
WEB APPLICATION SECURITY TESTING REPORT

Application Tested: OWASP Juice Shop

Document ID: FUTURE_CS_01

CIN: FIT/AUG25/CS3395

Tools Used: OWASP ZAP, Burp suite

Date of Testing: 10 September 2025

Test Conducted By: HARI PAVAN

Executive Summary:

Web Application Security Testing. The purpose of this assessment is to evaluate the security posture of the OWASP Juice Shop, an intentionally vulnerable application widely used for training and practice in penetration testing.

The testing methodology followed the OWASP Top 10 (2021) security risks and included the use of industry-standard tools such as OWASP ZAP for automated scanning, Burp Suite Community Edition for manual testing. During the course of this project, multiple vulnerabilities were identified, including SQL Injection, Broken Access Control, Reflected XSS.

Introduction

In today's digital landscape, securing web applications is crucial to prevent malicious attacks, data breaches, and service disruptions. As part of this cybersecurity project, we conducted a security assessment of the OWASP Juice Shop, a deliberately insecure web application used for testing and training. The aim of this assessment is to identify security flaws, analyze their potential risks, and recommend effective mitigation strategies using industry-standard tools and techniques. The findings presented in this report include detailed descriptions of each vulnerability, reproduction steps, screenshots, associated OWASP Top 10 mappings, impact analysis, and recommended mitigations. This deliverable simulates a real-world client security assessment and is intended to demonstrate practical skills in ethical hacking, vulnerability assessment, and penetration testing.

Scope and Methodology

The scope of this assessment was limited to the OWASP Juice Shop application, a deliberately insecure web application designed for security training. The objective was to perform a vulnerability assessment and penetration test based on OWASP Top 10 security risks. Tools Used • OWASP ZAP – Automated vulnerability scanning

- Burp Suite Community Edition – Manual exploitation of vulnerabilities
- Browser Developer Tools / Manual Payloads – For custom testing (SQLi, XSS)

Testing Methodology:

The methodology adopted for this assessment was based on industry standards and the OWASP Testing Guide:

1. Reconnaissance & Information Gathering – Identified technologies and potential attack surfaces.
2. Automated Scanning – Used OWASP ZAP to detect possible vulnerabilities.
3. Manual Testing – Performed SQL injection, XSS, and access control testing using Burp Suite.
4. Vulnerability Analysis – Validated findings and mapped them to OWASP Top 10.
5. Reporting – Documented vulnerabilities with screenshots, impacts, and mitigations

Severity:

Each finding is categorized as:

High – Immediate security threat. Likely to be exploited.

Medium – Could be exploited, requires moderate effort

Low – Information leakage or minor misconfiguration.

Informational – Does not pose direct risk but may assist an attacker.

Conclusion:

This assessment revealed several medium and low-risk vulnerabilities within the OWASP Juice Shop. While the application is intentionally vulnerable, the findings reflect real-world risks present in poorly secured web applications. Implementing proper security headers, access controls, and content restrictions is essential to protecting user data and maintaining trust. The remediation steps outlined for each issue serve as best practices that can be applied in production-grade applications.

Details findings:

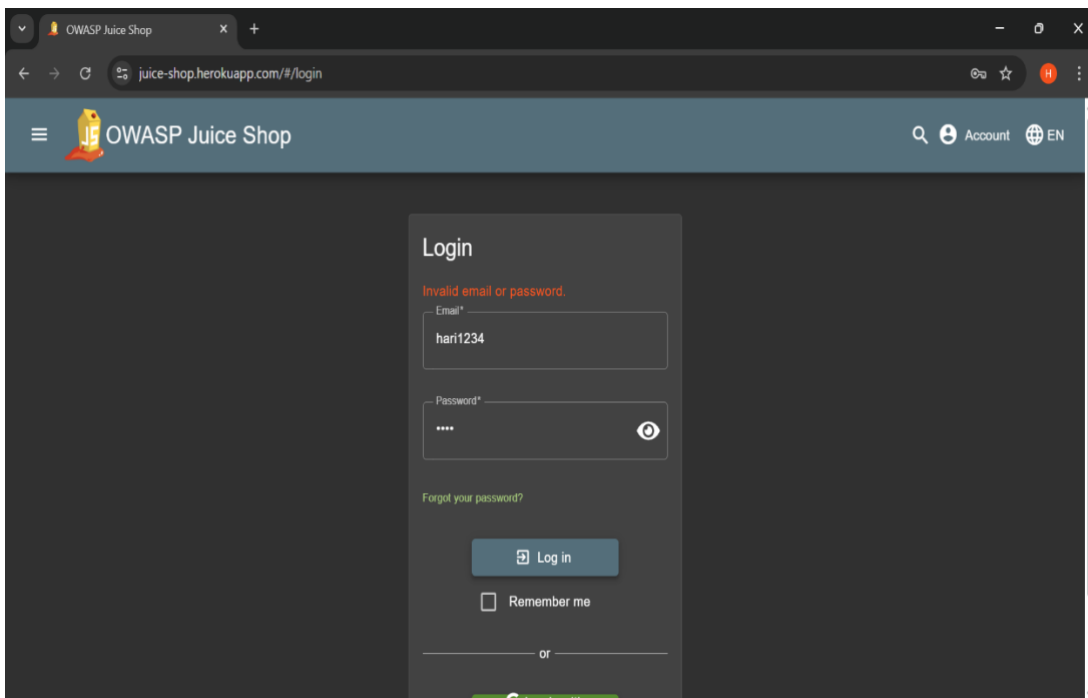
1. SQL Injection -Authentication by pass

Severity: High **Description:**

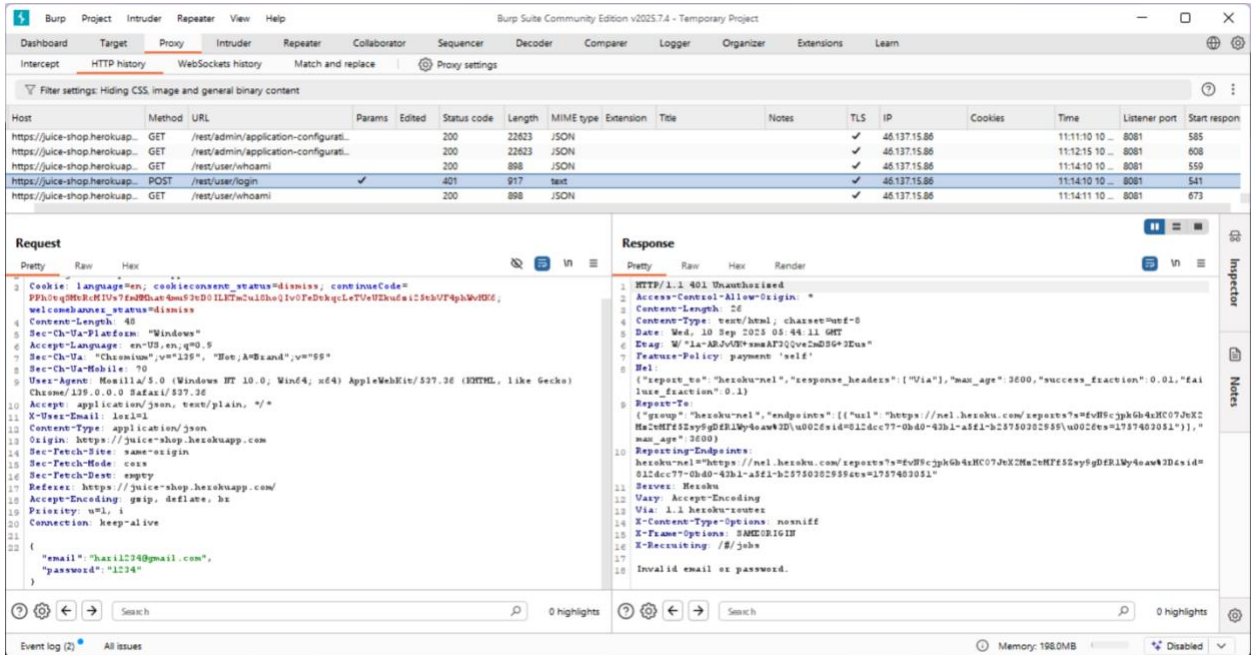
During testing of the login functionality, it was identified that the application is vulnerable to SQL Injection. By inserting a specially crafted payload in the username field, the authentication query was manipulated, allowing login without valid credentials.

