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

По дисциплине «Информационная безопасность»

Лабораторная работа №4

Аудит безопасности веб-приложения

Студент:
Дениченко Александр Олегович P3412
Практик:
Маркина Татьяна Анатольевна

Санкт-Петербург
2025 г.

Цель

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

1 Вводная часть

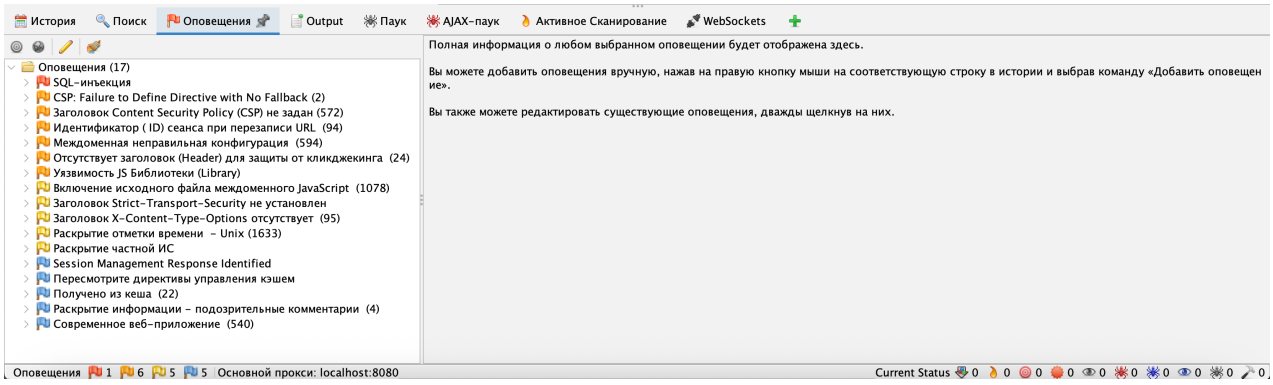
Установил OWASP Juice Shop и запустил его локально.

```
┌─[alexalex@Alexs-MacBook-Pro] - [~] - [4556]
└─[$] docker run --rm -p 3000:3000 bkimminich/juice-shop:snapshot    [20:15:54]
info: Detected Node.js version v22.19.0 (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 styles.css is present (OK)
info: Required file tutorial.js is present (OK)
info: Required file runtime.js is present (OK)
info: Required file main.js is present (OK)
info: Required file vendor.js 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
```

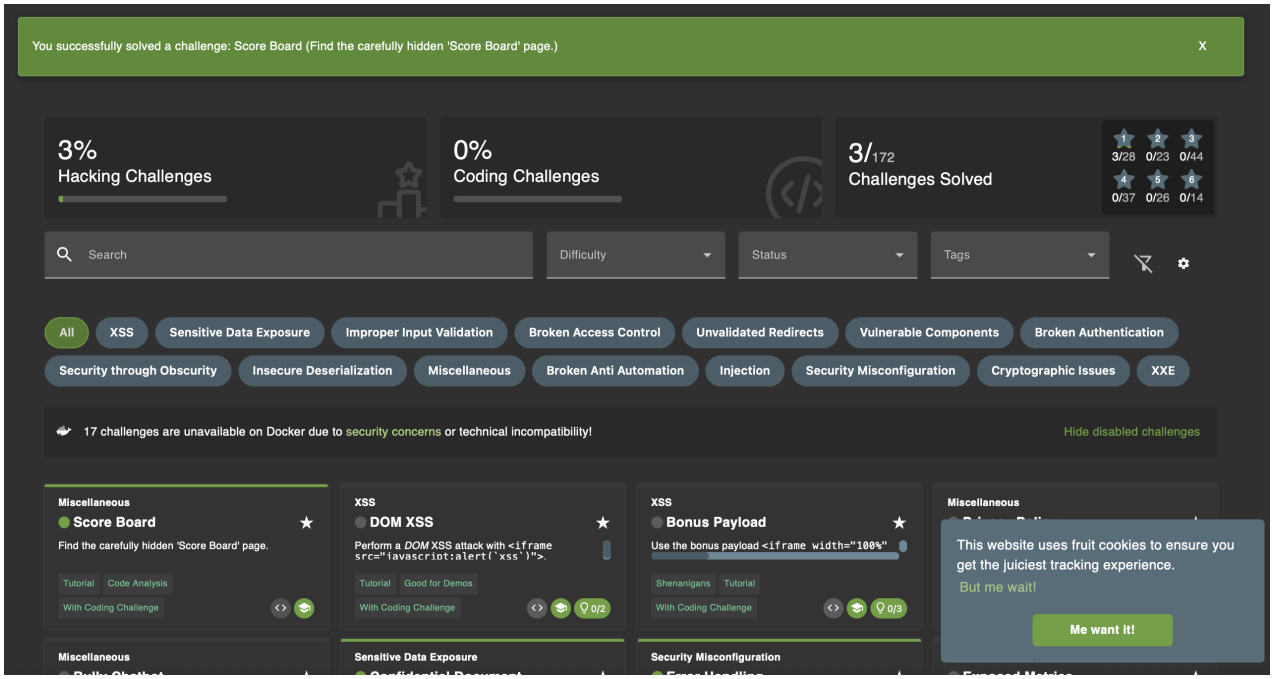
2 Первичное автосканирование

Настройки выставил – по стандарту использование традиционного паука + if modern – использование firefox. Далее сделана атака.

Сделал Атаку, получил следующие результаты:



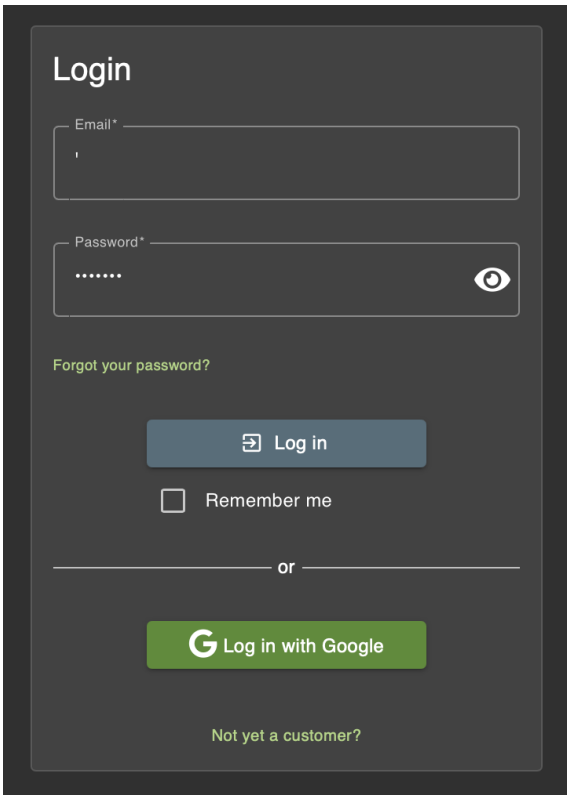
Было решено провести доп исследование через "Score Board чтобы получить больше уязвимостей уровня HIGH.



