

Федеральное государственное автономное образовательное  
учреждение высшего образования «Национальный  
исследовательский университет ИТМО»

*Факультет программной инженерии и компьютерной техники*  
*Информационная безопасность*

## **Отчёт по лабораторной работе №3**

### **Аудит безопасности веб-приложения**

Выполнил:  
Бабенко Даниил  
Александрович,  
Р3434

Санкт-Петербург  
2025г

## Отчёт об аудите безопасности:

### Краткое резюме

Для проведения аудита безопасности веб-приложения была скачана утилита OWASP ZAP. С её помощью было проведено DAST-сканирование требуемого приложения (OWASP Juice Shop).

По завершении сканирования утилита смогла обнаружить следующие уязвимости:

The screenshot shows the OWASP ZAP interface. On the left, the 'Alerts' (Оповещения) tab is selected, displaying a list of 13 alerts. The first alert, 'SQL-инъекция' (SQL Injection), is highlighted with a red box. The details for this alert are shown on the right:

- URL-адрес:** http://localhost:3000/rest/products/search?q=%27%28
- Риск:** High
- Достоверность:** Low
- Параметр:** q
- Атака:** '{
- Доказательства:** HTTP/1.1 500 Internal Server Error
- CWE ID:** 89
- WASC ID:** 19
- Источник:** Активная (40018 - SQL-инъекция)
- Input Vector:** URL Query String
- Описание:** SQL injection may be possible.

The top right of the interface shows the HTTP response details for the selected alert, indicating an 'HTTP/1.1 500 Internal Server Error'.

Для дальнейшей верификации найденных уязвимостей также была использована встроенная в сайт Таблица результатов (Score Board), которая предлагает пользователям самим найти уязвимости (в том числе и те, которые утилита OWASP ZAP не обнаружить).

The screenshot shows the 'Score Board' section of the OWASP Juice Shop. It displays progress on various challenges:

- Hacking Challenges:** 13% completed.
- Задание на программирование:** 0% completed.
- Challenges Solved:** 14/172.

