

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Национальный исследовательский университет ИТМО»

*Факультет Программной инженерии и компьютерной техники*

Дисциплина: Информационная безопасность

### **Лабораторная работа №3**

Аудит безопасности веб-приложения

Группа: Р3432

Выполнили:  
Готов Егор Дмитриевич

Преподаватель:  
Рыбаков Степан Дмитриевич

г. Санкт-Петербург

2025 г.

## Содержание

Назначение .....	3
Задание .....	4
Автоматизированное сканирование (DAST) .....	7
Верификация найденных уязвимостей после автоматизированного сканирования .....	10
Моделирование угроз с помощью STRIDE .....	15
Построение диаграммы потока данных (DFD).....	15
Анализ угроз по методике STRIDE .....	15
Таблица уязвимостей .....	24
Рекомендации по устранению рисков .....	27
Вывод.....	28

## **Назначение**

Освоить методику комплексного анализа защищенности веб-приложения, сочетая автоматизированное сканирование (DAST) и проактивное моделирование угроз (Threat Modeling). Получить навыки документирования результатов в виде профессионального отчета.

## Задание

Выполнить полный цикл аудита безопасности для тестового приложения OWASP Juice Shop.

### 1. Подготовка тестового стенда:

- a. Установите и запустите OWASP Juice Shop. Самый простой способ – через Docker: `docker run --rm -p 3000:3000 juicyshop/juice-shop`.
- b. Убедитесь, что приложение доступно по адресу <http://localhost:3000>
- c. Установите и настройте OWASP ZAP (Zed Attack Proxy). Рекомендуется использовать автономную версию (Standalone).

### 2. Автоматизированное сканирование (DAST)

- a. Настройка ZAP
  - i. Запустите ZAP.
  - ii. В поле “URL to attack” укажите адрес Juice Shop (<http://localhost:3000>)
  - iii. Нажмите “Attack”. ZAP начнет автоматическое сканирование (Ajax spider и Active scan)
- b. Анализ результатов
  - i. После завершения сканирования перейдите на вкладку “Alerts”. Отсортируйте уязвимости по степени риска (High, Medium, Low).
  - ii. Проведите верификацию найденных уязвимостей. Для этого найдите соответствующую уязвимость в Juice Shop и убедитесь, что ее можно эксплуатировать (например, реально украсть cookie через XSS или получить не санкционируемый доступ к API).

- iii. Сфокусируйтесь на нахождении и подтверждении как минимум 5 уязвимостей, среди которых должны быть SQLi и XSS
  - c. Дополнительное исследование: Используйте встроенный в Juice Shop “Score Board” для поиска дополнительных уязвимостей, которые ZAP мог пропустить
3. Моделирование угроз (Threat Modeling) с помощью STRIDE:
- a. Построение диаграммы потока данных (Data Flow Diagram - DFD)
    - i. Упрощенно визуализируйте ключевые компоненты Juice Shop: браузер, пользователя, веб-сервер, базу данных
    - ii. Отобразите на диаграмме основные потоки данных: аутентификация пользователя, поиск товаров, отправка отзывов, оформление заказов.
  - b. Анализ угроз по методике STRIDE:
    - i. Spoofing (Маскировка): можно ли impersonate другого пользователя ? (например, подменить сессию или JWT-токен)
    - ii. Tampering (Изменение данных): можно ли изменить цену товара, отзывы или данные профиля?
    - iii. Repudiation (Отказ от операций): можно ли отрицать совершение действия (например, покупки)? Есть ли логи?
    - iv. Information Disclosure (Раскрытие информации): можно ли получить доступ к данным других пользователей, API-ключам, исходному коду?
    - v. Denial of Service (Отказ в обслуживании): можно ли «положить» приложение одной HTTP-посылкой?

- vi. Elevation of Privilege (Повышение привилегий): можно ли из роли обычного пользователя получить права администратора
- с. Пример: на потоке данных “Аутентификация” угроза Spoofing может быть реализована через кражу сессионного токена (найденную вами уязвимость XSS)

#### 4. Подготовка финального отчета

- а. Создайте структурированный документ
- б. Таблица уязвимостей: для каждой найденной уязвимости (как через ZAP, так и через Threat Modeling) заполните таблицу со столбцами
  - i. Название (например, “Reflected XSS в поисковом запросе”)
  - ii. Описание (краткое описание и шаги воспроизведения)
  - iii. Уровень риска (CVSS) (воспользуйтесь онлайн калькулятором CVSS, например, от First.org)
  - iv. Категория OWASP Top 10 (например, A03:2021-Injection)
  - v. Предложение по исправлению (конкретная рекомендация: “Валидировать ввод на стороне сервера”, “Экранировать ввод” и т. д.)
- с. Рекомендации по устранению рисков: на основе анализа угроз дайте 3-5 общих рекомендаций по усилению безопасности приложения (например, “Внедрить строгую политику CSP”, “Регулярно обновлять зависимости”)

# Автоматизированное сканирование (DAST)

Запустим инструмент для автоматизированного сканирования ZAP (Zed Attack Proxy) со следующими настройками

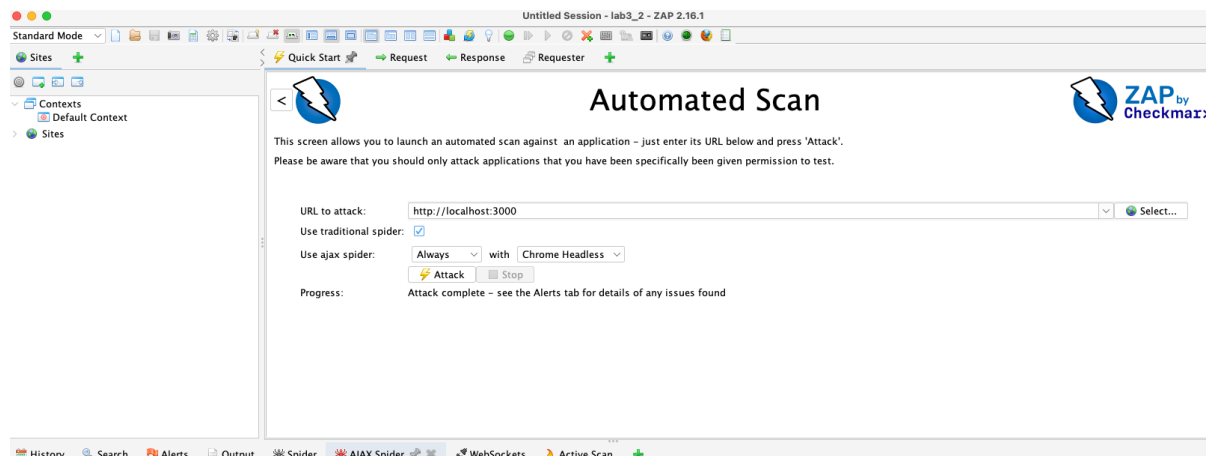


Рисунок 1-Настройка ZAP

При нажатии кнопки “Attack” начнется поиск всевозможных URL на веб-сайте сначала при помощи обычных Spider, потом при помощи Ajax Spider. После этого начнется активное сканирование, в котором ZAP сам генерирует специальные запросы к сайту с целью обнаружения таких уязвимостей как SQLi, XSS, CSRF и так далее.

