Web Application Security Assessment Report – OWASP Juice Shop

Created By :- Nishant
Date :- 08 Sept 2025
Internship Program: Future Interns Cyber Security Internship

Web Application Security Assessment Report – OWASP Juice Shop

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, and **Nikto** for server misconfiguration analysis. During the course of this project, multiple vulnerabilities were identified, including **SQL Injection, Broken Access Control, Reflected XSS, and Information Disclosure** through exposed directories.

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
- **Nikto** – Web server misconfiguration scanning
- **Browser Developer Tools / Manual Payloads** – For custom testing (SQLi, XSS)

Testing Methodology

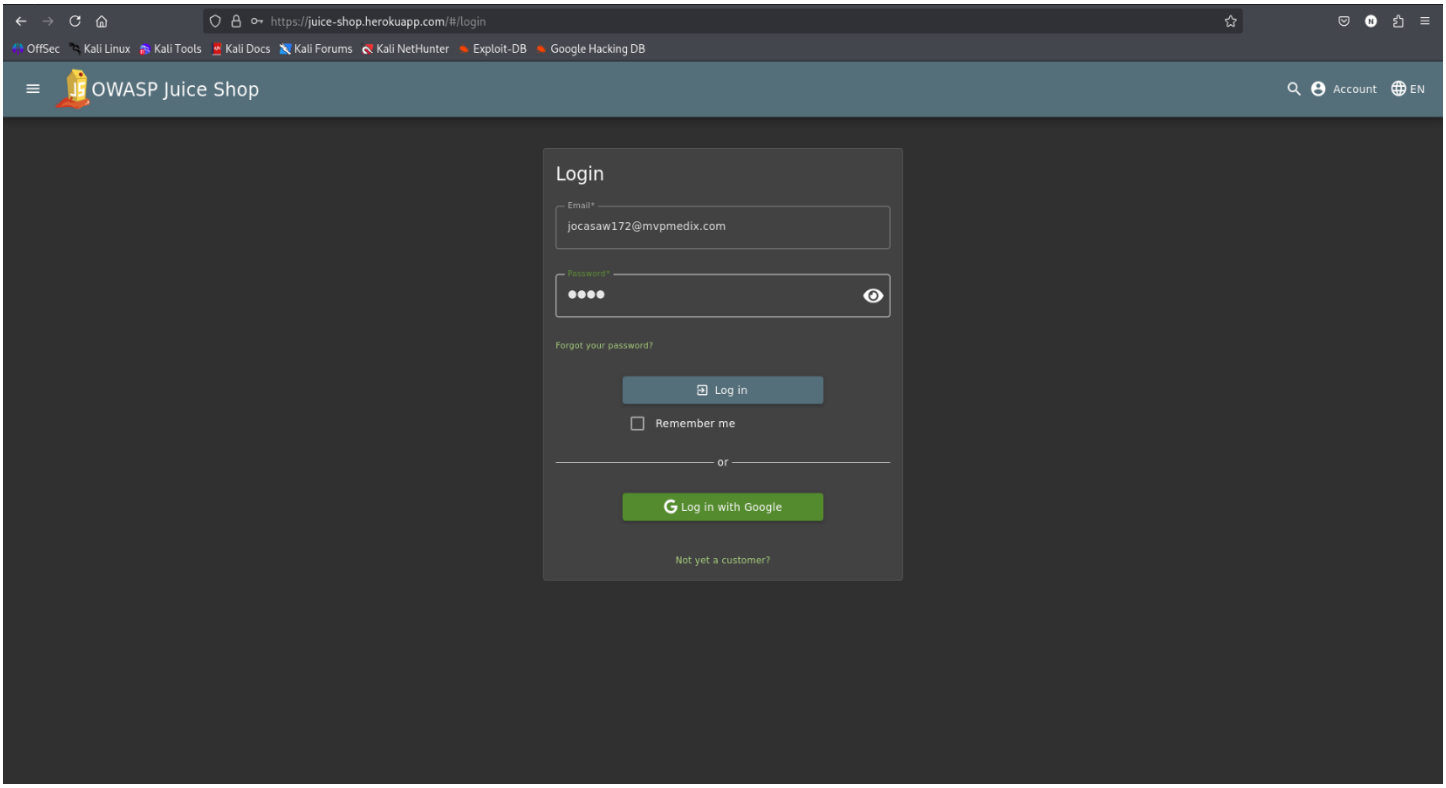
The methodology adopted for this assessment was based on industry standards and the OWASP Testing Guide:

1. **Reconnaissance & Information Gathering** – Identified technologies and potential attack surfaces.
2. **Automated Scanning** – Used OWASP ZAP and Nikto 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.

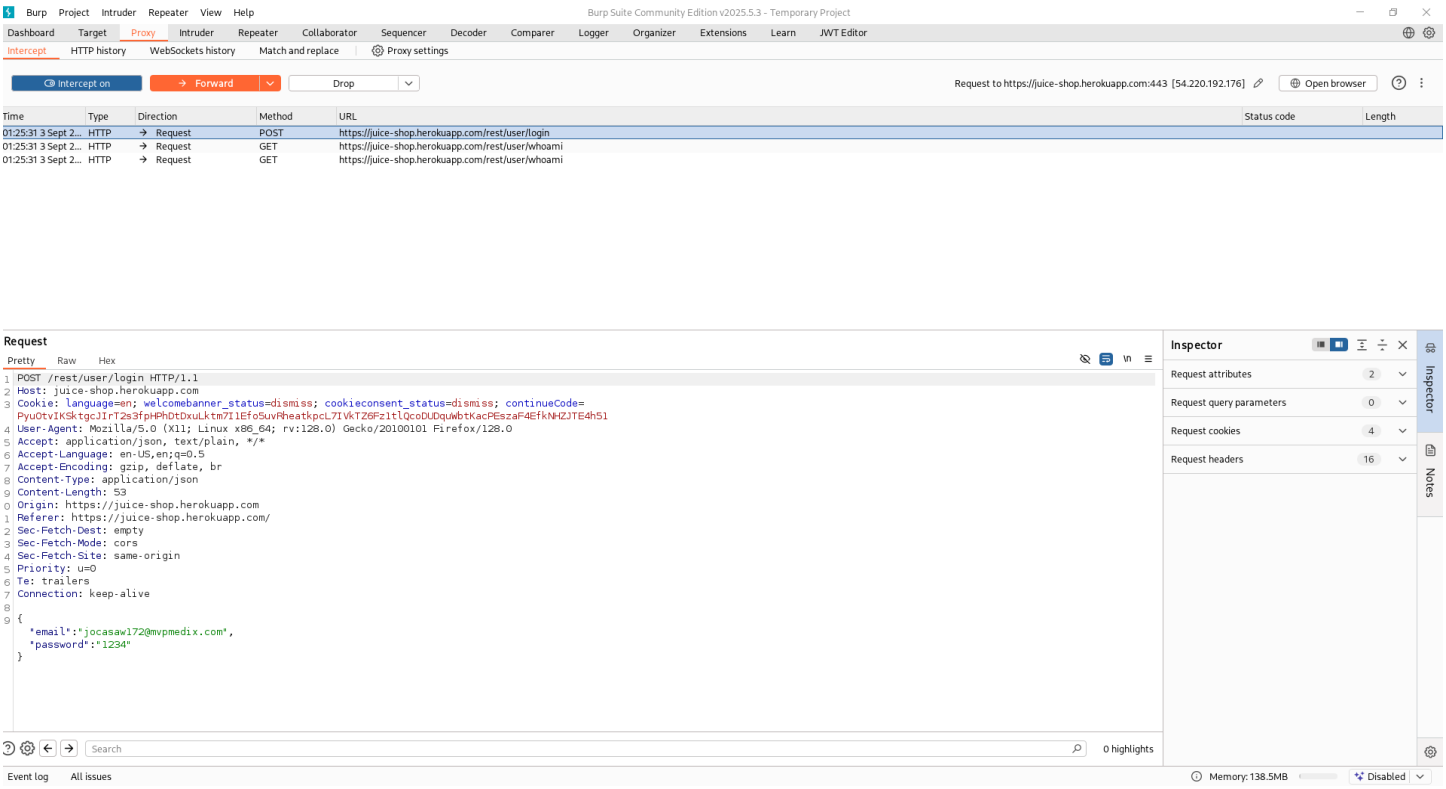
Finding 1: SQL Injection – Authentication Bypass

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 Evidence
Step 1: Opened the login page and entered random credentials in the email and password fields. The login attempt failed as expected.