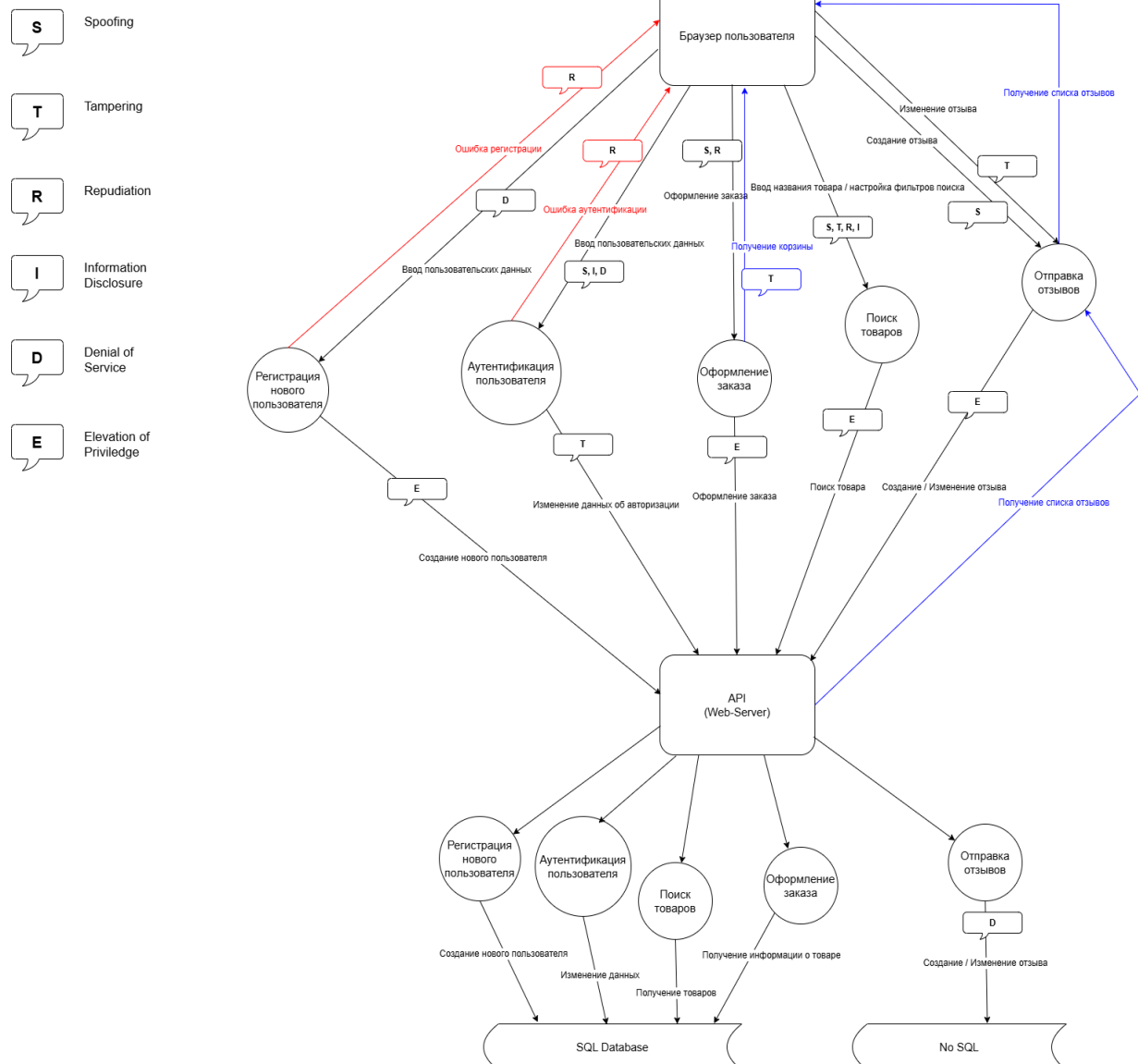
Below the progress bars, there are filters for different types of challenges, including XSS, Sensitive Data Exposure, Improper Input Validation, Broken Access Control, Unvalidated Redirects, Vulnerable Components, Broken Authentication, Security through Obscurity, Insecure Deserialization, Miscellaneous, Broken Anti Automation, Injection, Security Misconfiguration, Cryptographic Issues, and XXE.

At the bottom, there are four challenge cards:

- Score Board:** Find the hidden secret page 'Table'.
- DOM XSS:** Project DOM XSS attack with a crafted payload.
- Bonus Payload:** Use the provided payload to get a flag.
- Privacy Policy:** Read the privacy policy and find the hidden secret.

# 

### 



## 

### 

- “Аутентификация пользователя” - есть возможность получить доступ к чужой учётной записи, воспользовавшись SQL-инъекцией (закончить ввод логина “ -- “, тем самым можно будет ввести любой пароль и войти в чужой аккаунт).
- “Оформление заказа” - есть возможность совершения CSRF-атаки, благодаря чему возможно оформить заказ без ведома настоящего владельца аккаунта.

- “Поиск товаров” - изменение токена JWT и возможность подделать его без проверки подписи позволяет проводить поиск товаров от лица пострадавшего.
- “Отправка отзывов” - путём манипуляций с PUT-запросами на сервер можно отправлять отзывы от лица других пользователей.

## Tampering

- “Аутентификация пользователя” - есть возможность изменения пароля чужой учётной записи через SQL-инъекцию, Forget Password или через надлежащий запрос HTTP GET, позволяющий заменить пароль.
- “Оформление заказа” - есть возможность изменить bid пользователя в SessionStorage. Таким образом, можно получить доступ к чужой корзине, изменяя её содержимое.
- “Поиск товаров” - данная секция уязвима к SQL-инъекциям и XSS, поэтому манипуляции с запросами в строке поиска могут привести вредоносный код в приложение.
- “Отправка отзывов” - есть возможность модификации чужих отзывов при редактировании запросов на сервер.

