

Cybersecurity Internship Report

Intern Name: Ehmaan Shafqat

**Project Title: Strengthening Security Measures for a Web
Application**

Submitted to: Faizan Khan

Date: 29 May, 2025

Week 1: Security Assessment

Executive Summary

This report documents the Week 1 security assessment of the intentionally vulnerable web application OWASP Juice Shop. The objectives included:

- Setting up the test environment.
- Conducting automated and manual vulnerability scans.
- Identifying critical security flaws (XSS, SQLi).
- Recommending mitigations.

1. Setup the Application

For testing, we'll use a vulnerable Node.js app like:

OWASP Juice Shop (<https://github.com/juice-shop/juice-shop>)

This app is intentionally vulnerable, making them ideal for security testing.

Environment Configuration

Tool Used: OWASP Juice Shop (Node.js-based vulnerable app).

Setup Steps:

```
git clone https://github.com/juice-shop/juice-shop.git
```

```
cd juice-shop
```

```
npm install
```

```
npm start
```

```
C:\Users\PC-a>git clone https://github.com/juice-shop/juice-shop.git
Cloning into 'juice-shop'...
remote: Enumerating objects: 138040, done.
remote: Counting objects: 100% (84/84), done.
remote: Compressing objects: 100% (44/44), done.
remote: Total 138040 (delta 82), reused 40 (delta 40), pack-reused 137956 (from 3)
Receiving objects: 100% (138040/138040), 246.44 MiB | 1.65 MiB/s, done.
Resolving deltas: 100% (107910/107910), done.
Updating files: 100% (1155/1155), done.

C:\Users\PC-a>cd juice-shop
```

```

Run `npm audit` for details.
npm verbose cwd C:\Users\PC-a\juice-shop
npm verbose os Windows_NT 10.0.19045
npm verbose node v22.16.0
npm verbose npm v10.9.2
npm verbose exit 0
npm info ok

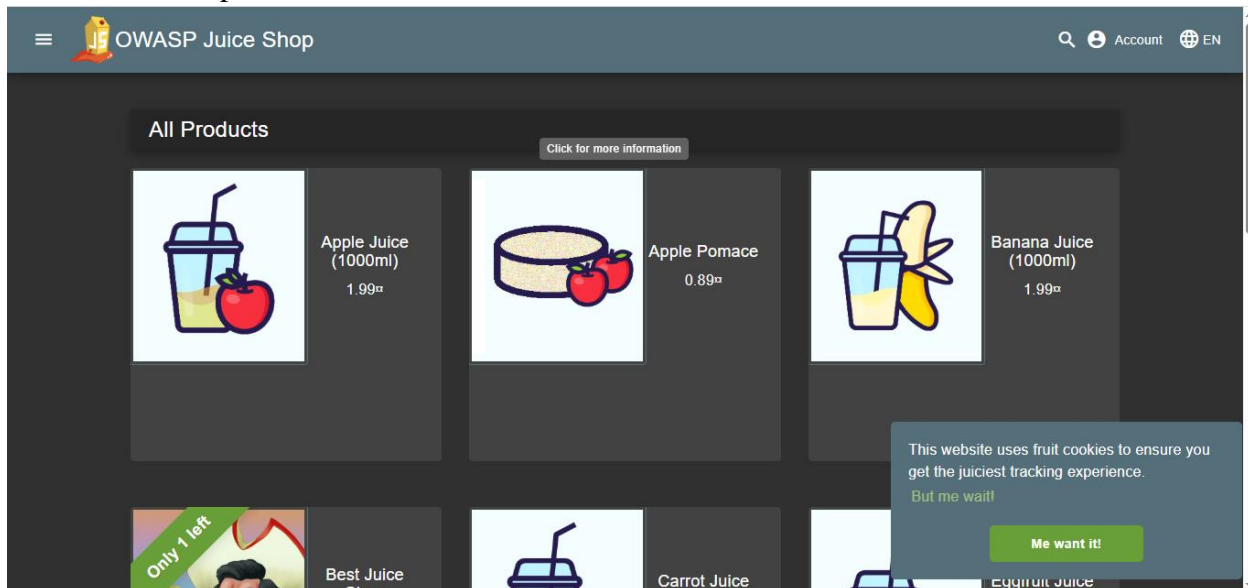
C:\Users\PC-a\juice-shop>npm start

> juice-shop@17.3.0 start
> node build/app

info: Detected Node.js version v22.16.0 (OK)
info: Detected OS win32 (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Entity models 19 of 19 are initialized (OK)
info: Required file server.js is present (OK)
info: Required file styles.css is present (OK)
info: Required file main.js is present (OK)
info: Required file tutorial.js is present (OK)
info: Required file index.html is present (OK)
info: Required file runtime.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