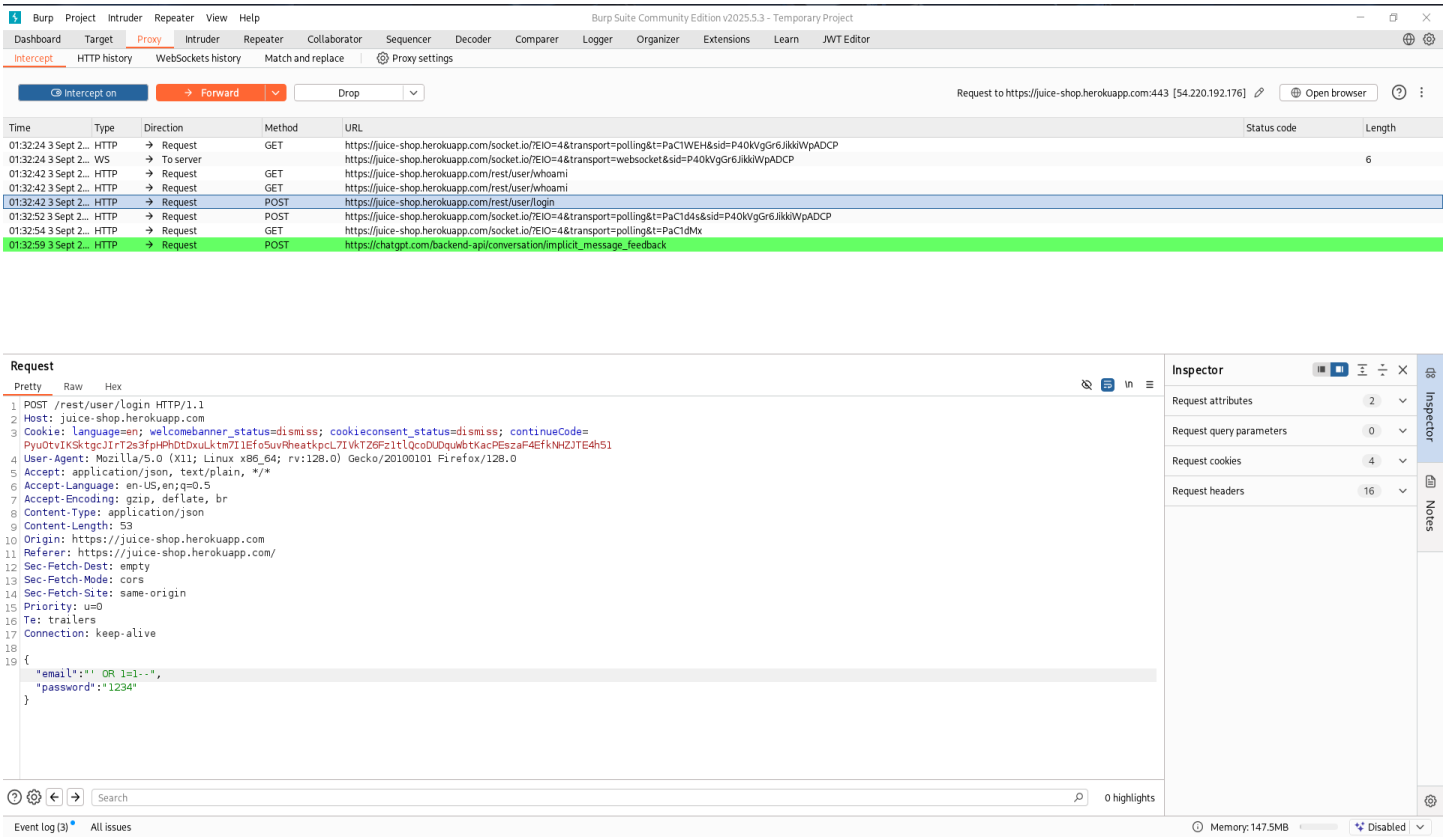
Step 2: Intercepted the login request using **Burp Suite**. The request contained the email and password parameters.



Step 3: Modified the email parameter to include a SQL Injection payload:

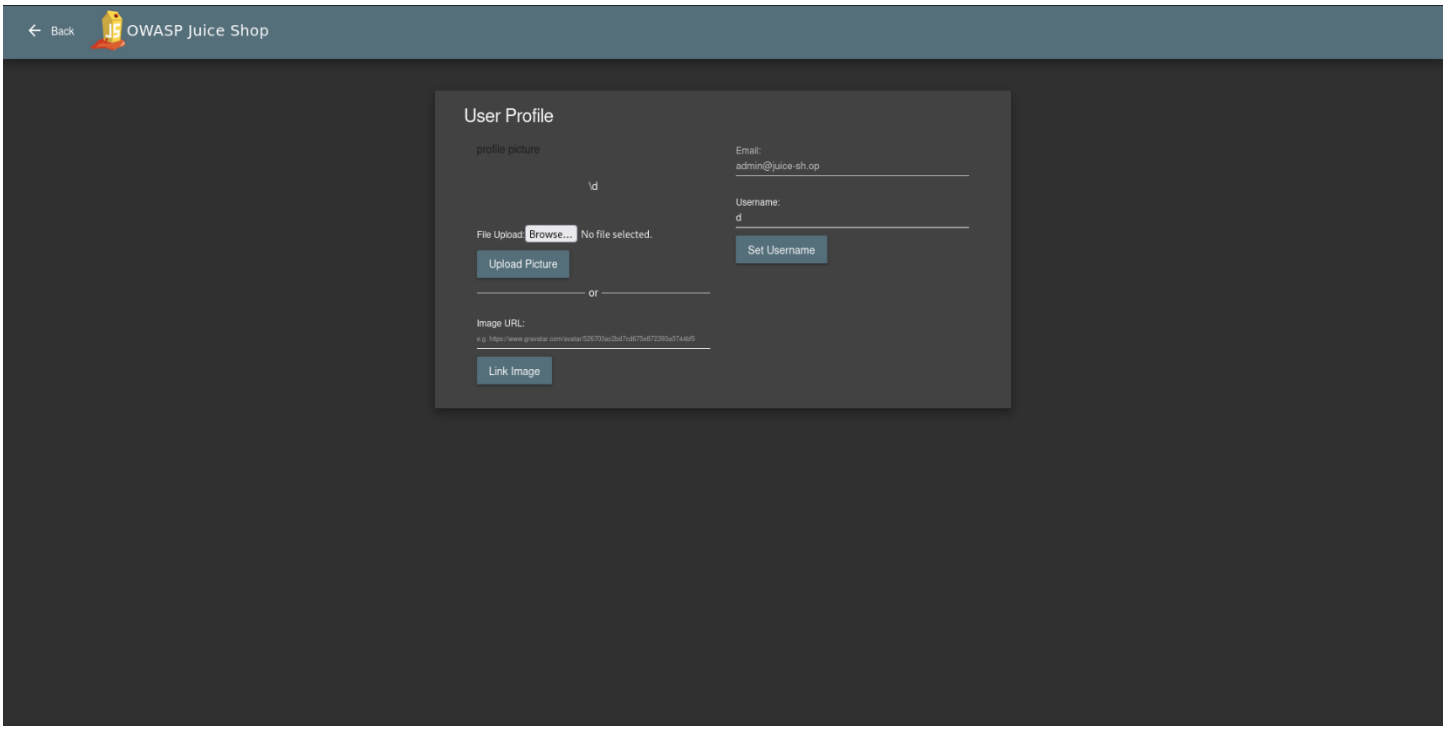
' OR 1=1--

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:

- Full administrative access can be achieved.
- Attackers may extract sensitive information from the database.
- Leads to complete compromise of the application and stored user data.