## Repudiation

- “Аутентификация пользователя” - возможность изменения пользователями JWT-токенов позволяет искажать информацию, записываемую в логи приложения.
- “Регистрация нового пользователя” - так как нет проверок валидности используемого в поле email при регистрации, отрицание факта регистрации пользователя с данным логином.
- “Поиск товаров” - поисковая строка подвержена различного рода XSS-атакам, которые могут добавить вредоносный код на сайт. Однако из-за отсутствия детального логирования будет трудно установить пользователя, который внёс в систему вредоносный скрипт.
- “Оформление заказа” - есть возможность совершения CSRF-атак, из-за чего система может считать незаконную покупку как совершённую легально самим пользователем, что невозможно будет опровергнуть.

## Information Disclosure

- “Аутентификация пользователя” - имеет место жёсткое кодирование данных в клиентской части приложения. Таким образом, можно узнать действующие credentials для входа в систему, проинспектировав код.
- “Поиск товаров” - данная секция уязвима к SQL-инъекциям. Данные о товарах и о пользователях находятся в одном и том же хранилище SQL, что позволяет через строку поиска вводить запросы и о базе данных с пользователями.

## Denial of Service

- “Аутентификация пользователя” - ограничений на частоту запросов и количество попыток входа в систему не имеется, поэтому возможно нагрузить систему до состояния, когда она будет отказывать в авторизации настоящим пользователям.
- “Регистрация нового пользователя” - нет кроме наличия знака @ в email'e, на сайте не имеется никакой проверки валидности email. Таким образом, один пользователь может создать бесчисленное множество аккаунтов, нагружая систему до неработоспособного состояния.
- “Отправка отзывов” - с помощью манипуляций с бек-эндом API можно увидеть, что секция с отзывами уязвима к DoS-атакам, и подставив в URL команду sleep(), можно заставить сервер “спать”.

## Elevation of Privilege

- “Регистрация нового пользователя” - при регистрации система допускает подмену атрибута роли, что позволяет получить права администратора без надлежащей валидации.
- “Поиск товаров” - манипуляция с JWT-токенами позволяет изменить токен и получить права администратора, что даёт доступ к скрытым функциям сайта.
- “Оформление заказа” - есть возможность незаконного получения клиентом “премиум-статуса”, что улучшает некоторые условия обслуживания.

## Таблица уязвимостей

Название	Описание	Уровень риска (CVSS)	Категория OWASP Top 10	Предложения по устранению
SQL-инъекция	<p>Злоумышленник может вставить вредоносный SQL-код через пользовательские входные данные, манипулируя запросами к БД.</p> <p>В пользовательских запросах к БД с продуктами сайта параметр q уязвим к инъекциям. Внедрив в данный параметр SQL-скрипт, можно получить из БД данные пользователей.</p>	8.6 (High)	A03:2021-Injection	<p>Использовать параметризованные запросы для взаимодействий с БД;</p> <p>Использовать ORM;</p> <p>Применить whitelisting для параметра q.</p>
XSS	<p>Политика безопасности содержимого (CSP) помогает обнаруживать и смягчать определенные типы атак, включая XSS и атаки с внедрением данных. Эти атаки используются для всего: от кражи данных до порчи сайта или распространения вредоносных программ.</p> <p>Поисковая строка позволяет вставлять внутрь себя различные скрипты JavaScript, которые тут же исполняются при нажатии на кнопку</p>	6.1 (Medium)	A05:2021-Security Misconfiguration	<p>Экранировать пользовательский ввод с помощью программных библиотек;</p> <p>Внедрить политику безопасности CSP;</p> <p>Установить заголовок X-Content-Type-Option, равным "nosniff";</p> <p>Применять валидацию и санацию данных на сервере перед отображением.</p>