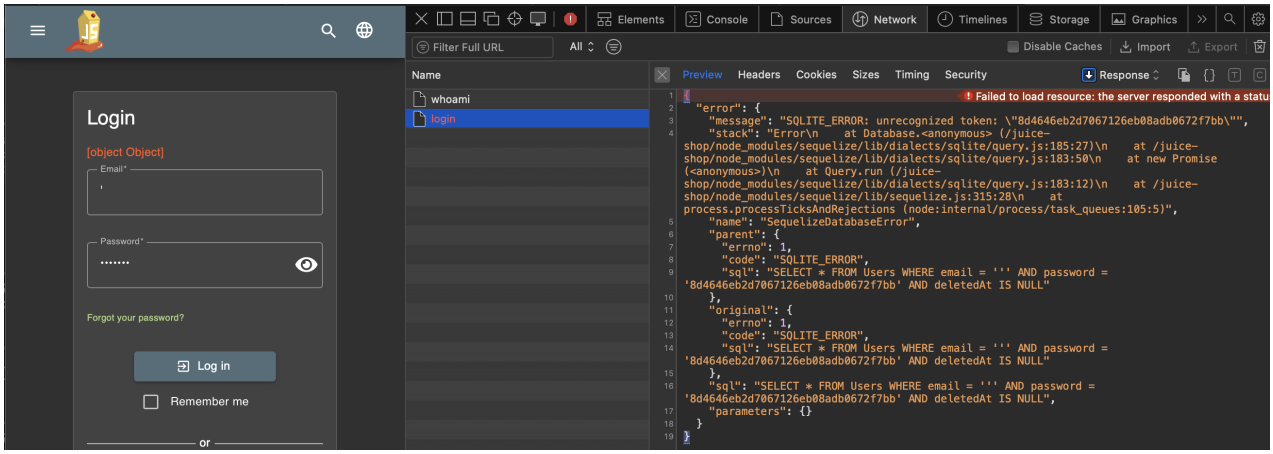
3 Нахождении и подтверждение уязвимостей

3.1 SQL Injection 1

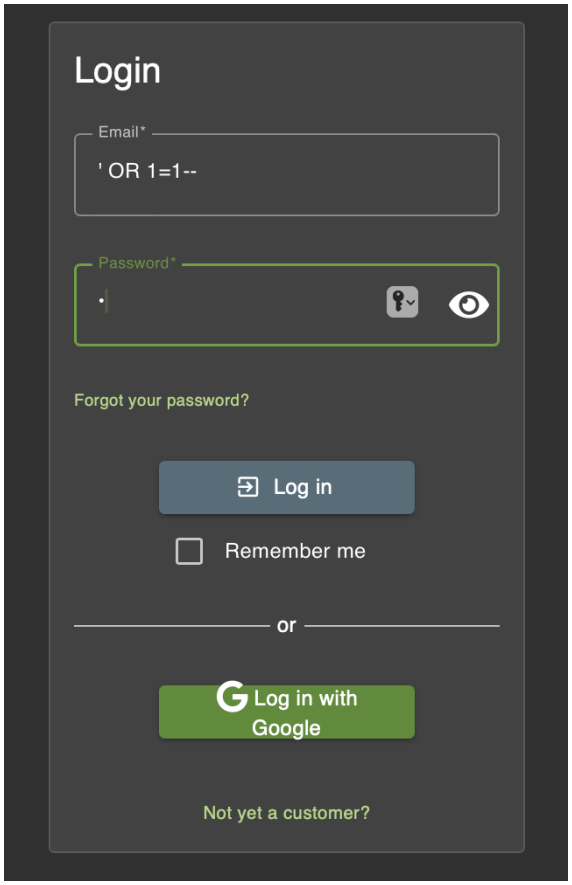
Попробуем залогиниться и ввести следующие данные:



Как можно увидеть получилось провести SQL Injection так как символ из логина не заэкранировался.



Зайдём под админом



Выполнится следующий запрос:

SELECT * FROM Users WHERE email = '' OR 1=1-- ' AND password = '...';