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:

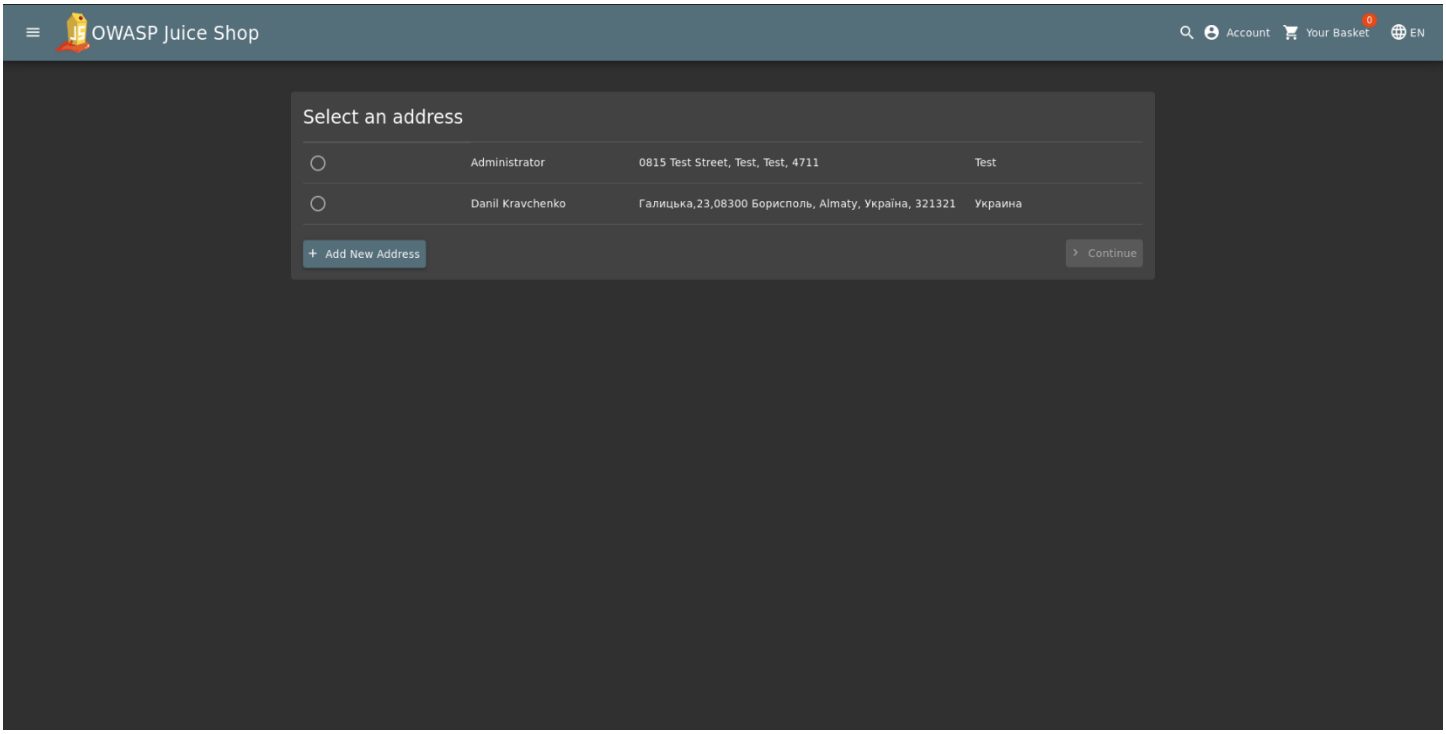
- A03: Injection

Finding 2: Broken Access Control – Saved Cards Misuse

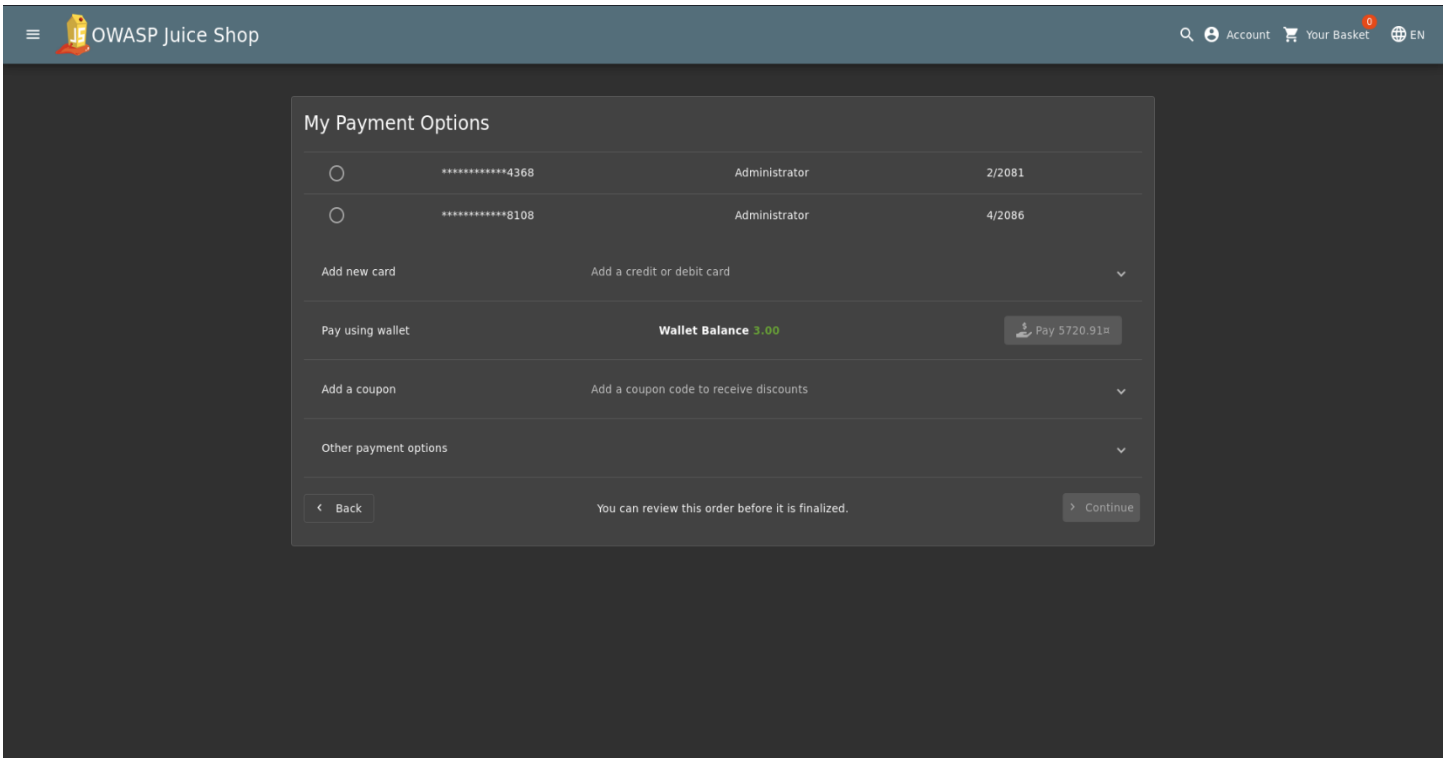
Severity: High

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.