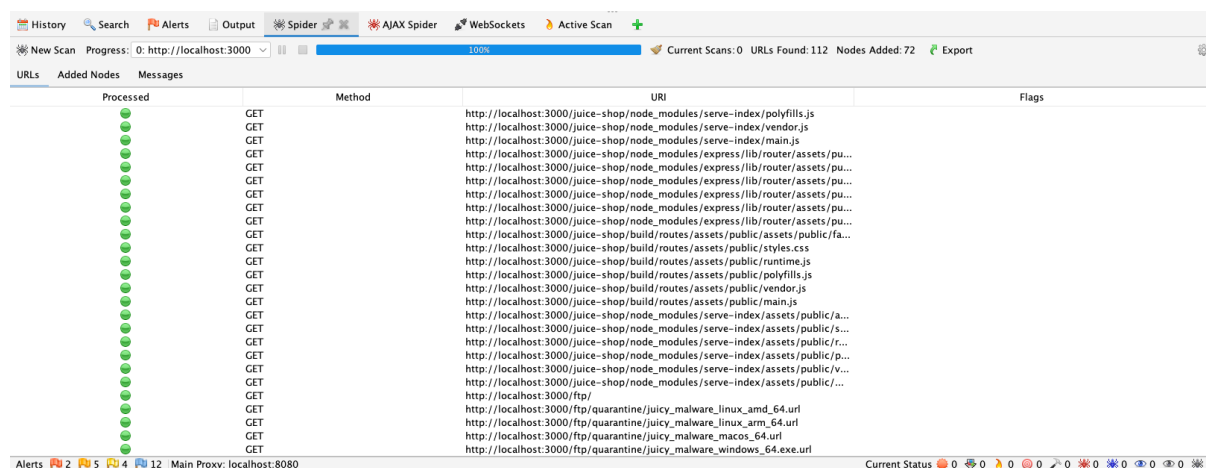


Рисунок 2 – Результат работы Spider

History	Search	Alerts	Output	Spider	AJAX Spider	WebSockets	Active Scan					
New Scan	Crawled URLs:358	Export										
Processed	ID	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	Note	Tags
	552	10/27/25, 8:42:26 PM	GET	http://localhost:3000/assets/public/images/pr...	200	OK	7 ms	431 bytes	17,038 bytes	Medium		
	553	10/27/25, 8:42:26 PM	GET	http://localhost:3000/assets/public/images/pr...	200	OK	11 ms	431 bytes	17,080 bytes	Medium		
	554	10/27/25, 8:42:26 PM	GET	http://localhost:3000/assets/public/images/pr...	200	OK	10 ms	431 bytes	15,910 bytes	Medium		Comment
	555	10/27/25, 8:42:26 PM	GET	http://localhost:3000/assets/public/images/pr...	200	OK	10 ms	431 bytes	21,524 bytes	Medium		
	556	10/27/25, 8:42:26 PM	GET	http://localhost:3000/assets/public/images/pr...	200	OK	24 ms	432 bytes	93,641 bytes	Medium		
	557	10/27/25, 8:42:26 PM	GET	http://localhost:3000/assets/public/images/pr...	200	OK	9 ms	431 bytes	26,934 bytes	Medium		
	558	10/27/25, 8:42:27 PM	GET	http://localhost:3000/rest/products/search?q=	304	Not Modified	72 ms	306 bytes	0 bytes	Medium		
	559	10/27/25, 8:42:27 PM	GET	http://localhost:3000/api/Quantities/	304	Not Modified	116 ms	306 bytes	0 bytes	Medium		
	560	10/27/25, 8:42:27 PM	GET	https://localhost:3000/socket.io/?EIO=4&transp...	400	Bad Request	3 ms	230 bytes	41 bytes	Medium		JSON
	561	10/27/25, 8:42:27 PM	GET	https://content-autofill.googleapis.com/v1/pa...	403	Forbidden	0 ms	130 bytes	40 bytes	Medium		
	562	10/27/25, 8:42:30 PM	POST	http://localhost:3000/socket.io/?EIO=4&transp...	200	OK	22 ms	213 bytes	2 bytes	Medium		
	563	10/27/25, 8:42:30 PM	GET	http://localhost:3000/socket.io/?EIO=4&transp...	101	Switching Pr...	18 ms	129 bytes	0 bytes	Medium		
	564	10/27/25, 8:42:30 PM	GET	http://localhost:3000/socket.io/?EIO=4&transp...	200	OK	20 ms	230 bytes	34 bytes	Medium		
	565	10/27/25, 8:42:30 PM	GET	http://localhost:3000/rest/admin/application-c...	304	Not Modified	13 ms	306 bytes	0 bytes	Medium		
	566	10/27/25, 8:42:32 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	7 ms	392 bytes	0 bytes	Medium		
	567	10/27/25, 8:42:33 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	3 ms	392 bytes	0 bytes	Medium		
	568	10/27/25, 8:42:33 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	5 ms	392 bytes	0 bytes	Medium		
	569	10/27/25, 8:42:33 PM	POST	http://localhost:3000/socket.io/?EIO=4&transp...	200	OK	3 ms	213 bytes	2 bytes	Medium		
	570	10/27/25, 8:42:34 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	10 ms	392 bytes	0 bytes	Medium		
	571	10/27/25, 8:42:34 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	13 ms	392 bytes	0 bytes	Medium		
	572	10/27/25, 8:42:34 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	9 ms	392 bytes	0 bytes	Medium		
	573	10/27/25, 8:42:34 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	9 ms	392 bytes	0 bytes	Medium		
	574	10/27/25, 8:42:34 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	11 ms	392 bytes	0 bytes	Medium		
	575	10/27/25, 8:42:34 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	12 ms	393 bytes	0 bytes	Medium		
	576	10/27/25, 8:42:33 PM	GET	http://localhost:3000/socket.io/?EIO=4&transp...	200	OK	465 ms	229 bytes	1 bytes	Medium		
	577	10/27/25, 8:42:34 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	7 ms	392 bytes	0 bytes	Medium		
	578	10/27/25, 8:42:34 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	7 ms	392 bytes	0 bytes	Medium		
	579	10/27/25, 8:42:34 PM	GET	http://localhost:3000/assets/public/images/pr...	304	Not Modified	10 ms	392 bytes	0 bytes	Medium		
Alerts	2	5	4	12	Main Proxy: localhost:8080					Current Status	0	0

Рисунок 3-Результат работы Ajax Spider

http://localhost:3000 Scan Progress						
Progress		Response Chart				
Host:		http://localhost:3000				
	Strength	Progress	Elapsed	Reqs	Alerts	Status
Analyser			00:00.862	61		
Plugin						
Path Traversal	Medium		01:51.737	2646	0	✓
Remote File Inclusion	Medium		01:01.163	1580	0	✓
Source Code Disclosure – /WEB-INF Folder	Medium		00:01.157	60	0	✓
Heartbleed OpenSSL Vulnerability	Medium		00:00.047	3	0	✓
Source Code Disclosure – CVE-2012–1823	Medium		00:06.323	29	0	✓
Remote Code Execution – CVE-2012–1823	Medium		00:01.937	272	0	✓
External Redirect	Medium		00:55.962	1422	0	✓
Server Side Include	Medium		00:24.996	632	0	✓
Cross Site Scripting (Reflected)	Medium		00:31.194	830	0	✓
Cross Site Scripting (Persistent) – Prime	Medium		00:06.562	158	0	✓
Cross Site Scripting (Persistent) – Spider	Medium		00:06.491	136	0	✓
Cross Site Scripting (Persistent)	Medium		00:00.648	0	0	✓
SQL Injection	Medium		02:15.085	3443	2	✓
SQL Injection – MySQL (Time Based)	Medium		01:01.328	1580	0	✓
SQL Injection – Hypersonic SQL (Time Bas...	Medium		01:01.310	1580	0	✓
SQL Injection – Oracle (Time Based)	Medium		00:30.967	790	0	✓
SQL Injection – PostgreSQL (Time Based)	Medium		00:31.355	789	0	✓
SQL Injection – SQLite (Time Based)	Medium		00:19.213	1417	2	✓
Cross Site Scripting (DOM Based)	Medium		18:55.783	7726	0	✓
SQL Injection – MsSQL (Time Based)	Medium		01:01.346	1580	0	✓
Log4Shell	Medium		00:00.019	0	0	✗
Spring4Shell	Medium		00:06.943	284	0	✓
Server Side Code Injection	Medium		00:49.476	1264	0	✓
Remote OS Command Injection	Medium		01:56.276	3002	0	✓
XPath Injection	Medium		00:18.687	474	0	✓
XML External Entity Attack	Medium		00:00.473	0	0	✓
Generic Padding Oracle	Medium		00:00.306	0	0	✓
Cloud Metadata Potentially Exposed	Medium		00:00.426	9	0	✓
Server Side Template Injection	Medium		01:25.936	2212	0	✓
Server Side Template Injection (Blind)	Medium		01:14.553	1896	0	✓
Remote OS Command Injection (Time Based)	Medium		01:38.568	2530	0	✓
Directory Browsing	Medium		00:06.622	136	0	✓
Buffer Overflow	Medium		00:06.526	154	0	✓
Format String Error	Medium		00:18.785	462	0	✓
CRLF Injection	Medium		00:43.143	1106	0	✓
Parameter Tampering	Medium		00:43.194	884	0	✓
ELMAH Information Leak	Medium		00:00.070	1	0	✓
Trace.axd Information Leak	Medium		00:00.744	40	0	✓
.htaccess Information Leak	Medium		00:01.088	35	0	✓
.env Information Leak	Medium		00:00.840	35	0	✓
Hidden File Finder	Medium		00:00.256	52	0	✓
Spring Actuator Information Leak	Medium		00:00.046	2	0	✓
XSLT Injection	Medium		00:49.439	1120	0	✓
GET for POST	Medium		00:00.588	0	0	✓
User Agent Fuzzer	Medium		01:13.783	1632	148	✓
Script Active Scan Rules	Medium		00:00.021	0	0	✗
SOAP Action Spoofing	Medium		00:00.601	0	0	✓
SOAP XML Injection	Medium		00:00.320	0	0	✓
Totals			42:35.007	45344	152	
Copy		Close				

