2025

Developers Hub Corporation

Security Assessment Report Week 1

MUHAMMAD SHAHAB QAMAR

ID: DHC-3478

Cybersecurity Internship

Executive Summary

This report documents the Week 1 security assessment of the OWASP Juice Shop, a deliberately insecure web application designed for cybersecurity training. The assessment focused on identifying vulnerabilities in the signup, login, and profile functionalities accessible at **http://localhost:3000.**

Key findings include a critical SQL injection vulnerability in the login form, weak password storage using MD5 hashes, and security misconfigurations identified through automated scanning. While tested fields were not vulnerable to basic Cross-Site Scripting (XSS) attacks, further testing is recommended due to Juice Shop's design. This report details the tools used, vulnerabilities, their impacts, evidence, and remediation recommendations.

Application Overview

OWASP Juice Shop is an open-source web application intended to simulate real-world vulnerabilities for educational purposes. It was set up locally using **npm install** and **npm start**, and its core functionalities **signup, login**, **and profile pages** were explored at http://localhost:3000 to understand user flows and identify potential attack surfaces.

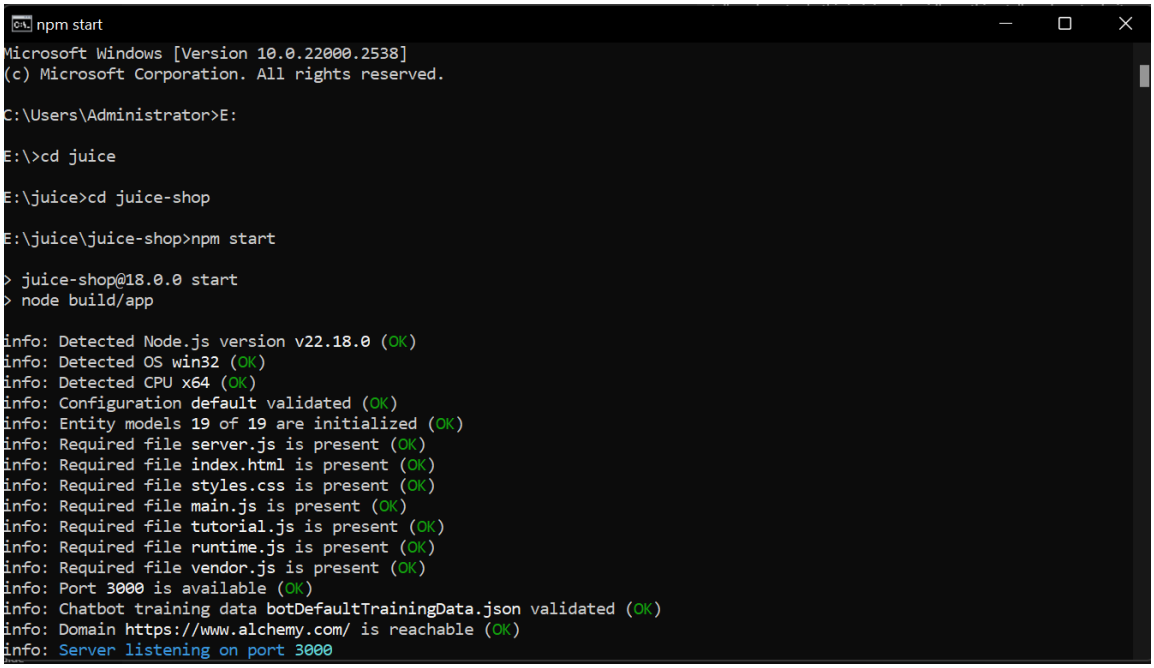


Fig 1 Starting the server

Tools Used

The following tools were utilized to conduct the security assessment:

- **OWASP ZAP**: An automated web application security scanner used to detect vulnerabilities such as SQL injection, XSS, and security misconfigurations.
- **Browser Developer Tools**: Utilized to inspect elements and simulate XSS attacks by injecting scripts into input fields (e.g., feedback form, profile bio, search).
- **Manual Input Testing**: Employed to test SQL injection vulnerabilities by crafting malicious inputs for the login form.
- **Database Inspection**: Conducted to analyze password storage mechanisms, either through direct database access or by observing application behavior.