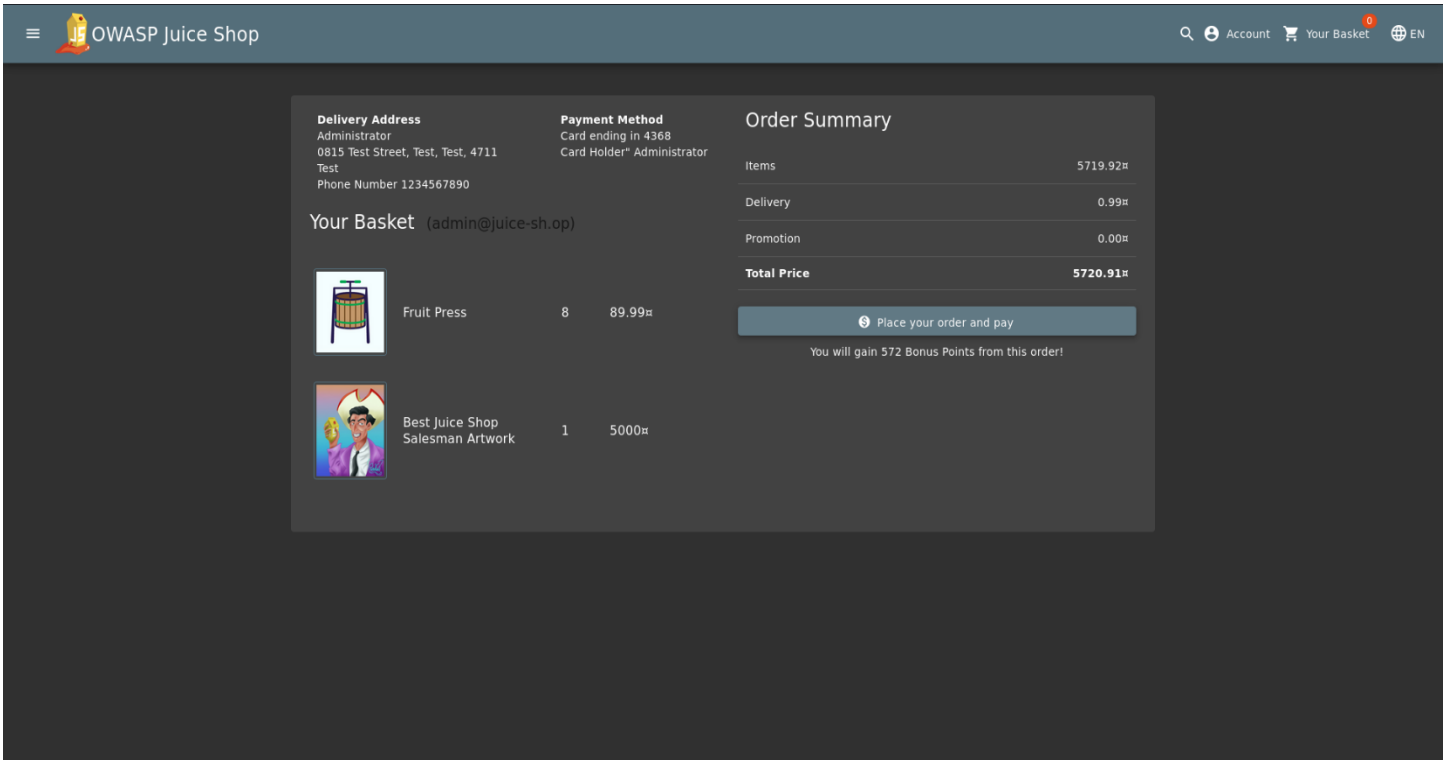
Steps with Evidence
Step 1: Added an item to the shopping cart and proceeded to checkout.



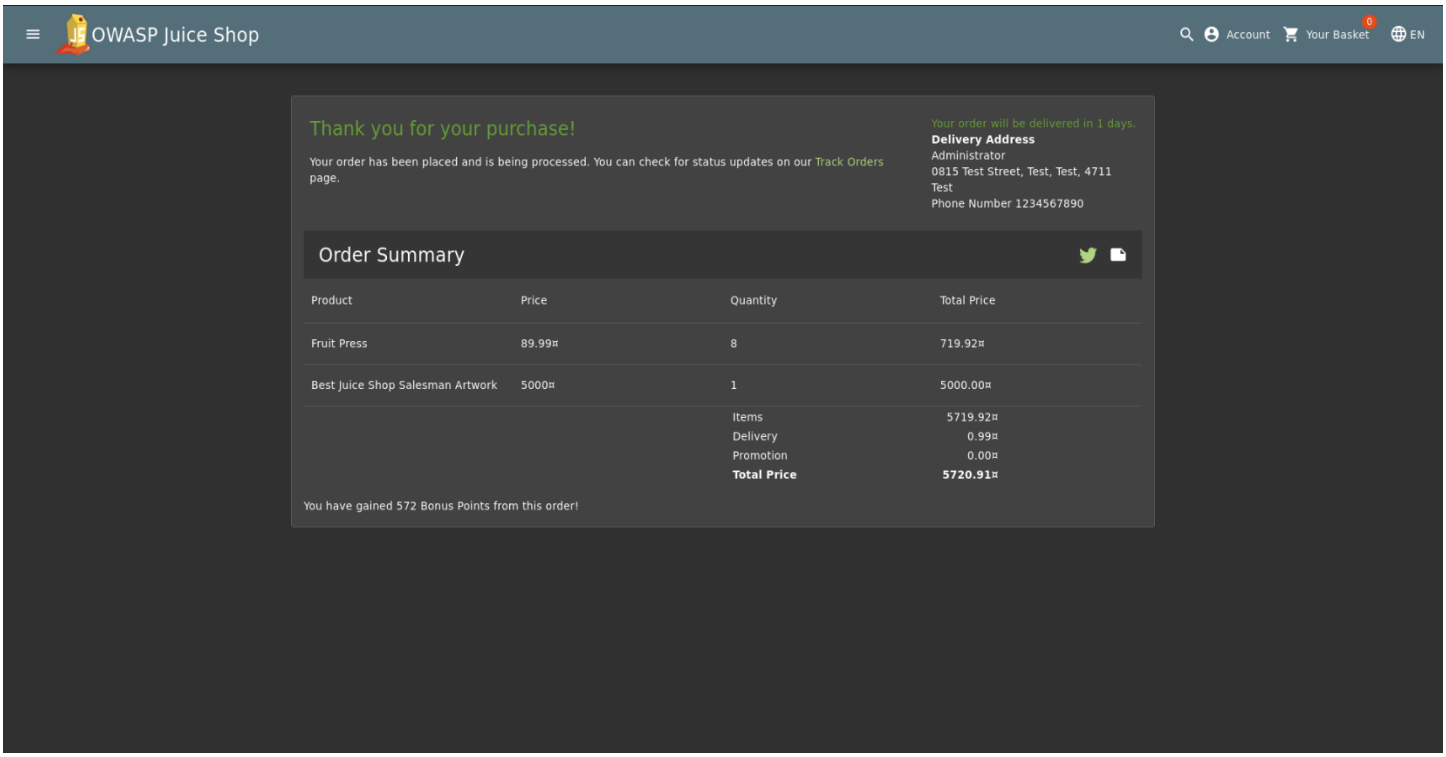
Step 2: Reached the payment page. Two saved credit cards and a digital wallet option were available for selection.



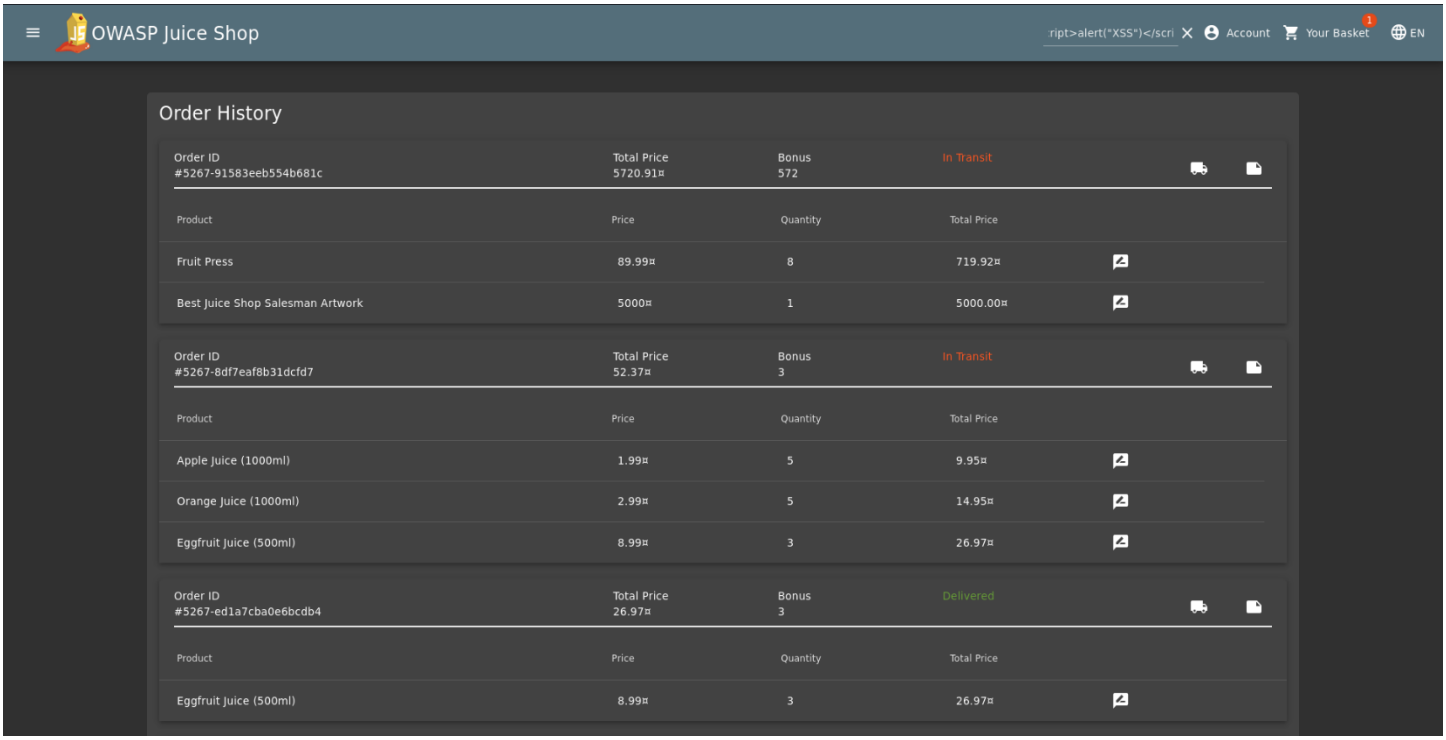
Step 3: Selected one of the saved cards and continued. No CVV or OTP verification was required to complete the purchase.