Рисунок 4 – Результат работы активного сканирования



После сканирования получился следующий отчет с найденными уязвимостями (вкладка “Alerts”)

The screenshot shows the ZAP Alerts tab. On the left, a list of 23 alerts is shown, with 'SQL Injection (2)' selected. The main panel displays details for an 'SQL Injection' alert. The URL is 'http://localhost:3000/rest/languages?query=query%27+AND+%271%27%3D%271%27+---+'. The risk is 'High' and confidence is 'Medium'. The parameter is 'query' and the attack is 'query' OR '1'='1' --'. The evidence shows the alert was triggered by an active scan. The description states 'SQL injection may be possible'. The 'Other Info' section explains that the page results were successfully manipulated using the boolean conditions [query' AND '1'='1' -- ] and [query' OR '1'='1' -- ]. The solution advises not to trust client side input and to check all data on the server side. The reference is 'https://cheatsheetseries.owasp.org/cheatsheets/SQL\_injection\_Prevention\_Cheat\_Sheet.html'. At the bottom, there is an 'Alert Tags' table with columns 'Key' and 'Value'.

Key	Value
POLICY: CA_BULL	

Рисунок 5-Вкладка “Alerts” в ZAP

# Верификация найденных уязвимостей после автоматизированного сканирования

## 1. SQL Injection

Стандартная SQLi, которая присутствует в веб-приложении.

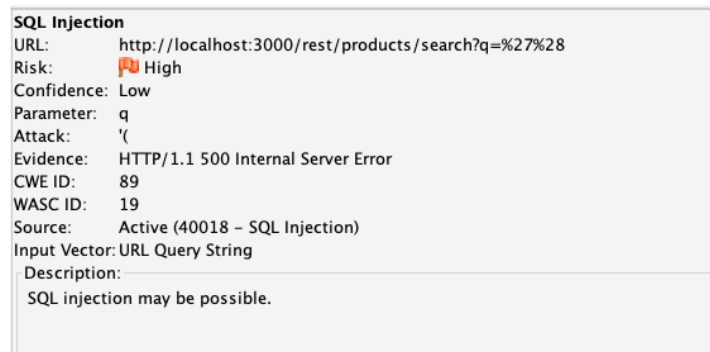


Рисунок 6-Уязвимость SQLi

Для проверки уязвимости перейдем по ссылке, которую сформировал ZAP во время атаки.

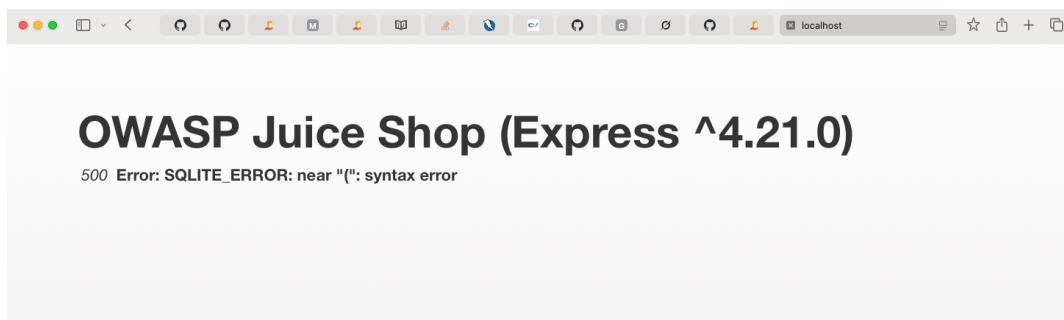


Рисунок 7-Верификация уязвимости SQLi

И действительно, сервер падает с ошибкой 500 SQLITE\_ERROR. Это происходит из-за того, что query-параметр встраивается непосредственно в SQL-запрос без какой-либо санитизации пользовательского ввода

## 2. Отсутствие заголовка CSP

Заголовок CSP сообщает браузеру откуда разрешено загружать ресурсы (script-src, font-src, style-src, media-src и так далее) и какие типы выполнения контента допустимы (inline-скрипты, CSS, JavaScript и так далее). Данный заголовок создает дополнительный уровень защиты, даже если XSS-

уязвимости присутствуют в приложении. К тому же можно отправлять отчеты о попытках XSS-атак при помощи параметра CSP заголовка report-uri.

ZAP нашел 73 ресурсов, где отсоветует данный заголовок. Обычно такой заголовок вставляется сервером при отправке веб-страниц пользователю.

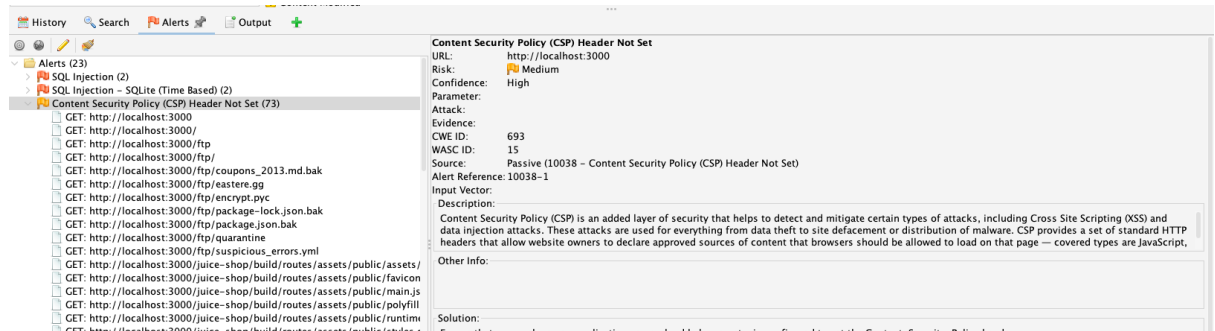


Рисунок 8-Описание уязвимости CSP из ZAP

Проверим, что заголовок правда нет, к примеру, при открытии стартовой страницы



Рисунок 9-Запрос для верификации уязвимости CSP

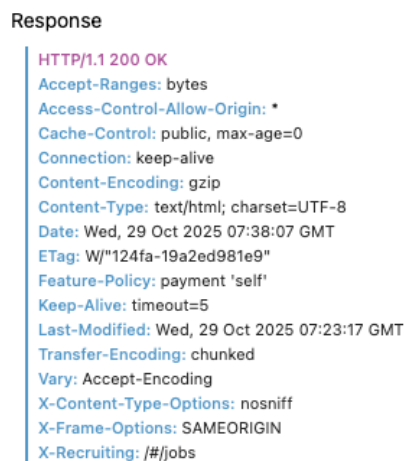


Рисунок 10-Верификация отсутствия CSP-заголовка

Видим, что действительно заголовка CSP в ответе от сервера нет.