Успешно

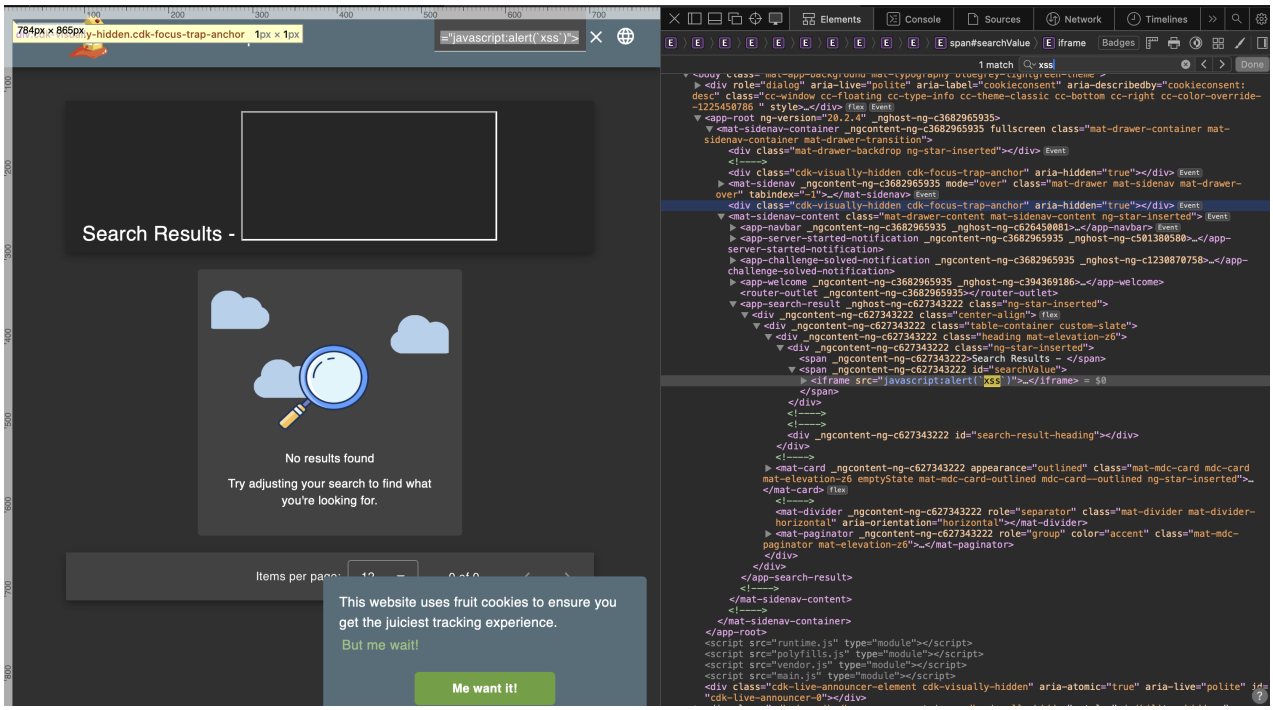
```
{
    "authentication": {
        "token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwi
ZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLX
NoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUw
MCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOi
IiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2Rl
ZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsIm
NyZWF0ZWRBdCI6IjIwMjUtMDktMjggMTc6MTY6NDAuNzcwICswMDowMCIsInVwZGF0
ZWRBdCI6IjIwMjUtMDktMjggMTc6MTY6NDAuNzcwICswMDowMCIsImRlbGV0ZWRBdC
I6bnVsbH0sImlhdCI6MTc1OTA4MzU0M30.CVhIHBuUHTcKWgK6zWB4tBVRY9VjGYJK
F6N1gaLqfsFBWXFNbtwY8P0aDi-
M9G5V3pjBDmxjsy6xWp9kFb0eKTCW9gj85L4RqnpqRw3YflgKlvEm8IV6MAo1XzImA
r2VZZba-wCVklgx8U0I6O60X5BC83WWL8Qc8w9lLehHmPQ",
        "bid": 1,
        "umail": "admin@juice-sh.op"
    }
}
```


3.2 XSS

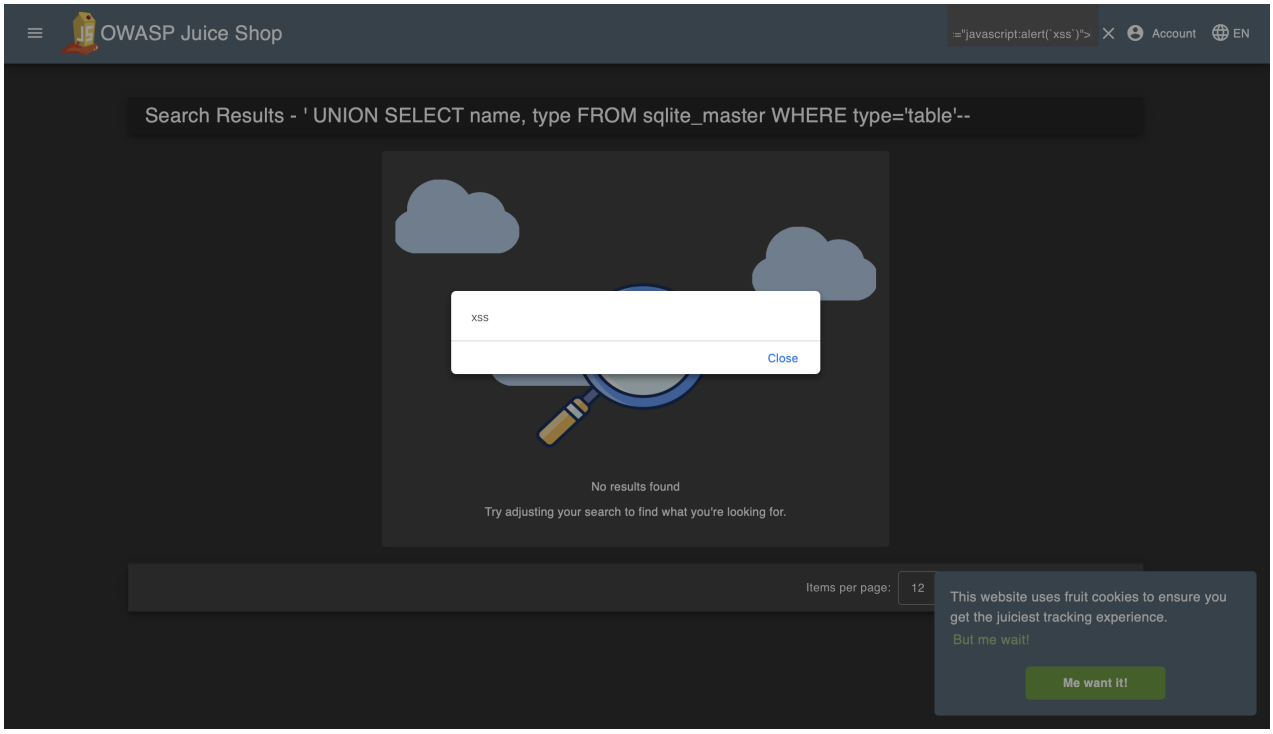
Попробуем ввести следующие данные поле поиска:

```
<iframe src="javascript:alert(`xss`)">
```

Наш код встроился в страницу.