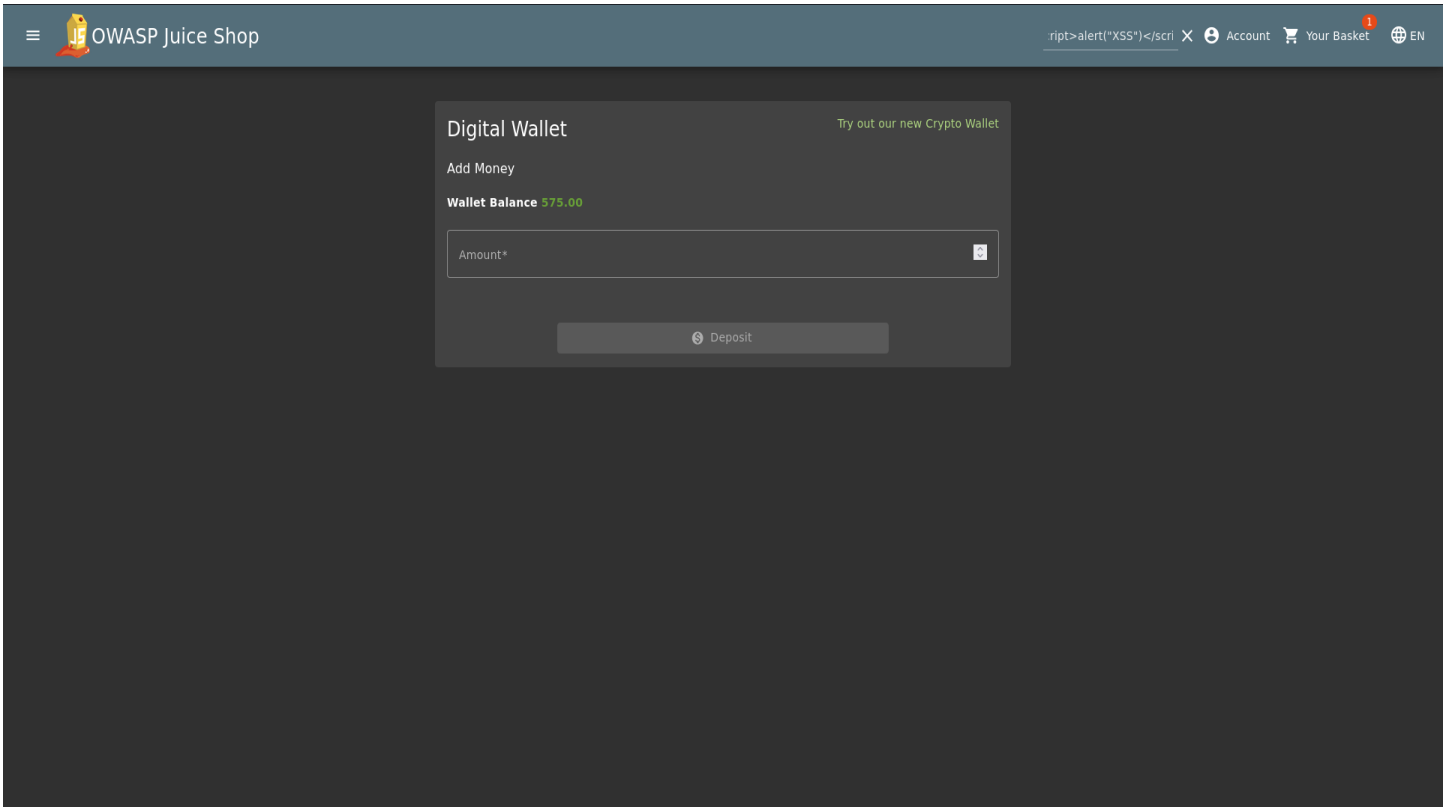
Step 4: Completed the transaction successfully. The application displayed a confirmation message along with the order summary.



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



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



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

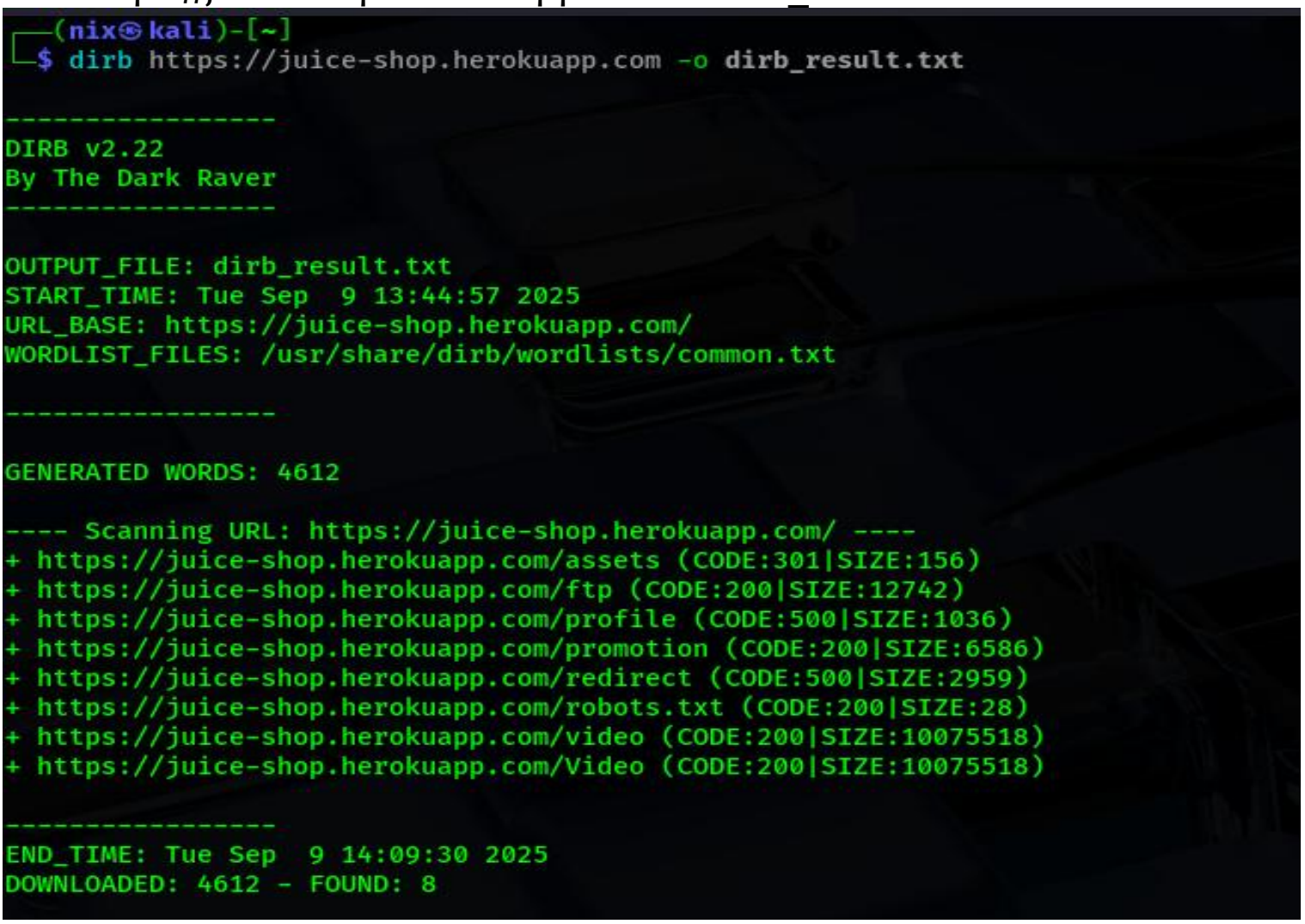
Finding 3: Information Disclosure – /ftp Directory Exposure

Severity: Medium
Description: Using **DirBuster**, a hidden /ftp directory was discovered. It exposes sensitive files including backup configs, error logs, encrypted announcements, compiled Python code, and a KeePass database (incident-support.kdbx).