### 3. Неправильная настройка CORS

Уязвимость заключается в том, сторонний сайт может прочитать ответы нашего сервера из браузера жертвы из-за слишком гибкого разрешения в заголовке Access-Control-Allow-Origin.

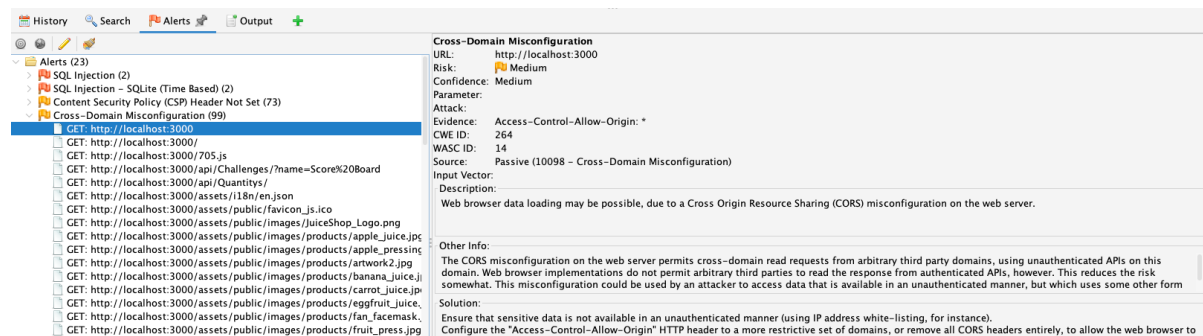


Рисунок 11-Описание уязвимости CORS из ZAP

Данная уязвимость была найдена для 99 ресурсов на веб-сайте.

Проверим это в браузере



Рисунок 12-Запрос для верификации уязвимости CORS

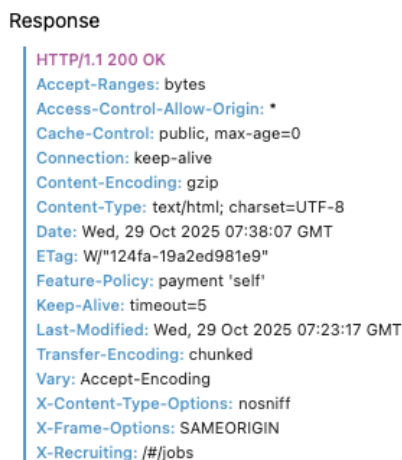


Рисунок 13-Верификация уязвимости CORS

Видим, что заголовок Access-Control-Allow-Origin ставится \*, что означает, что любой сторонний сайт может не только инициировать запросы, но и получать содержимое ответов. При условии, если у жертвы есть активная сессия на этом сайте.

#### 4. Отсутствие заголовка, защищающего от Clickjacking

Clickjacking – вид атаки, при которой злоумышленник обманывает пользователя и заставляет его нажать на скрытые элементы другого сайта, не осознавая, что он взаимодействует именно с ними.

Для защиты от такой уязвимости используются заголовки X-Frame-Options (контролирует загрузку фреймов на странице) и CSP, который был описан выше.

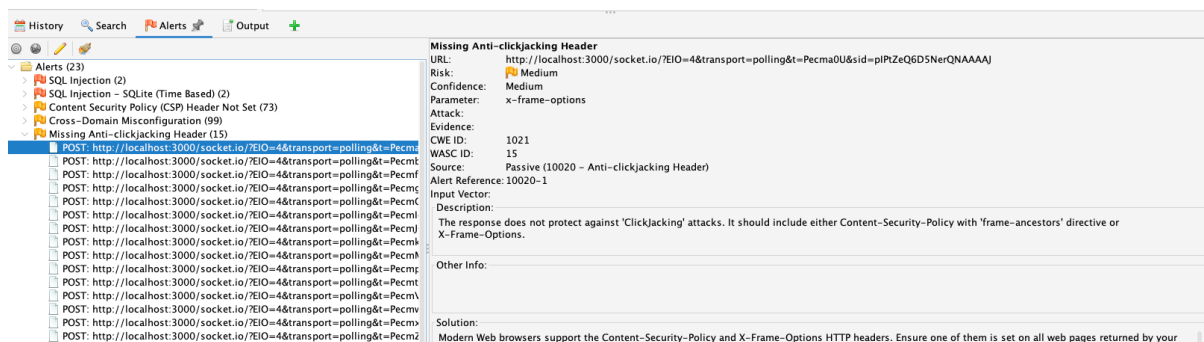


Рисунок 14-Описание уязвимости Clickjacking

##### Request

```
POST /socket.io/ HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en;q=0.9
Connection: keep-alive
Content-Length: 2
Content-Type: text/plain;charset=UTF-8
Cookie: continueCode=jxkMXgwP5nR7a6VzLKm2lW4oGMMtkf7LhKjGpq9jQ1DbJxyENOrB8Ye3vZME; cookieconsent_status=dismiss; welcomebanner_status=dismiss; language=en
Origin: http://localhost:3000
Priority: u=3, i
Referer: http://localhost:3000/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/26.0.1 Safari/605.1.15
```