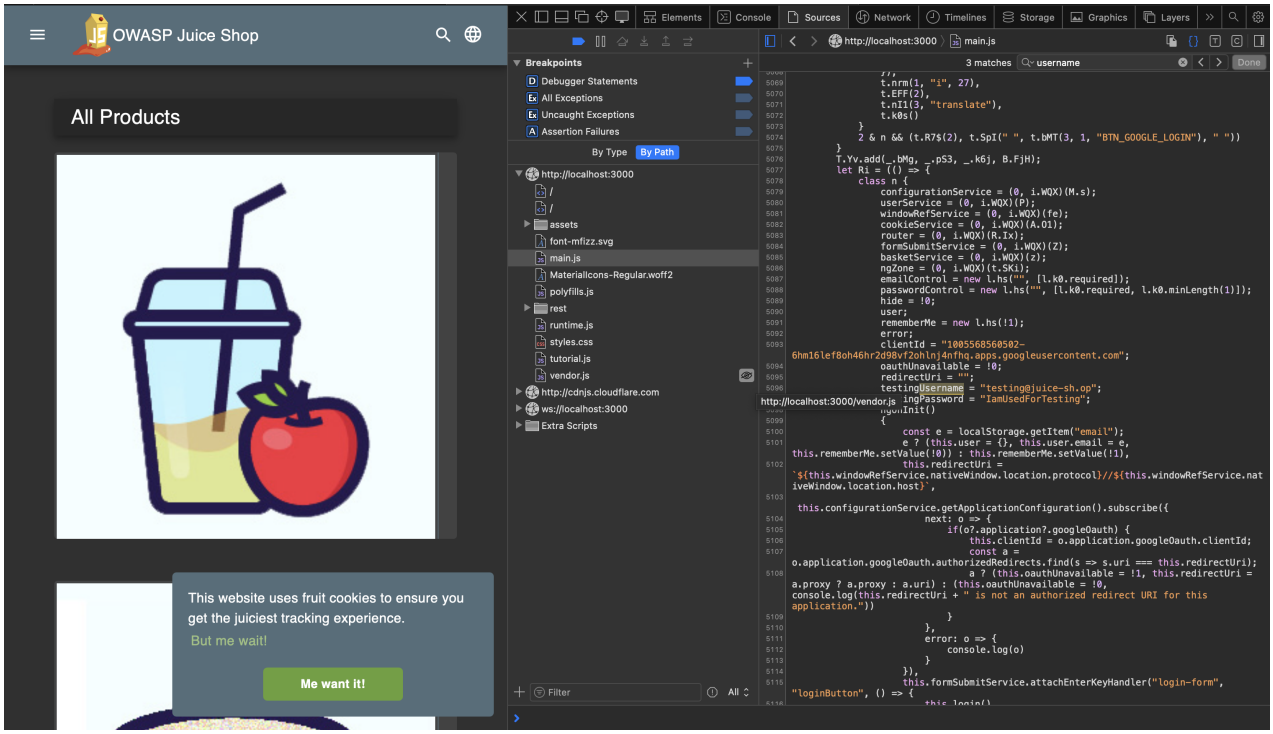


При открытии страницы теперь у всех пользователей будет выполняться XSS этого вида.

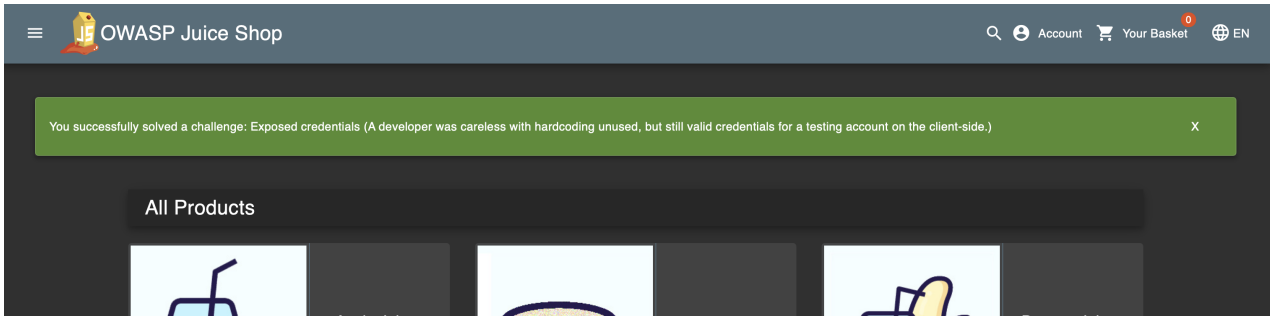


3.3 Sensitive Data Exposure. Exposed credentials

Попробуем получить пароль и логин из исходного кода. Перейдём во вкладку "Sources"в dev tools. И сделаем поиск по строке "username".



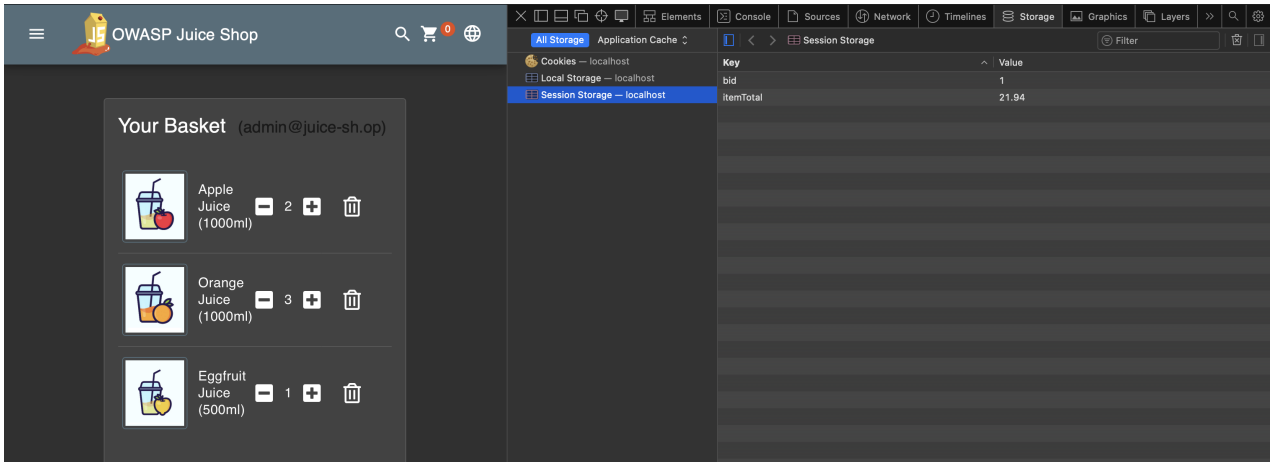
Мы видим, что пароль и логин находятся в файле "main.js"и его легко использовать для входа в систему. Проверим вход.



