Name: Abdul Basit Rizwan
ID: DHC-251
Field: Cyber Security
Task: 1

Cybersecurity Testing Documentation – Juice Shop Application

1. Introduction

This documentation presents the security assessment and testing of the Juice Shop web application. The objective is to identify vulnerabilities using automated and manual methods, confirm them with tests, and provide solutions to strengthen the application's security.

2. OWASP ZAP Scan

2.1 Overview

To assess the security of the Juice Shop web application, a comprehensive scan was performed using OWASP ZAP (Version 2.16.1). The workflow included:
1. Manual Exploration: Accessed the application at http://localhost:3000 using the ZAP-provided browser and interacted with pages such as signup, login, search, feedback, and profile to capture requests.
2. Spidering: Used ZAP Spider to automatically crawl all reachable pages and input fields.
3. Active Scanning: Performed Active Scan to identify vulnerabilities including SQL Injection, XSS, security misconfigurations, and cookie issues.
4. Alerts Inspection: Reviewed the Alerts tab to analyze all discovered security issues.

2.2 Summary of Alerts

Risk Level	Number of Alerts
High	6
Medium	12
Low	14
Informational	11

2.3 High-Risk Alerts

External Redirect – 1
Off-site Redirect – 1
SQL Injection – 3
SQL Injection – Oracle – Time Based – 1
SQL Injection – SQLite – 1
Vulnerable JS Library – 1

Impact: These issues can allow attackers to bypass authentication, steal sensitive information, or manipulate user sessions.

2.4 Medium-Risk Alerts

Absence of Anti-CSRF Tokens – 3
Content Security Policy (CSP) misconfigurations – multiple instances
Cross-Domain Misconfiguration – 453
Format String Error – 1
Missing Anti-clickjacking Header – 14
Session ID in URL Rewrite – 20
Vulnerable JS Library – 2

Impact: Could allow session hijacking, content injection, or client-side attacks.