Steps with screenshots:

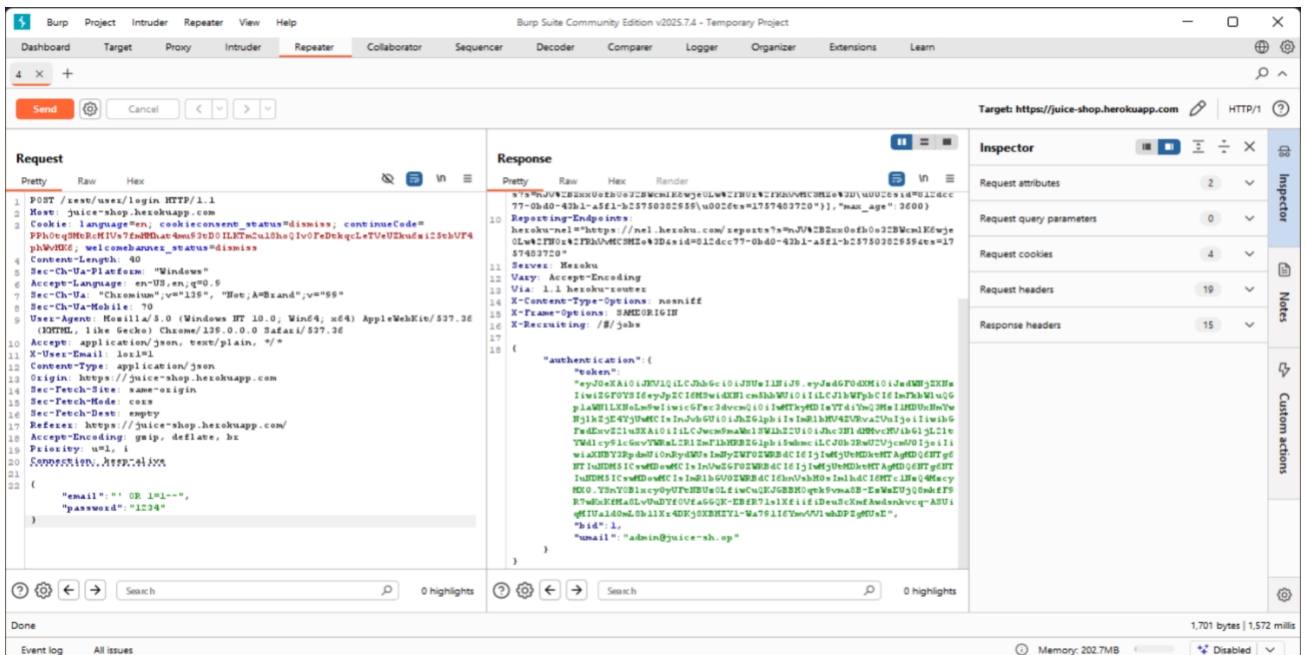
Step1: open the OWASP juice shop login page and entered to random credentials in the mail and passwords. the login page failed as expected.



Step 2: Intercepted the login request using **Burp Suite**. The request contained the email and password parameters.

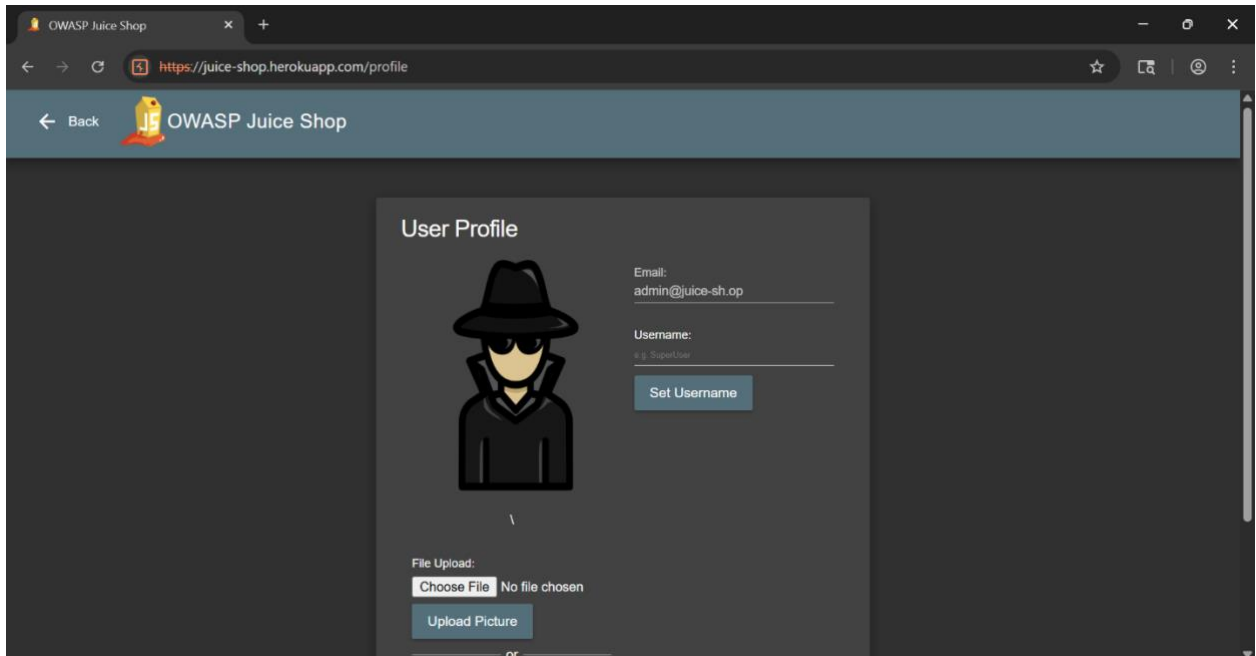


Step 3: Modified the email parameter to include a SQL Injection payload: **' OR 1=1--** This payload forces the query condition to always return true.



Step 4: Forwarded the modified request to the server. Instead of rejecting, the application processed the login and redirected to the authenticated area.

Step 5: Verified that the login was successful and administrative access was granted without valid credentials.



Impact:

- Bypass of login authentication
- Unauthorized access to user or administrator accounts
- Potential access to sensitive data
- Increased risk of privilege escalation and data breaches.

Mitigation:

- Use parameterized queries (Prepared Statements).
- Implement strong server-side input validation and sanitization.
- Apply the principle of least privilege on database accounts.

OWASP Top 10 Mapping:

*A03: Injection.