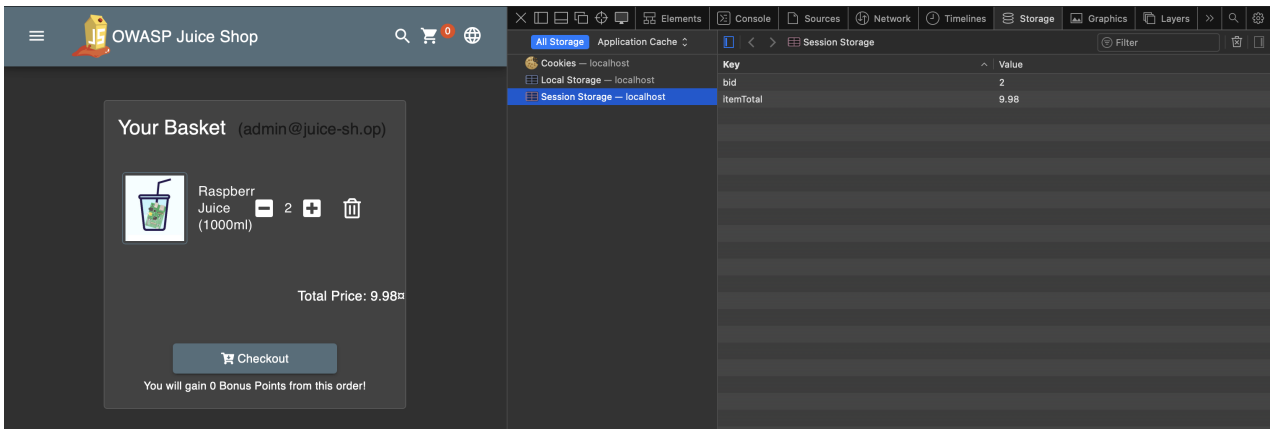
Всё работает.

3.4 Broken Access Control

Попробуем получить доступ к чужой корзине.



Сейчас мы в своей корзине, через локальное хранилище можно увидеть где хранится id корзины. Попробуем его поменять на id корзины другого пользователя.

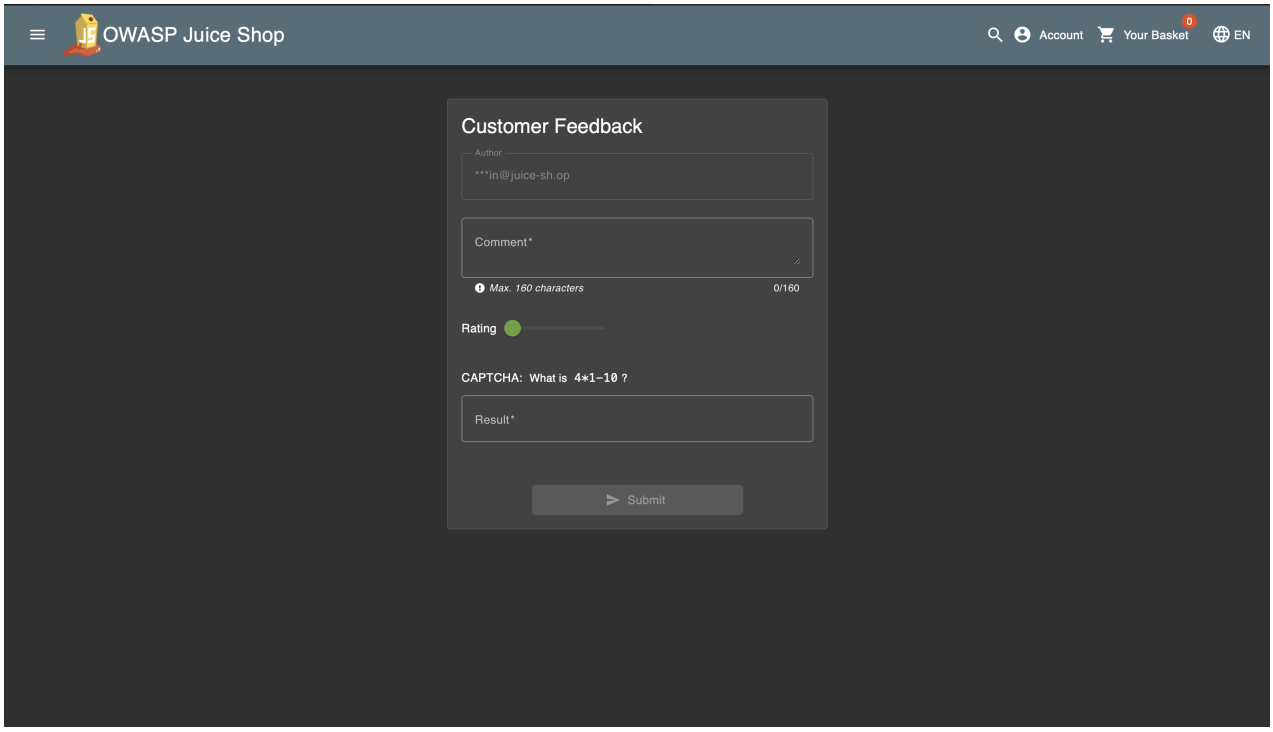


Мы успешно получили доступ к чужой корзине и можем её редактировать и далее вернуться в свою корзину.

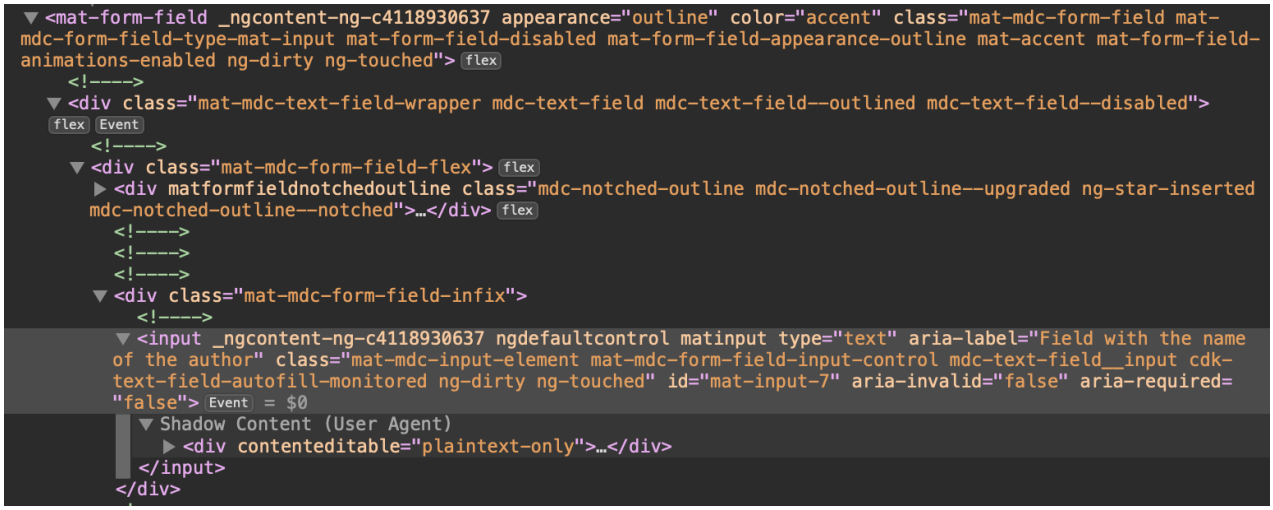
3.5 Broken Access Control

Попробуем оставить фидбэк от не своего имени, так как возможно, что логин отправителя фидбэка будет вычислен не на сервере, а на клиенте.

