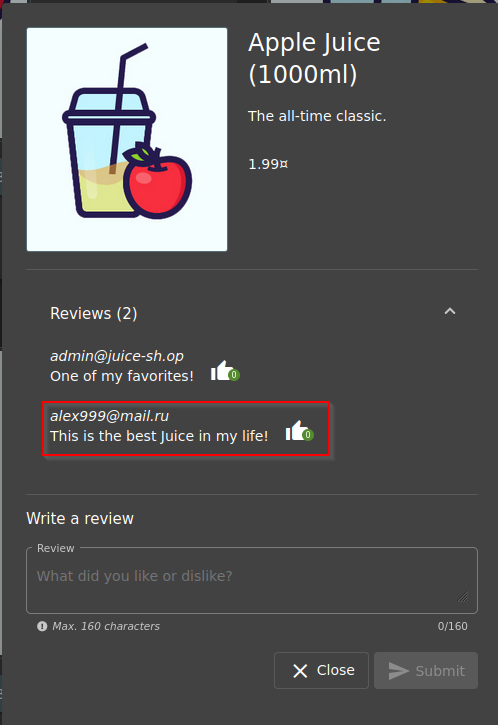


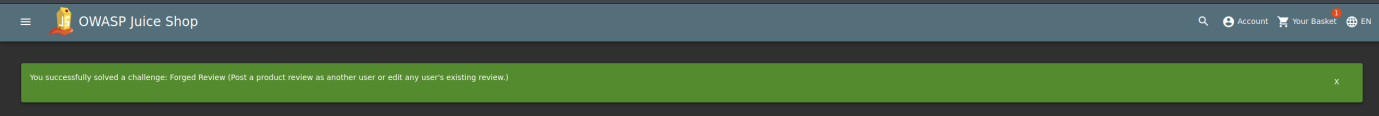
**4.2 A01\_2021-Broken\_Access\_Control, Forged Review**

Я создал обычного пользователся с e-mail [vasya@mail.ru](mailto:vasya@mail.ru) в приложении Joice shop. Включил режим перехвата в Burpsuite. Зашел на страничку товара Apple Joice и оставил там комментарий.



