Web Penetration Testing Report

Executive Summary

Purpose and Overview

The purpose of this test was to evaluate the security posture of the OWASP Juice Shop application by identifying and exploiting potential vulnerabilities. This assessment uncovered critical vulnerabilities, including **Cross-Site Scripting (XSS)** and **Brute-Force attacks**, which can severely impact the application's security and user data.

Key Findings

1. **Reflected XSS:** Attackers can execute malicious scripts in users' browsers.
2. **Brute-Force Vulnerability:** Weak login security allowed unauthorized access to the admin panel.
3. **Improper Input Validation:** Lack of input sanitization leaves the application exposed to various injection attacks.

High-Level Impact

- **User Data Compromise:** Unauthorized access to sensitive user information.
- **Arbitrary Code Execution:** Malicious scripts can run on users' browsers.
- **Service Disruption:** Exploiting these vulnerabilities can lead to system compromise or downtime.

Summary of Recommendations

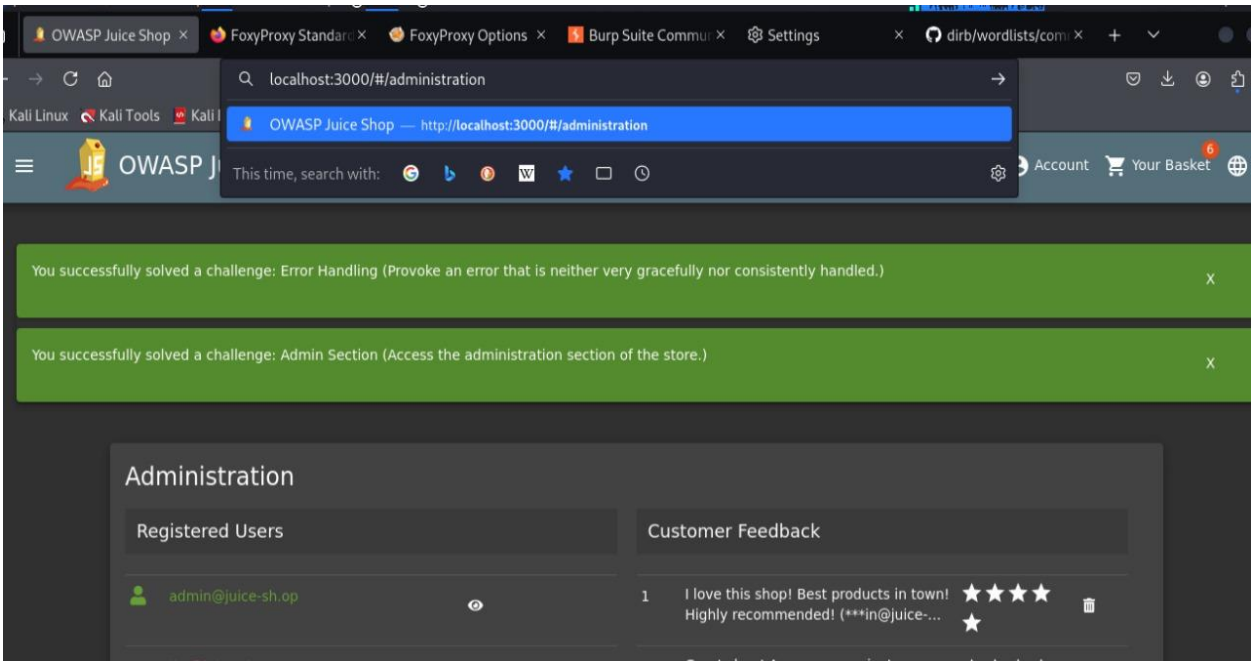
- Implement input sanitization and output encoding to prevent XSS.
- Enforce strong authentication mechanisms (rate limiting, CAPTCHA, account lockout).
- Conduct regular security assessments and training for developers.

1:Scope and Methodology

Scope:
Owasp Juice Shop is tested on: http: // Localhost: 3000.

Approach:
Black-Box Testing was used.

