Team: https://github.com/hatemharby98/DEPI_R3_ONL3_ISS3_S1_FINAL_PROJECT

Penetration Testing Project Documentation

1. Overview

- **Project Name:**
    - Educational Penetration Test of OWASP Juice Shop.
- **Description:**
    - **Project Summary:** An analysis and security assessment of the "OWASP Juice Shop" web application, which is intentionally designed to be insecure.
    - **Goal:** To identify and exploit as many vulnerabilities as possible listed on the application's "Scoreboard" and to understand their exploitation methods.
    - **Problem Solved:** This project does not solve a problem in a real-world product. Instead, it serves as a training tool to enhance the skills of penetration testers in identifying common vulnerabilities (like the OWASP Top 10) in a safe and legal environment.
- **Main Objectives:**
    - Practical application of web application penetration testing methodologies.
    - Discover and exploit vulnerabilities such as SQL Injection, XSS, Broken Access Control, etc.
    - Understand the application's business logic and how it can be abused.
    - Practice writing professional reports on testing findings.
- **Scope of Testing:**
    - The OWASP Juice Shop web application (assuming it's running on a local server or a dedicated hosted instance for testing, e.g., http://juice-shop.local).
- **Tools and Technologies:**
    - Burp Suite (Community/Pro)

- Nmap
- sqlmap
- Browser Developer Tools
- (Optional: Metasploit, Wireshark)

2. Testing Methodology

- **Methodology Type:**
  - The methodology will be primarily based on the **OWASP Testing Guide (OTGv4)** and the **OWASP Top 10** risks, as the application is designed around these principles.

3. Requirements

- **a. Functional Requirements:**
  - **Reconnaissance:** Identify the technologies used by the application (e.g., Angular, Node.js) and understand the application's structure and pages.
  - **Vulnerability Scanning:** Use tools like Burp Suite Scanner (if available) and Nmap to scan for open ports and services.
  - **Exploitation:** Manually exploit vulnerabilities and use semi-automated tools (like sqlmap) to access sensitive data or control other user accounts.
  - **Reporting:** Document all discovered vulnerabilities with clear "Steps to Reproduce" and classify their severity.

4. Test Plan and Design

- **Scope Definition:**
  - **In-Scope:** All functions, pages, and APIs belonging to the Juice Shop application itself.
  - **Out-of-Scope:** The host server OS, network infrastructure, any other applications hosted on the same server, and Denial of Service (DoS/DDoS) attacks.
- **Rules of Engagement (RoE):**
  - As this is a training application, testing can be conducted at any time.
  - Testing is performed on an isolated (sandboxed) instance to ensure no impact on other services.

- **Attack Vectors:**
    - **Web Application:**
        - SQL Injection (e.g., on the login page).
        - Cross-Site Scripting (XSS) (e.g., in the search bar or product reviews).
        - Broken Access Control / IDOR (e.g., attempting to access another user's shopping basket).
        - Sensitive Data Exposure (e.g., searching for files containing passwords or data).
    - **Network:**
        - Port Scanning (using Nmap) to discover services running on the server.

5. Implementation Details

- **Tool Configuration:**
    - Configure Burp Suite as a proxy to intercept and analyze all requests/responses between the browser and the Juice Shop server.
- **Custom Scripts/Modules:**
    - (Hypothetical) A simple Python script could be used for directory brute-forcing to find hidden files or folders not present in common wordlists.

6. Execution and Findings

- **Testing Phases:**
    - The documented phases were followed: Recon, Scanning, Gaining Access, Maintaining Access (by gaining admin privileges). (Covering Tracks is not applicable in this educational scenario).
- **Tools Used:**
    - **Burp Suite:** To intercept and modify requests (e.g., modifying a user ID to access another user's basket).
    - **Nmap:** To discover the web service (HTTP) running on port 3000 (the Juice Shop default).
    - **sqlmap:** To confirm and exploit an SQLi vulnerability in the search field.

- **Key Findings (Common examples from Juice Shop):**
    - **Critical:** SQL Injection in the login API, allowing login as any user (including the admin) using a payload like: ' OR 1=1 --.
    - **High:** Broken Access Control (IDOR) allowing any user to view another user's shopping basket by manipulating the Basket ID in the API request.
    - **Medium:** Stored XSS in the product review page, allowing the injection of JavaScript code that executes in the browser of any user visiting that product page.
    - **Low:** Sensitive Data Exposure revealing the framework version (Node.js/Angular) in HTTP headers, which could aid an attacker in finding known exploits.

7. Reporting and Delivery

- **Report Format:**
    - A PDF document containing an Executive Summary for management and a Technical Details section for developers, including each vulnerability and its exploitation steps.
- **Delivery Instructions:**
    - The report will be delivered to the academic supervisor/course instructor via email.
- **Remediation Recommendations:**
    - **For SQLi:** Use Parameterized Queries (or Prepared Statements) to prevent user input from being interpreted as part of an SQL command.
    - **For IDOR/BAC:** Implement mandatory server-side checks to verify that the requesting user has the proper authorization to access the requested resource (e.g., their *own* basket).
    - **For XSS:** Implement context-aware output encoding for all user-supplied data before rendering it on a page.

Contributors

- **Team Members:**
    - (Your Name) – Pentester / Report Writer
- **Supervisor:**

- (Supervisor's/Instructor's Name) – Project Supervisor

References

- OWASP Top 10 - 2021
- OWASP Testing Guide (OTGv4)
- OWASP Juice Shop Official GitHub (For challenge validation)