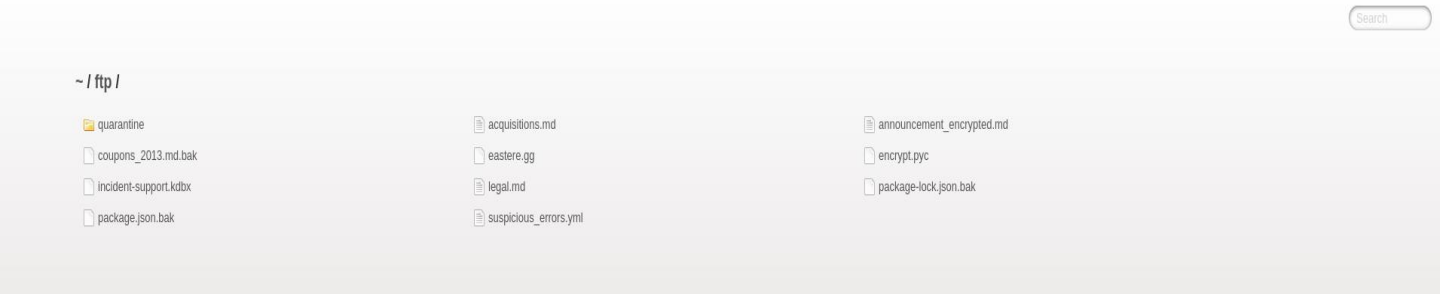
Steps with Evidence

Step 1: Executed a directory brute-force scan with Dirb using a common wordlist.

dirb https://juice-shop.herokuapp.com -o dirb_result.txt



Step 2: Accessed the /ftp directory in the browser. The server displayed an open directory listing with sensitive files.



Step 3: Observed files such as:

- incident-support.kdbx (KeePass DB, may store credentials)
- package.json.bak and package-lock.json.bak (backup configs)
- suspicious_errors.yml (error logs)
- encrypt.pyc (compiled Python code)

Impact:

- Disclosure of configs and credentials.
- Files like .kdbx and .pyc may lead to full compromise.

Mitigation:

- Disable **directory listing** on the web server.
- Remove backup and test files before deploying to production.
- Store sensitive files outside the web root.
- Enforce strict file access permissions.

OWASP Top 10 Mapping:

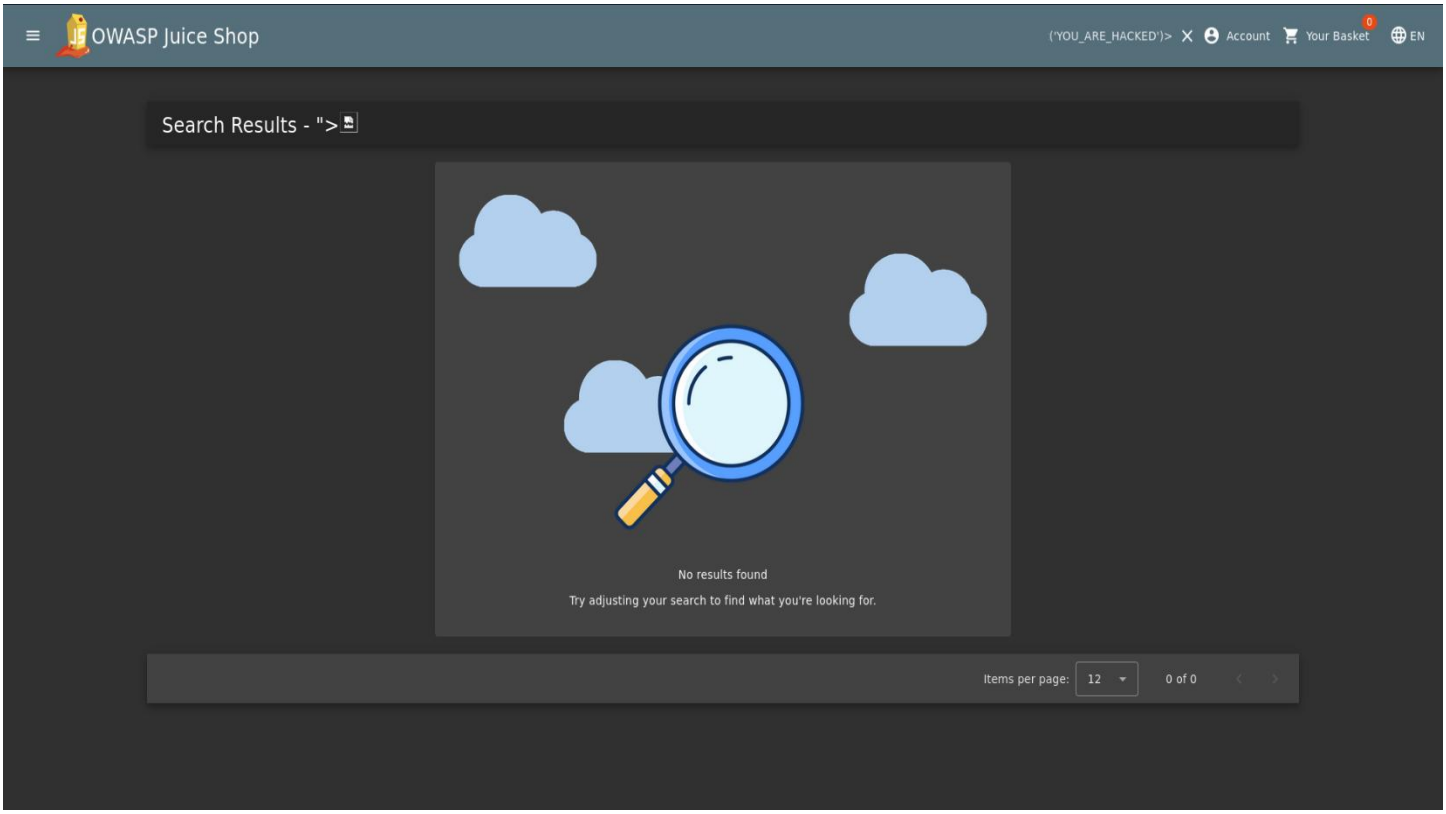
- A01: Broken Access Control
- A05: Security Misconfiguration