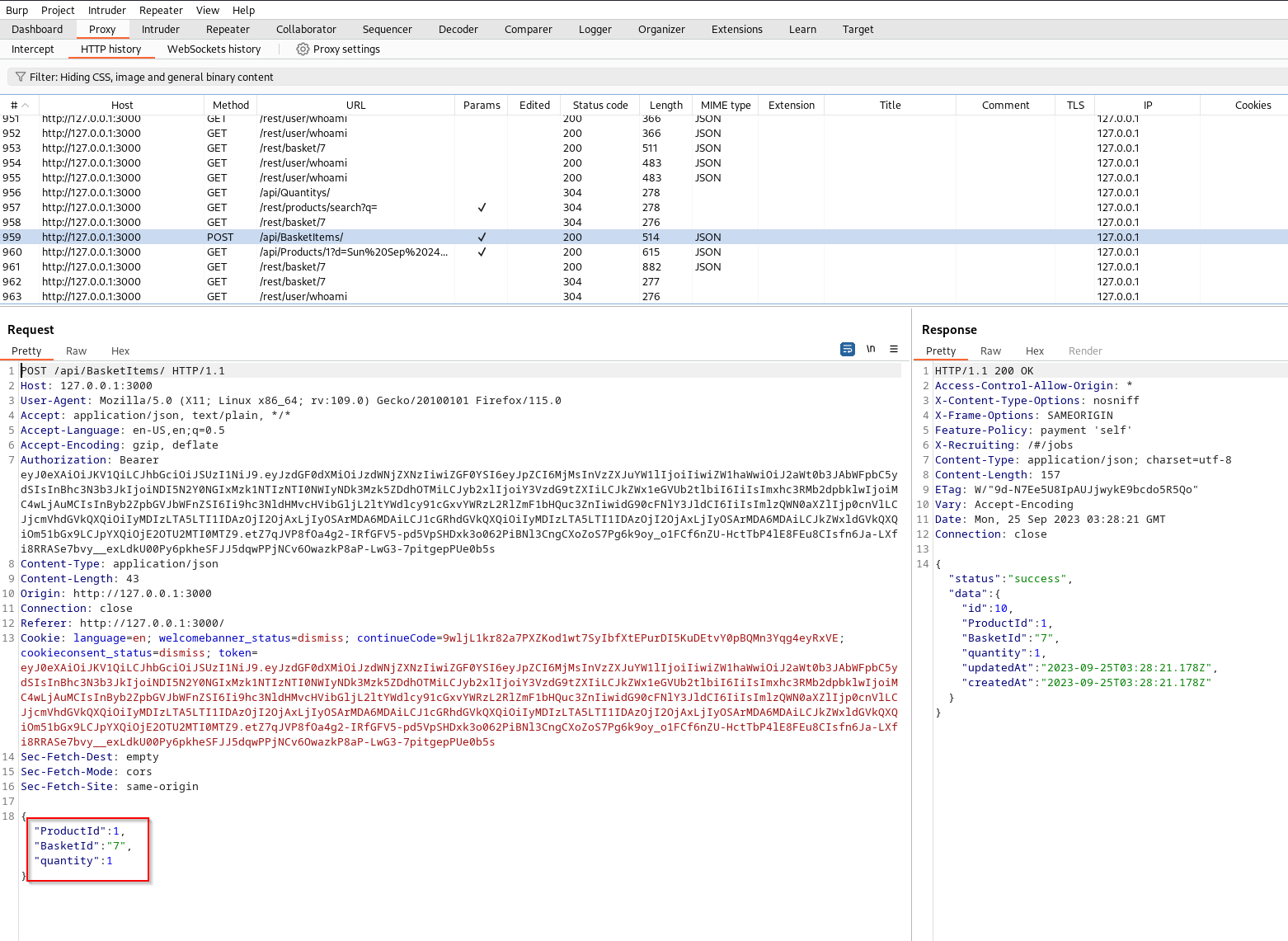
В перехваченном сообщении исправил e-mail на [alex999@mail.ru](mailto:alex999@mail.ru) и отправил сообщение дальше. В результате, мой комментарий был написан от имени [alex999@mail.ru](mailto:alex999@mail.ru).