```

Access URL: <http://localhost:3000>



Functional Exploration

Tested Modules:

- User registration (/signup).
- Authentication (/login).

2. Vulnerability Assessment

Automated Scanning with OWASP ZAP

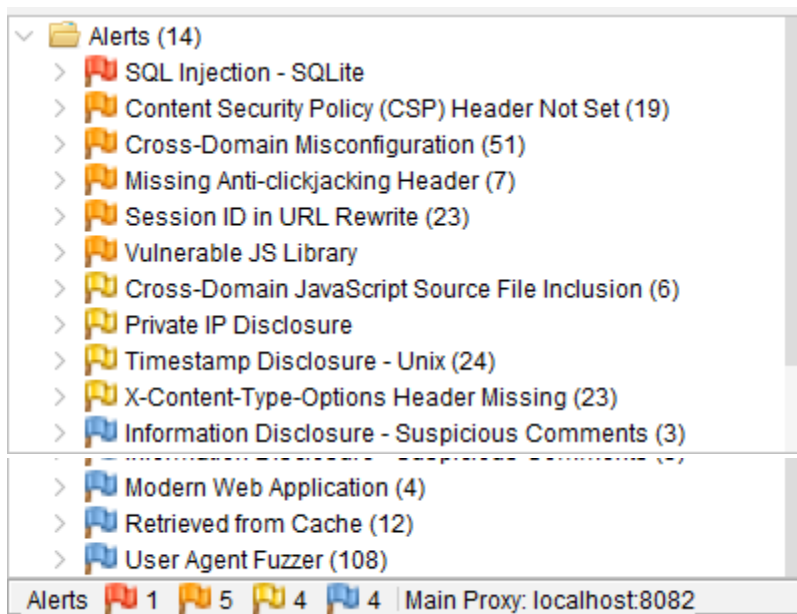
Scope: Full application crawl.

Findings:

Missing security headers (e.g., Content-Security-Policy).

Susceptible endpoints flagged for XSS and SQLi.

Screenshot:

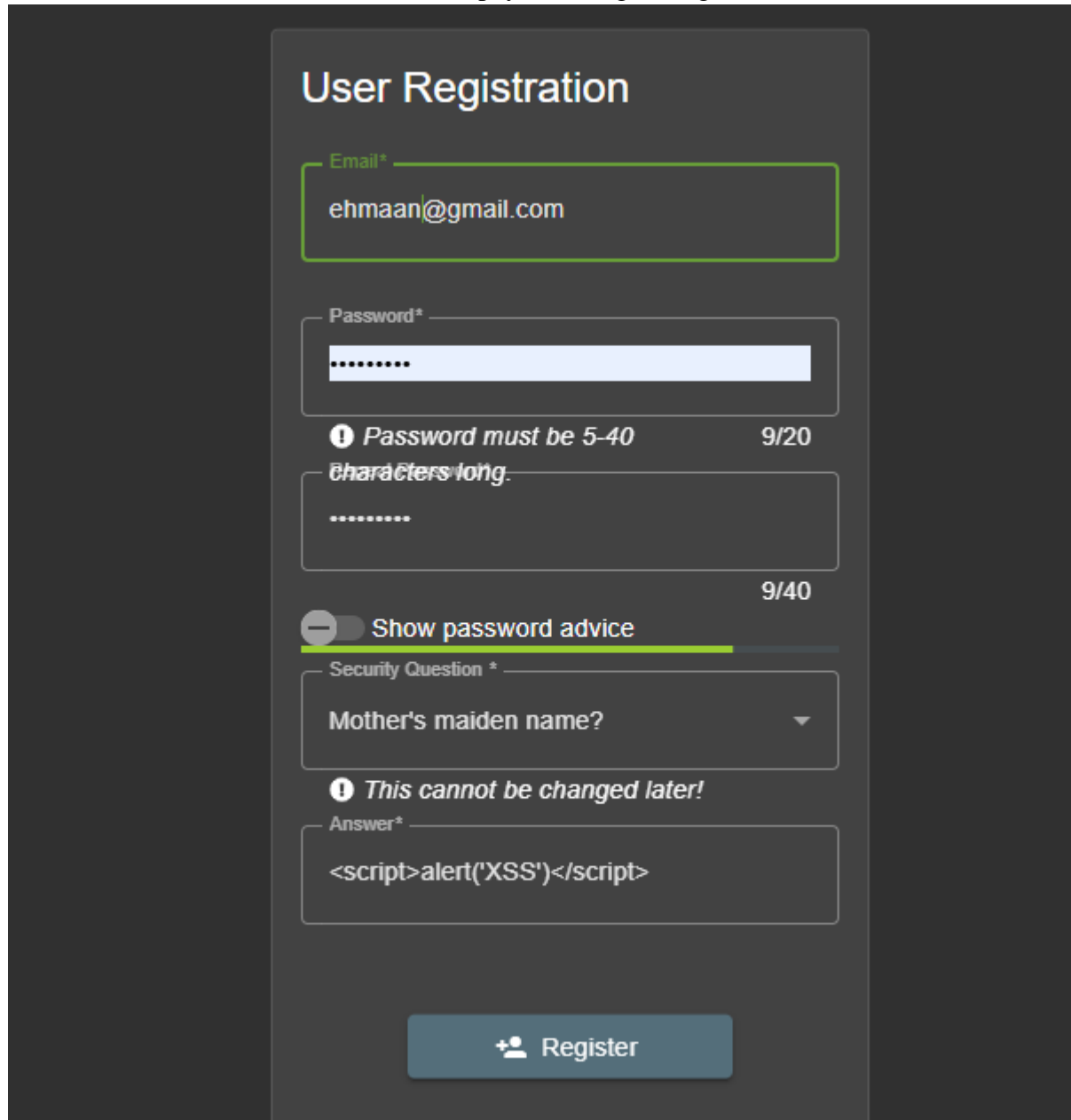


Manual Testing

A. Cross-Site Scripting (XSS)

Test Case:

- Payload: `<script>alert('XSS')</script>`
- Input Field: Search bar, comment section.
- Result: No alert triggered (input sanitization detected).
- Conclusion: Basic XSS mitigated.
- Recommendation: Test advanced payloads (e.g., ``).



The image shows a 'User Registration' form on a dark background. The form contains several input fields and a 'Register' button. The 'Email*' field contains 'ehmaan@gmail.com'. The 'Password*' field is masked with dots and has a strength indicator showing '9/20' and a message 'Password must be 5-40 characters long.' Below this is another masked field with a strength indicator of '9/40'. A toggle switch labeled 'Show password advice' is currently turned off. The 'Security Question *' dropdown menu is set to 'Mother's maiden name?'. Below this, a message states 'This cannot be changed later!'. The 'Answer*' field contains the XSS payload: `<script>alert('XSS')</script>`. At the bottom is a blue 'Register' button with a user icon.