Finding 4: Reflected XSS – Search Bar

Severity: Medium

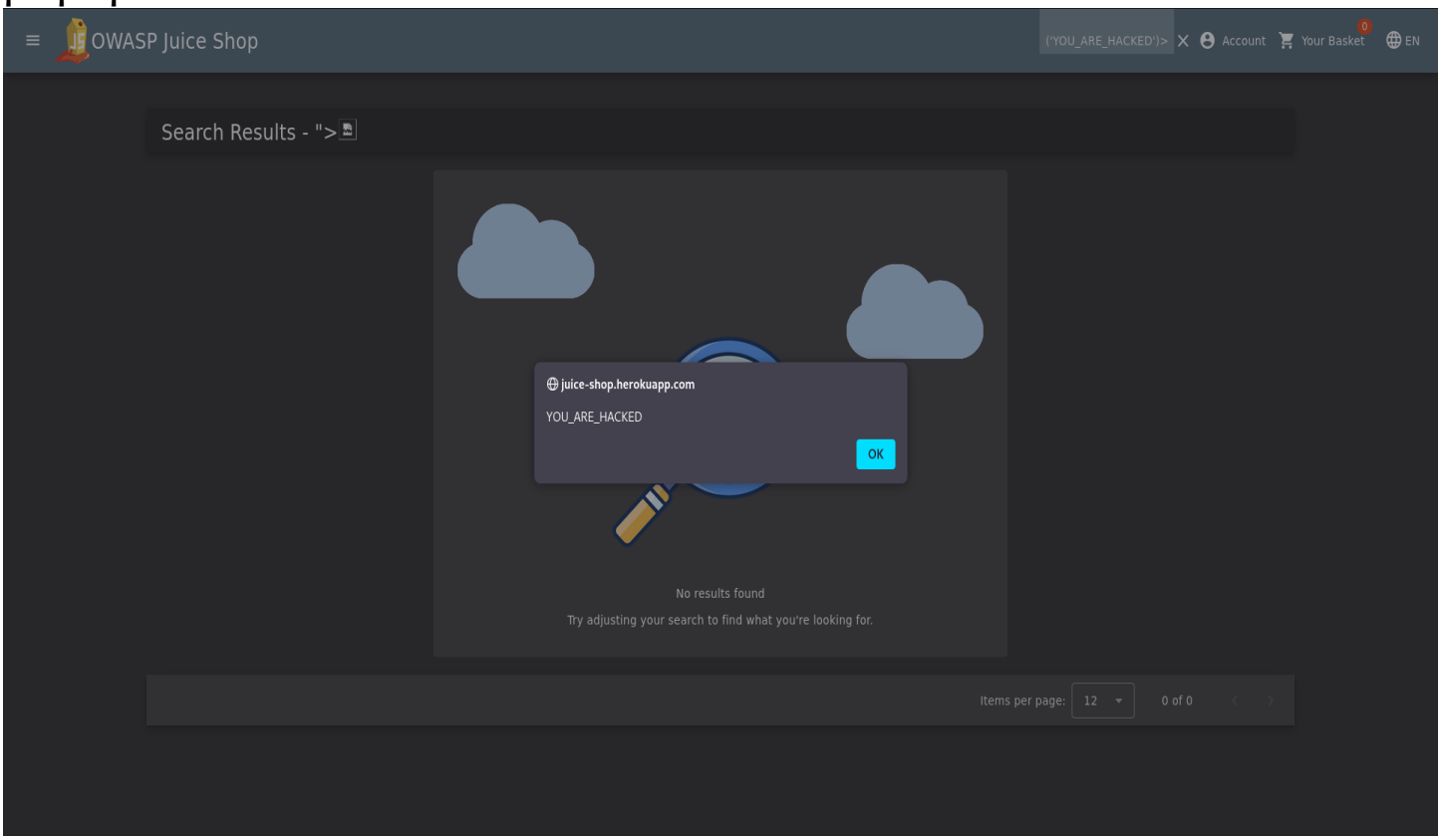
Description:
The search functionality reflects unsensitized user input in the response. By injecting a crafted payload, JavaScript code was executed in the browser, confirming a Reflected Cross-Site Scripting (XSS) vulnerability. This type of issue can be exploited by attackers through malicious links,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

Steps with Evidence
Step 1: Navigated to the application's search bar and entered a crafted XSS payload:
"><img src=x onerror=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 DOMPurify for input validation.

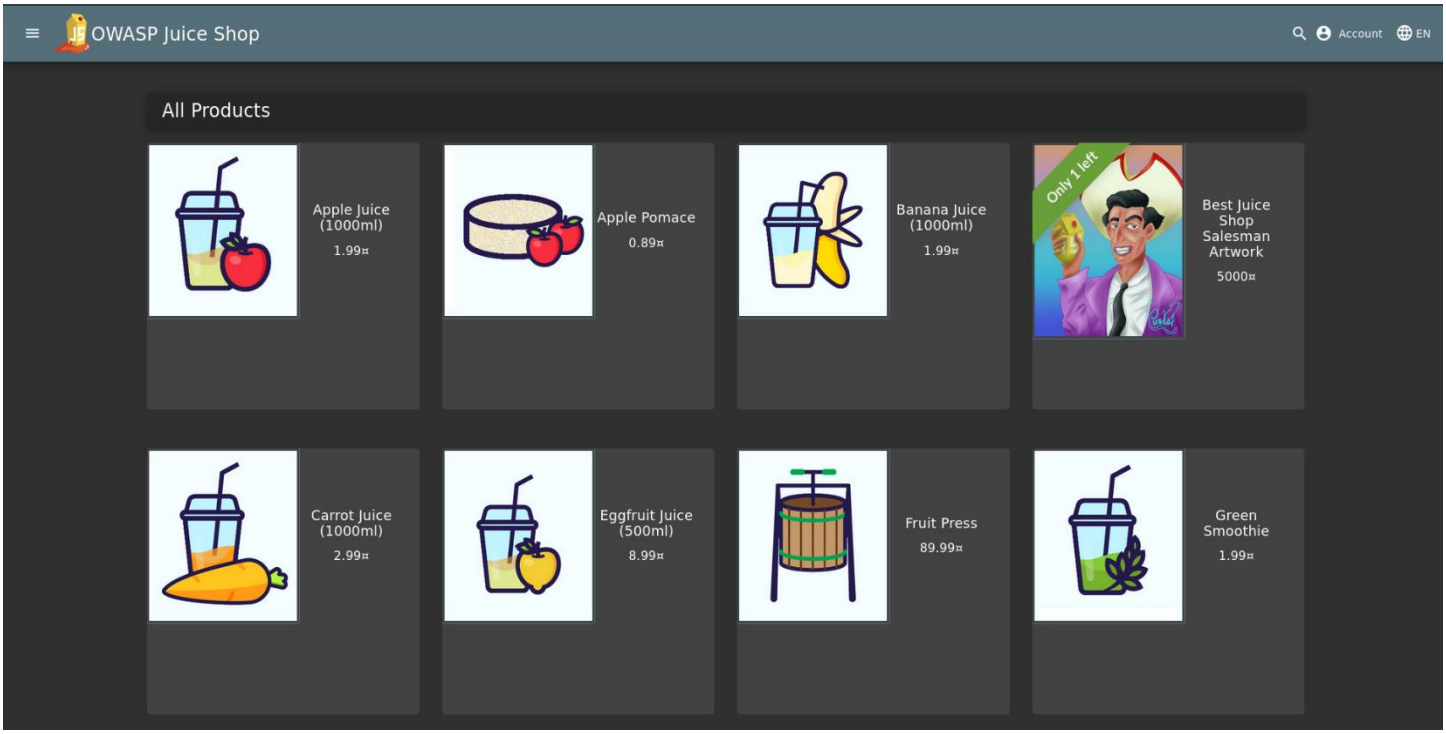
OWASP Mapping:

- A03: Injection

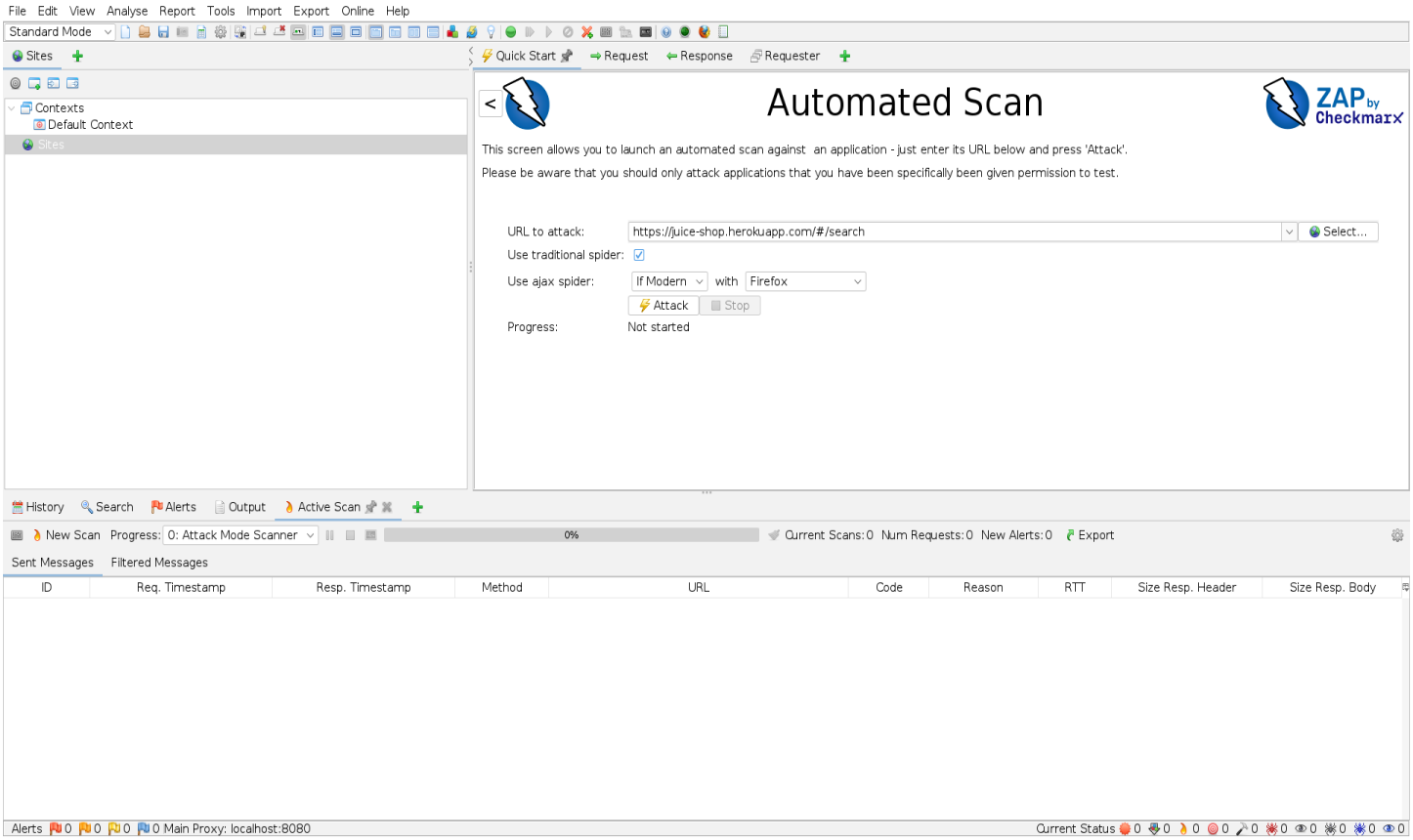
Finding 5: Automated Scan – OWASP ZAP

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.

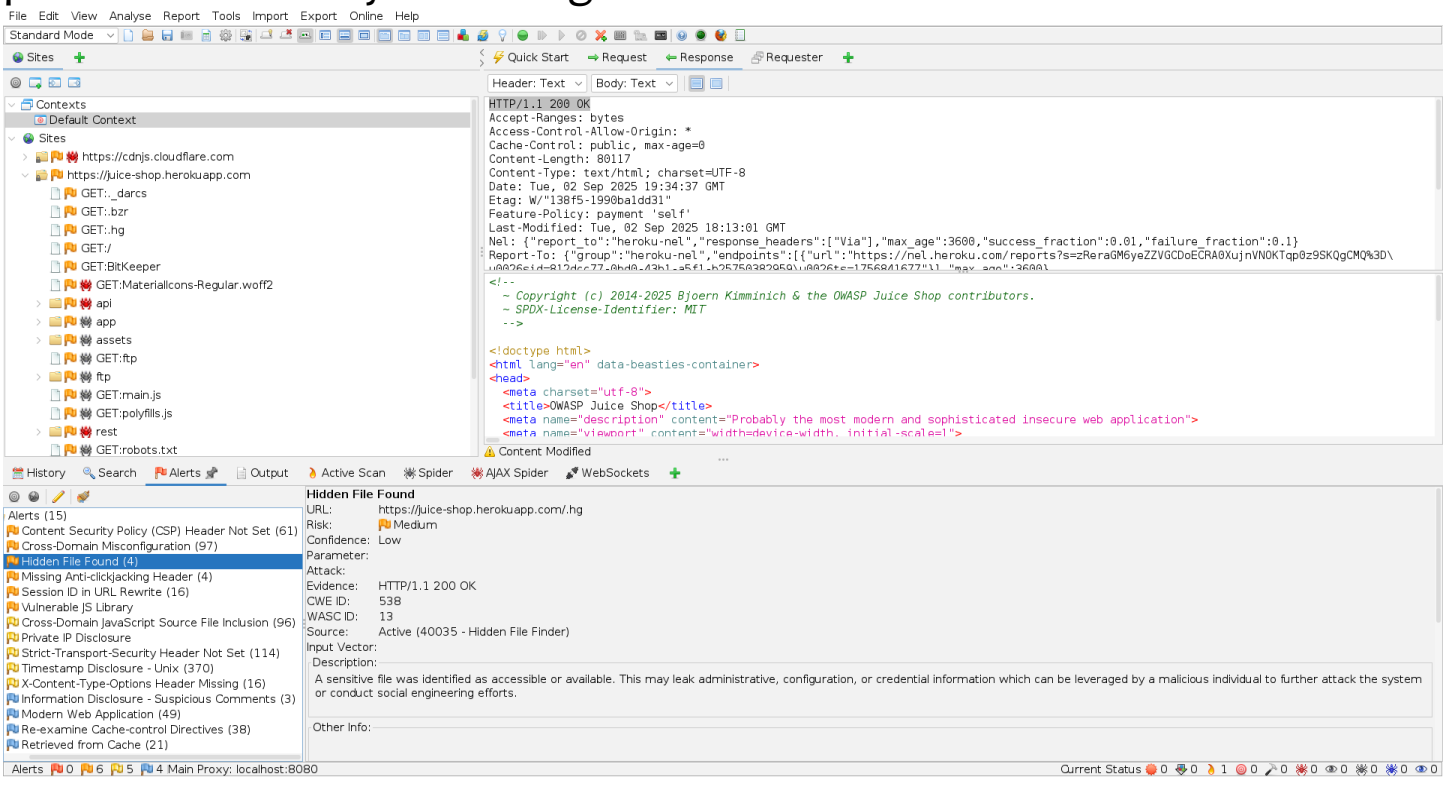
Steps with Evidence
Step 1: Launched OWASP ZAP and provided the target Juice Shop application URL for scanning. The target application's initial view was verified.



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



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:

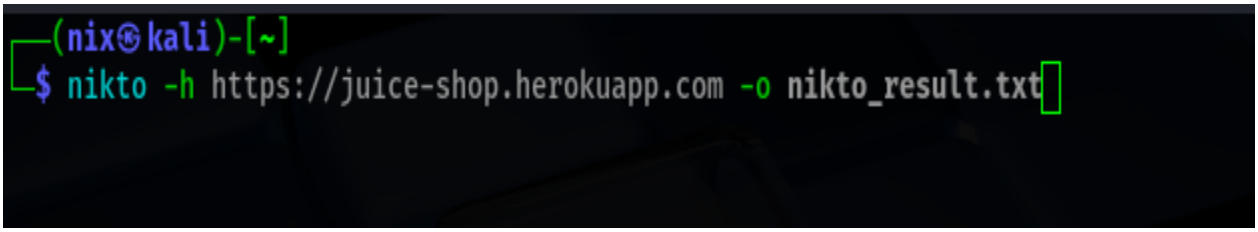
- Multiple categories (A01: Broken Access Control, A03: Injection, A05: Security Misconfiguration) depending on flagged issue.

Finding 6: Nikto Scan – Informational Results

Severity: Informational

Description:
A Nikto web server scan was performed against the OWASP Juice Shop application to identify server misconfigurations, outdated components, and known vulnerabilities. The scan completed successfully but did not reveal any critical security issues.

Steps with Evidence
Step 1: Executed the Nikto scan against the target URL:
nikto -h https://juice-shop.herokuapp.com -o nikto_result.txt



Step 2: The scan results listed several observations, including:

- Presence of unusual HTTP headers (Cross-Domain, Reporting-Endpoints).
- Missing **X-Content-Type-Options** header.
- Robots.txt file exposing /ftp path.
- Multiple references to potential backup/certificate files.

Impact:

- No exploitable high or medium-severity issues were identified.
- General server information such as headers and directories were listed.

Mitigation:

- Regularly update web server and application dependencies.
- Periodically rerun vulnerability scans as part of routine security testing.

OWASP Mapping:

- Not directly mapped – Informational only.

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, Nikto) with manual verification (Burp Suite, payload testing).

A total of 4 confirmed vulnerabilities were identified:

- SQL Injection (Authentication Bypass)
- Broken Access Control (Saved Cards Misuse)
- Information Disclosure (/ftp Directory)
- Reflected Cross-Site Scripting (XSS in Search Bar)

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, and Nikto scans did not reveal critical findings, showing some protective mechanisms are in place. 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	A01: Broken Access Control
/ftp Directory Exposure	Medium	A05: Security Misconfiguration
Reflected XSS (Search Bar)	Medium	A03: Injection
Owasp Zap Scan Results	Informational	Not Applicable (No critical findings)
Nikto Scan Results	Informational	Not Applicable (No critical findings)

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, Burp, Nikto) 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.

Thank You!