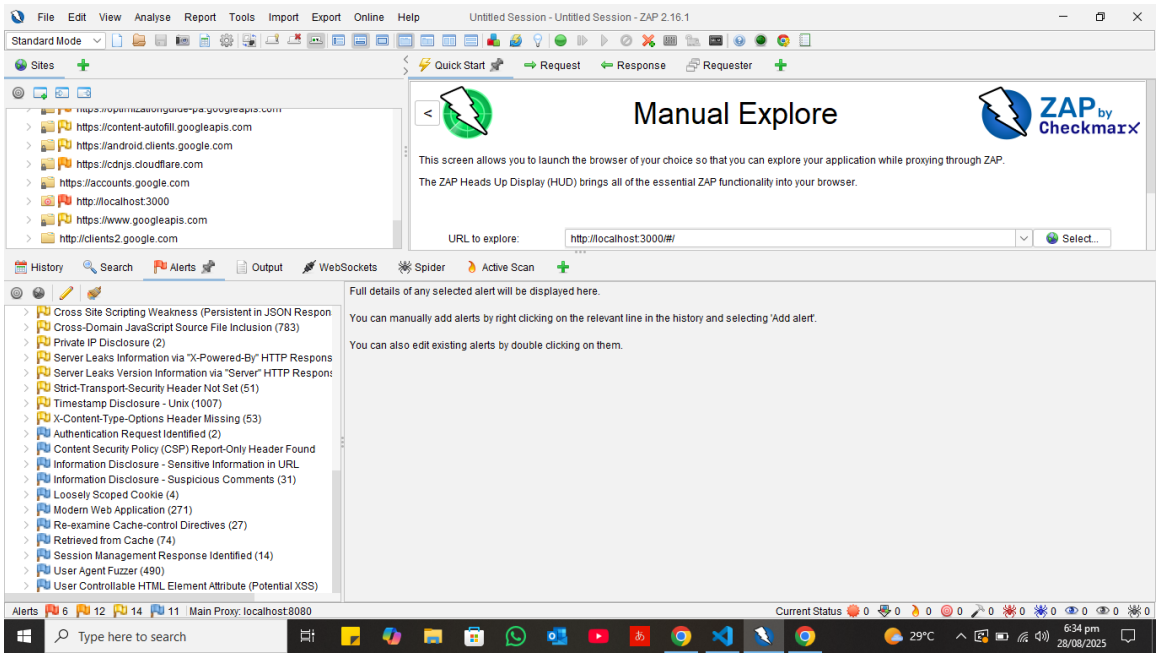
2.5 Low & Informational Alerts

Application Error Disclosure – 13
Cookies missing HttpOnly or Secure flags
Server information leaks via headers
Timestamp disclosures

Impact: Reveals sensitive server/application information that could be leveraged in targeted attacks.

2.6 Passing Rules

ZAP verified active scan rules were configured correctly, including:
- Directory Browsing, CRLF Injection, Path Traversal
- SQL Injection variants (MySQL, Oracle, PostgreSQL, SQLite, MsSQL)
- Cross-Site Scripting (Reflected, Persistent, DOM-Based)
- Remote Code Execution indicators



