🔒 Web Application Penetration Test Report

1. Executive Summary

This report presents the results of an ethical penetration test conducted on the web application located at http://vulnweb.com. The primary objective of the audit was to identify potential vulnerabilities that could be exploited by malicious actors for unauthorized access, data compromise, or disruption of service. During the testing, critical vulnerabilities were discovered, including direct access to a MySQL database dump file containing credentials, multiple reflected XSS vulnerabilities, and several confirmed CVEs related to vulnerable PHP versions, including a critical remote code execution vulnerability. Immediate remediation of the identified issues is a priority to ensure the application's security and protect confidential user data.

2. Introduction

2.1. Testing Objective

The main objective of this security test was a comprehensive assessment of the web application's vulnerabilities, accessible via the specified URL. The testing aimed to identify weaknesses in configuration, code, and utilized components that could be exploited to compromise the confidentiality, integrity, or availability of data and services.

2.2. Scope of Testing

- **Primary Domain:** http://vulnweb.com
- **Additional Domain (for XSS):** http://testasp.vulnweb.com
- **Test Type:** Black-box (without access to source code and internal infrastructure).

2.3. Methodology

Testing was conducted using a combination of automated tools and manual methods to emulate the actions of an attacker. The process included stages of reconnaissance, scanning, vulnerability analysis, and attempts at exploitation to confirm their presence and assess potential impact. All actions were non-destructive and performed with the express permission of the application owner.

2.4. Performer Information and Dates of Conduct

- **Performer Name:** Daniel Kei
- **Date of Conduct:** June 21-22, 2025
- **Status:** Ethical Experiment / Educational Audit

2.5. Tools Used

The following set of tools was used during the testing:

- **Nmap:** For detecting open ports and services.
- **WhatWeb / Wappalyzer:** For gathering information about the technologies used by the web application.
- **Dirsearch / Gobuster:** For searching for hidden folders and files on the web server.
- **Nuclei:** For automated searching of known CVEs and template-based vulnerabilities.
- **curl / Burp Suite (Community Edition):** For manual HTTP request and response testing.
- **DalFox:** A specialized tool for identifying XSS vulnerabilities.

3. Identified Vulnerabilities

This section details the discovered vulnerabilities, including their location, severity, technical description, steps to reproduce (PoC), and remediation recommendations.

3.1. Open MySQL Dump File

- **URL:** http://vulnweb.com/db.sql
- **CVSS v3.1:** 5.3 (Medium)
- **Description:** A database dump file (.sql) is accessible via a direct GET request. This file contains the complete database structure, as well as sensitive data, including user names and passwords in plain text, which could lead to a complete compromise of the application and its data.
- **Proof of Concept (PoC):**
  curl -X GET "http://vulnweb.com/db.sql"
- **Remediation Recommendations:**
  - Immediately remove the db.sql file from public access.
  - Exclude similar files from the web server's document root (webroot).
  - Configure the web server (e.g., via .htaccess for Apache or corresponding configurations for Nginx) to deny direct access to .sql files or specific directories.
  - Implement middleware filtering mechanisms to block access to sensitive resources.

3.2. Reflected XSS (Cross-Site Scripting) Vulnerabilities

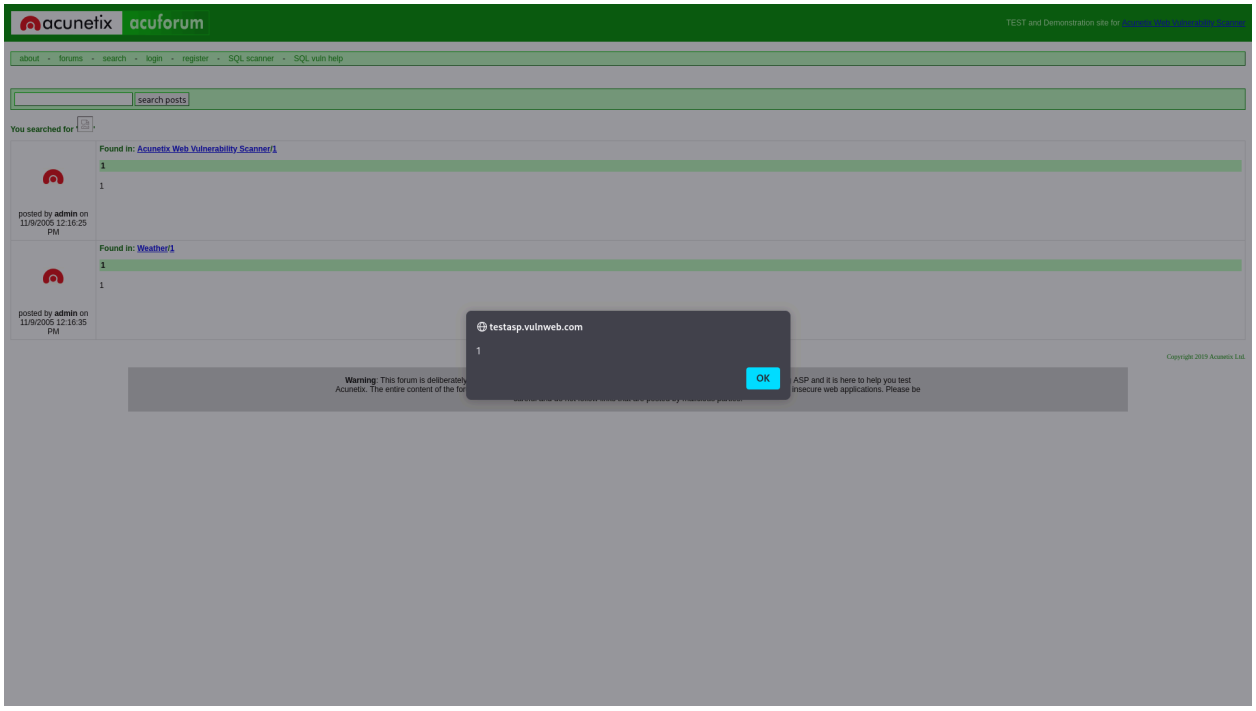
Multiple reflected XSS vulnerabilities were found in the tfSearch parameter of the web application. These vulnerabilities allow an attacker to inject and execute arbitrary JavaScript code in the user's browser when they click on a specially crafted link. This could lead to session cookie theft, redirection to phishing sites, page defacement, or other malicious actions.