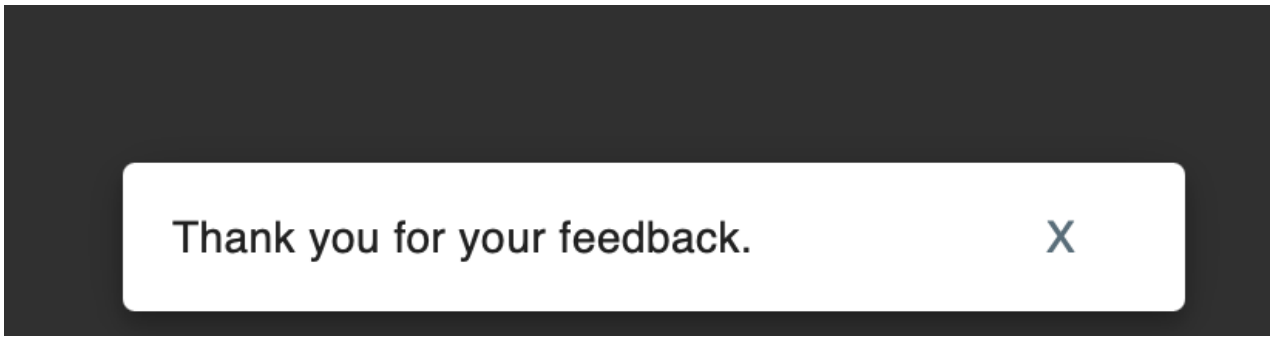
На данный момент видно, что поле с вводом логина для фидбэка отключено для ввода и там по умолчанию наш логин.



Попробуем ввести фидбэк от имени другого пользователя и для этого поменяем через dev tools дефолтный логин. Достаточно было отключить disable атрибут у input с логином и ввести другой логин.



Отправка фидбэка прошла успешно.

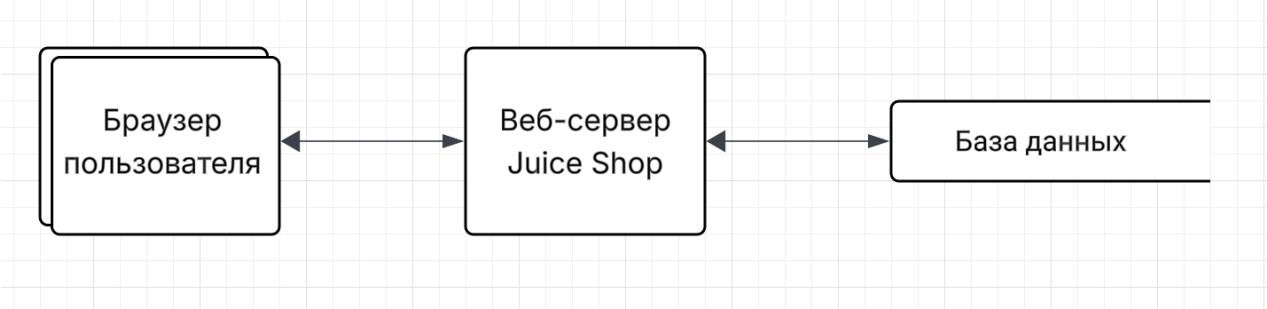


Мы отправили фидбэк от имени другого пользователя (anonimous). Вот ответ от сервера, что он сохранил в базу данных.

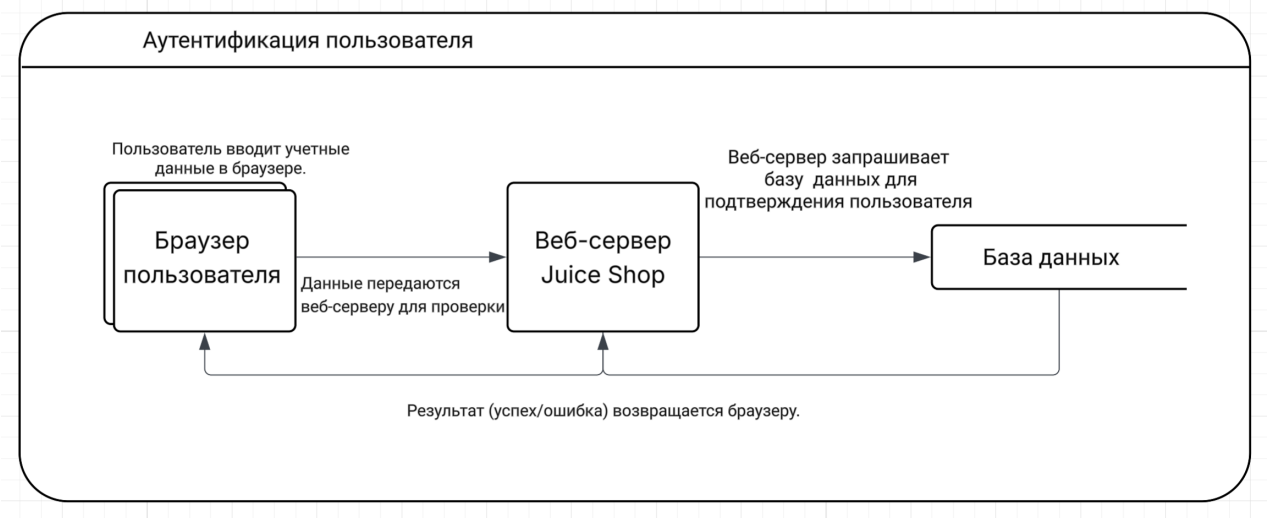
```
{
    "status": "success",
    "data": {
        "id": 11,
        "comment": "12312 (anonymous)",
        "rating": 2,
        "updatedAt": "2025-09-28T21:02:40.435Z",
        "createdAt": "2025-09-28T21:02:40.435Z",
        "UserId": null
    }
}
```