Vulnerability Assessment Methodology

The assessment involved:

- Automated scanning with OWASP ZAP to identify common web vulnerabilities.
- Manual XSS testing by injecting scripts into input fields using browser developer tools.
- Manual SQL injection testing to attempt authentication bypass in the login form.
- Analysis of password storage practices to evaluate hashing mechanisms.

Findings

Finding #1: SQL Injection in Login Form

Description: The login form is vulnerable to SQL injection due to direct concatenation of user inputs into SQL queries, enabling authentication bypass without valid credentials.

Payload Used:

```
Username: admin' OR '1'='1--

Password: anything
```

Evidence:

- Successful login to the admin dashboard using the payload, bypassing authentication.

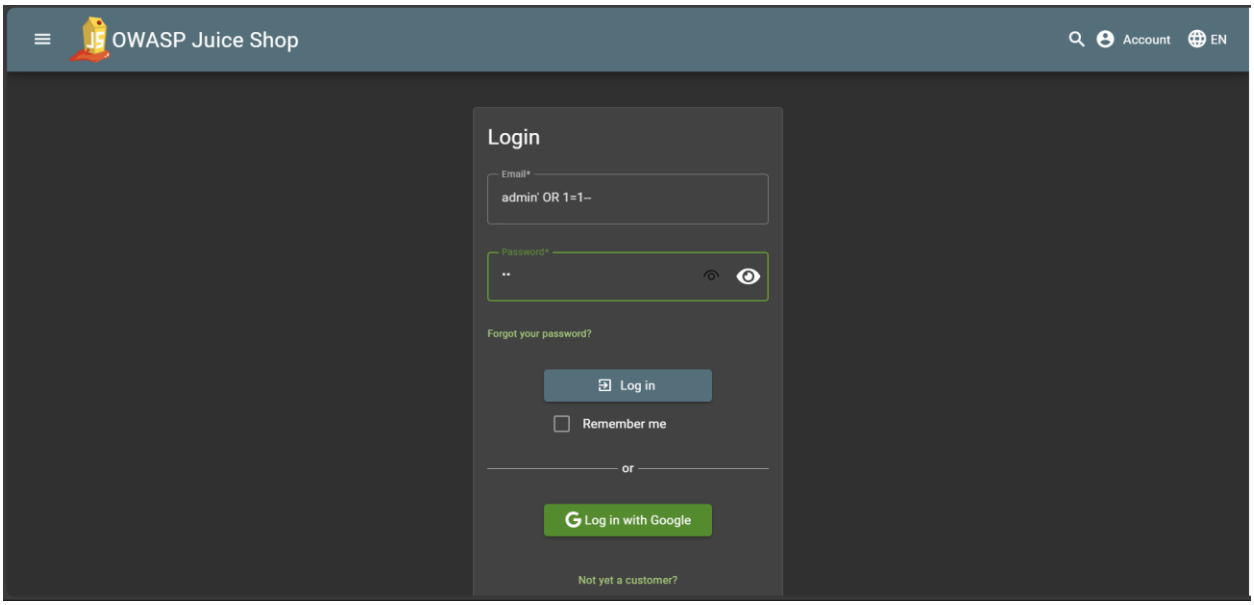


Fig 2 Trying to login

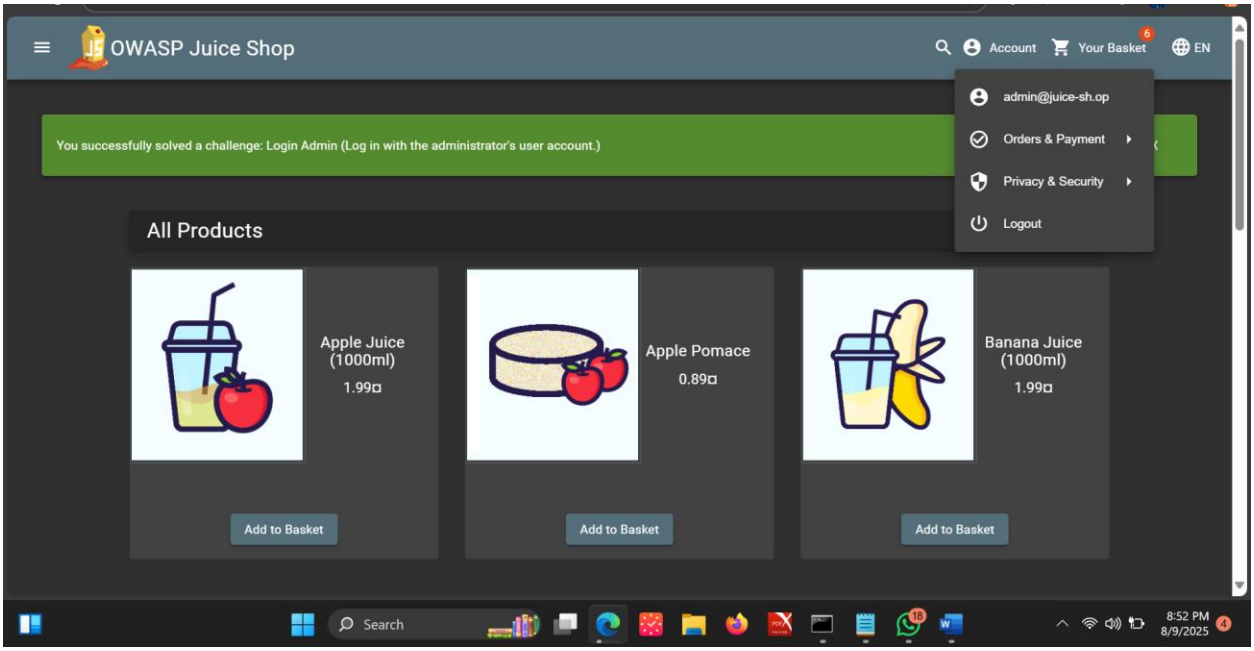


Fig 3 Successful login

Impact:

- Full authentication bypass, granting unauthorized access to admin-only functionalities.
- Potential exposure of sensitive user data or system compromises in a production environment. In real-world scenarios, this could lead to complete system takeover.

Recommendation:

- Implement parameterized queries or prepared statements to prevent SQL injection.
- Avoid direct string concatenation in SQL queries.
- Validate and sanitize all user inputs before processing.

Finding #2: Cross-Site Scripting (XSS)

Description: Input fields (feedback form, profile bio, search) were tested for XSS vulnerabilities using payloads such as **<script>alert('XSS');</script>** and **<script>alert(document.cookie)</script>.** No JavaScript execution occurred, indicating proper escaping or rejection of malicious inputs in these fields.

Tested Payloads:

```
<script>alert('XSS');</script>
```

```
<script>alert(document.cookie)</script>
```

Evidence:

- Inputs were rendered as plain text, with HTML tags escaped.