	(тестировалось на безобидном для системы примере, где внутри строки был записан запрос для вызова alert).			
Insecure Direct Object Reference (IDOR)	<p>Сайт предоставляет прямой доступ к внутренним объектам на основе предоставленного пользователем идентификатора без надлежащей проверки прав доступа.</p> <p>С помощью DevTools браузера можно заменить значение bid в SessionStorage, тем самым получив доступ к данным другого пользователя (например, к содержимому корзины) без необходимости входа в его аккаунт.</p>	6.5 (Medium)	A01:2021-Broken Access Control	<p>Хранить bid текущей сессии на сервере, а не на клиенте;</p> <p>Заменять предсказуемые значения id на криптографически стойкие UUID;</p> <p>Внедрить проверку прав доступа на уровне каждого запроса.</p>
Отсутствие требований по отношению к слабым паролям (CWE-521)	<p>Атака, основанная на подборе паролей. Чаще всего возникает, если приложение использует однофакторную аутентификацию для пользователей.</p> <p>При этом в системе нет никаких ограничений на создание пароля при регистрации нового юзера, поэтому могут</p>	9.8 (Critical)	A07:2021-Identification and Authentication Failures	<p>Использовать обязательной 2FA;</p> <p>Ограничение количества попыток на ввод верного пароля;</p> <p>Добавить обязательные требования к сложности паролей</p>

	использоваться любые, даже самые примитивные и ненадёжные пароли, чем легко можно воспользоваться (так, например, пароль админа сайта - admin123).			
JWT-уязвимость	<p>Уязвимость, позволяющая злоумышленнику повысить уровень своих прав в системе, тем самым, получая доступ к защищённым и важным данным путём манипуляций с JWT-токеном.</p> <p>С помощью DevTools браузера зарегистрированный пользователь может получить свой JWT-токен, декодировать его и на основании полученного результата изменить свой токен, выдав себе дополнительные права (заменив role на "admin" и включив проверку подписи)</p>	9.1 (Critical)	A02:2021- Crypto-graphic Failures	<p>Использовать механизм Access-Renew токенов;</p> <p>Явно запретить обработку неподписанных токенов;</p> <p>Валидировать подпись на стороне сервера;</p> <p>Разрешение использования только определённых алгоритмов (например, RS256, HS256).</p>

## Общие рекомендации по устранению рисков

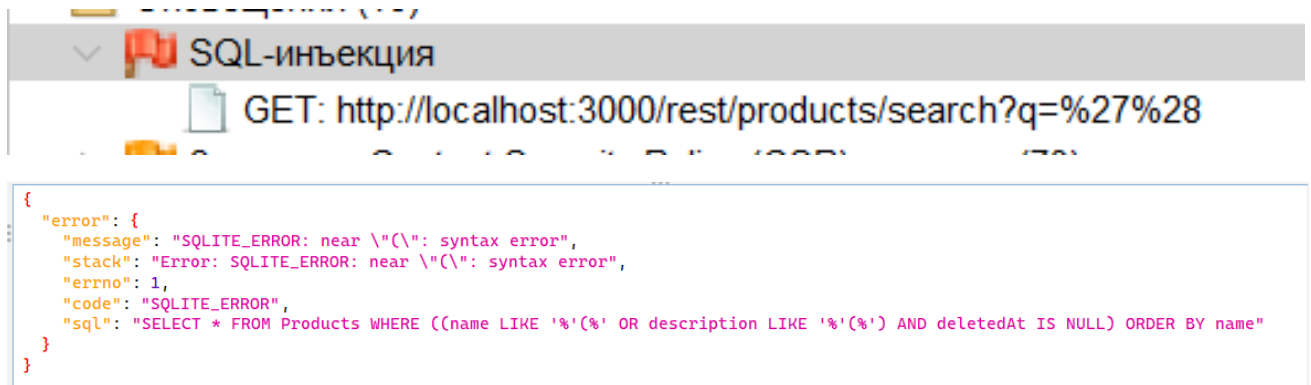
1. Внедрить единый строгий фреймворк для контроля доступа на основе роли и контексте пользователя.

2. Вся критичная бизнес-логика (включая проверку прав доступа, валидацию и санацию данных, должна выполняться на стороне сервера).
3. Реализовать многоуровневую защиту аутентификации.
4. Установить безопасные заголовки HTTP, политики и заголовки (CSP, HttpOnly, X-Frame-Options).