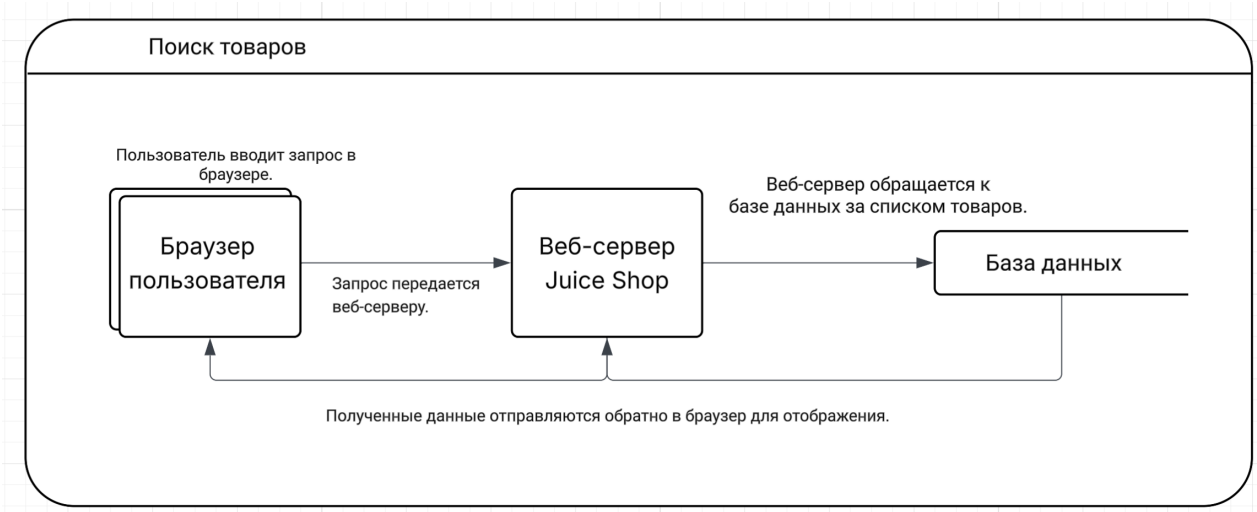

4 Data Flow Diagram — DFD

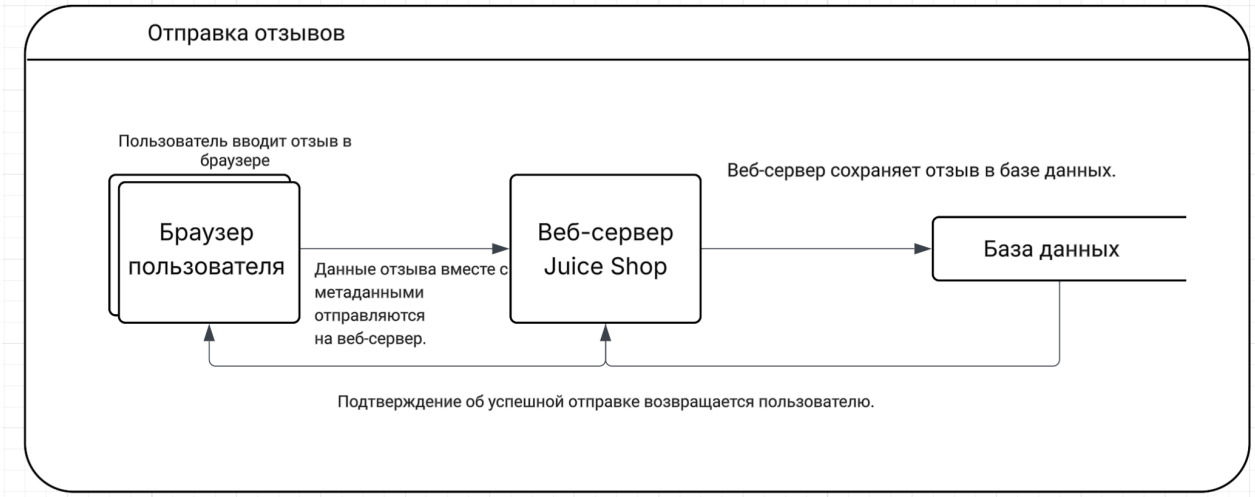
4.1 Диаграмма

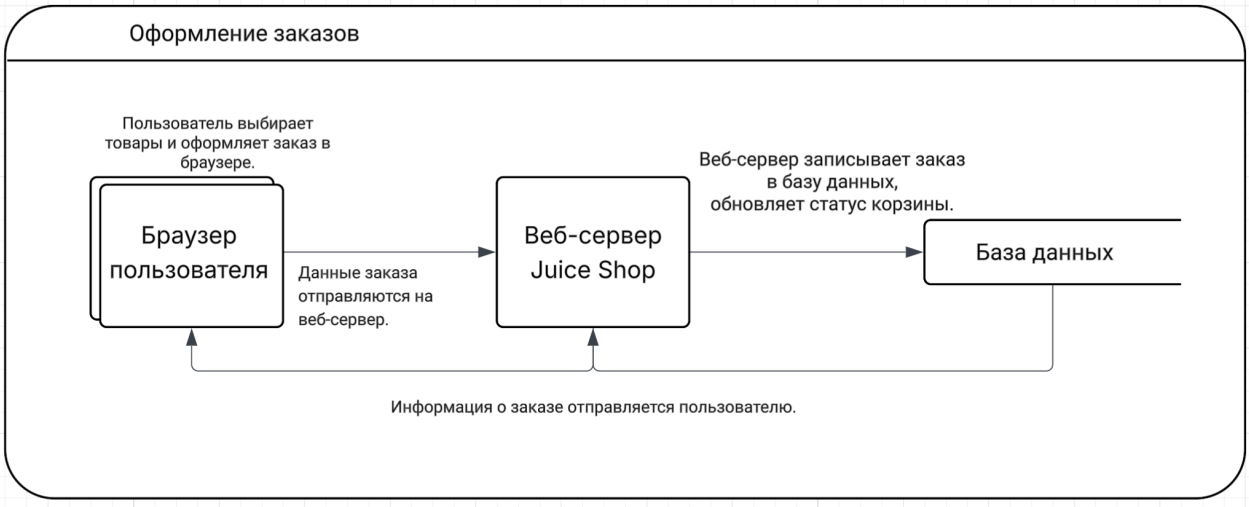


4.2 Потоки данных




Поиск товаров
Пользователь вводит запрос в браузере.
Браузер пользователя
Запрос передается веб-серверу.
Веб-сервер Juice Shop
Веб-сервер обращается к базе данных за списком товаров.
База данных
Полученные данные отправляются обратно в браузер для отображения.