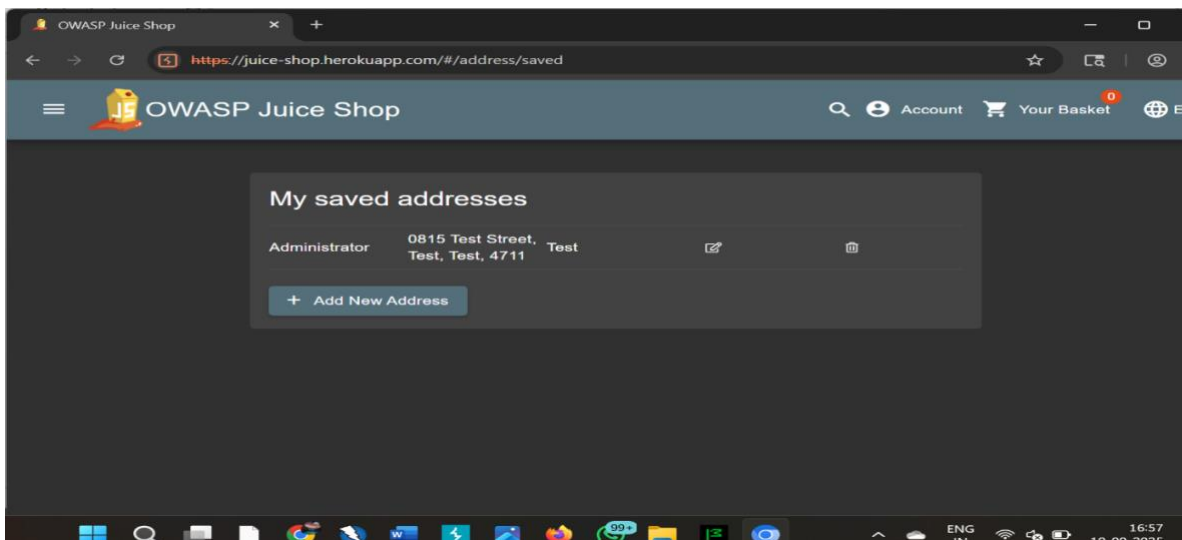
2: Broken Access Control – Saved Cards Misuse

Severity: High Description:

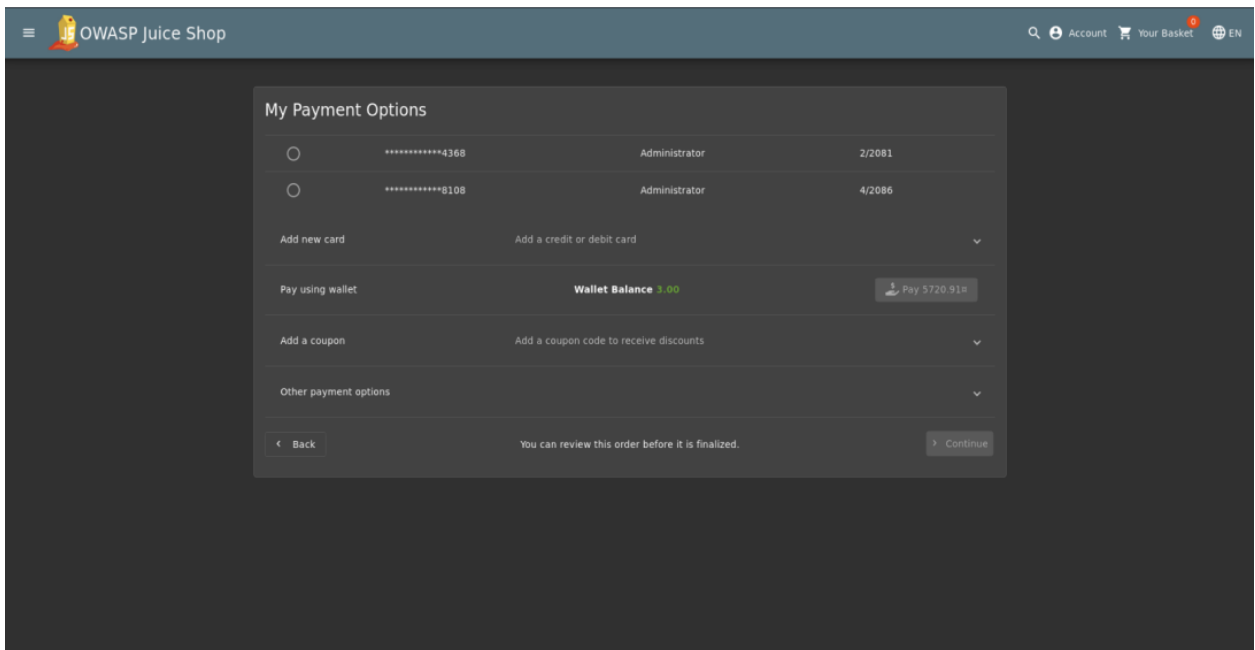
During testing of the checkout process, it was observed that previously saved payment methods (credit cards and digital wallet) could be used to make purchases without requiring OTP or CVV re-verification. This indicates a broken access control and weak payment authorization mechanism.

Steps with screenshots:

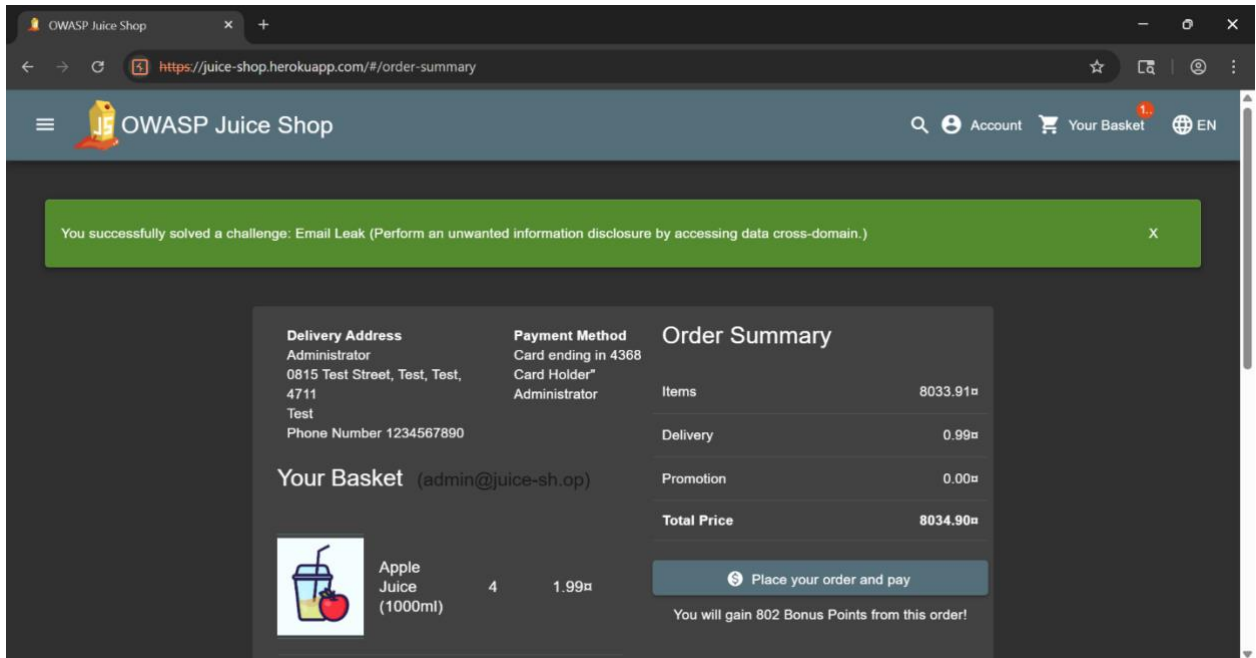
Step 1: Added an item to the shopping cart and proceeded to checkout.