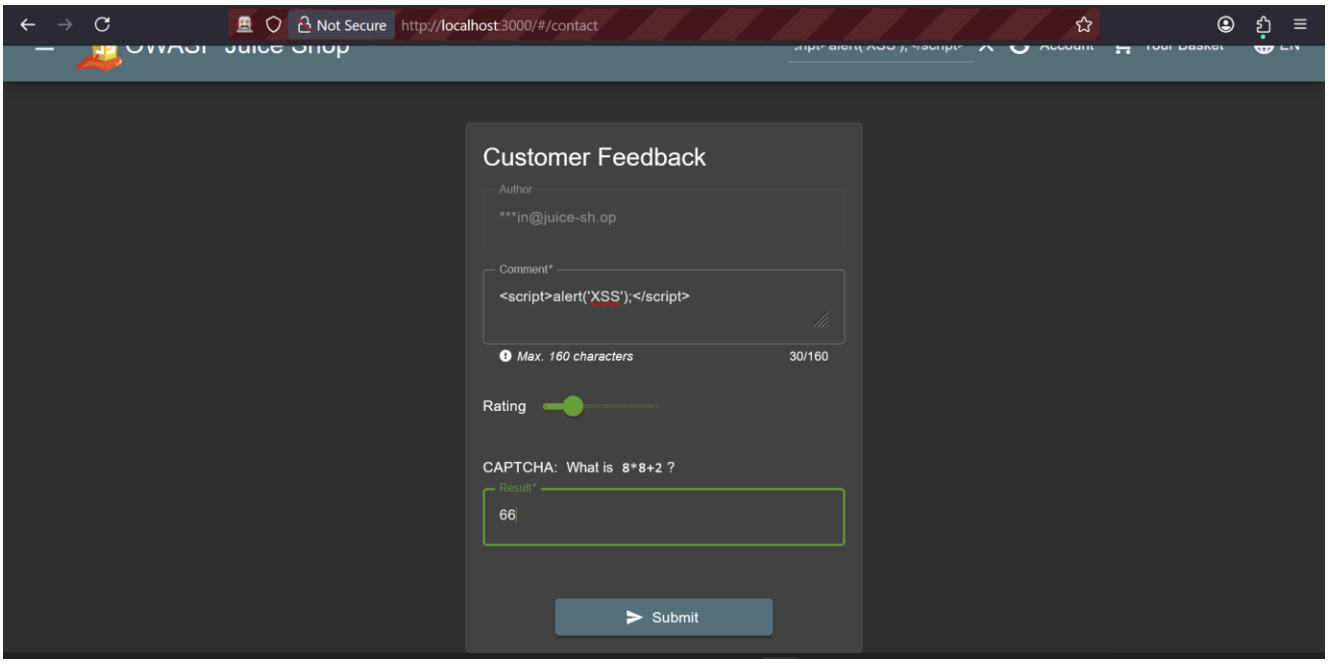


Fig 4 Trying the payload in Customer Feedback

This payload Didn't Worked

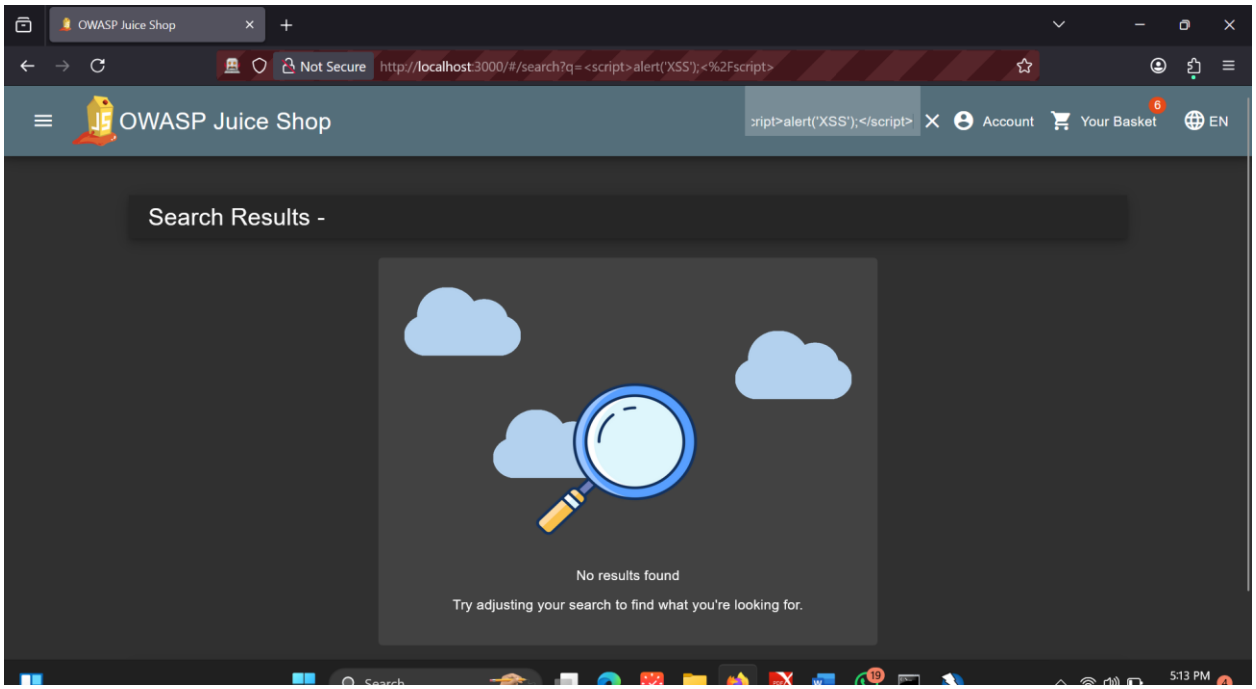


Fig 5 Trying the payload in Search

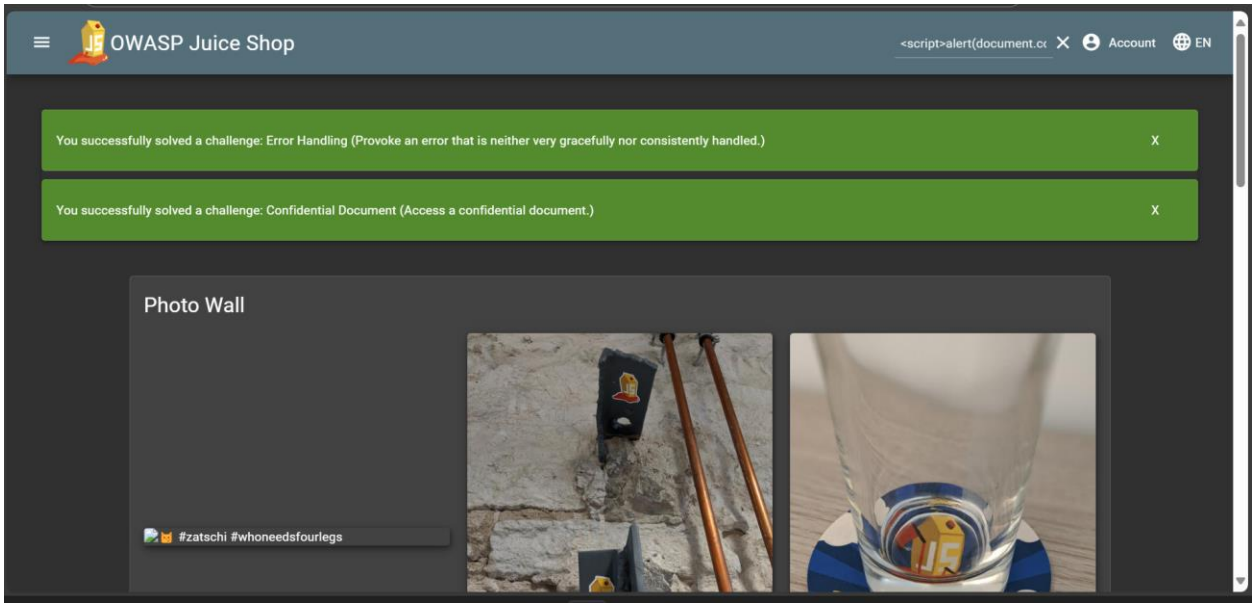


Fig 6 Trying the second payload in Customer Feedback

The payload For document revealed a flag

Impact:

- The tested fields are not vulnerable to basic reflected <script>alert('XSS');</script>

  The tested fields are vulnerable to <script>alert(document.cookie)</script>

- Proper input handling reduces the risk of malicious script execution in these areas.

Recommendation:

- Continue to sanitize and escape all user inputs across the application.
- Implement a Content Security Policy (CSP) to provide an additional layer of protection against XSS.

Finding #3: Weak Password Storage

Description: Passwords are stored using MD5 hashes, which are cryptographically weak and susceptible to rainbow table attacks. No salting was observed, and user roles are stored in plaintext, increasing the risk of credential compromise and targeted attacks if the database is exposed.

For this purpose I registered a user and observed the data stored in database using **SQL database**

Evidence:

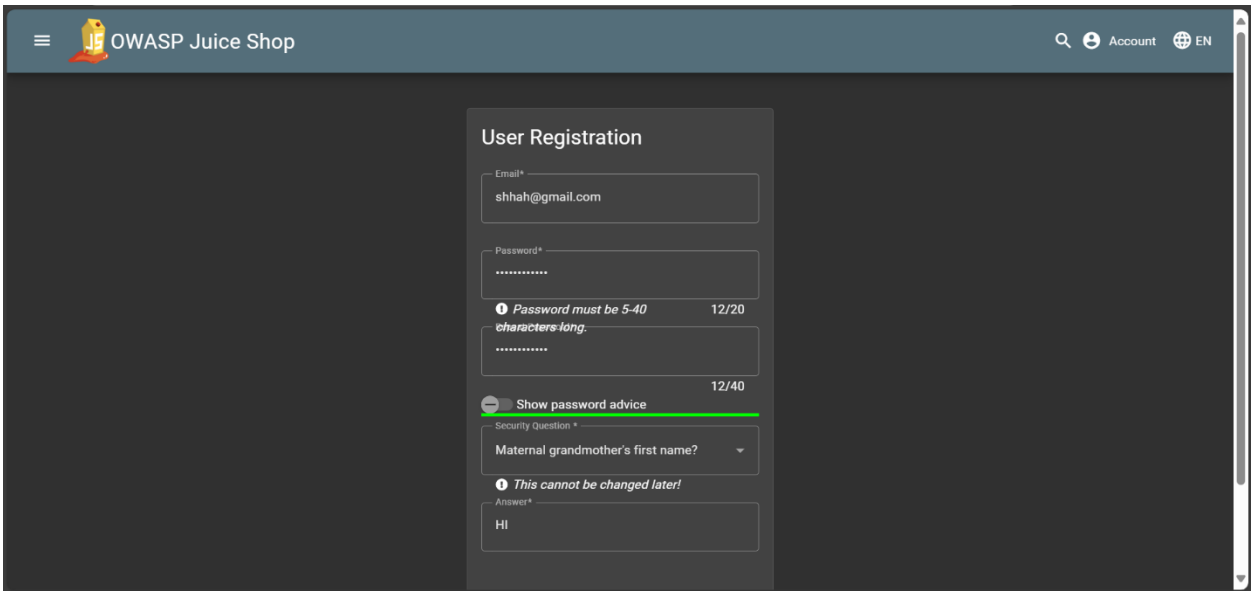


Fig 7 Creating the user

- Example hash: demo → fe01ce2a7fbac8fafaed7c982a04e229 (MD5). **Show in Fig 8**

- Database inspection or application behavior revealed unsalted MD5 hashes and exposed user roles.

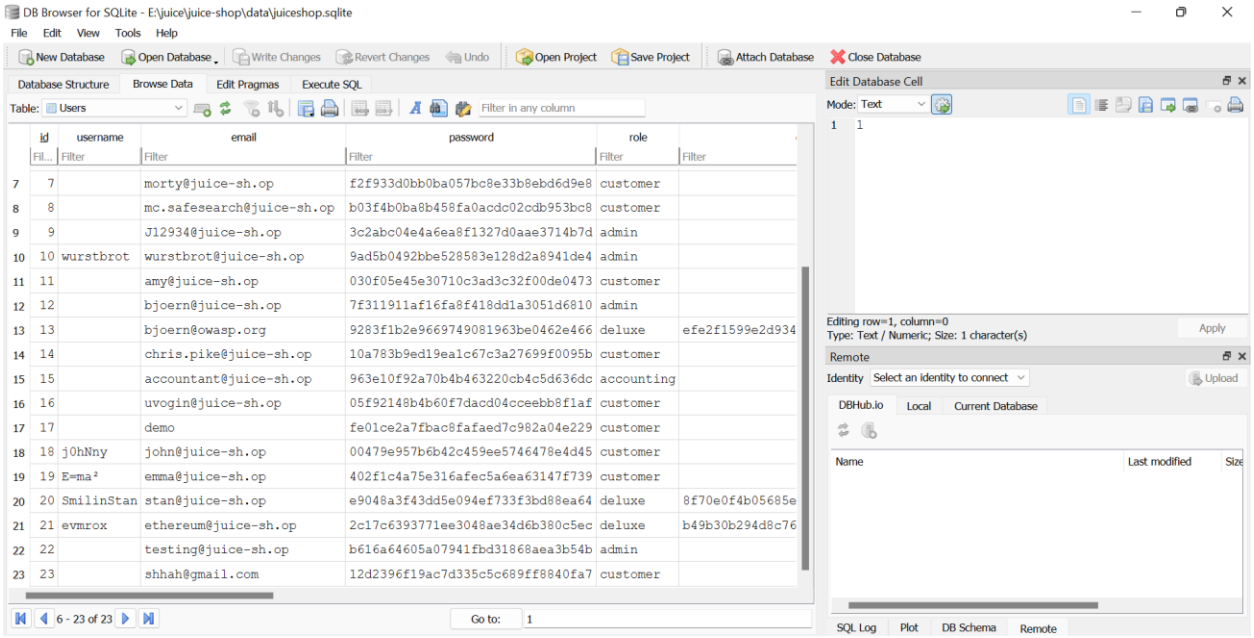


Fig 8

Impact:

- MD5 hashes can be cracked easily using precomputed rainbow tables, compromising user credentials.
- Exposed user roles could assist attackers in identifying high-value targets (e.g., admin accounts).
- Weak hashing undermines the security of the authentication system.

Recommendation:

- Upgrade password storage to bcrypt or Argon2 with proper salting.
- Discontinue the use of MD5 or SHA1 for password hashing in production systems.
- Minimize exposure of user roles in the database or application outputs.

