# WEB APPLICATION SECURITY ASSESSMENT REPORT

*OWASP Juice Shop*

**Prepared For: Future Interns**
**Prepared By: Devulapally Bhanu Prasad**
Email: bhanudevulapally342@gmail.com
Phone: +91-9133511394
Date: 19 October 2025
Version: 1.0

## 1. Executive Summary

This report presents the findings of a comprehensive Web Application Security Assessment conducted on the OWASP Juice Shop application. The primary objective was to identify vulnerabilities that could be exploited to compromise the confidentiality, integrity, or availability of the system. The engagement revealed several critical and high-severity findings, including a SQL Injection vulnerability allowing administrative access and reflected Cross-Site Scripting (XSS) issues. Immediate remediation is recommended to mitigate these risks and improve the application's security posture.

Overall Security Posture: CRITICAL

Summary of Findings:
- Critical: SQL Injection vulnerability enabling admin authentication bypass
- High: Reflected Cross-Site Scripting (XSS) in search functionality
- Medium: Broken Access Control leading to sensitive file exposure
- Medium: Missing Content Security Policy (CSP) header

## 2. Scope and Methodology

Scope: The assessment focused exclusively on the OWASP Juice Shop instance accessible at http://localhost:3000. All other systems, services, or infrastructure assets were explicitly out of scope.

Methodology: Testing was performed following the OWASP Top 10 framework using a hybrid approach combining manual penetration testing and automated scanning techniques.

Tools Used:

- OWASP ZAP (v2.x)
- Burp Suite Community Edition
- Kali Linux
- Docker
- Mozilla Firefox Developer Edition

## 3. Detailed Findings

### Finding 1: SQL Injection – Administrator Authentication Bypass

Risk Level: CRITICAL
OWASP Mapping: A03:2021 – Injection

Description: The login form fails to sanitize user input in the email field, allowing an attacker to inject SQL commands to manipulate backend queries.

Proof of Concept (PoC):

1. Navigate to http://localhost:3000/#/login
2. Enter `' OR 1=1 --` in the Email field.
3. Enter any password and click 'Log In'.
4. The attacker gains access as an administrator.

Impact: Successful exploitation grants full administrative privileges, allowing complete control over user accounts, products, and data.

Recommended Mitigation: Implement parameterized queries and use ORM frameworks to prevent direct SQL query manipulation. Validate and sanitize all user inputs.

### Finding 2: Reflected Cross-Site Scripting (XSS) in Search Bar

Risk Level: HIGH
OWASP Mapping: A03:2021 – Injection

Description: The search functionality reflects unsanitized user input back into the page, allowing execution of arbitrary JavaScript in the user's browser.

PoC: Enter `<script>alert('XSS')</script>` in the search field to trigger a JavaScript alert.

Impact: Exploitation may lead to session hijacking, credential theft, and defacement.

Recommended Mitigation: Apply context-aware output encoding and sanitize user input before rendering it in the response. Implement a strong Content Security Policy (CSP).

### Finding 3: Broken Access Control – Sensitive File Exposure

Risk Level: MEDIUM
OWASP Mapping: A01:2021 – Broken Access Control

Description: Directory browsing is enabled, exposing sensitive files under /ftp, which should not be publicly accessible.

PoC: Navigate to http://localhost:3000/ftp to view exposed directories.

Impact: Unauthorized users may download or modify internal files leading to data disclosure.

Recommended Mitigation: Disable directory listing and restrict access to sensitive directories using authentication and authorization controls.

### Finding 4: Missing Content Security Policy (CSP) Header

Risk Level: MEDIUM
OWASP Mapping: A05:2021 – Security Misconfiguration

Description: The application does not enforce a Content Security Policy (CSP), increasing the risk of XSS and data injection attacks.

PoC: Detected using OWASP ZAP scan results.

Impact: Allows untrusted scripts to execute in the user's browser environment.

Recommended Mitigation: Implement a strict CSP header defining trusted content sources (scripts, styles, frames, etc.).