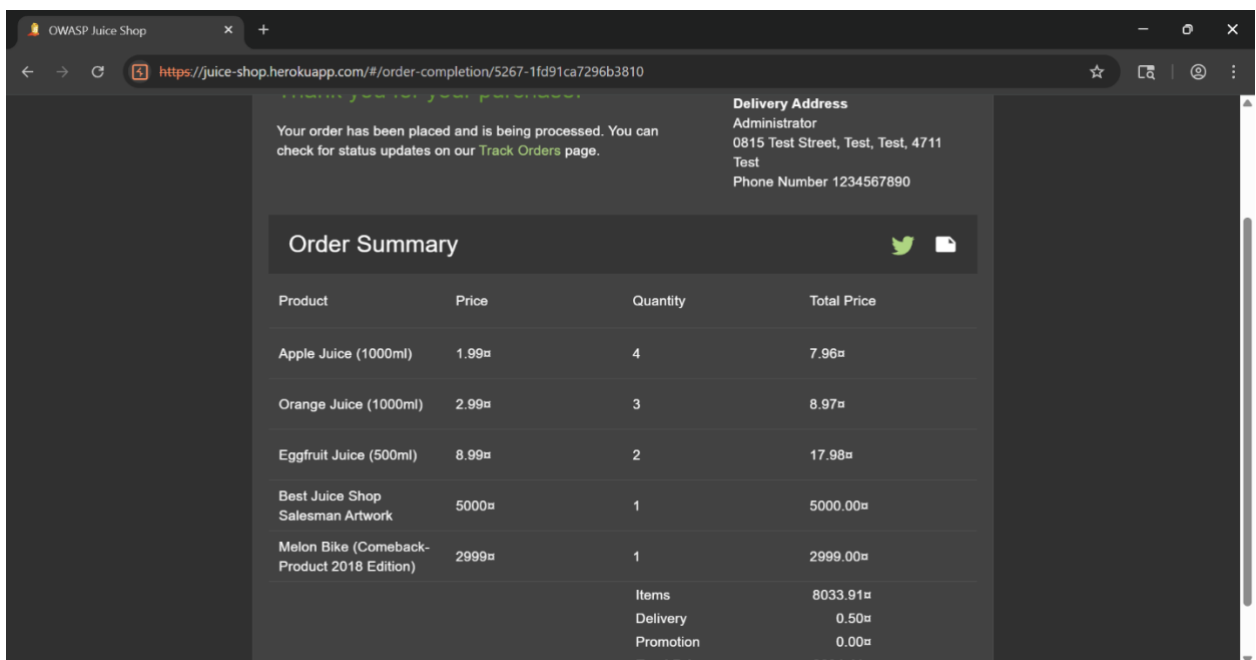
Step 2: Reached the payment page. Two saved credit cards and a digital wallet option were available for selection.



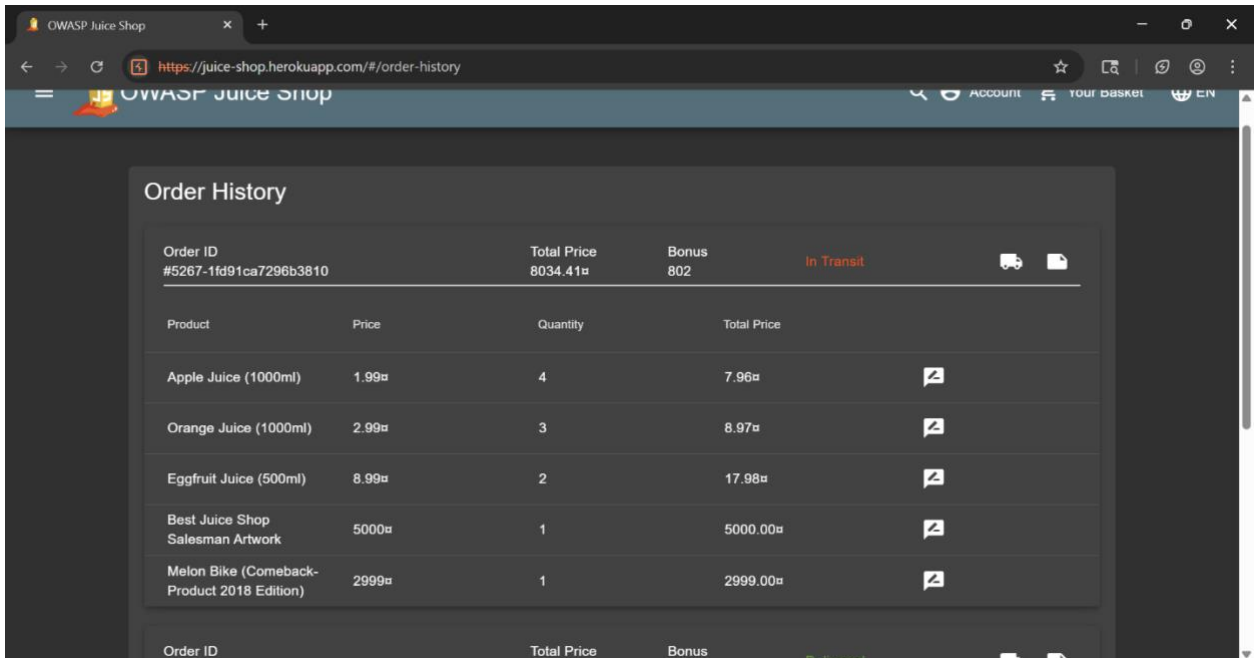
Step 3: Selected one of the saved cards and continued. No CVV or OTP verification was required to complete the purchase.



Step 4: Completed the transaction successfully. The application displayed a confirmation message along with the order summary.



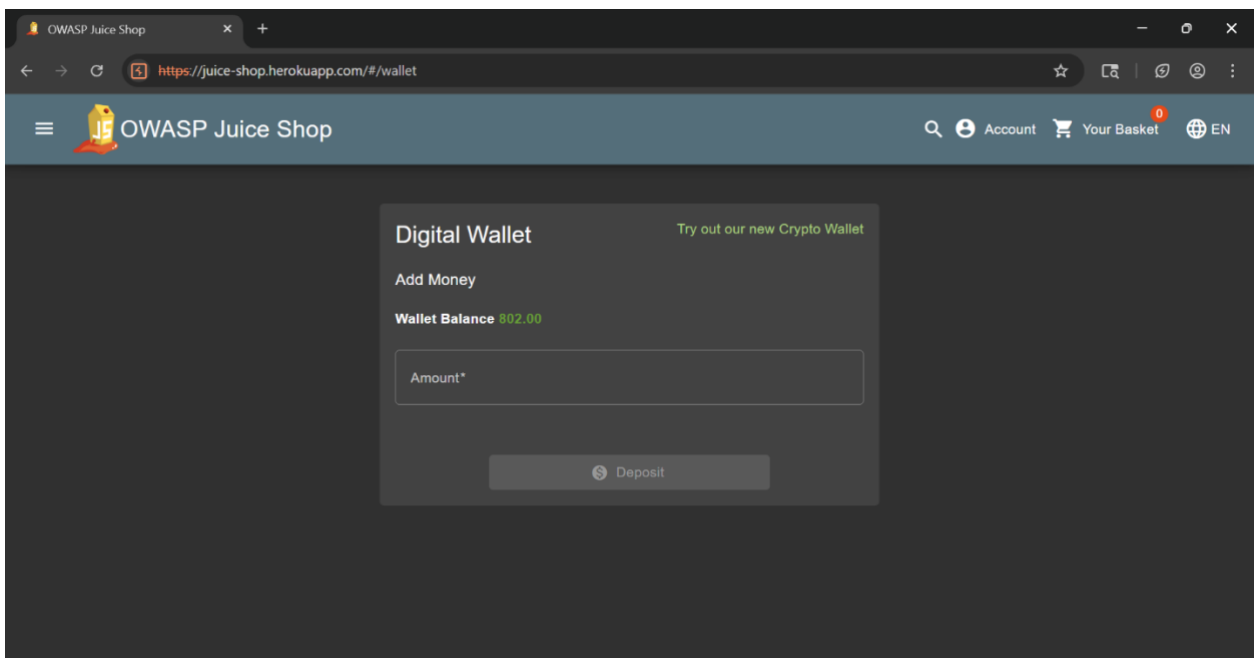
Step 5: Navigated to the order history page. All past purchases, including the unauthorized transaction, were listed.