## Скриншоты, подтверждающие наличие уязвимостей

## 1. SQL-инъекция.

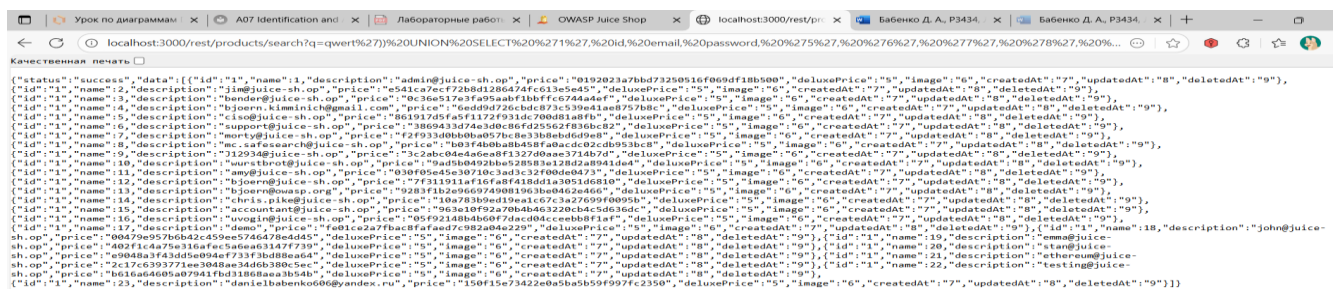
OWASP ZAP даёт подсказку, где можно найти SQL-инъекцию, а именно в параметре q при переходе на адрес /rest/products.



Также тут видно, каким образом происходит запрос к базе данных, и поэтому, зная кол-во столбцов в БД с продуктами, в текст запроса возможно вставить вредоносный SQL-скрипт.

Например, вставив данную строку параметр q, можно получить список всех юзеров вместе с их закодированными паролями:

```
qwert')) UNION SELECT '1', id, email, password, '5', '6', '7', '8', '9' FROM
Users--
```

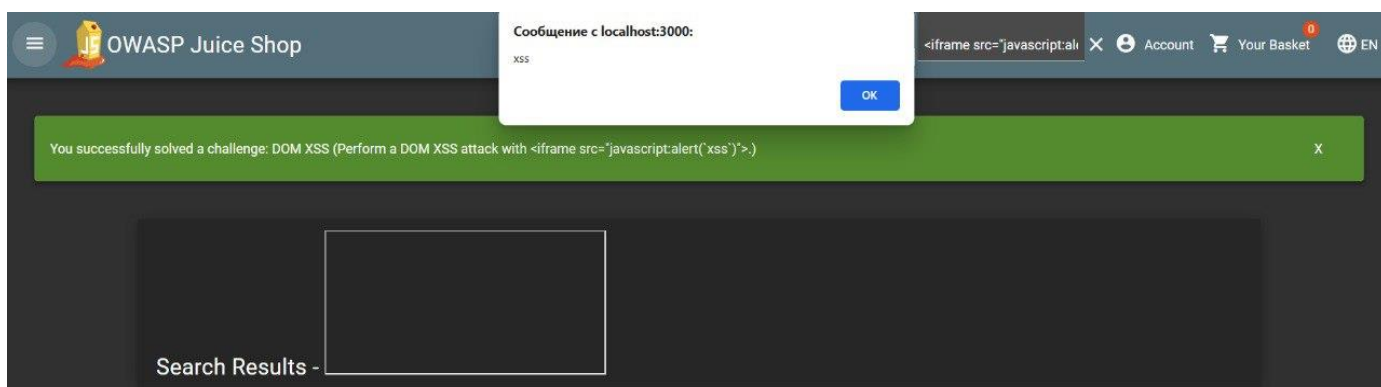


Примечание: так как я не являюсь экспертом в хакинге, то в подтверждении большинства уязвимостей в целях самопроверки я сверялся с заданиями в Score Board.