##### Response

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:4200
Connection: keep-alive
Content-Length: 2
Content-Type: text/html
Date: Wed, 29 Oct 2025 08:09:35 GMT
Keep-Alive: timeout=5
Vary: Origin
```

Рисунок 15-Запрос и ответ для подтверждения отсутствия заголовка X-Frame-Options

Видим, что CSP установлен, но X-Frame-Options нет. Таким образом, можно замаскировать, к примеру, любую кнопку на веб-сайте под какой-

нибудь фрейм, позволяющий отправлять запросы на сторонние веб-сайты под именем пользователя.

## 5. XSS

Так как ZAP по какой-то причине не нашел ни одной XSS уязвимости, найдем ее самостоятельно.

По такой ссылке [http://localhost:3000/api/products/{product\\_id}](http://localhost:3000/api/products/{product_id}) и методу PUT мы можем поменять описание товара на странице. Попробуем в поле description товара поместить iframe и создать stored XSS

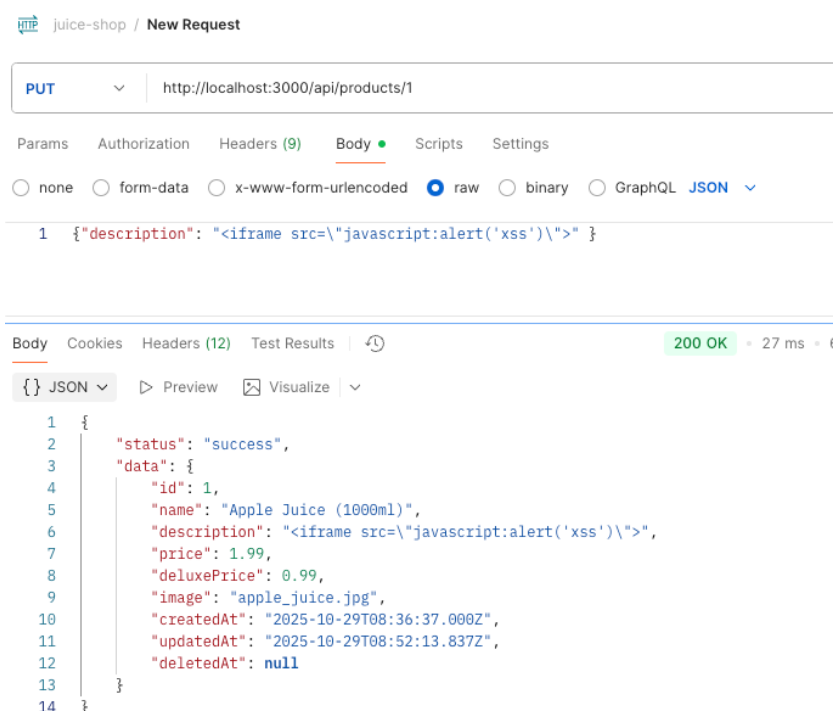


Рисунок 16-XSS-атака

Запрос выполнен успешно. Теперь при нажатии на товар каждый пользователь будет подвергнут XSS атаке.

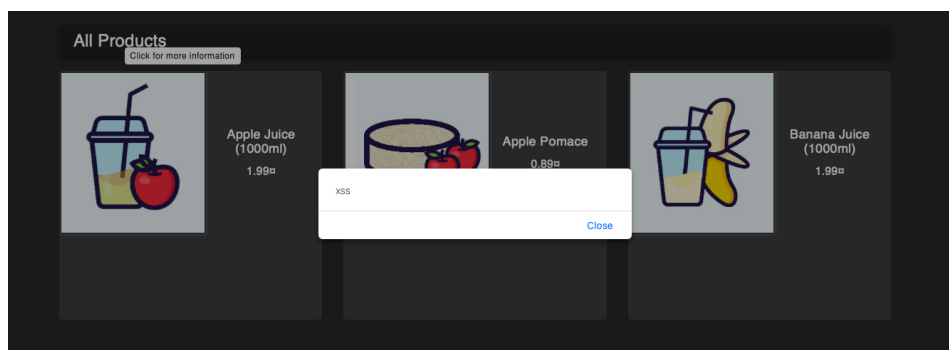


Рисунок 17-Подтверждение XSS-атаки

# Моделирование угроз с помощью STRIDE

## Построение диаграммы потока данных (DFD)

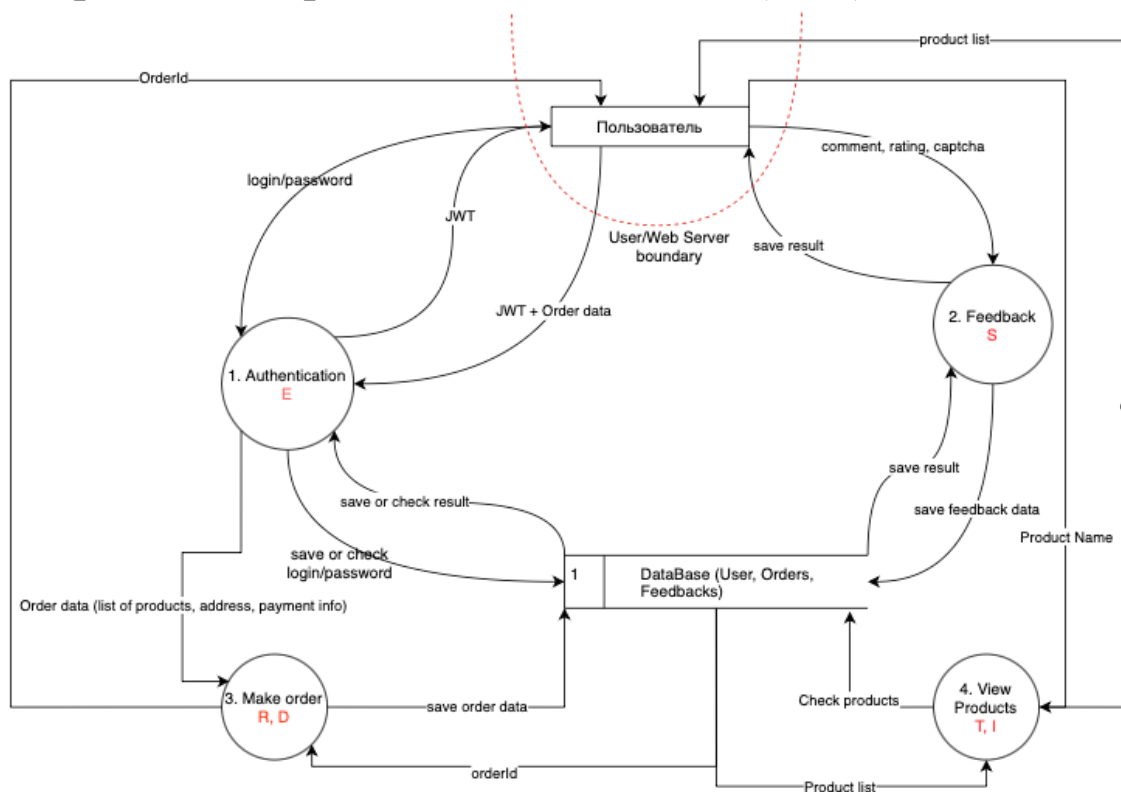


Рисунок 18-DFD-диаграмма

## Анализ угроз по методике STRIDE

- **Spoofing (Маскировка)**

На потоке данных “Добавление отзыва о сайте” угроза маскировки может быть реализована через подмену id пользователя и подмену JWT токена в запросе так как сервер не проверяет авторизованный ли пользователь отправляет отзыв.

Рисунок 19-Форма для отзывов