The screenshot shows the 'Order History' page of the OWASP Juice Shop. The order summary at the top indicates an order ID of #5267-1fd91ca7296b3810, a total price of 8034.41€, a bonus of 802, and a status of 'In Transit'. Below this, a table lists the items purchased:

Product	Price	Quantity	Total Price
Apple Juice (1000ml)	1.99€	4	7.96€
Orange Juice (1000ml)	2.99€	3	8.97€
Eggfruit Juice (500ml)	8.99€	2	17.98€
Best Juice Shop Salesman Artwork	5000€	1	5000.00€
Melon Bike (Comeback-Product 2018 Edition)	2999€	1	2999.00€

Step 6: Accessed the digital wallet feature and confirmed that funds could be viewed and added without additional authentication.



The screenshot shows the 'Digital Wallet' page of the OWASP Juice Shop. The wallet balance is displayed as 802.00€. There is an 'Add Money' section with a text input field labeled 'Amount*' and a 'Deposit' button.

Amount*

Deposit

Impact:

- Attackers can misuse stored cards for unauthorized purchases.
- Risk of financial fraud and loss of customer trust.
- May lead to PCI-DSS non-compliance. Mitigation:
 - Enforce re-authentication (OTP, CVV) for every transaction.
 - Mask and encrypt saved card details.
- Implement role-based access control (RBAC) to restrict admin misuse.
- Ensure compliance with PCI-DSS standards.

OWASP Top 10 Mapping:

- A01: Broken Access Control

3: Reflected XSS – Search Bar

Severity: Medium

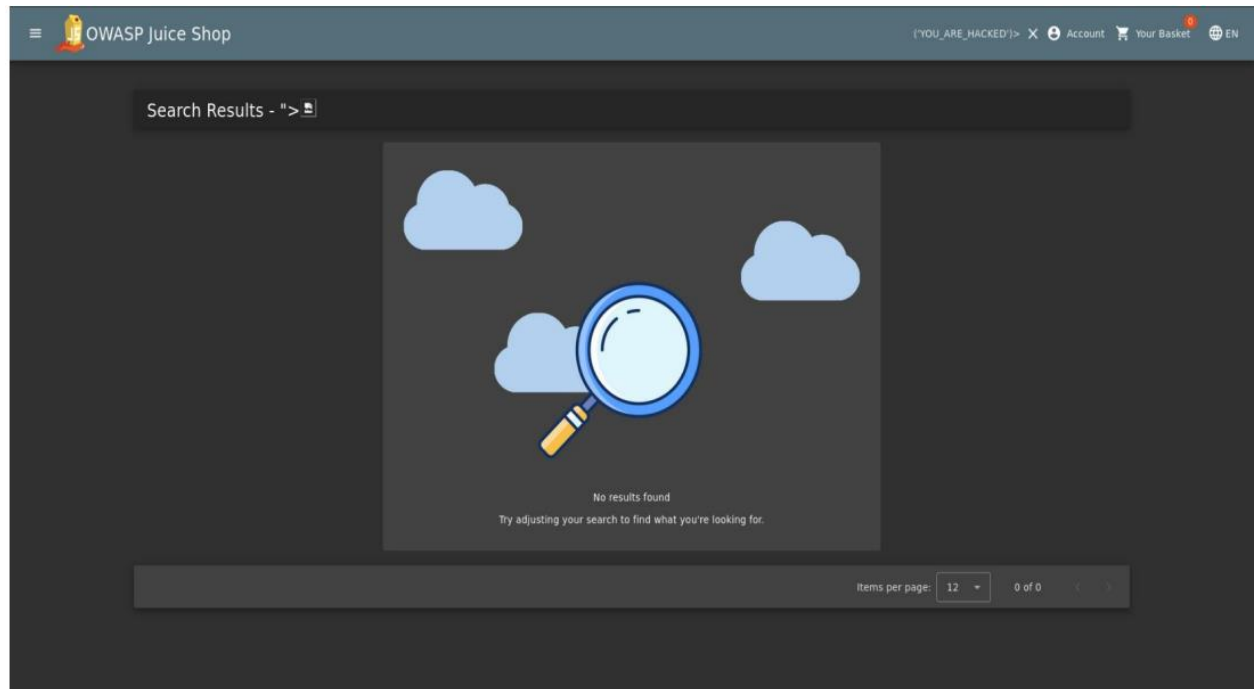
Description:

The search functionality reflects unsensitized user input in the response. By injecting a crafted payload, JavaScript code was executed in the browser, confirming a Reflected Cross-Site Scripting (XSS) vulnerability. This type of issue can be exploited by attackers through malicious links, making unsuspecting users run harmful scripts in their browsers.

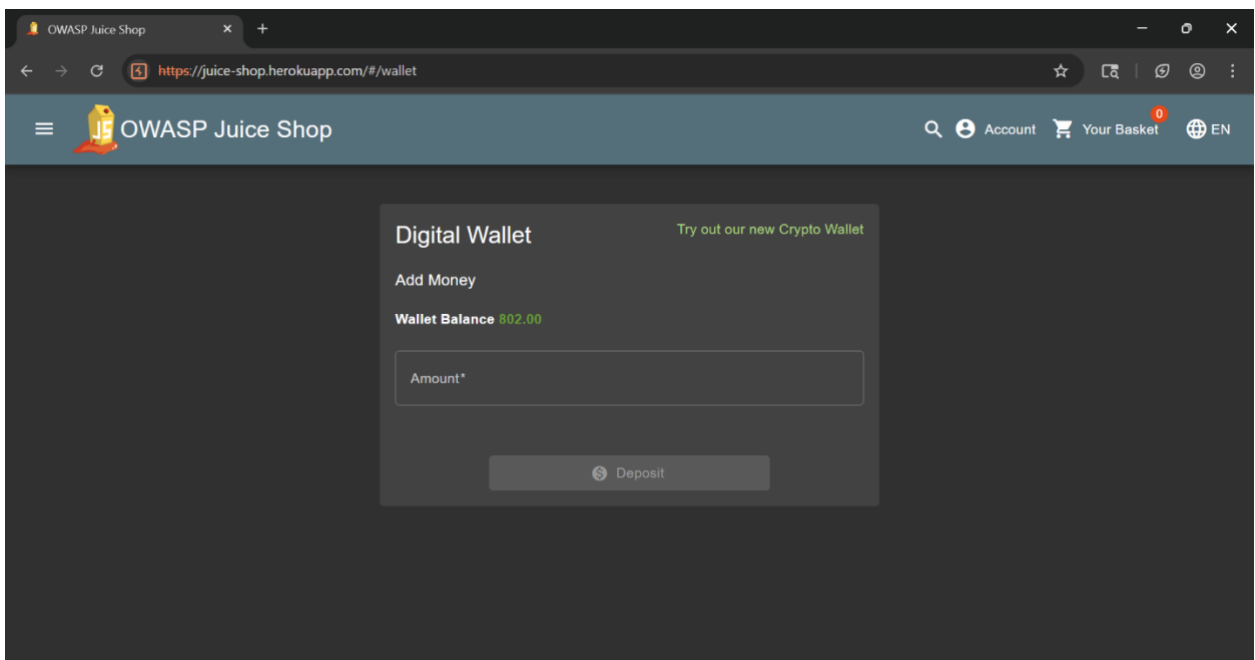
Steps with screenshots:

Step 1: Navigated to the application's search bar and entered a crafted XSS payload: ">".

