SECURITY ASSESSMENT

Juice Shop Report

Submitted to: << Requester>>

Security Analyst:

Mohamed Awad Mohamed
Mary Alfons Shokry
Mohamed Gamal
Yomna Mohamed Abdelfatah
Mahmoud Mohamed Ismail

Date of Testing: 20/12/2024
Date of Report Delivery: 24/12/2024

Table of Contents

Contents

Security Engagement Summary

Engagement Overview

This engagement was part of a penetration testing training program aimed at understanding web application vulnerabilities using the OWASP Juice Shop. The primary goal was to identify vulnerabilities within the application, exploit them, and document findings for educational purposes. The engagement was conducted by a collective effort from our team of four members, focusing on practical learning in a controlled environment. This was a one-time analysis designed to enhance our understanding of web security.

Scope

The scope of this engagement involved the OWASP Juice Shop web application, which is intentionally designed to be vulnerable. The focus was on identifying common web application vulnerabilities, including SQL injection, cross-site scripting (XSS), and insecure direct object references. This scope is appropriate as it provides a hands-on environment to practice exploiting real-world vulnerabilities and understanding their implications.

Executive Risk Analysis

The OWASP Juice Shop challenge presents a medium risk level. The identified vulnerabilities, such as SQL injection and XSS, expose the application to potential attacks that could lead to unauthorized data access or manipulation.

Executive Recommendation

Remediation efforts are warranted for the vulnerabilities discovered in the OWASP Juice Shop. It is crucial to address these issues to prevent potential exploitation. The highest priority should be mitigating SQL injection vulnerabilities, as they can lead to significant data breaches.

Recommended Remediation Steps:

1. Parameterized Queries: Implement prepared statements to prevent SQL injection.
2. Input Validation: Validate and sanitize user input to prevent XSS and other injection attacks.
3. Access Control Measures: Apply strict access controls to prevent insecure direct object references.

Significant Vulnerability Summary

High Risk Vulnerabilities

- SQL Injection
- Cross Site Scripting (XSS)

Medium Risk Vulnerabilities

- Insecure Direct Object Reference
- Server-Side Request Forgery (SSRF)
- Cross-Site Request Forgery (CSRF)

Low Risk Vulnerabilities

- Information Disclosure

Significant Vulnerability Details

SQL Injection

Risk Level: **HIGH**

Vulnerability detail

- Summary: This vulnerability was identified through testing input fields on the Juice Shop web application. It allowed the execution of arbitrary SQL queries against the database.
- Probability of Exploit: High—SQL injection is one of the most commonly exploited vulnerabilities in web applications.
- Impact: Exploiting this vulnerability could lead to unauthorized access to sensitive data, affecting all users and potentially the business's reputation.
- Remediation: Use parameterized queries or ORM frameworks to mitigate SQL injection risks.

Cross-Site Scripting (XSS)

Risk Level: **HIGH**

Vulnerability detail

- Summary: XSS vulnerabilities were identified in various input fields, allowing attackers to inject and execute scripts in users' browsers.
- Probability of Exploit: High—XSS attacks are prevalent and can easily be automated.
- Impact: This vulnerability can compromise user sessions, leading to data theft and loss of user trust.
- Remediation: Implement output encoding and proper input validation to prevent XSS.

Insecure Direct Object References

Risk Level: MEDIUM

Vulnerability detail

- Summary: This vulnerability was identified by manipulating URL parameters to access unauthorized resources.
- Probability of Exploit: Medium—While this requires some knowledge, it is a straightforward attack vector.
- Impact: Unauthorized access to sensitive information could occur, leading to data leakage.
- Remediation: Implement proper access controls and validate user permissions for resource access.

Server-Side Request Forgery (SSRF)

Risk Level: MEDIUM

Vulnerability detail

- Summary: The application was found to allow users to specify URLs for image uploads, which could be manipulated to perform SSRF attacks against internal services.
- Probability of Exploit: Medium—If internal services are accessible, this can be exploited with crafted requests.
- Impact: Exploiting this vulnerability could lead to exposure of internal services and sensitive information.
- Remediation: Restrict outbound requests to known safe hosts and validate user input for URLs.

Cross-Site Request Forgery (CSRF)

Risk Level: MEDIUM

Vulnerability detail

- The Juice Shop application lacks CSRF tokens, allowing attackers to trick authenticated users into making unwanted requests.
- Probability of Exploit: Medium—While the attack requires user interaction, it is a common vulnerability in web applications.
- Impact: This can lead to unauthorized actions being performed on behalf of users without their consent.
- Remediation: Implement CSRF tokens for state-changing requests and validate them on the server side.

Information Disclosure

Risk Level: LOW

Vulnerability detail

- Summary: The application discloses version information and stack traces that can aid attackers in understanding the underlying technologies.
- Probability of Exploit: Low—This typically aids in reconnaissance but does not lead directly to exploitation.
- Impact: Attackers can gather information to formulate more targeted attacks, increasing the overall risk profile.
- Remediation: Hide detailed error messages and remove version information from responses.

Methodology

The methodology for the OWASP Juice Shop challenge is structured for technical practitioners, including security analysts and engineers. It follows a chronological approach to assess vulnerabilities effectively, starting from tool selection through to execution, analysis, and validation.

Assessment Toolset Selection

The following tools were utilized in the assessment:

1. Burp Suite: For intercepting and analyzing web traffic.
2. OWASP ZAP: To perform automated vulnerability scans.
3. Nikto: For scanning the application for security issues.
4. Nmap: For network and service enumeration.
5. Metasploit: To validate and exploit identified vulnerabilities.

Assessment Methodology Detail

1. Tool Execution:
   - Burp Suite: Intercepted HTTP requests to identify vulnerabilities like XSS. An example of a modified request is:

     GET /product?id=1&<script>alert('XSS')</script> HTTP/1.1

     Host: juice-shop.com
   - OWASP ZAP: Automated scanning was performed to detect vulnerabilities in the application.

2. Evidence of Tool Execution: Screenshots and command-line outputs were taken during the assessment. A typical OWASP ZAP scan result might be represented as:
3. Manual Validation:

   Significant vulnerabilities identified were manually tested. For example, XSS was validated by injecting a script into a user input field.

   The validation process included:

   Screenshots of successful attempts.

   Commands documented for the Metasploit framework:

   Use exploit/multi/http/xxx

   Set RHOST target_ip

   Run

4. Analysis of Tool Output: Each tool's output was analyzed to identify vulnerabilities. Significant findings were manually validated, with documentation of the process including screenshots and outputs to support the findings.

   Significant vulnerabilities identified were manually tested. For example, XSS was validated by injecting a script into a user input field.

This concluded the vulnerability assessment methodology portion of this report.