Есть такая форма, которая отправляется на сервер. Введем тестовые данные, чтобы посмотреть какой формат запроса отправляется.



Рисунок 20-Запрос, который отправляется при отправке формы

А также найдем JWT и откроем любой сайт, где можно декодировать payload токена, чтобы его поменять.

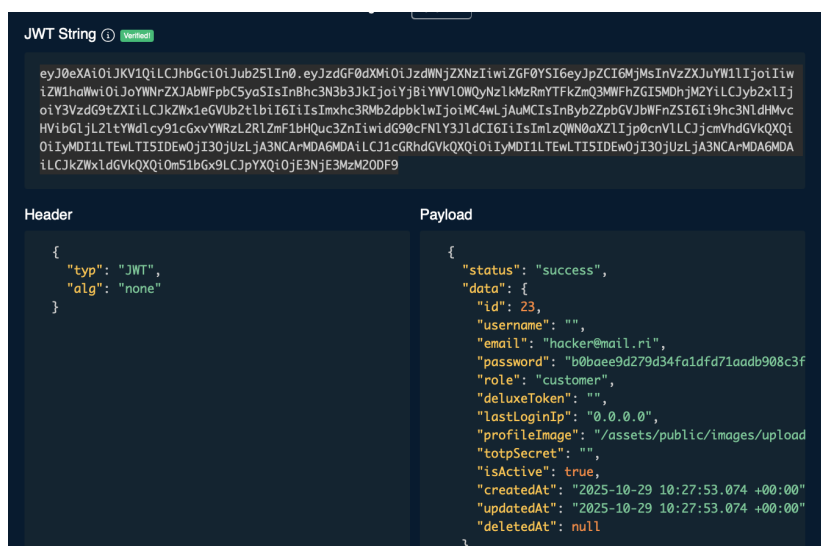


Рисунок 21-Подмена JWT для реализации атаки



Меняем поле `alg` на `none` и поле `email` на ту почту пользователя, от лица которого мы хотим отправить отзыв. Заходим в Postman и отправляем запрос, предварительно настроив `cookie`.

Перед этим убедимся, что мы не указываем собственный `userId`

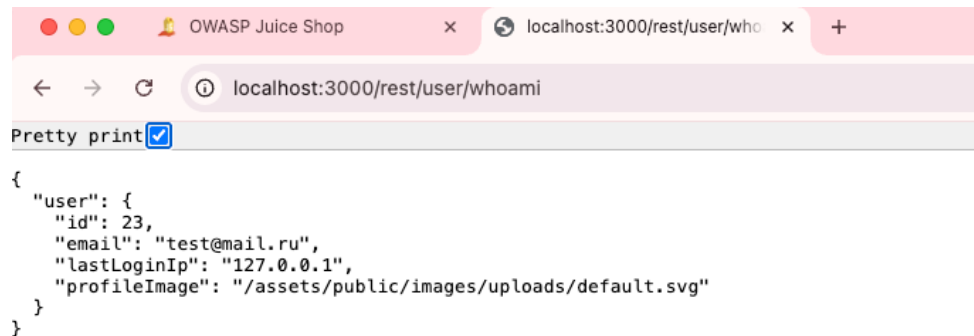


Рисунок 22-Данные текущего пользователя

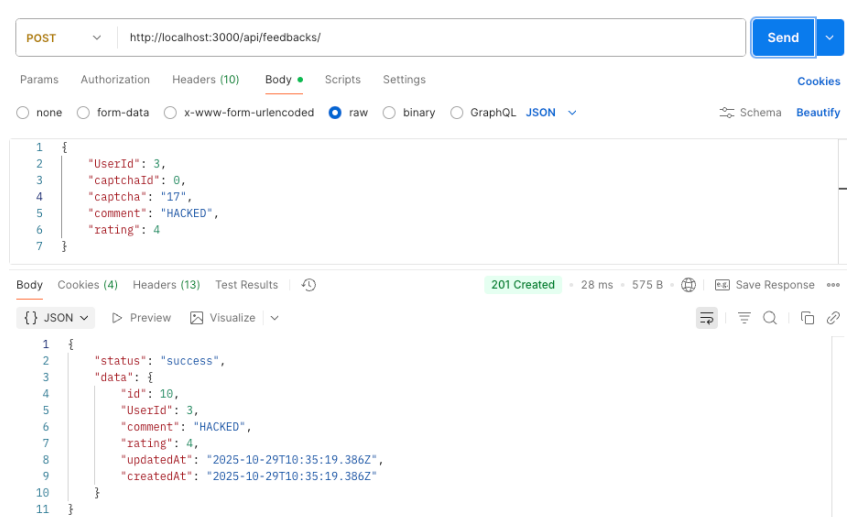


Рисунок 23-Реализация Spoofing угрозы

Таким образом, мы отправили отзыв от лица другого пользователя.

- **Tampering (изменение данных)**

На потоке “Просмотр карточки товара” угроза изменение данных может быть реализована через изменение цены на товар посредством PUT запроса.

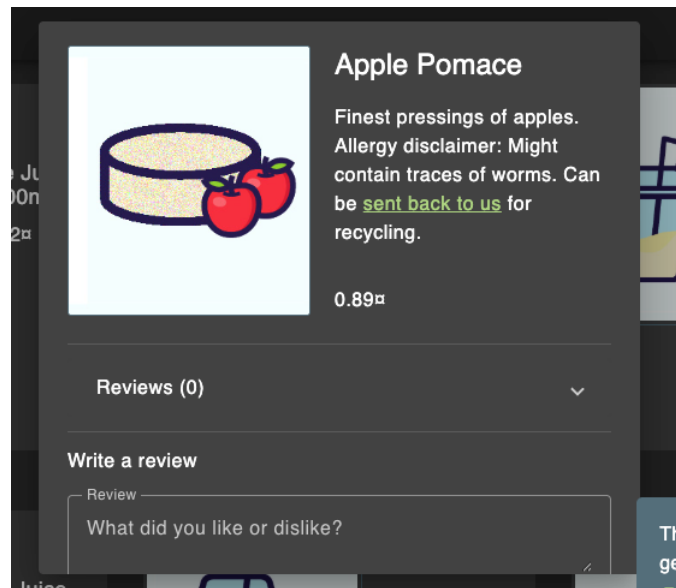


Рисунок 24-Карточка товара

На данной карточке товара цена указана 0.89. Давайте поменяем на 0.

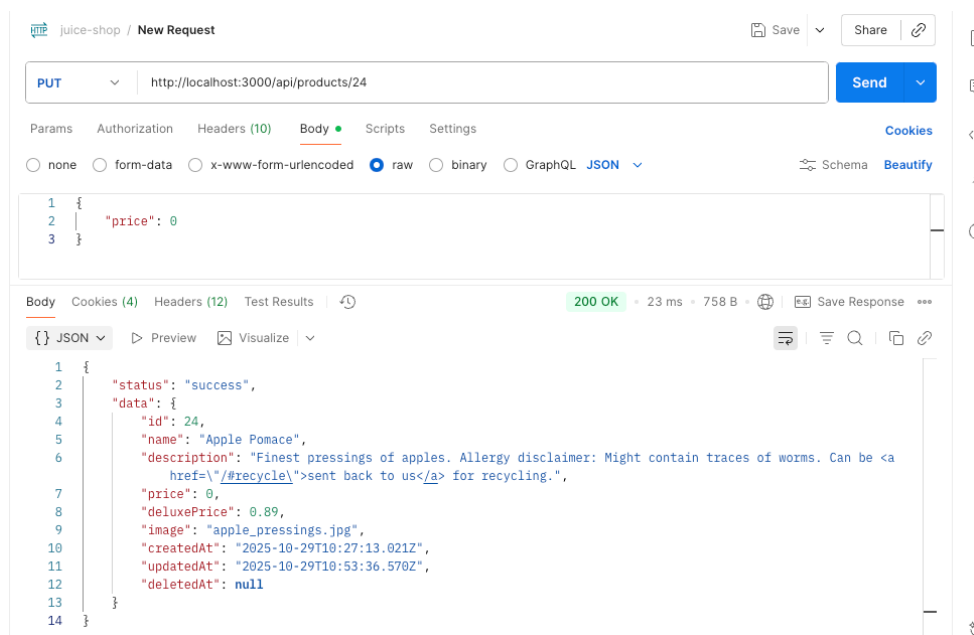


Рисунок 25-Изменение цены товара, реализация Tampering

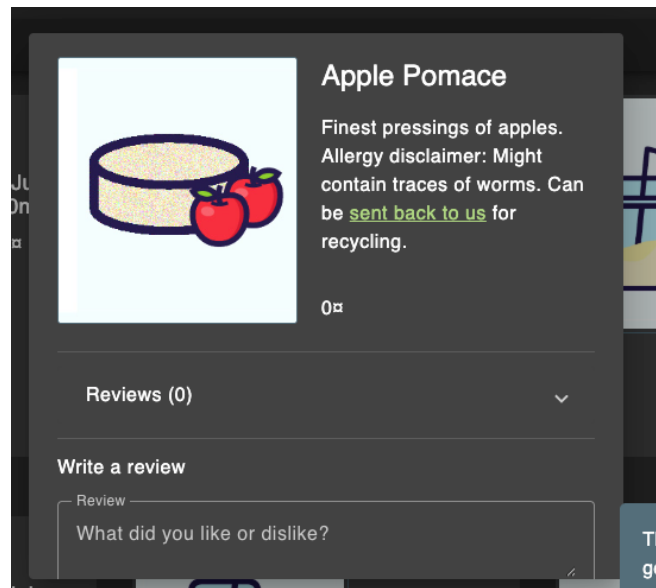


Рисунок 26-Подтверждение атаки

Запрос выполнен успешно, и мы без проблем подменили цену товара на сайте.

- **Repudiation (Отказ от операций)**

На потоке данных “Оформление заказа” угроза отказа от операции может быть реализована при использовании уже просроченного купона и отсутствия логирования. Из файла main.js мы можем найти список купонов