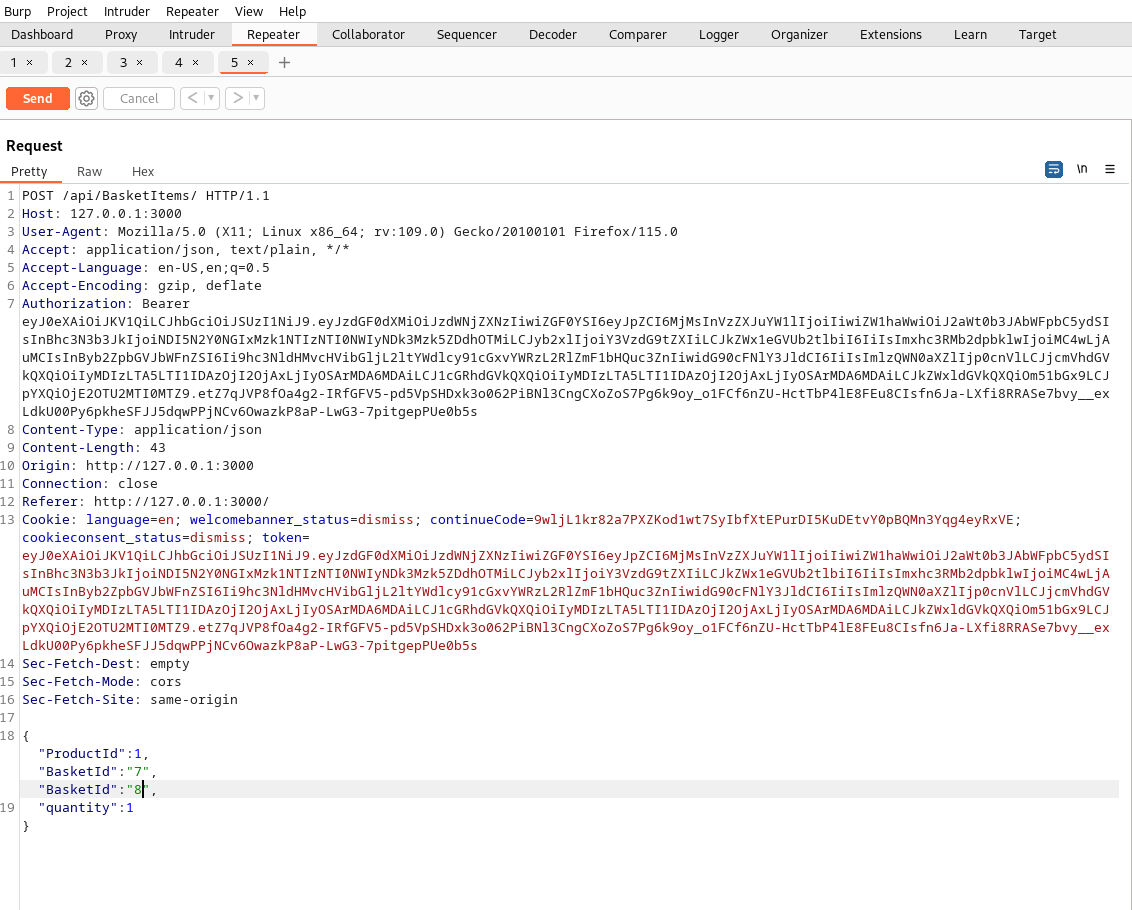


**4.2 A01\_2021-Broken\_Access\_Control, Manipulate Basket**

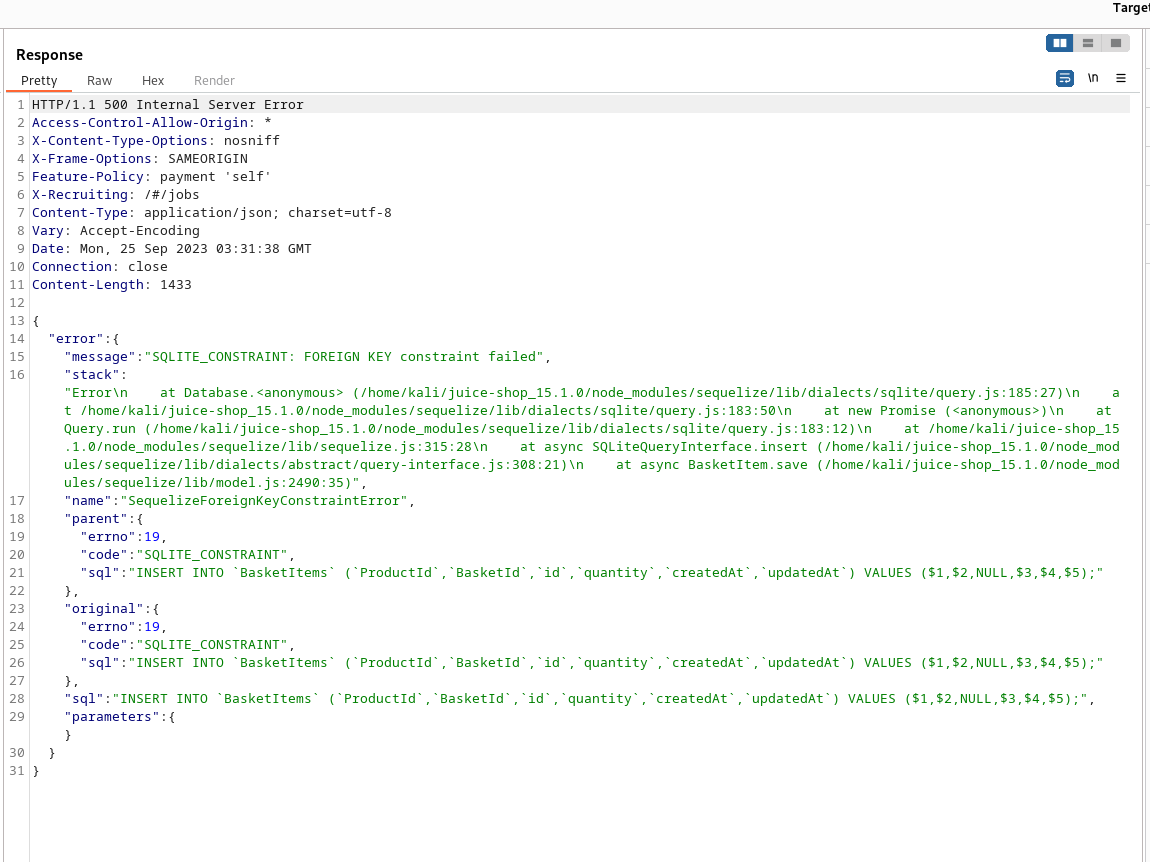
Создаём нового пользователя, логинимся и добавляем какой-нибудь товар в корзину. Затем находим это действие в Burpsuite