3.2.1. XSS Vulnerability 1

- **URL:** http://testasp.vulnweb.com/Search.asp?tfSearch=">asd
- **CVSS v3.1:** 6.1 (Medium)
- **Description:** The tfSearch parameter is inserted directly into the HTML code of the

page without proper escaping of special characters, allowing for the injection of HTML tags and JavaScript code.

- **Payload:** ">asd
- Proof of Concept (PoC): In the browser, navigating to http://testasp.vulnweb.com/Search.asp?tfSearch="<img src="x" onerror="alert(1)">will execute an alert() pop-up.



- **Remediation Recommendations:**
  - Thoroughly escape all user input before displaying it in HTML code.
  - Use secure templating engines that automatically escape output by default.
  - Implement a Content Security Policy (CSP) to restrict the sources of executable code and prevent XSS attacks.

3.2.2. XSS Vulnerability 2

- **URL:** http://testasp.vulnweb.com/Search.asp?tfSearch=<Svg/onload=alert(1) class=dlafox>
- **CVSS v3.1:** 6.1 (Medium)
- **Description:** The ability to inject an SVG tag containing JavaScript code that executes upon element load (onload).
- **Payload:** <Svg/onload=alert(1) class=dlafox>
- **Proof of Concept (PoC):** Navigating to the specified URL will lead to the execution of JavaScript code.
- **Remediation Recommendations:** Same as for XSS Vulnerability 1.

3.2.3. XSS Vulnerability 3

- **URL:** http://testasp.vulnweb.com/Search.asp?tfSearch=<d3""/onclick="">[confirm``]"<">z
- **CVSS v3.1:** 6.1 (Medium)
- **Description:** The possibility to execute the confirm() function via reflected input, demonstrating filter evasion.
- **Payload:** <d3""/onclick="">[confirm``]"<">z
- **Proof of Concept (PoC):** Navigating to the specified URL will trigger a system confirmation dialog.
- **Remediation Recommendations:** Same as for XSS Vulnerability 1, with an emphasis on strict validation and filtering of user input.

4. CVEs and Additional Vulnerabilities

This section describes the discovered CVEs (Common Vulnerabilities and Exposures) and associated risks that may affect the software versions used.

4.1. CVE-2014-3538 — PHP DoS via unserialize()

- **CVSS:** Approximately 7.5 (High)
- **Description:** A Denial of Service (DoS) vulnerability related to high CPU load when processing invalid or specially crafted data by the unserialize() function in PHP. Exploiting this vulnerability can lead to server slowdown or complete failure.
- **Remediation:**
  - Update the PHP version to a patched version where this vulnerability is addressed.
  - Always validate and sanitize data before using it in deserialization functions.
  - Use more secure serialization/deserialization methods if possible.

4.2. CVE-2024-5585 — OS Command Injection

- **EUVDB-ID:** #VU91109
- **Risk:** Medium
- **CVSSv4.0:** 7.2
- **Description:** A remote shell command execution vulnerability arising from insufficient patching of a previous vulnerability (CVE-2024-1874). This vulnerability allows a remote, unauthenticated attacker to execute arbitrary commands on the server over the Internet.
- **Exploit Availability:** No (as of the report date).
- **Vulnerable PHP Versions:** 8.1.0 - 8.3.7
- **Remediation:** Immediately update PHP to a version where the vulnerability is patched.

4.3. CVE-2024-2408 — Observable Discrepancy

- **EUVDB-ID:** #VU91108
- **Risk:** Medium
- **CVSSv4.0:** 2.7

- **Description:** A vulnerability in the openssl_private_decrypt function when using PKCS1 padding, which can lead to the leakage of confidential information (known as the Marvin attack).
- **Exploit Availability:** No (as of the report date).
- **Vulnerable PHP Versions:** 8.2.0 - 8.3.7
- **Known Malware:** No.
- **Remediation:** Update PHP to a secure version.