```

- paymentMode = "card";
- campaigns = {
-   WMNSDY2019: {
-     validOn: 15519996e5,
-     discount: 75
-   },
-   WMNSDY2020: {
-     validOn: 1583622e6,
-     discount: 60
-   },
-   WMNSDY2021: {
-     validOn: 1615158e6,
-     discount: 60
-   },
-   WMNSDY2022: {
-     validOn: 1646694e6,
-     discount: 60
-   },
-   WMNSDY2023: {
-     validOn: 167823e7,
-     discount: 60
-   },
-   ORANGE2020: {
-     validOn: 15885468e5,
-     discount: 50
-   },
-   ORANGE2021: {
-     validOn: 16200828e5,
-     discount: 40
-   },
- }

```

Рисунок 27-Купоны из исходного кода

Если мы поменяем дату на устройстве на 8 марта 2019, то купон WMNSDY2019 применится и мы сможем оформить заказ со скидкой

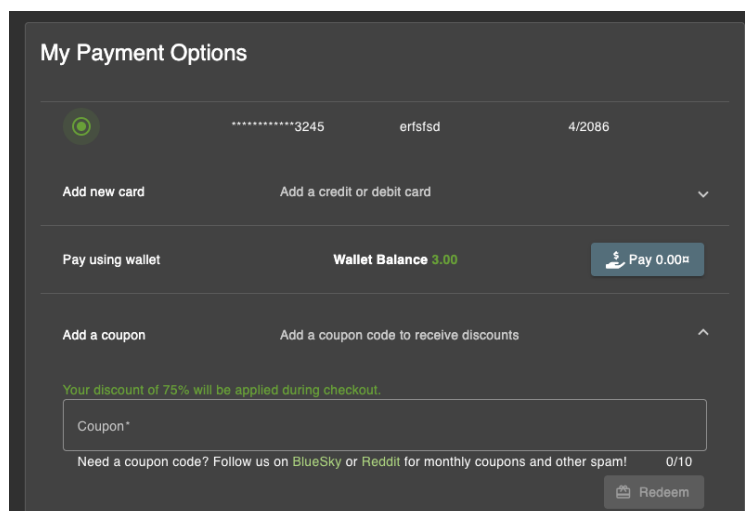


Рисунок 28-Реализация использования устаревшего купона

Купон применился, и мы можем перейти к оплате заказа. Таким образом, на сервере нет информации о том, что купон был просрочен и использован и пользователь может отрицать это действие. Со стороны магазина так же нет доказательств, так как на сервере нет логов, которые могут подтверждать или отрицать это действие. Единственный логи, которые удалось найти представлены ниже.