Пересылаем его в Repeater и добавляем ещё один BasketId внизу запроса



Отправляем исправленный запрос. В результате наш товар успешно добавился другому пользователю к корзину



C:\Users\Aginor\Documents\task13\16.res2.png

**5. Рекомендации по устранению уязвимостей.**

**5.1. Broken Access Control.**

Контроль доступа как мера эффективен в системе trusted server-side code or server-less API, где атакующий не способен модифицировать проверку access control check или metadata.

- Для всех ресурсов, кроме публичных: deny by default.

- Реализовать механизмы контроля доступа один раз на этапе разработки и использовать их в приложении, вклющающем минимизирование Cross-Origin Resource Sharing (CORS).

- Настройки контроля доступа должны установить разрешения на уровне объектов, не давая пользователю создавать, читать, обновлять или удалять любые записи.

- Уникальные настройки и требования организации должны быть установлены на уровне домена.

- Запретить web server directory listing и убедиться в недоступности метаданных для корневого каталога веб-сервера.

- Логгирование всех попыток неверного входа или использования учётных записей, уведомление администраторов при повторных срабатываниях.

- Установить API limit для минимизирования вреда от автоматизированых атак.

- Идентификаторы сессий должны быть обнулены на сервере после выхода из учётной записи. Stateless JWT tokens должны иметь короткий срок жизни.

- Разработчики и сотрудники QA обязаны тестировать функциональный контроль доступа и модели интеграции элементов.

**5.2. Database Injection.**

Считается, что для предотвращения данного типа атак необходимо использовать следующие два уровня обороны.

1. Параметризацию – где возможно, использовать структурные механизмы, которые обязывают разделятъ данные и команды. Этот механизм обеспечивается соответствующим кодированием строк, использованием кавычек, и т.д.

2. Проверка ввода – значений для команд и относящихся к ним аргументов. Существуют разные подходы к проверке верности команд и их аргументов:

- При использовании команд, они сверяются со списком допустимых.

- Аргументы сверяются со списком позволенных и чётко определённых символов при вводе.

- Список разрешённых выражений использующих символы с заданной длинной.

- Необходимо убедиться, что метасимволы & | ; $ > < \ \ !` и пробелы не являются частью Regular Expression. Например, следующее выражение позволяет только символы нижнего регистра и числа и не содержит метасимволы, длина ограничена 3-10 символами: ^[a-z0-9]{3,10}$.