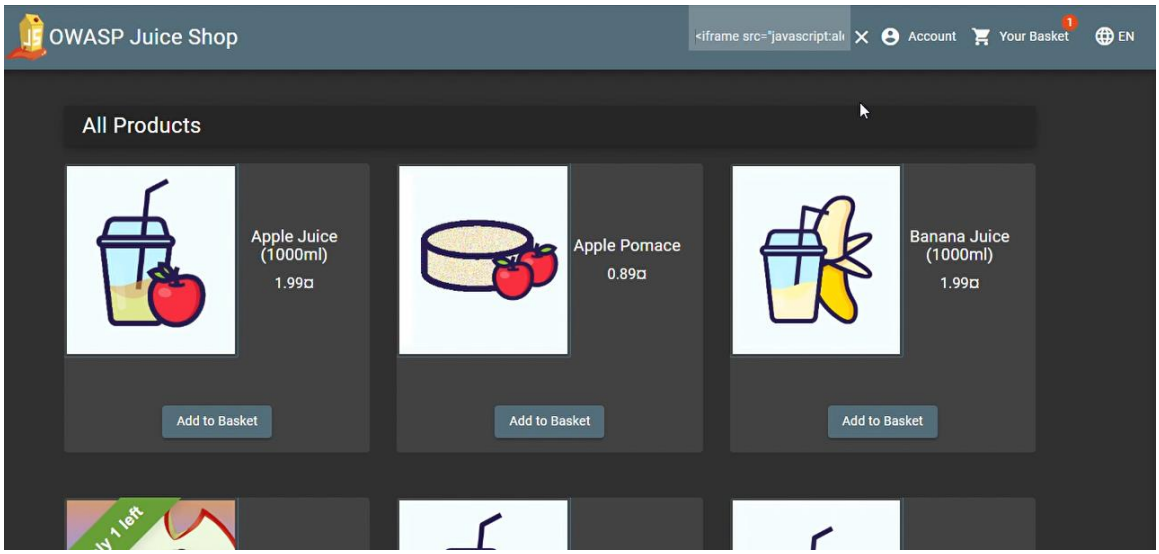
3. XSS Testing

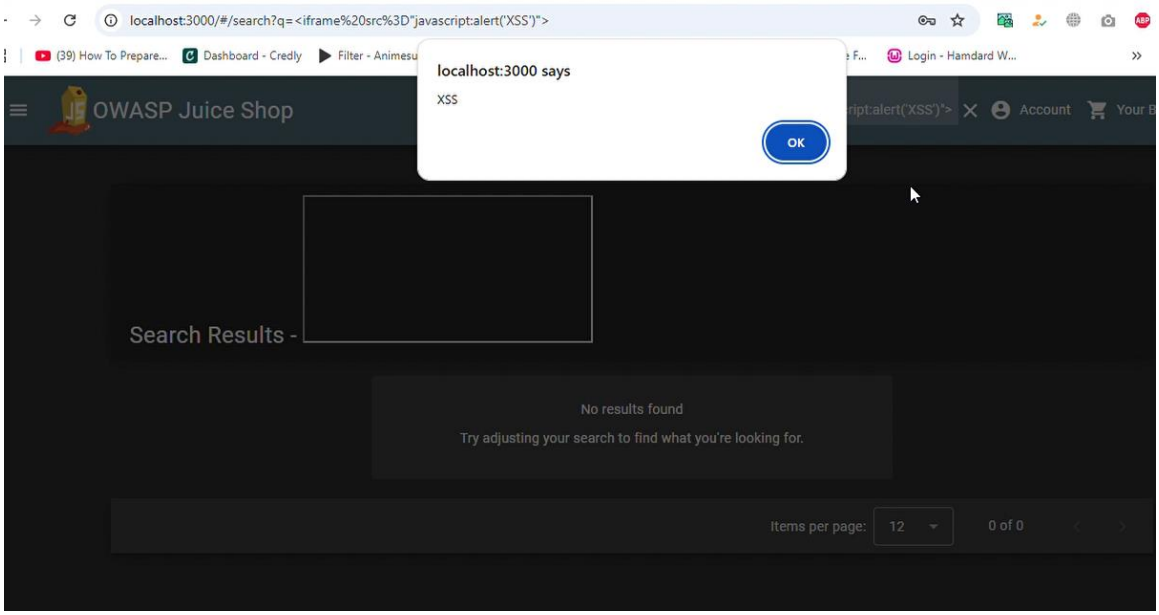
3.1 Test Conducted

Tested Reflected XSS on the search box.
Input used:
<iframe src="javascript:alert('XSS')">
Result: Triggered an alert pop-up, confirming an XSS vulnerability.

3.2 Impact

Attackers can execute arbitrary JavaScript in user browsers, which can lead to cookie theft, session hijacking, and phishing attacks.





