Penetration Testing Report on OWASP Juice Shop

Executive Summary

This report provides a comprehensive overview of the penetration testing conducted on OWASP Juice Shop, a purposely vulnerable web application designed for learning and practicing web security. The primary goal of this exercise was to identify and exploit vulnerabilities within the application while evaluating its overall security posture. The report documents all findings, their impacts, and recommendations for mitigating the identified issues.

1. Introduction

1.1 **Objective**

- Assess the security vulnerabilities within the OWASP Juice Shop application.
- Simulate real-world attack scenarios to identify potential threats.
- Provide actionable recommendations to improve the security of the application.

1.2 **Scope**

- Testing included application functionalities, user input fields, authentication mechanisms, and API endpoints.
- The assessment adhered to OWASP Testing Guide principles to maintain structured and ethical testing practices.

1.3 **Methodology** The penetration testing process followed these key phases:

- **Reconnaissance**: Gathering information about the application.
- **Scanning**: Identifying vulnerabilities and weak points.
- **Exploitation**: Attempting to exploit identified weaknesses.
- **Post-Exploitation**: Analyzing the impact of successful attacks.
- **Reporting**: Documenting findings and proposing remediations.

2. Tools Used

The following tools were utilized during the assessment:

- **Burp Suite**: For intercepting and manipulating web traffic.
- **OWASP ZAP**: To automate the scanning of vulnerabilities.
- **SQLmap**: For identifying and exploiting SQL injection vulnerabilities.
- **Nmap**: Used for network scanning and reconnaissance.
- **Browser Developer Tools**: To inspect and manipulate client-side web elements.

- **Kali Linux Tools**: Leveraged for exploitation techniques and payload crafting.

3. Vulnerability Assessment

3.1 **Authentication Bypass**

- **Finding**: Weak password policies and lack of account lockout mechanisms.
- **Exploitation**: Conducted brute force attacks using a wordlist.
- **Impact**: Enabled unauthorized access to user accounts.
- **Mitigation**: Enforce strong password policies and implement account lockout mechanisms.

3.2 **SQL Injection**

- **Finding**: Input fields were vulnerable to SQL injection due to improper sanitization.
- **Exploitation**: SQL queries retrieved sensitive database information.
- **Impact**: Compromised user credentials and application data.
- **Mitigation**: Use parameterized queries and validate user inputs.

3.3 **Cross-Site Scripting (XSS)**

- **Finding**: Input fields accepted and executed malicious scripts.
- **Exploitation**: Scripts executed in user browsers, stealing cookies and sessions.
- **Impact**: Session hijacking and user impersonation.
- **Mitigation**: Implement robust input validation and output encoding mechanisms.

3.4 **Broken Access Control**

- **Finding**: Unauthorized users accessed restricted resources.
- **Exploitation**: Direct access to admin functionalities via hidden endpoints.
- **Impact**: Privilege escalation and unauthorized actions.
- **Mitigation**: Apply strict role-based access controls and secure all endpoints.

4. Exploitation Techniques

4.1 **Reconnaissance**

- Conducted application mapping using Nmap and Burp Suite.
- Identified exposed endpoints and entry points for attacks.

4.2 **Payload Injection**

- Crafted and tested malicious payloads to exploit vulnerabilities such as SQL injection and XSS.
- Utilized automated tools to streamline testing.

4.3 **Brute Force Attacks**

- Performed brute force attacks on login pages using tools like Hydra.
- Identified weak credentials, gaining unauthorized access.

4.4 **Post-Exploitation**

- Extracted sensitive data, including user credentials and application configurations.
- Analyzed application logs to assess the extent of exploitation.

5. Results and Findings

Vulnerability	**Severity**	**Impact**
SQL Injection	High	Data exfiltration and database compromise
Cross-Site Scripting (XSS)	High	User data theft
Authentication Bypass	Medium	Unauthorized access to accounts
Broken Access Control	High	Privilege escalation
Information Disclosure	Medium	Leakage of sensitive application details

6. Recommendations

6.1 **General Security Measures**

- Update dependencies and third-party libraries regularly.
- Conduct periodic security audits and penetration testing.
- Employ Content Security Policies (CSP) to mitigate XSS attacks.

6.2 **Specific Mitigations**

- Validate and sanitize all user inputs to prevent SQL injection and XSS.
- Implement strong authentication and role-based authorization mechanisms.
- Encrypt sensitive data both in transit and at rest.
- Monitor logs for unusual activities and suspicious patterns.

6.3 **Development Practices**

- Adhere to secure coding guidelines, such as those provided by OWASP.
- Train developers on security best practices and awareness programs.

7. Conclusion

The penetration testing exercise uncovered multiple critical vulnerabilities in the OWASP Juice Shop application, underlining the importance of regular security assessments and adherence to secure coding practices. Addressing the identified issues will significantly improve the application's resilience against attacks.

8. References

- OWASP Testing Guide: https://owasp.org/
- CWE Database: https://cwe.mitre.org/
- Burp Suite Documentation: https://portswigger.net/burp

Appendices

A. Screenshots of Exploits

- Screenshots of exploited vulnerabilities, including SQL injection results and XSS payloads.

B. Tools Configuration

- Details on tool configurations, such as Burp Suite and Nmap setups, for replication.

C. Vulnerability Details

- Step-by-step instructions to reproduce each identified vulnerability, supplemented with screenshots.