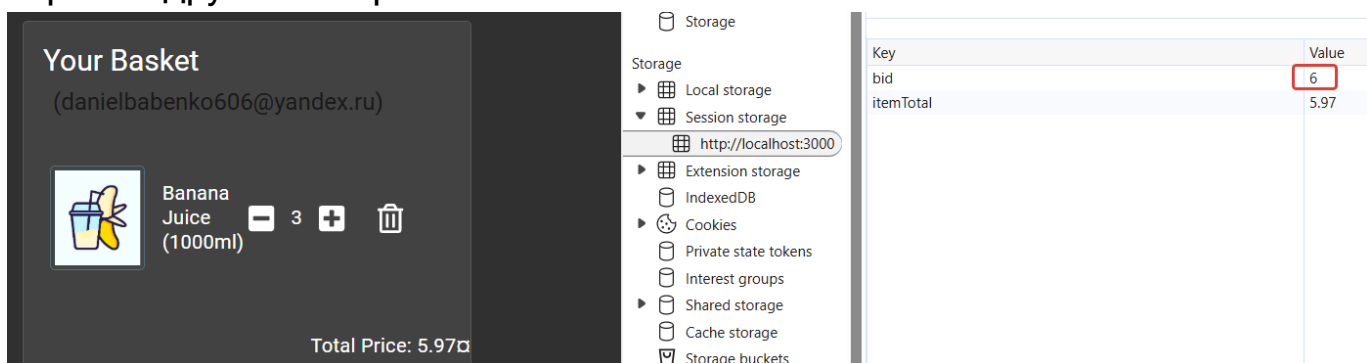


2. XSS - базовый пример, который показывает, как можно вставить собственный скрипт на сайт через строку поиска (DOM XSS).

Необходимо вставить в поисковую строку данный запрос: `<iframe src="javascript:alert('xss')">`.



3. В качестве проверки уязвимости, принадлежащей категории Broken Access Authentication попытаемся просмотреть содержимое корзины другого юзера.



Заменив вручную bid текущего юзера, осуществляю перезаход в корзину и, по сути, получаю доступ к корзине другого юзера.

You successfully solved a challenge: View Basket (View another user's shopping basket.)

Your Basket

(danielbabenko606@yandex.ru)

Total Price: 00

Checkout

You will gain 0 Bonus Points from this order!

Storage

Local storage

Session storage

http://localhost:3000

Extension storage

IndexedDB

Cookies

Private state tokens

Interest groups

Shared storage

Cache storage

Storage buckets

Background services

Back/forward cache

Background fetch

Background sync

Bounce tracking miti...

Key	Value
bid	7
itemTotal	5.97

4. Пароль администратора сайта является до смешного простым, так что злоумышленникам не составит труда войти в аккаунт администратора простым перебором паролей (можно использовать также и SQL-инъекцию, но ей уже уделили внимание ранее).

Login

Email\*

admin@juice-sh.op

Password\*

admin123

Forgot your password?

Log in

☐ Remember me

or

Log in with Google

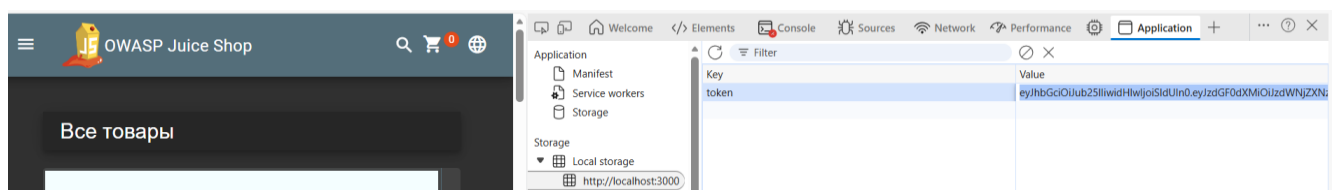
Not yet a customer?