Отправка отзывов
Пользователь вводит отзыв в браузере
Браузер пользователя
Данные отзыва вместе с метаданными отправляются на веб-сервер.
Веб-сервер Juice Shop
Веб-сервер сохраняет отзыв в базе данных.
База данных
Подтверждение об успешной отправке возвращается пользователю.


Оформление заказов
Пользователь выбирает товары и оформляет заказ в браузере.
Браузер пользователя
Данные заказа отправляются на веб-сервер.
Веб-сервер Juice Shop
Веб-сервер записывает заказ в базу данных, обновляет статус корзины.
База данных
Информация о заказе отправляется пользователю.

5 Анализ угроз по методике STRIDE

5.1 Spoofing (Маскировка)

- **SQL Injection в аутентификации** - возможность входа под любым пользователем через SQL-инъекцию `OR 1=1-`
- **Подмена отправителя фидбэка** - возможность отправки фидбэка от имени другого пользователя через изменение параметров в DevTools

Описание: SQL-инъекция позволяет обойти аутентификацию и войти в систему под любым пользователем, включая администратора. Изменение параметров фидбэка через DevTools позволяет отправлять сообщения от имени других пользователей.

5.2 Tampering (Изменение данных)

- **Доступ к чужим корзинам** - возможность изменения содержимого корзины другого пользователя через изменение ID в localStorage
- **Подмена отправителя фидбэка** - возможность изменения автора фидбэка

Описание: Через изменение ID корзины в localStorage можно получить доступ к корзине другого пользователя и изменять её содержимое. Также можно изменять автора фидбэка, что приводит к искажению данных в системе.

5.3 Repudiation (Отказ от операций)

- **Отсутствие логирования действий** - нет возможности отследить, кто именно совершил действия в системе
- **Подмена отправителя фидбэка** - пользователь может отрицать отправку фидбэка, так как система не проверяет подлинность отправителя

Описание: Из-за отсутствия надлежащего логирования и проверки подлинности пользователей, невозможно доказать, кто именно совершил определенные действия в системе.

5.4 Information Disclosure (Раскрытие информации)

- **Учетные данные в исходном коде** - логин и пароль администратора хранятся в открытом виде в файле main.js
- **SQL Injection** - возможность получения доступа к данным всех пользователей через SQL-инъекции
- **Доступ к чужим корзинам** - возможность просмотра содержимого корзины других пользователей

Описание: Критические учетные данные администратора находятся в исходном коде приложения, что позволяет любому пользователю получить полный доступ к системе. SQL-инъекции позволяют извлекать данные из базы данных.

5.5 Denial of Service (Отказ в обслуживании)

- **XSS атаки** - выполнение произвольного JavaScript кода может привести к нестабильной работе приложения
- **SQL Injection** - некорректные SQL-запросы могут привести к перегрузке базы данных

Описание: XSS атаки могут нарушить работу пользовательского интерфейса, а SQL-инъекции могут вызвать перегрузку базы данных, что приведет к отказу в обслуживании.

5.6 Elevation of Privilege (Повышение привилегий)

- **SQL Injection в аутентификации** - получение прав администратора через SQL-инъекцию
- **Использование учетных данных из исходного кода** - вход под тестовым аккаунтом с найденными в коде учетными данными

Описание: SQL-инъекция позволяет обойти систему аутентификации и получить права администратора. Также учетные данные администратора в исходном коде позволяют любому пользователю получить полные права в системе.

6 Таблица уязвимостей

Калькулятор уровня риска использовал https://bdu.fstec.ru/calc3

Название	**Описание**	**Уровень риска (CVSS)**	**OWASP Top 10**	**Предложение по исправлению**
SQL Injection в аутентификации	Возможность обхода аутентификации через SQL-инъекцию `OR 1=1-` в поле логина. Позволяет войти под любым пользователем, включая администратора.	9.8 (Critical)	A03:2021 - Injection	Использовать параметризованные запросы, валидировать и экранировать пользовательский ввод на сервере
Stored XSS в поиске	Возможность внедрения произвольного JavaScript кода через поле поиска. Код сохраняется и выполняется для всех пользователей.	8.8 (High)	A03:2021 - Injection	Валидировать и экранировать пользовательский ввод, внедрить Content Security Policy (CSP)
Sensitive Data Exposure	Учетные данные (логин и пароль) хранятся в открытом виде в файле main.js	7.5 (High)	A02:2021 - Cryptographic Failures	Удалить учетные данные из исходного кода, использовать переменные окружения, внедрить безопасное хранение паролей
Broken Access Control - корзины	Возможность доступа к корзине другого пользователя через изменение ID в localStorage	6.5 (Medium)	A01:2021 - Broken Access Control	Проверять права доступа на сервере, не полагаться на клиентские данные для авторизации
Broken Access Control - фидбэк	Возможность отправки фидбэка от имени другого пользователя через изменение параметров в DevTools	5.3 (Medium)	A01:2021 - Broken Access Control	Проверять подлинность пользователя на сервере, использовать серверные сессии

Таблица 1: Таблица найденных уязвимостей

7 Рекомендации по устранению рисков

1. **Внедрить строгую валидацию данных:** Все пользовательские данные должны валидироваться и экранироваться на стороне сервера. Использовать параметризованные запросы для предотвращения SQL-инъекций.
2. **Реализовать Content Security Policy (CSP):** Ограничить выполнение JavaScript кода только из доверенных источников.

3. **Усилить систему аутентификации и авторизации:** Удалить учетные данные из исходного кода, реализовать проверку прав доступа на сервере для всех операций.
4. **Внедрить логирование и мониторинг:** Реализовать детальное логирование всех действий пользователей для обеспечения возможности аудита и обнаружения подозрительной активности. Добавить триггеры для отправки уведомлений на email администратора об странных дейсвиях.
5. **Регулярно обновлять зависимости:** Внедрить процесс регулярного обновления всех используемых библиотек для устранения известных уязвимостей.
6.
7. **Доработка пайплайнов:** Внедрить в пайпланы статические, динамические проверки на уязвимости.

Вывод

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