## 4. OWASP Top 10 Mapping

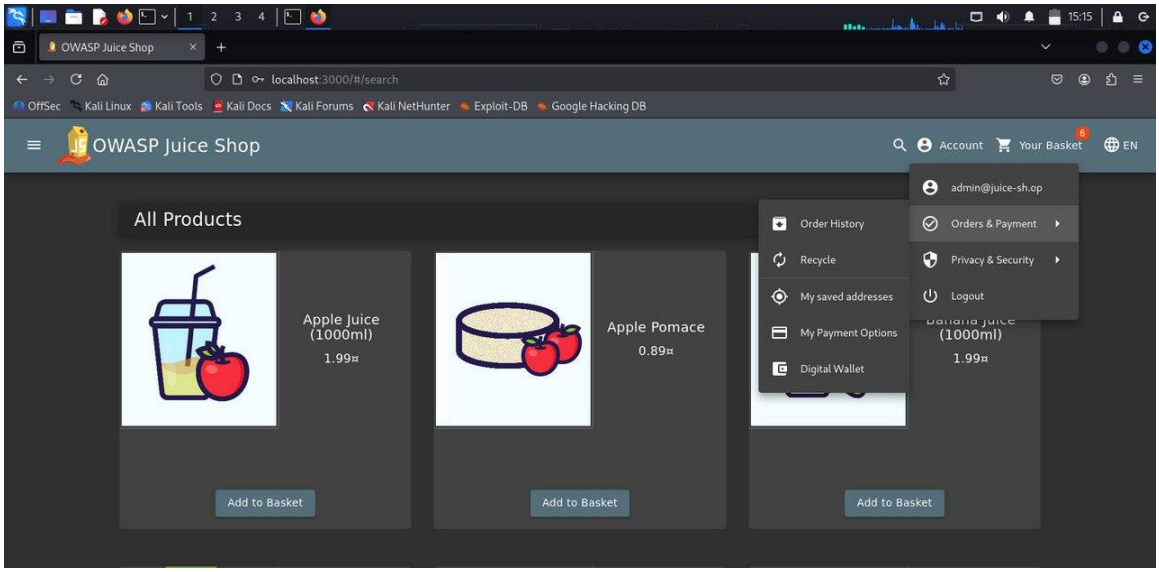
| OWASP Top 10 Category | Status | Corresponding Findings |
|---|---|---|
| A01: Broken Access Control | Vulnerable | Sensitive File Exposure (/ftp) |
| A02: Cryptographic Failures | Not Tested | - |
| A03: Injection | Vulnerable | SQL Injection, Reflected XSS |
| A04: Insecure Design | Not Tested | - |
| A05: Security Misconfiguration | Vulnerable | CSP Header Not Set |