User Registration

Email*
ehmaan@gmail.com

Password*
.....
! Password must be 5-40 characters long. 9/20

.....
9/40

— Show password advice

Security Question *
Mother's maiden name? ▼

! This cannot be changed later!

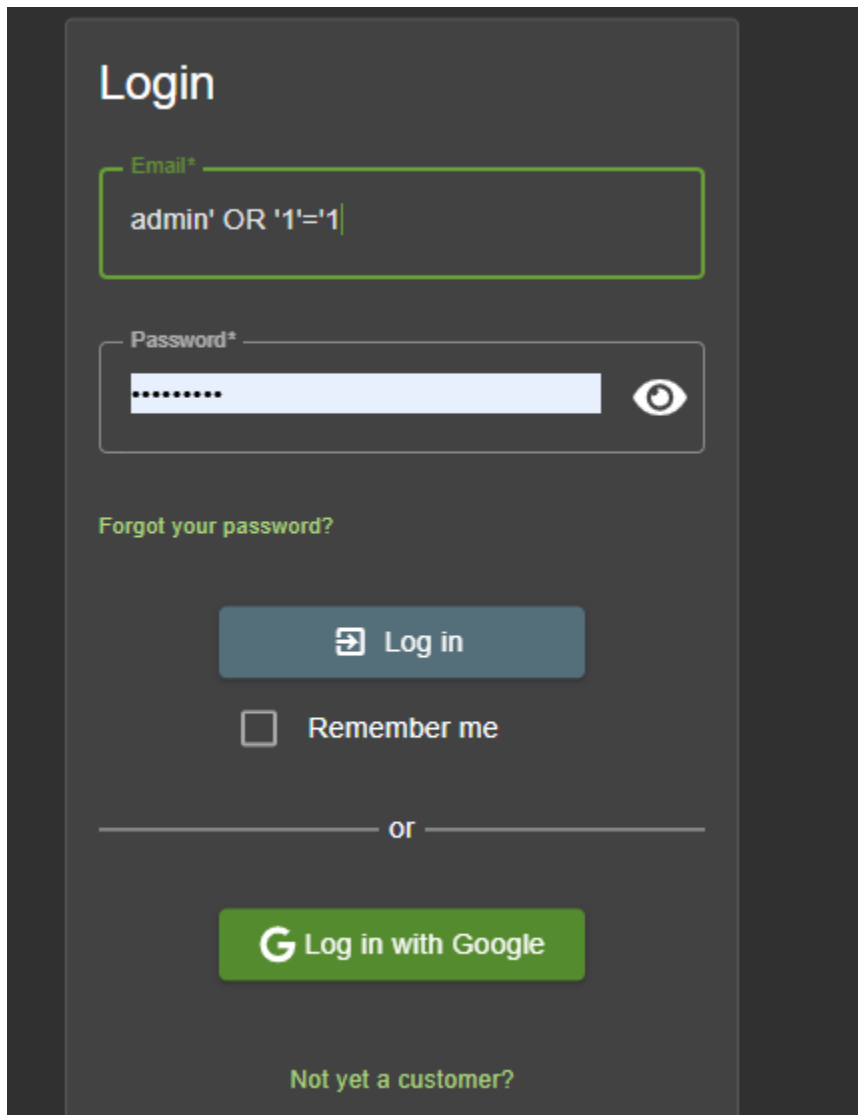
Answer*
<script>alert('XSS')</script>

+ Register

SQL Injection (SQLi)

Test Case:

- Payload: admin' OR '1'='1
- Location: Login form.
- Result: Successful authentication bypass.
- Severity: Critical (CWE-89).
- **Screenshot:**



Recommendation:

- Use parameterized queries (e.g., sequelize).
- Implement WAF rules.

Findings Summary:

Vulnerability	Location	Risk Level	Mitigation
SQL Injection	Login endpoint	Critical	Parameterized queries, input validation
Missing CSP	All pages	Medium	Add Content-Security-Policy header