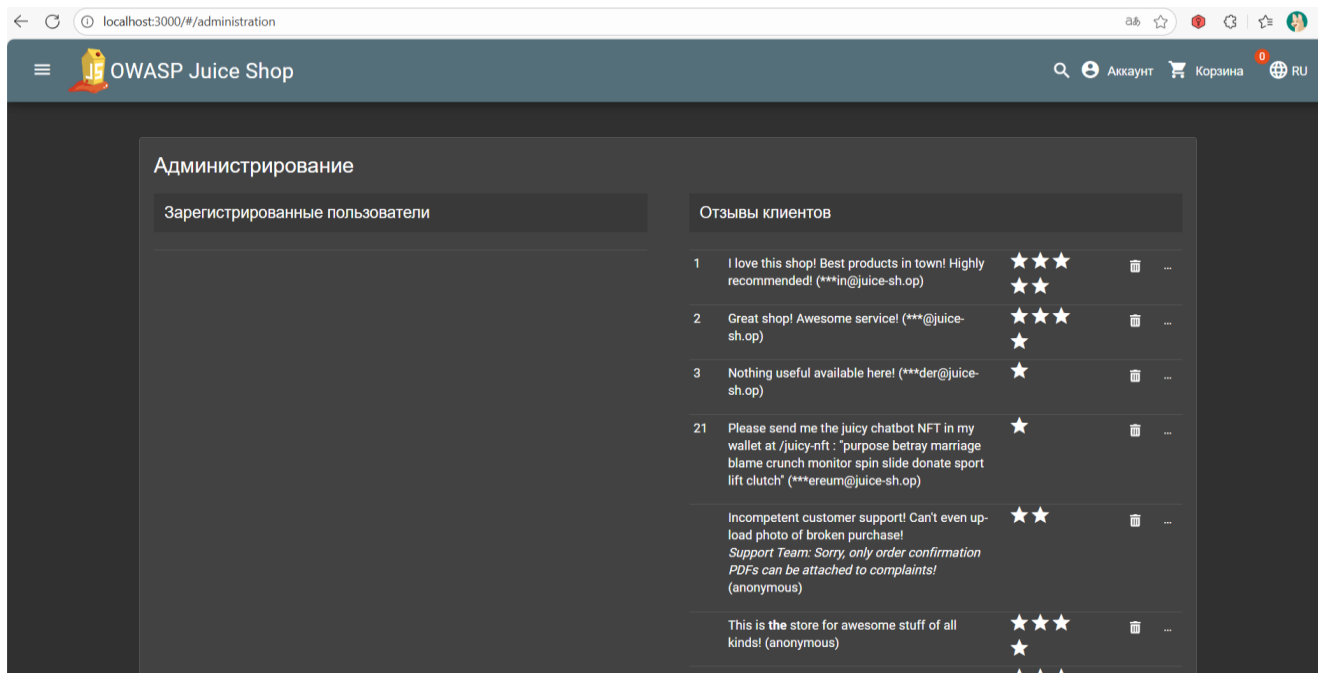


## DECODED PAYLOAD

```
JSON CLAIMS TABLE COPY ↗  
{  
  "status": "success",  
  "data": {  
    "id": 23,  
    "username": "",  
    "email": "danielbabenko606@yandex.ru",  
    "password": "150f15e73422e0a5ba5b59f997fc2350",  
    "role": "customer",  
    "deluxeToken": "",  
    "lastLoginIp": "",  
    "profileImage": "/assets/public/images/uploads/default.svg",  
    "totpSecret": "",  
    "isActive": true,  
    "createdAt": "2025-11-08 15:43:23.628 +00:00",  
    "updatedAt": "2025-11-08 15:49:45.231 +00:00",  
    "deletedAt": null  
  },  
  "iat": 1762782817  
}
```

Затем кодируем токен назад, изменяя значение `role` на `“admin”`, а также меняя значения поля алгоритма проверки подписи на `“none”`, и вставляем данный токен в качестве токена юзера.





## Отчёт OWASP ZAP

<https://github.com/DanielBabenko/Infobez-LR3-Babenko/blob/main/2025-11-12-ZAP-Report-.html>