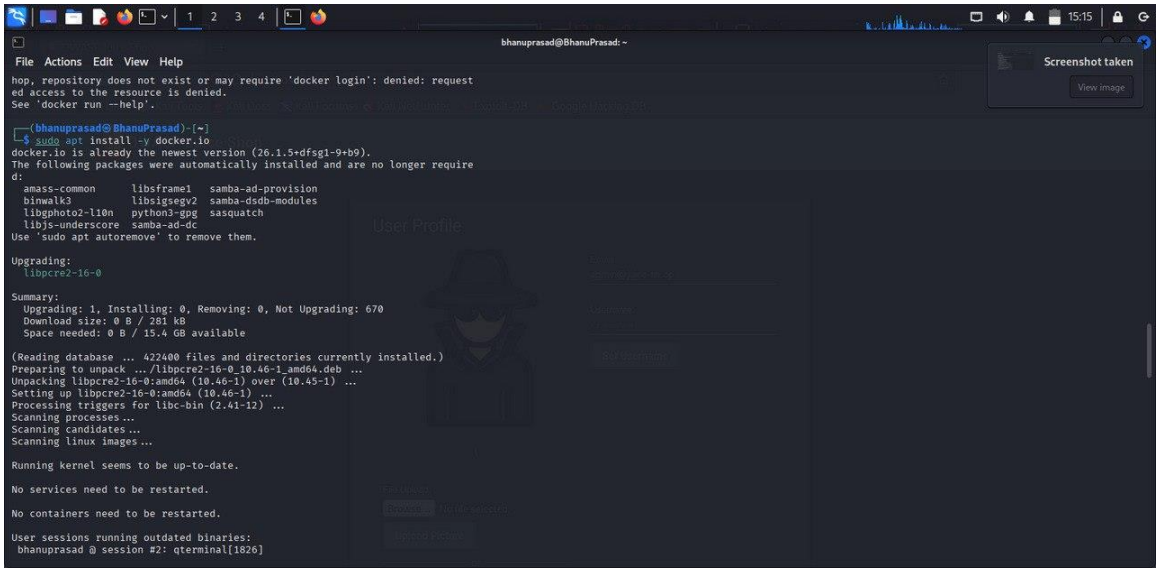
## 5. Conclusion and Recommendations

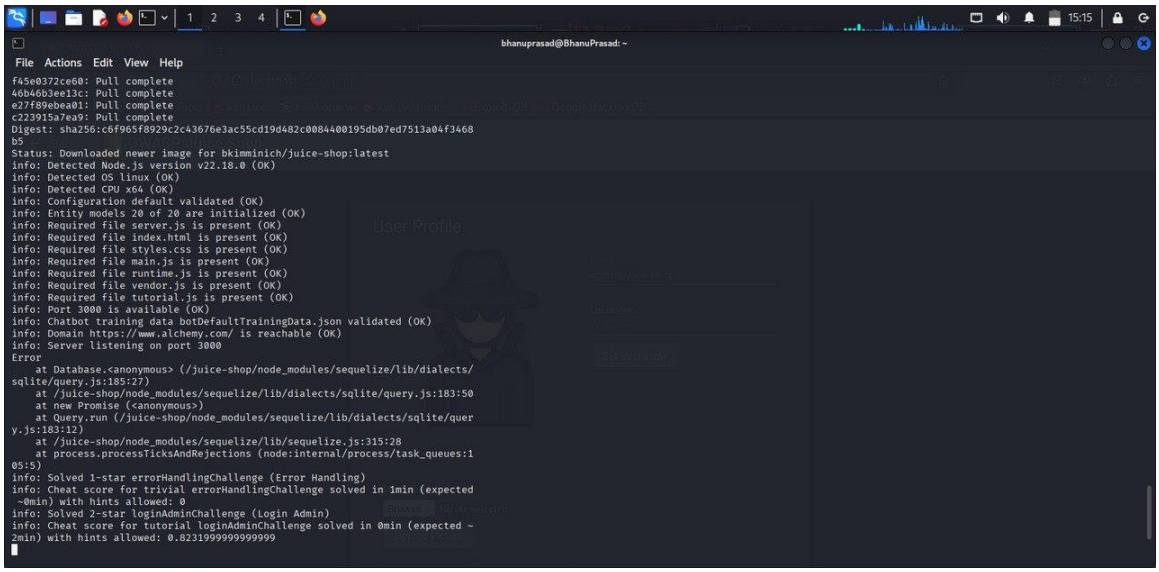
The OWASP Juice Shop application was found to contain multiple security weaknesses, primarily in input validation and access control. The SQL Injection and XSS vulnerabilities pose the highest risks, potentially allowing attackers to gain full control over the application. It is imperative that the development team implements the mitigations outlined for each issue and adopts secure coding practices.

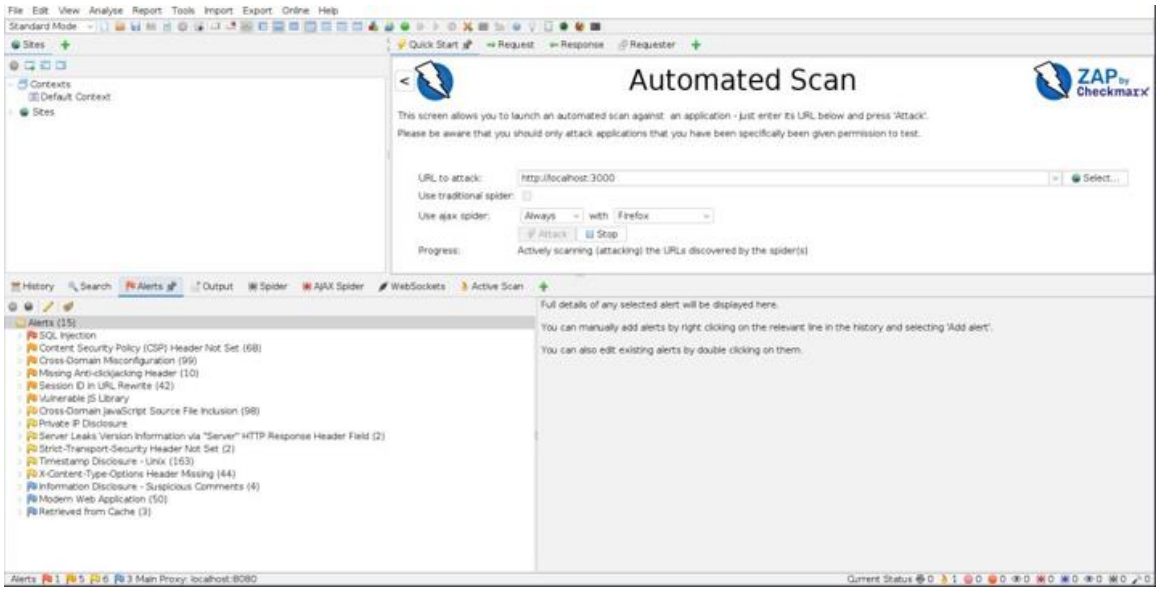
Recommended Actions:

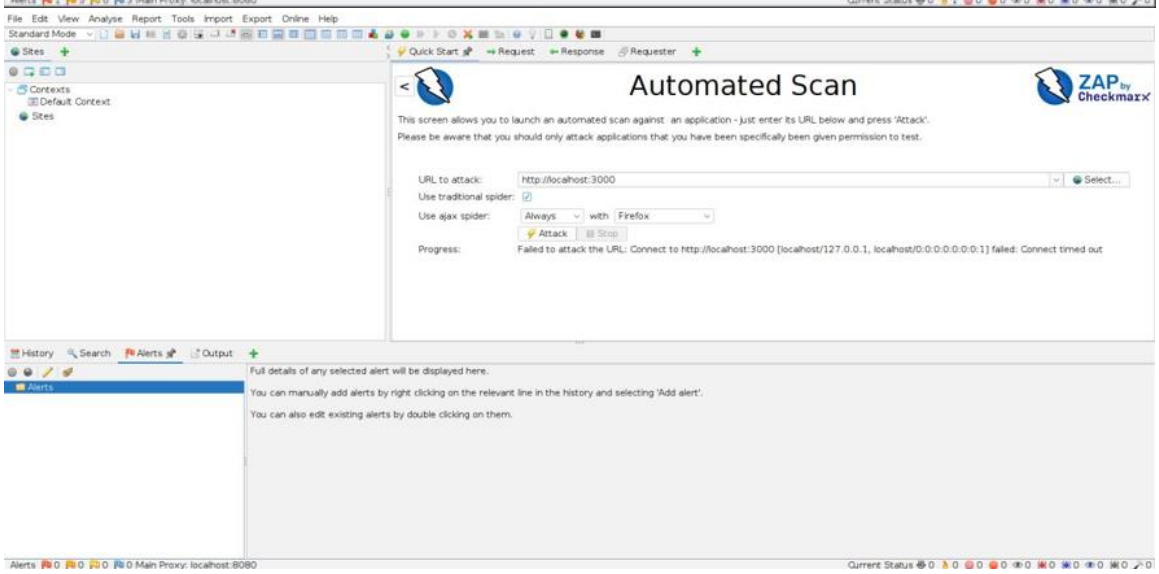
1. Patch all critical and high-severity vulnerabilities immediately.
2. Conduct a follow-up assessment after remediation.
3. Integrate automated security testing in the CI/CD pipeline.
4. Enforce security headers and strict validation mechanisms.
5. Provide regular security awareness training to the development team.




bhanuprasad@BhanuPrasad: ~
File Actions Edit View Help
f45e0372ce60: Pull complete
46b46b3ee13c: Pull complete
e27f89ebea01: Pull complete
c223915a7ea9: Pull complete
Digest: sha256:c6f965f8929c2c43676e3ac55cd19d482c0084400195db07ed7513a04f3468
b5
Status: Downloaded newer image for bkimminich/juice-shop:latest
info: Detected Node.js version v22.18.0 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (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 main.js is present (OK)
info: Required file runtime.js is present (OK)
info: Required file vendor.js is present (OK)
info: Required file tutorial.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
Error
at Database.<anonymous> (/juice-shop/node_modules/sequelize/lib/dialects/
sqlite/query.js:185:27)
at /juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:50
at new Promise (<anonymous>)
at Query.run (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/quer
y.js:183:12)
at /juice-shop/node_modules/sequelize/lib/sequelize.js:315:28
at process.processTicksAndRejections (node:internal/process/task_queues:1
05:5)
info: Solved 1-star errorHandlingChallenge (Error Handling)
info: Cheat score for trivial errorHandlingChallenge solved in 1min (expected
~0min) with hints allowed: 0
info: Solved 2-star loginAdminChallenge (Login Admin)
info: Cheat score for tutorial loginAdminChallenge solved in 0min (expected ~
2min) with hints allowed: 0.8231999999999999


Automated Scan
ZAP by Checkmarx
This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.
URL to attack: http://localhost:3000
Use traditional spider:
Use ajax spider: Always with Firefox
Attack Stop
Progress: Actively scanning (attacking) the URLs discovered by the spider(s)
Alerts (15)
SQL Injection
Content Security Policy (CSP) Header Not Set (68)
Cross-Domain Misconfiguration (99)
Missing Anti-clickjacking Header (10)
Session ID in URL Rewrite (42)
Vulnerable JS Library
Cross-Domain JavaScript Source File Inclusion (98)
Private IP Disclosure
Server Leaks Version Information via "Server" HTTP Response Header Field (2)
Strict-Transport-Security Header Not Set (2)
Timestamp Disclosure - Unix (163)
X-Content-Type-Options Header Missing (44)
Information Disclosure - Suspicious Comments (4)
Modern Web Application (50)
Retrieved from Cache (3)
Full details of any selected alert will be displayed here.
You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.
You can also edit existing alerts by double clicking on them.


Automated Scan
ZAP by Checkmarx
This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.
URL to attack: http://localhost:3000
Use traditional spider:
Use ajax spider: Always with Firefox
Attack Stop
Progress: Failed to attack the URL: Connect to http://localhost:3000 [localhost/127.0.0.1, localhost/0:0:0:0:0:0:0:1] failed: Connect timed out
Full details of any selected alert will be displayed here.
You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.
You can also edit existing alerts by double clicking on them.

End of Report.