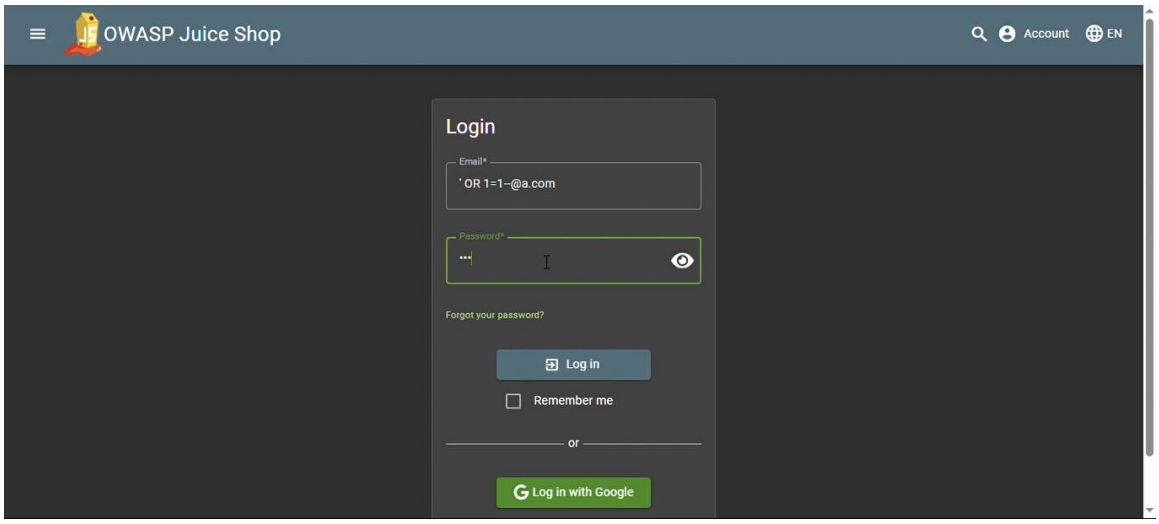
4. SQL Injection Testing

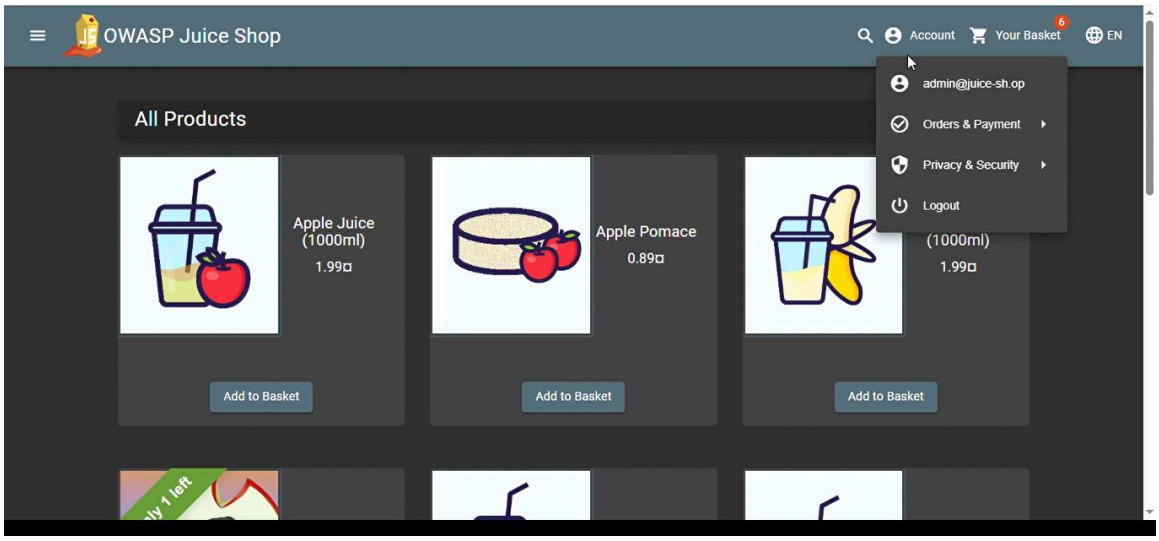
4.1 Test Conducted

Tested login form for SQL Injection.
Email input: ' OR 1=1--@a.com
Dummy password entered.
Result: Successfully logged in without valid credentials, confirming vulnerability.

4.2 Impact

Attackers can bypass authentication and gain unauthorized access, potentially leading to data theft or account manipulation.





5. Solutions / Remediation

5.1 OWASP ZAP Vulnerabilities

- SQL Injection: Use prepared statements / parameterized queries.
- XSS: Sanitize and escape all user input before rendering. Implement Content Security Policy (CSP) headers.
- CSRF: Add Anti-CSRF tokens in all forms.
- Cookie Security: Set HttpOnly, Secure, and proper SameSite attributes.
- Header & Server Information Leaks: Hide server versions and sensitive headers; configure proper security headers.
- Other Security Misconfigurations: Regularly update libraries, validate input, and follow secure coding practices.

5.2 XSS

- Escape special characters (<, >, ", ') in output.
- Use frameworks or libraries that auto-encode user input.
- Implement CSP to restrict script execution.

5.3 SQL Injection

- Always use parameterized queries or ORMs.
- Validate and sanitize user input.
- Limit database permissions; avoid using high-privilege accounts for application access.

6. Conclusion

The Juice Shop application demonstrated multiple vulnerabilities via OWASP ZAP scan, XSS, and SQL Injection tests. Implementing the recommended remediation steps will significantly strengthen the application’s security and protect it from common web attacks.