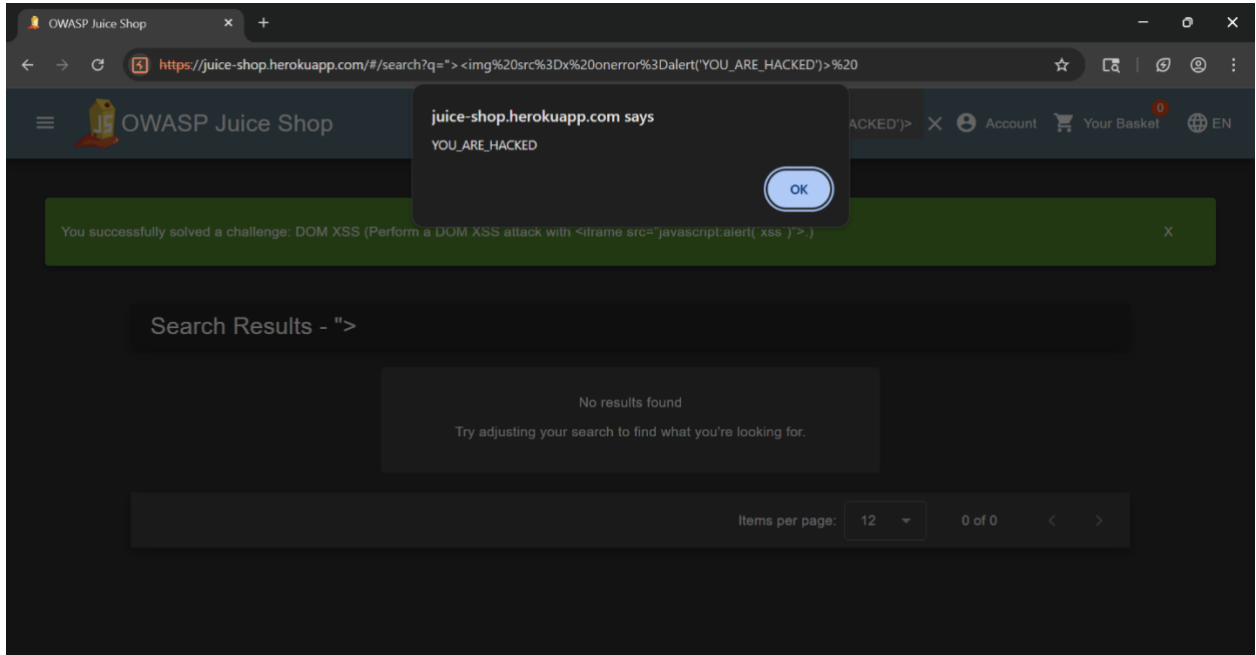
```
"><img src= xoneerror=alert('you_are_hacked')>.
```



Step 2: Submitted the search request. The payload was reflected back in the response without sanitization.



Step 3: The browser executed the injected JavaScript, displaying a popup alert box.



Impact:

- Attackers can steal session cookies.
- Users may be redirected to phishing sites.
- Enables account compromise if exploited with social engineering. Mitigation:
- Sanitize and encode user inputs before rendering.
- Implement a strong Content Security Policy (CSP).
- Use libraries like DOM Purify for input validation.

OWASP Mapping:

- A03: Injection.

4: Automated Scan – OWASP ZAP

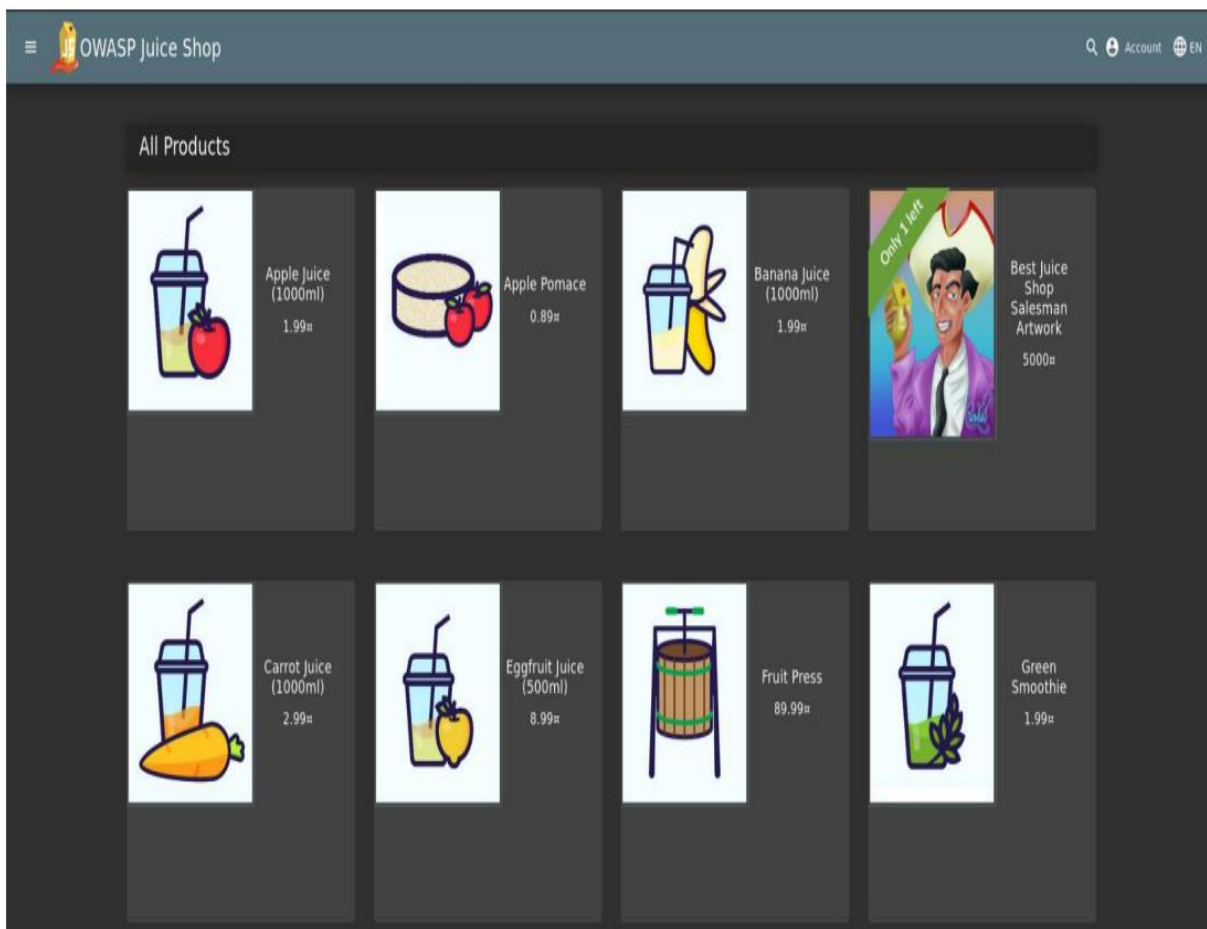
Severity: Informational

Description:

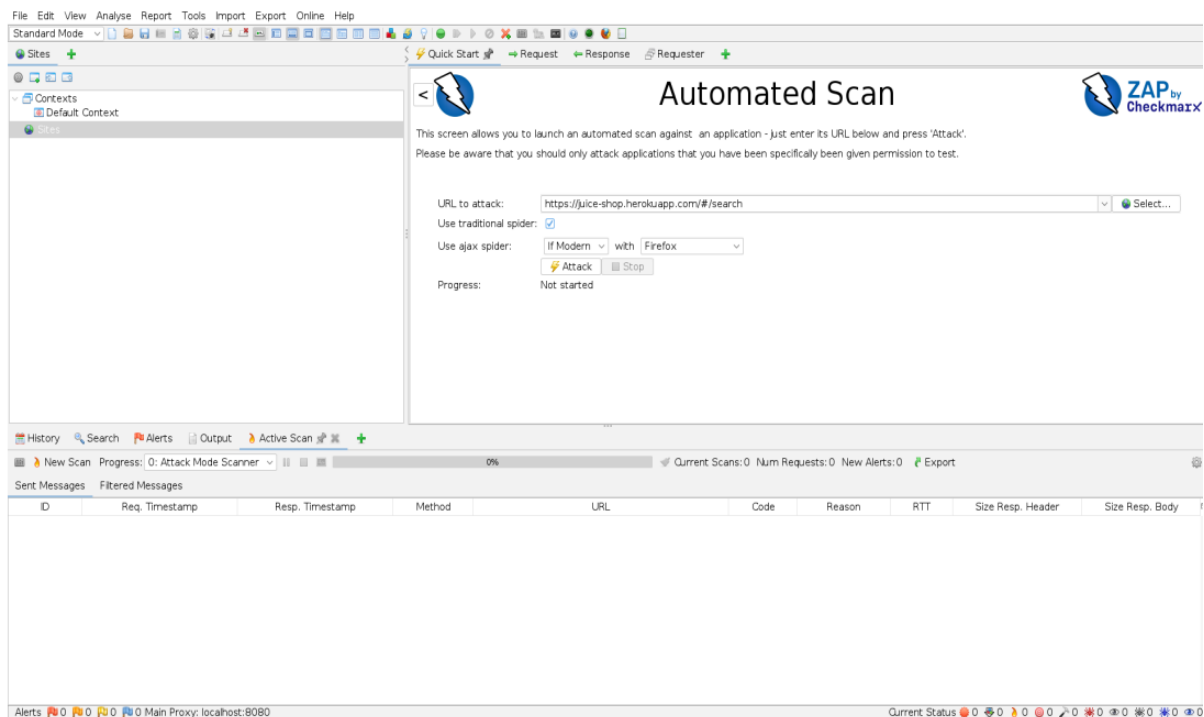
An automated vulnerability scan was performed using OWASP ZAP against the OWASP Juice Shop application. The scan identified several potential issues, including possible SQL Injection, XSS, and security misconfigurations. These results were further analyzed and manually validated with Burp Suite to confirm actual exploitable vulnerabilities.

Steps with screenshots:

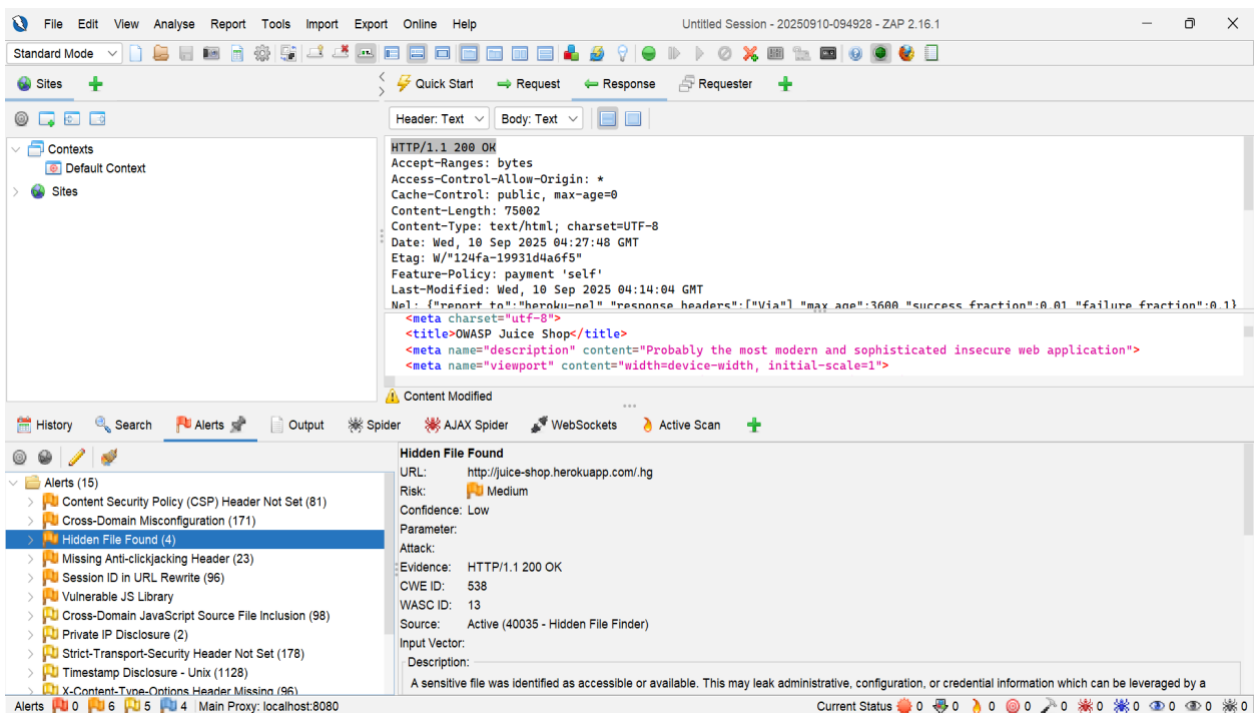
Step 1: Launched OWASP ZAP and provided the target Juice Shop application URL for scanning. The target application's initial view was verified.



Step 2: Entered the Juice Shop URL into the Automated Scan section of ZAP's Quick Start ta



Step 3: After the scan completed, ZAP displayed a list of potential vulnerabilities in the Alerts tab. These included possible injection points and security misconfigurations.



Impact:

- Provides an initial vulnerability overview.
- Helps identify potential weak points quickly
- Automated results should not be considered final without manual verification.

Mitigation:

- Regularly perform automated scans with OWASP ZAP.
- Always combine automated scanning with manual penetration testing for accuracy.

OWASP Mapping:

- Multiple categories (A01: Broken Access Control,
A03: Injection,
A05: Security Misconfiguration) depending on flagged issue.

5: Content Security Policy (CSP) Header Not Set

Severity: Medium

Description: The Content Security Policy header is missing. This makes the application more vulnerable to Cross-Site Scripting (XSS) and data injection attacks.

Explanation:

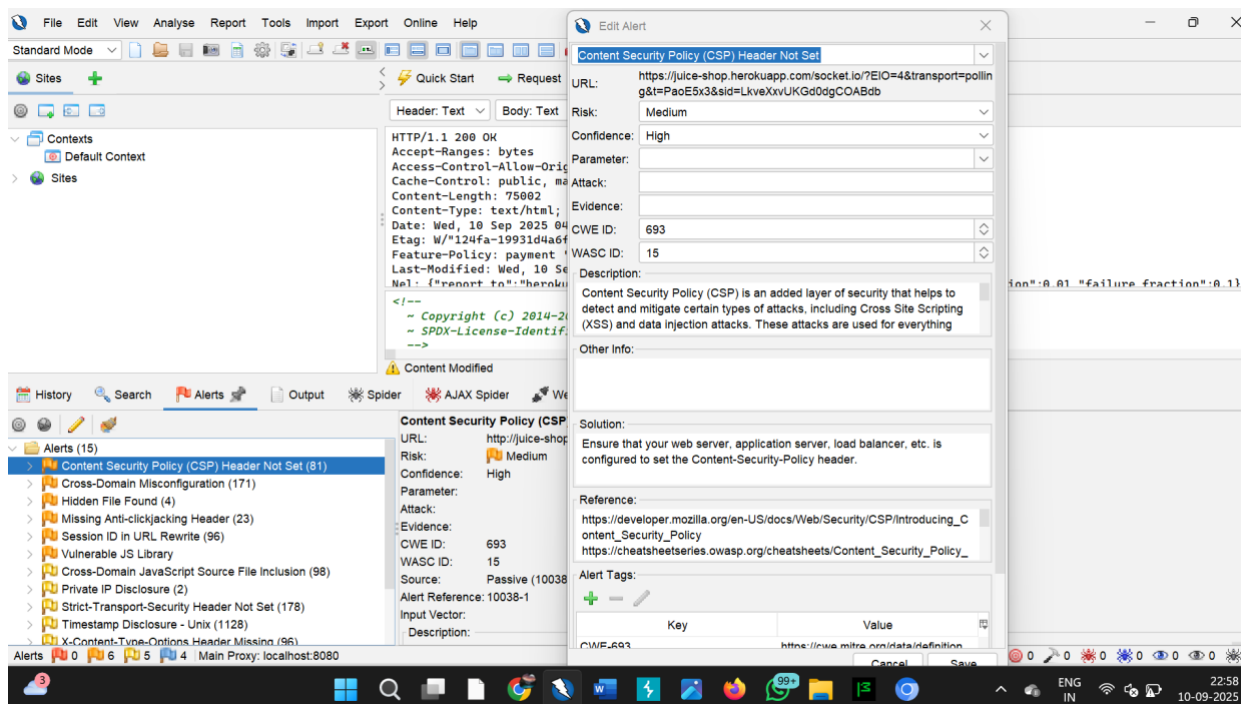
The server does not send a Content-Security-Policy HTTP header, which defines allowed content sources (e.g., scripts, styles). Without it, the app is more vulnerable to Cross-Site Scripting (XSS) attacks.

Impact:

Attackers may inject malicious scripts into the application and steal sensitive user data.

OWASP Mapping: A5 – Security Misconfiguration

Screenshot:



6. Cross-Domain Misconfiguration

Severity: Medium

Description:

The application allows resources to be shared across different domains, which may lead to Cross-Site Request Forgery (CSRF) or data leakage issues.

Explanation:

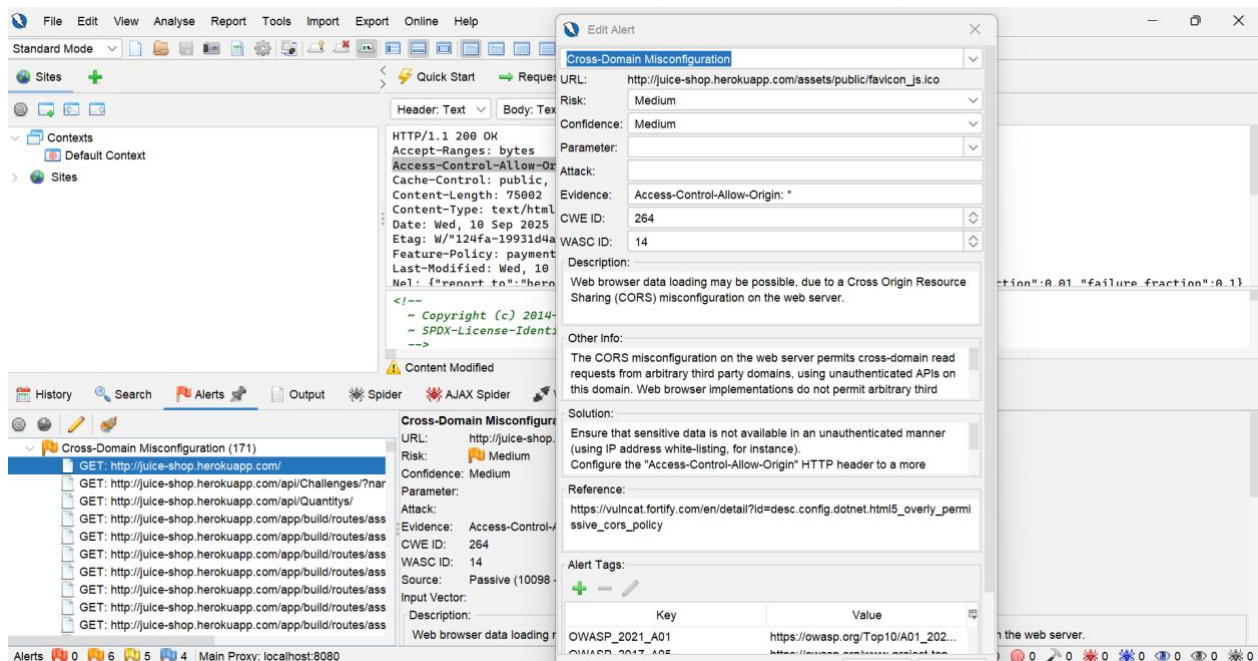
The application allows resources to be accessed from other domains without proper control, which is a common misconfiguration.

Impact:

Enables attackers to perform **Cross-Site Request Forgery (CSRF)** or data leakage.

OWASP Mapping: A8 – Cross-Site Request Forgery (CSRF)

Screenshot:



Conclusion:

The security assessment of the OWASP Juice Shop application was conducted as part of the Future Interns Cyber Security Internship Program (Task 1). The testing combined automated scans (OWASP ZAP, (Burp Suite, payload testing).

A total of 4 confirmed vulnerabilities were identified:

- SQL Injection (Authentication Bypass)
- Broken Access Control (Saved Cards Misuse)
- Reflected Cross-Site Scripting (XSS in Search Bar)
- Security misconfiguration (depending on flagged issue).
- Cross site request forgery (CSRF).

These issues highlight weaknesses in input validation, access control, and server configuration. While the application is intentionally vulnerable, the findings demonstrate the importance of secure coding practices and robust configuration management. The application resisted CSRF attacks, Implementing the suggested mitigations will significantly reduce the risk of exploitation and strengthen the security posture of the application.

OWASP Top 10 Mapping

Vulnerability	Severity	OWASP Top 10 Mapping
SQL Injection (login bypass)	high	A03 Injection
Saved Cards Misuse (Broken Access)	high	A A01: Broken Access Control
Reflected XSS (Search Bar)	medium	A03: injection
OWASP Zap Scan Results	Informal	Not Applicable (No critical findings)
Content security policy (CSP)	medium	A05: (security misconfiguration)
Cross domain misconfiguration)	medium	A08: cross site request forgery (CSRF) Or data leakage

Final Recommendations & Next Steps

Based on the findings from this security assessment, the following recommendations are provided to improve the security posture of the OWASP Juice Shop application and prevent real-world exploitation:

General Recommendations:

- **Input Validation:** Apply strict server-side validation and sanitization for all user inputs to prevent SQL Injection and XSS.
- **Authentication & Access Control:** Enforce re-authentication for sensitive actions and ensure proper role-based access restrictions.
- **Configuration Hardening:** Remove backup/test files and disable directory listing to prevent information disclosure.
- **Encryption Practices:** Securely store sensitive data (such as saved cards) and ensure compliance with data protection standards.
- **Regular Updates & Patching:** Keep all frameworks, libraries, and server components up to date with the latest security patches.

Next steps:

- Conduct a retest after implementing the recommended fixes to validate remediation.
- Integrate security testing tools (ZAP, Burpsuite) into the development lifecycle (DevSecOps approach).
- Perform periodic penetration testing to identify new vulnerabilities.
- Provide security awareness training for developers to align with OWASP secure coding practices.