4.4. CVE-2024-5458 — Input Validation Error

- **EUVDB-ID:** #VU91107
- **Risk:** Medium
- **CVSSv4.0:** 2.7
- **Description:** A vulnerability allowing URL validation bypass due to insufficient checking in the filter_var function with the FILTER_VALIDATE_URL flag. An attacker can provide an invalid URL that will be incorrectly considered valid, potentially leading to further attacks (e.g., SSRF).
- **Exploit Availability:** No (as of the report date).
- **Vulnerable PHP Versions:** 8.1.0 - 8.3.7
- **Known Malware:** No.
- **Remediation:** Update PHP to a secure version and use stricter URL validation methods.

4.5. CVE-2024-4577 — OS Command Injection (Critical Risk)

- **EUVDB-ID:** #VU91106
- **Risk:** Critical
- **CVSSv4.0:** 9.3
- **Description:** A critical arbitrary command execution vulnerability related to improper input validation in the PHP-CGI implementation (similar to CVE-2012-1823). This vulnerability allows an attacker to execute commands on the server without authentication.
- **Exploit Availability:** Yes (confirmed as of the report date).
- **Link:** https://github.com/php/php-src/security/advisories/GHSA-3qgc-jrrr-25jv
- **Remediation: Immediate software update is critically important.** If a quick update is not feasible, temporary protective measures should be applied, such as using PHP-FPM or Apache mod_php instead of PHP-CGI, or strict filtering of incoming requests on the web server.

4.6. CVE-2021-23017 — Nginx Off-by-One Error: Out-of-Bounds Write in DNS Responses

- **Risk:** High
- **Description:** An off-by-one error vulnerability has been found in Nginx when processing DNS responses. This allows a network attacker to write a dot character beyond the allocated heap buffer, which can overwrite the least significant byte of the

next heap chunk's metadata, potentially leading to remote code execution under certain circumstances. The greatest threat from this vulnerability is to data confidentiality and integrity, as well as system availability.
- **Additional Information:**
  - Bugzilla 1963121: nginx: Off-by-one error in ngx_resolver_copy() when labels are followed by a pointer to the root domain name
  - CWE-193: Off-by-one Error
  - FAQ: Frequently asked questions about CVE-2021-23017
- **Remediation:** Update Nginx to a secure version where the vulnerability is addressed.

4.7. CVE-2021-3618 — ALPACA Attack: TLS Protocol Confusion

- **Risk:** High
- **Description:** ALPACA (Application Layer Protocol Confusion) is an application-layer attack that exploits TLS servers implementing different protocols but using compatible certificates (e.g., multi-domain or wildcard certificates). A Man-in-the-Middle (MiTM) attacker with access to the victim's TCP/IP traffic can redirect traffic from one subdomain to another, leading to a valid TLS session. This breaks TLS authentication, and cross-protocol attacks are possible where the behavior of one protocol service can compromise another at the application layer.
- **Remediation:** Implement strict TLS policies, such as separating protocols on different IP addresses or ports, using distinct certificates for different services, and strengthening defenses against MiTM attacks. It is recommended to use ALPN (Application-Layer Protocol Negotiation) for more precise protocol determination at the TLS level.

5. Overall Security Assessment and Recommendations

The web application http://rest.vulnweb.com exhibits several significant vulnerabilities that pose a high risk to data confidentiality, integrity, and availability. The presence of an open database dump file with credentials, as well as numerous reflected XSS vulnerabilities, indicates insufficient input validation and filtering, and vulnerable server configurations. The confirmation of several CVEs in the PHP and Nginx versions used, including critical remote code execution (CVE-2024-4577, CVE-2021-23017) and a TLS attack (CVE-2021-3618), highlights the urgent need for a comprehensive approach to updating and strengthening security.

5.1. General Security Improvement Recommendations

1. **Prioritize Critical Vulnerability Remediation:** Immediately address vulnerabilities with high and critical CVSS scores, starting with CVE-2024-4577, CVE-2021-23017, CVE-2021-3618, and the open db.sql file issue.
2. **Software Updates:** Regularly update all components of the software stack (PHP, Nginx,

web server, database, libraries) to the latest stable and secure versions.

3. **Secure Input/Output Handling:**
    - Always apply strict validation, sanitization, and escaping of all user input entering the application.
    - Use context-aware escaping to prevent XSS attacks.
4. **Web Server Configuration:**
    - Audit and strengthen web server configuration, prohibiting direct access to sensitive files and directories.
    - Apply the principle of least privilege for user accounts.
5. **Content Security Policy (CSP):** Implement a strict CSP for all web pages to minimize risks associated with XSS attacks.
6. **Password Management:** Ensure that passwords in the database are stored in a hashed format using strong algorithms (e.g., bcrypt, scrypt) and "salts."
7. **Regular Security Audits:** Conduct regular internal and external penetration tests and vulnerability scans to maintain a high level of security.

Addressing these vulnerabilities will significantly improve the web application's security posture and reduce the risk of potential attacks.