OWASP AUTOMATED SCAN

Finding #4: Security Misconfigurations (OWASP ZAP Scan)

The **ZAP by Checkmarx** scan conducted on August 13, 2025, identified multiple security misconfigurations at **http://localhost:3000** and **http://cdnjs.cloudflare.com**, including:

- **Content Security Policy (CSP) Header Not Set** (Risk: Medium, Confidence: High): Missing **CSP** header on **GET http://localhost:3000/sitemap.xml**.
- **Missing Anti-clickjacking Header** (Risk: Medium, Confidence: Medium): Absence of **X-Frame-Options** header on **POST http://localhost:3000/socket.io/**.
- **Session ID in URL Rewrite** (Risk: Medium, Confidence: High): Session IDs exposed **POST http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PYZNVnp&sid=itvE-VwXdav6bu7AAAAQ**.
- **Cross-Domain Misconfiguration** (Risk: Medium, Confidence: Medium): Detected on **GET http://localhost:3000/robots.txt**.
- **Hidden File Found** (Risk: Medium, Confidence: Low): Potential hidden file at **GET http://localhost:3000/.hg**.
- **Vulnerable JS Library** (Risk: Medium, Confidence: Medium): Use of outdated **jQuery 2.2.4** at **http://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js**.
- **Cross-Domain JavaScript Source File Inclusion** (Risk: Low, Confidence: Medium): Detected on **GET http://localhost:3000/sitemap.xml**.
- **Private IP Disclosure** (Risk: Low, Confidence: Medium): Found on **GET http://localhost:3000/rest/admin/application-configuration**.
- **X-Content-Type-Options Header Missing** (Risk: Low, Confidence: Medium): Absent on **GET http://localhost:3000/socket.io/**.
- **Timestamp Disclosure - Unix** (Risk: Low, Confidence: Low): Detected on **GET http://localhost:3000/**.
- **Informational Findings**: **Information Disclosure - Suspicious Comments** (Confidence: Low), **Modern Web Application** (Confidence: Medium), and **Retrieved from Cache** (Confidence: Medium).

Evidence:

- **ZAP scan** report (generated August 13, 2025, at 17:24:01) detailed 13 alerts across **Medium**, **Low**, and **Informational** risk levels.
- Manual inspection via **Browser Developer Tools** confirmed missing headers like **X-Frame-Options** and **Content-Security-Policy**.
- I attached some here and the detailed reference will be in zip file of this Report

		Confidence				
		User Confirmed	High	Medium	Low	Total
Risk	High	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
	Medium	0 (0.0%)	2 (15.4%)	3 (23.1%)	1 (7.7%)	6 (46.2%)
	Low	0 (0.0%)	0 (0.0%)	3 (23.1%)	1 (7.7%)	4 (30.8%)
	Informational	0 (0.0%)	0 (0.0%)	2 (15.4%)	1 (7.7%)	3 (23.1%)
	Total	0 (0.0%)	2 (15.4%)	8 (61.5%)	3 (23.1%)	13 (100%)

Fig 9 OWASP SCAN

Alerts

Risk=Medium, Confidence=High (2)

http://localhost:3000 **(2)**

Content Security Policy (CSP) Header Not Set (1)

▶ GET http://localhost:3000/sitemap.xml

Session ID in URL Rewrite (1)

▶ POST http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PYZNVnp&sid=itvE-VwXdav6bu7AAAAQ

Risk=Medium, Confidence=Medium (3)

http://cdnjs.cloudflare.com **(1)**

Vulnerable JS Library (1)

▶ GET http://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js

http://localhost:3000 **(2)**

Cross-Domain Misconfiguration (1)

▶ GET http://localhost:3000/robots.txt

Missing Anti-clickjacking Header (1)

▶ POST http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PYZNVnp&sid=itvE-VwXdav6bu7AAAAQ