6 **Screenshot:**



7

8 **Tools used:**

9 **Burp Suite: for requests for requests and testing attacks.**

10 **Kali Linux Tools: To help examine and exploit.**

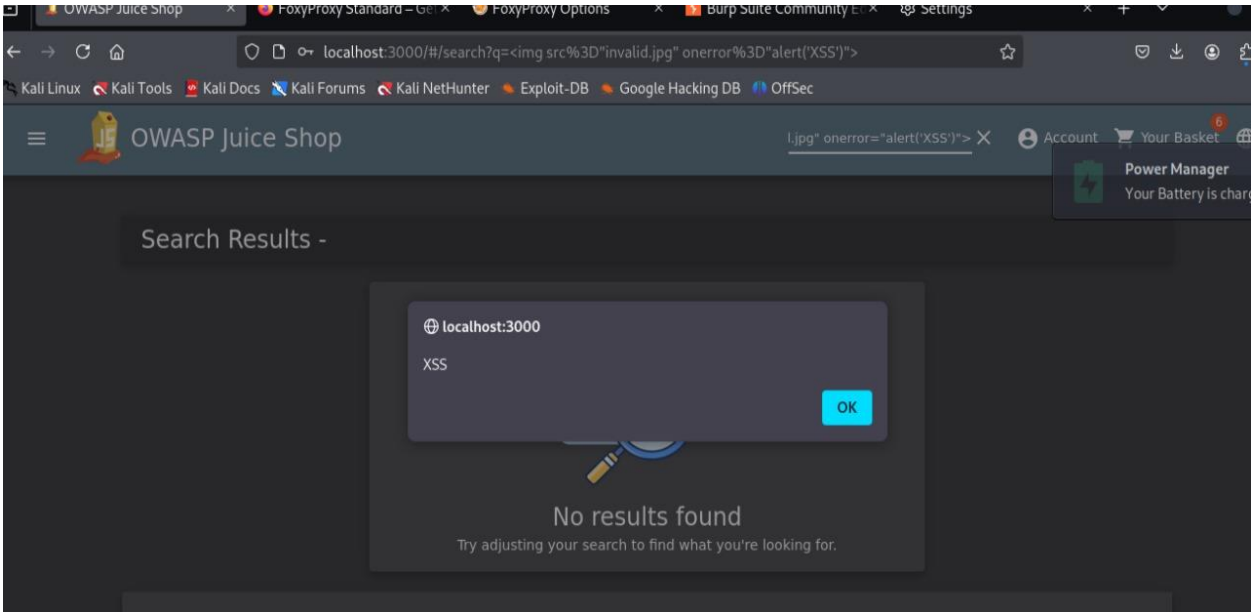
11 **Owasp Zap: to automatically test.**

Vulnerability Findings

2. Reflected Cross-Site Scripting (XSS)

- **Description:**
  A reflected XSS vulnerability was identified in the search functionality. An attacker can inject malicious JavaScript code that executes in the victim's browser when they perform a search.
- **Impact:**
  - Arbitrary script execution in the browser.
  - Stealing cookies, session tokens, or sensitive data.
  - Redirecting users to malicious websites.
- **Proof of Exploitation:**
  The following payload was used:

```
html
Copy code
<img src="invalid.jpg" onerror="alert('XSS')">
```