```
access.log.2019-03-08
::1 - - [08/Mar/2019:11:24:16 +0000] "GET /api/Deliverys/1 HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:24:28 +0000] "GET /api/Deliverys/1 HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:24:53 +0000] "POST /api/Cards/ HTTP/1.1" 201 202 "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:24:53 +0000] "GET /api/Cards HTTP/1.1" 200 128 "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:25:00 +0000] "GET /api/Deliverys/1 HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:25:00 +0000] "GET /rest/user/whoami HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:25:00 +0000] "GET /api/Cards/7 HTTP/1.1" 200 126 "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:25:00 +0000] "GET /rest/basket/6 HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:25:00 +0000] "GET /api/Addresses/7 HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:25:10 +0000] "GET /rest/continue-code HTTP/1.1" 200 79 "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:25:10 +0000] "POST /rest/basket/6/checkout HTTP/1.1" 200 45 "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:25:10 +0000] "GET /rest/track-order/cb69-23897e0a884461bc HTTP/1.1" 200 450 "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:25:10 +0000] "GET /rest/admin/application-configuration HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:25:10 +0000] "GET /rest/basket/6 HTTP/1.1" 200 154 "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
::1 - - [08/Mar/2019:11:25:10 +0000] "GET /api/Addresses/7 HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36"
```

Рисунок 29-Серверный лог

Однако, они не несут в себе достаточной информации, чтобы подтвердить то или иное действие пользователя.

- **Information Disclosure (раскрытие информации)**

На потоке данных “Поиск товаров” может быть реализована угроза раскрытия информации посредством выполнения SQLi в query-параметре (qwerty)) UNION SELECT id, email, password, '4', '5', '6', '7', '8', '9' FROM Users—

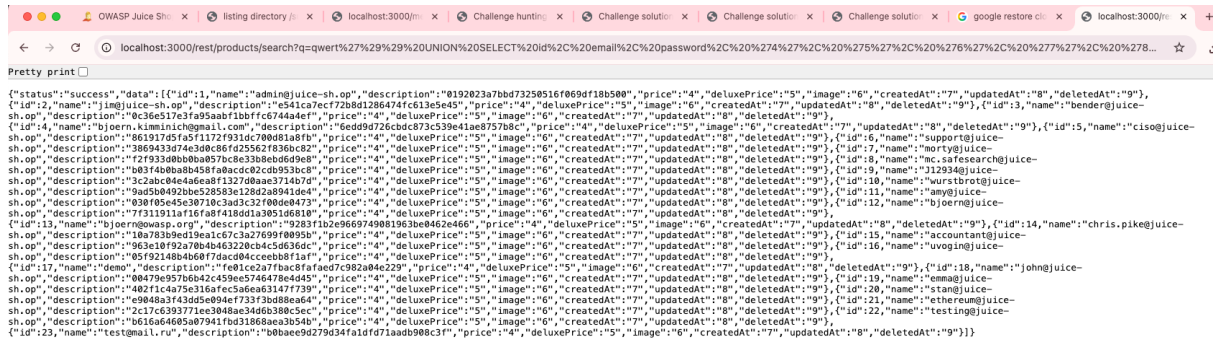


Рисунок 30-Реализация раскрытия конфиденциальной информации

Видим всех пользователей, которые зарегистрированных на сайте, включая их email, пароли и другие данные

- **Denial of Service (Отказ в обслуживании)**

На потоке “Оформление заказа” угроза отказа в обслуживании может быть реализована посредством использования уязвимой библиотеки десериализации JSON объектов.

Зайдем в Swagger данного веб-сайта и найдем endpoint для отправки заказа

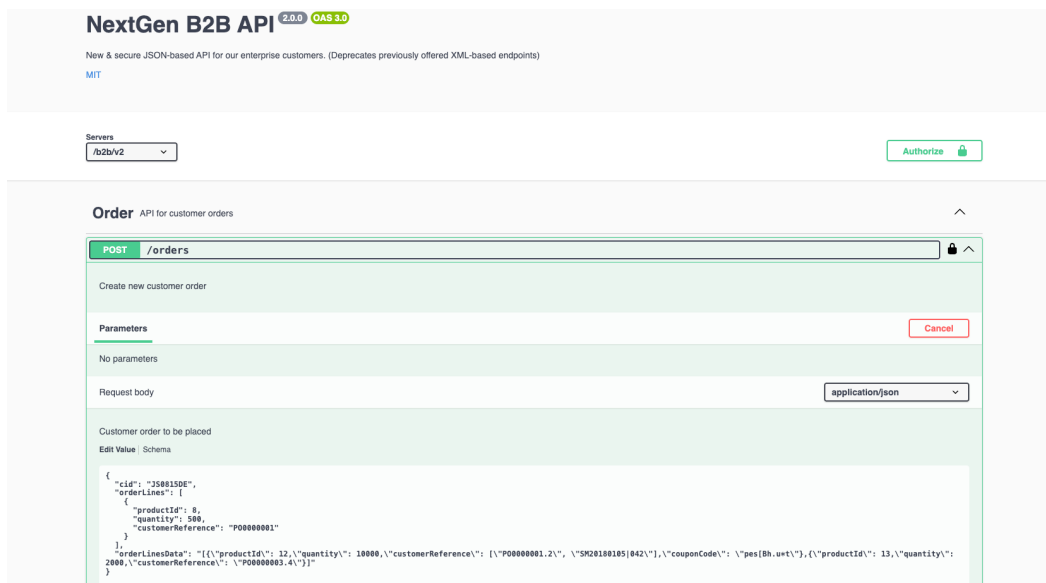


Рисунок 31-Swagger Juice Shop

В данном приложении используется библиотека, которая небезопасно парсит JSON формат тем самым позволяя исполнять любой валидный код, который будет передан как значение поля. Попробуем отправить следующий JSON объект, чтобы вызвать бесконечный цикл на сервере и остановить его работу



Рисунок 32-Внедрение уязвимости в тело запроса

Видим, что сервер через какое-то время вернул ошибку 500. Такая ошибка возвращается так как на сервере есть таймаут в случае обнаружения бесконечного цикла. Но сам факт, что сервер ложится при помощи одного запроса выполняется.

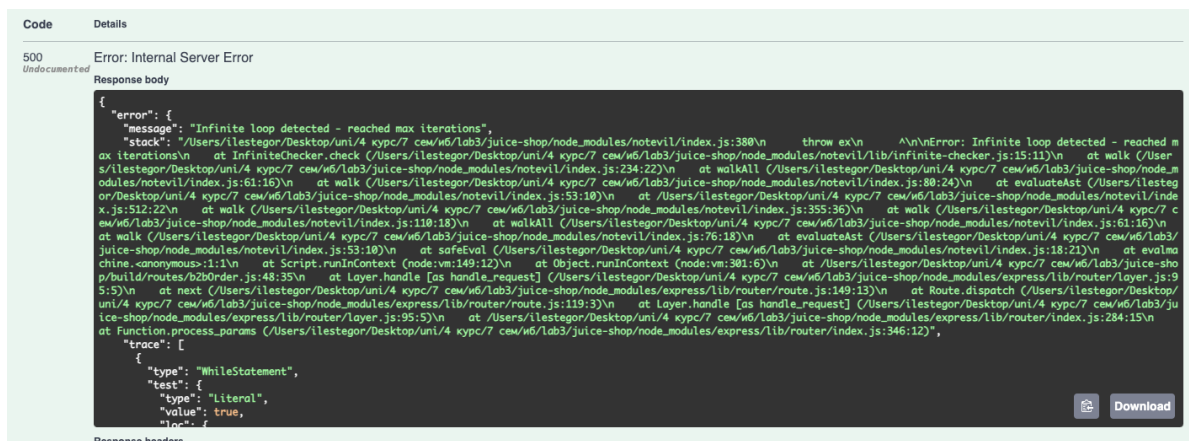


Рисунок 33-Результат атаки

- **Elevation of Privilege (Повышение привилегий)**

На потоке “Аутентификация” угроза повышения привилегий может быть реализована посредством SQLi.

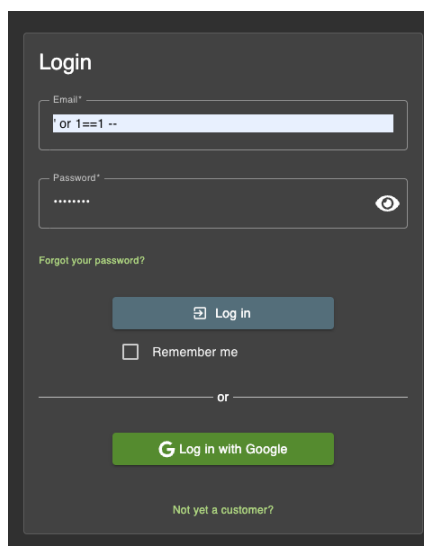


Рисунок 34-Форма логина

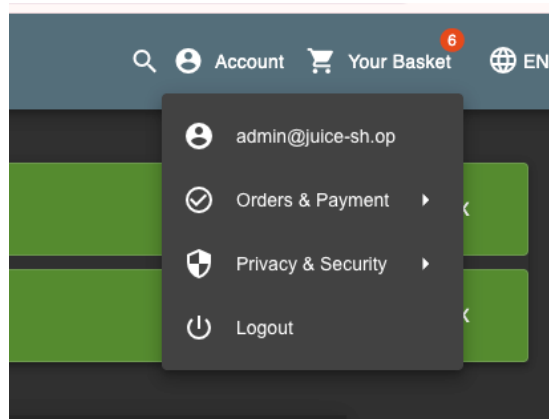


Рисунок 35-Подтверждение повышение привилегий

## Таблица уязвимостей

Таблица 1-Таблица уязвимостей

Название	Описание	Уровень риска (CVSS)	Категория OWASP Top 10	Предложение по исправлению
SQLi в поисковом запросе	Возможность положить сервер (получить 500) при определенном запросе в поисковой строке	8.8 (High)	A03:2021 - Injection	Не использовать конкатенацию строк при составлении запроса, санитизировать весь пользовательский ввод
CSP header not set	Отсутствует заголовок CSP, что может привести к XSS атакам	(6.1)Medium	A05:2021 - Security Misconfiguration	Настроить заголовок CSP, чтобы возвращался сервером в каждом ответе и запрещал встраивание и исполнение чужого JS-кода
Cross-Domain Misconfiguration	Заголовок Access-Control-Allow-Origin имеет слишком широкий контекст и может привести к XSS и CSRF	(6.5)Medium	A05:2021 – Security Misconfiguration	Настроить заголовок Access-Control-Allow-Origin, чтобы к странице не могли обращаться сторонние ресурсы
Missing Anti-clickjacking header	Отсутствие таких заголовков, как X-Frame-Options и CSP может привести к встраиванию вредоносного кода на страницу	(6.5)Medium	A05:2021 – Security Misconfiguration	Настроить заголовок X-Frame-Options: DENY и CSP: default-src 'self', чтобы нельзя было встраивать и исполнять чужой код на странице
XSS при изменении описания товара	Можно встроить вредоносный код, который будет исполняться при каждом нажатии на карточку товара для всех пользователей	8.5 (High)	A03:2021 - Injection	Чистить пользовательский ввод и не сохранять неочищенный ввод на странице в HTML.



Spoofing посредством подмены email и JWT	Нет проверки соответствия пользователя и токену, который передается. Можно писать отзывы о компании от любого лица, которое зарегистрирован о на сайте и имеет userId	6.8 (Medium)	A01:2021- Broken Access Control	Добавить правильную обработку JWT для проверки его валидности и принадлежности текущему пользователю
Tampering через изменение цены товара	Можно беспрепятственн о заменить поле price в карточке товара и заказать товар с новой ценой	6.5 (Medium)	A01:2021- Broken Access Control	Добавить проверку цены на стороне сервера, сделать ролевую модель, которая запрещала бы изменение цены даже не аутентифицированн ому пользователю
Repudiation через использовани е просроченног о купона при оформлении заказа и отсутствие логов	Система не ведет надлежащий лог за историей поведения пользователя на сайте. А также за историей купонов. Все это может привести к отказу от операции	7.5 (High)	A09:2021 – Security Logging and Monitoring Failures	Добавить хорошее логирование действий пользователей (время, IP, URL) во время серьезных операций таких как оформление заказа, оплата заказа и использование купонов.
Information Disclosure через SQLi при поиске товаров	При помощи SQLi можно получить доступ к конфиденциальн ой информации всех пользователей системы	7.7 (High)	A03:2021 - Injection	Не использовать конкатенацию строк при составлении запроса, санитизировать весь пользовательский ввод. Также можно встроить ролевую модель, запрещая доступ к списку пользователей помимо админа
Denial of Service через использовани е	Возможность выполнить произвольный	9.8 (Critical)	A06:2021 – Vulnerable and Outdated Components	Заменить библиотеку с уязвимостью. Чаше делать аудит

небезопасной библиотеки десериализации JSON при оформлении заказа	код в значении поля JSON			зависимостей при помощи SCA инструментов
Elevation of Privilege через SQLi в форме регистрации	При помощи SQLi можно получить доступ к аккаунту администратора без надобности пароля	9.8 (Critical)	A03:2021 - Injection	Не использовать конкатенацию строк при составлении запроса, санитизировать весь пользовательский ввод.

## **Рекомендации по устранению рисков**

1. Использовать Prepared Statement, ORM при составлении запросов к БД, чтобы избежать возможности SQLi, валидация всех входных данных от пользователя
2. Внедрять строгие политики CSP, Anti-clickjacking, CORS для устранения возможности XSS и CSRF
3. Пересчитывать все критичные поля (цена, скидки, валидность купонов) на сервере из БД
4. Создание надежной системы логирования, мониторинга и проведение аудитов. Логировать ключевые события в стандартизованном формате, который бы нес достаточно информации для принятия решения о ситуации. Хранить логи централизованно при помощи SIEM-инструментов. Настраивать уведомления о нестандартном поведении пользователей на основе логов
5. Проводить постоянные проверки зависимостей при помощи SCA. Внедрить эти инструменты в CI/CD

## **Вывод**

В результате выполнения лабораторной работы был проведен аудит тренировочного веб-приложения OWASP Juice Shop. Были использованы как средства автоматического тестирования безопасности ZAP (Zed Attack Proxy), который нашел немало уязвимостей (SQLi, отсутствие важных заголовков безопасности), так и ручное тестирование, при котором удалось найти еще больше уязвимостей (XSS, падение приложения при одном HTTP-запросе и другие). Найденные угрозы были проанализированы по модели STRIDE, а также была создана DFD диаграмма наглядно, показывающая потоки данных и места найденных уязвимостей в системе. На основе найденных и проанализированных уязвимостей были сформированы некоторые рекомендации по устранению проблем в системе.