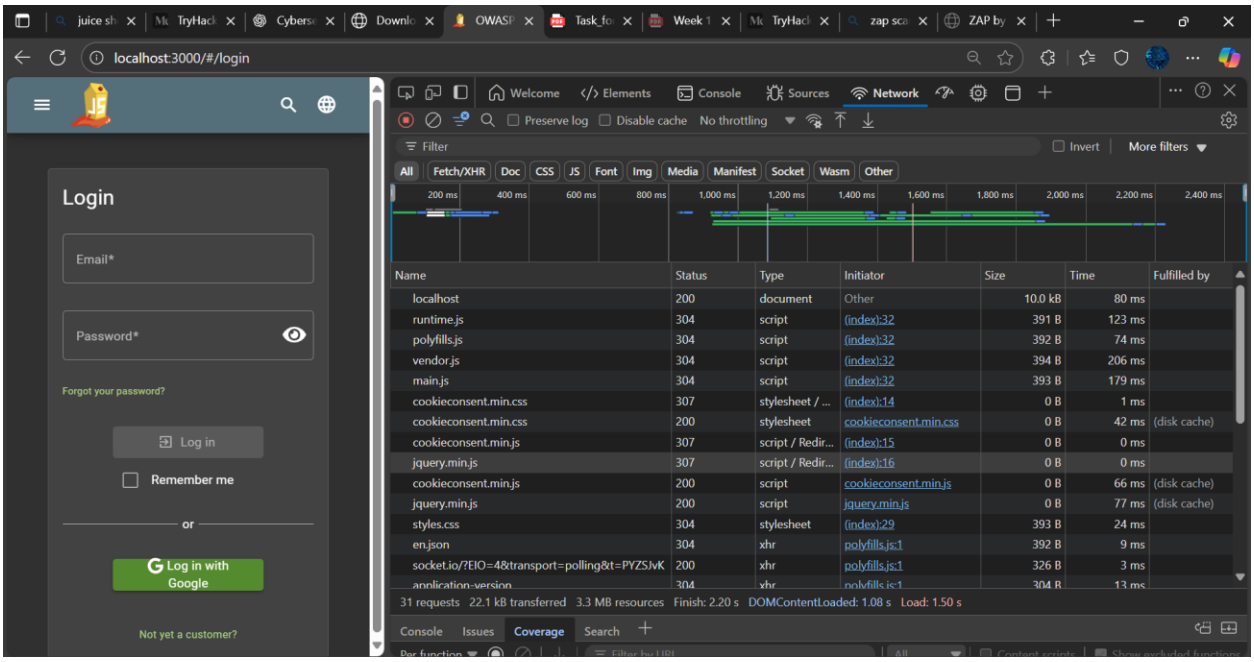
Fig 10 OWASP SCAN Alerts

Impact:

- Missing **CSP** and **X-Frame-Options** headers increase risks of **XSS** and **clickjacking**.
- **Session ID in URL Rewrite** could expose session identifiers, enabling session hijacking.
- **Vulnerable JS Library** (e.g., **jQuery 2.2.4**) may introduce known vulnerabilities (e.g., **CVE-2019-11358**).
- **Cross-Domain Misconfiguration** and **Hidden File Found** could allow unauthorized access to sensitive resources.

Summary Table

ID	**Vulnerability**	**Description**	**Observation**	**Risk**
1	**SQL Injection**	Login form allows authentication bypass via crafted input	Successful admin login with **admin' OR '1'='1--**	Critical
2	**XSS**	Tested fields not vulnerable to basic **XSS** payloads	Inputs escaped, no script execution for **<script>alert('XSS')</script>**	Low
3	Weak Password Storage	Passwords stored as unsalted **MD5** hashes	**MD5** hash: **fe01ce2a7fbac8fafaed7c982a04e229**	High
4	Security Misconfigurations	Missing headers (**CSP**, **X-Frame-Options**, **X-Content-Type-Options**), **Session ID in URL**, **Vulnerable JS Library**, etc.	**ZAP** scan flagged 13 alerts, e.g., missing **CSP** on **GET http://localhost:3000/sitemap.xml**	Medium



Browser Vulnerabilties

Conclusion

The Week 1 security assessment of OWASP Juice Shop, supported by the **ZAP by Checkmarx** scan, identified critical vulnerabilities, including **SQL injection** in the login form, weak password storage using **MD5** hashes, and multiple security misconfigurations (e.g., missing **CSP**, **X-Frame-Options**, and **Vulnerable JS Library**). While tested fields were not vulnerable to basic **XSS** attacks, Juice Shop's design suggests potential vulnerabilities in untested areas. The recommendations—adopting **parameterized queries**, upgrading to **bcrypt**, implementing **Helmet.js**, and updating **jQuery**