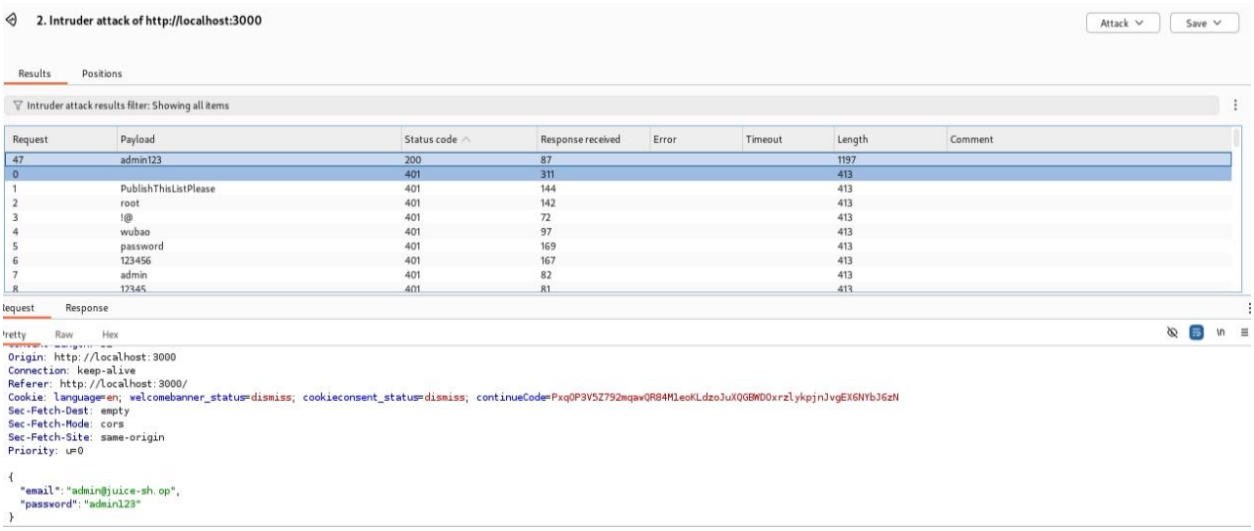
- **Remediation Steps:**
  - Implement server-side validation to sanitize input.
  - Use libraries like `DOMPurify` for cleaning HTML and JavaScript inputs.
  - Apply Content Security Policy (CSP) headers to restrict JavaScript execution.

---

3. Brute-Force Attack on Login Panel

- **Description:**
  The login page was vulnerable to brute-force attacks due to the absence of rate-limiting or account lockout mechanisms.
- **Impact:**
  - Unauthorized access to sensitive accounts, including admin credentials.
  - Potential exposure of user data stored in the admin panel.
- **Proof of Exploitation:**
  Using **Burp Suite Intruder**, the following credentials were discovered:
  - **Email:** admin@juice-sh.op
  - **Password:** admin123

Screenshot of Successful Brute-Force Attack:



- **Remediation Steps:**
  - Add rate-limiting to restrict the number of login attempts.
  - Implement CAPTCHA to prevent automated login attempts.
  - Lock accounts temporarily after a set number of failed login attempts.

Exploitation and Attack Simulation

Tools and Techniques Used

- **Burp Suite:** For intercepting and manipulating HTTP requests.
- **Kali Linux Tools:** For executing brute-force attacks.
- **OWASP ZAP:** For identifying vulnerabilities such as XSS.
- **Browser DevTools:** For testing JavaScript payloads and manual injections.

Outcome of the Attack

1. Successfully exploited **XSS** to execute arbitrary JavaScript code in the browser.
2. Discovered **admin credentials** through brute-force attack.
3. Gained unauthorized access to the **admin panel**, exposing sensitive user data.

Recommendations

Short-Term Recommendations

1. **Fix XSS Vulnerabilities:**

    - Sanitize all user inputs.
    - Use output encoding when displaying user-provided data.
    - Enforce CSP headers to limit JavaScript execution.
2. **Strengthen Authentication Mechanisms:**
    - Implement CAPTCHA on login forms.
    - Enforce account lockout after a set number of failed login attempts.
    - Use rate-limiting to restrict login attempts.

Long-Term Recommendations

1. **Conduct Regular Security Assessments:**
    - Schedule penetration tests and vulnerability scans regularly.
    - Include security testing in the development lifecycle.
2. **Train Developers:**
    - Provide training on secure coding practices.
    - Encourage the use of secure libraries and frameworks.
3. **Adopt a Web Application Firewall (WAF):**
    - Deploy a WAF to filter and block malicious traffic.
    - Monitor for common attack patterns like SQL injection and XSS.

Summary of Findings

Vulnerability	Risk Level	Impact
Reflected XSS	High	Arbitrary script execution.
Brute-Force Attack	High	Unauthorized access to admin panel.
Weak Input Validation	Medium	Potential for injection attacks.

Conclusion

Based on the findings, the OWASP Juice Shop application has significant security weaknesses that require immediate attention. While the application has several critical vulnerabilities, implementing the recommended measures will significantly improve its security posture.

عبدالرحمن حسام الدين شعت /2305573

احمد رضا /2305078

احمد مهران /2305051