SecureCorp Cybersecurity Report

Project Title: Vulnerability Assessment and Penetration Testing of SecureCorp's Web Applications

Prepared by: Saurav Dhapola

Date: 20-10-2024

Executive Summary

This report details the results of a comprehensive vulnerability assessment and penetration test conducted on SecureCorp's web application using the Damn Vulnerable Web Application (DVWA). The goal was to identify weaknesses in the system, exploit them, and propose actionable mitigation strategies. The following vulnerabilities were found and exploited:

- SQL Injection
- Cross-Site Scripting (XSS)
- File Upload Exploits
- Brute Force Attacks
- Command Injection

This report includes detailed findings, attack simulations, and remediation steps for each vulnerability.

---

1. Reconnaissance and Information Gathering

Objective: Gather information on the target system to build a profile of SecureCorp's infrastructure.

- **Tools Used:** Nmap, Whois, DNS Recon, Burp Suite
- **Findings:**
    - **Open Ports:** 80 (HTTP), 443 (HTTPS)
    - **Services Running:** Apache Web Server, MySQL
    - **Web Application Frameworks Identified:** PHP 7.4.3
    - **Server Information:** Linux (Ubuntu 20.04)

---

2. Vulnerability Assessment

A detailed assessment was performed to identify potential vulnerabilities in SecureCorp's web applications. The following vulnerabilities were found:

2.1 SQL Injection (SQLi)

Description: SQL Injection occurs when unsanitized user inputs are directly used in SQL queries, allowing attackers to manipulate the query and retrieve sensitive data.

- **Exploited in:** Login forms, search fields.
- **Tool Used:** SQLMap, manual injection testing.
- **Attack Simulation:**
    - Bypassed login authentication using `' OR '1'='1`.
    - Retrieved admin user credentials from the database.

Recommendation:

- Use prepared statements (parameterized queries) to prevent SQL injection.
- Sanitize user inputs and apply input validation.

2.2 Cross-Site Scripting (XSS)

Description: XSS allows attackers to inject malicious scripts into web pages viewed by other users.

- **Exploited in:** Comment forms and search bars.
- **Tool Used:** Burp Suite, manual testing.
- **Attack Simulation:**
    - Injected `<script>alert('XSS');</script>` into input fields.
    - The script executed on page reload, demonstrating a successful attack.

Recommendation:

- Implement proper input validation and output encoding.
- Apply Content Security Policy (CSP) headers to restrict script execution.

2.3 File Upload Exploits

Description: Unrestricted file uploads allow attackers to upload malicious files, potentially gaining access to the server.

- **Exploited in:** File upload functionality.
- **Tool Used:** Burp Suite.
- **Attack Simulation:**

  - Uploaded a malicious PHP script disguised as an image.
  - Accessed the uploaded script to gain remote shell access.

Recommendation:

- Restrict file types that can be uploaded (e.g., images only).
- Validate file extensions and MIME types server-side.
- Use antivirus scanning on uploaded files.

2.4 Command Injection

Description: Command injection occurs when user input is used in system commands without proper sanitization.

- **Exploited in:** Ping command feature in DVWA.
- **Tool Used:** Manual testing.
- **Attack Simulation:**
  - Injected commands such as `; ls` to list server directories.
  - Gained access to sensitive files.

Recommendation:

- Use proper input validation.
- Avoid executing system commands based on user input.

2.5 Brute Force Attacks

Description: Brute force attacks are used to guess passwords by repeatedly trying different combinations.

- **Exploited in:** Login page.
- **Tool Used:** Hydra.
- **Attack Simulation:**
  - Conducted a successful brute force attack on the admin login page using a dictionary of common passwords.

Recommendation:

- Enforce strong password policies.
- Implement account lockout mechanisms after a certain number of failed attempts.
- Enable two-factor authentication.

---

3. Client-Side Attacks

Objective: Identify vulnerabilities that can be exploited by manipulating client-side code.

Findings:

- **XSS Vulnerabilities:** As described above, client-side script injection was possible.
- **Cookie Manipulation:** SecureCorp's cookies were not flagged as secure or HTTPOnly, making them vulnerable to session hijacking.

Recommendation:

- Set `HttpOnly` and `Secure` flags for cookies.
- Validate and sanitize all user inputs on the client and server sides.

---

4. Social Engineering Awareness

Objective: Simulate phishing or social engineering attacks targeting SecureCorp employees.

Scenario:

- A phishing email was simulated, requesting employees to reset their passwords via a fake login page.
- Several employees fell for the attack, entering credentials into the fake page.

Recommendation:

- Conduct regular social engineering awareness training.
- Implement email filtering and alert employees to common phishing techniques.

---

5. Wi-Fi Network Fortification

Objective: Suggest ways to secure SecureCorp's Wi-Fi network from attacks.

Recommendations:

- **Upgrade to WPA3:** Use WPA3 encryption to enhance Wi-Fi security.
- **Segment Networks:** Separate guest and internal networks to limit access.
- **Strong Passwords:** Use complex Wi-Fi passwords and rotate them regularly.
- **Firewall Configuration:** Ensure proper firewall rules are in place to limit unauthorized access.

---

6. Conclusion

This assessment identified several critical vulnerabilities within SecureCorp's web applications, which, if exploited, could lead to significant data breaches or system compromises. By following the outlined remediation steps, SecureCorp can improve its security posture and mitigate the risks posed by these vulnerabilities.