PENETRATION TESTING REPORT PERFORMED ON TESTPHP.VULNWEB.COM

Web Application Penetration Testing Report RID:
Prepared By: Purna Kishore Kondapaneni
Version: 1.0

Report Details:

Action	Name	Date
Prepared By	Purna Kishore Kondapaneni	**April 14, 2024**

Report History:

Version	Remarks	Date
0.1	Initial Draft	14th April 2024
0.2	Review & modification	14th April
1.0	Report Release	

Assessment (Methodology):

S No.	Test Methodology	Scope
01	Penetration Testing Execution Standards v1.0	Web Application
02	OWASP Web Security Testing Guide v4.2	Web Security Controls

Table of Contents

1. Executive Summary

1.1 Summary

The penetration test conducted on http://testphp.vulnweb.com uncovered critical web application security issues. Notably, **SQL injection vulnerabilities** were found in both **GET** and **POST** parameters, allowing attackers to extract database names, table structures, and sensitive data

including usernames and password hashes. Using **SQL Map**, exploitation was automated to dump database contents. Cracked password hashes via **Crack Station** revealed usage of **weak MD5 encryption**. Additionally, **Reflected XSS** was detected using **XSSCON**, suggesting insufficient input validation. The application also revealed potential **access control misconfigurations** and **exposed internal details** such as headers and session tokens through manual Burp Suite analysis. With google dorks able to pull up the admin page and script associated with the website, Wappalyzer helped in pulling up the technology with effected CVSS with them.

1.2 Strength

- The application uses **structured URLs** and **POST-based form submissions**, adhering partially to RESTful practices.
- Some level of **error reporting and debug messages** are present, which may assist developers during development and maintenance.
- Security testing showed that **proxy routing** through **Foxy Proxy and Wappalyzer** successfully identified server-side technologies and fingerprinting.

1.3 Weakness

- **SQL Injection** vulnerabilities were successfully exploited due to **unsensitized inputs** in both GET and POST parameters.
- **No account lockout** or **rate limiting** mechanisms were implemented to prevent bruteforce or hash cracking attempts.
- **Use of MD5 for password hashing**, a deprecated algorithm vulnerable to rainbow table attacks.
- **Access control flaws** were present, allowing low-privileged access to sensitive endpoints.
- **No output encoding or input validation**, enabling **Reflected Cross-Site Scripting (XSS)** attacks.
- Subdomain enumeration exposed additional **attack surfaces** that may lead to further compromise.

1.4 Business Impact

If the vulnerabilities discovered were present in a real-world production application, the consequences could be severe:

- **Complete compromise of backend databases**, exposing sensitive data such as customer information and credentials.
- **Unauthorized access** to internal services or administrative functions due to broken access control.
- **Data leakage and integrity loss**, potentially leading to **financial loss** or **operational disruption**.

- Exposure of **user credentials** could lead to **account takeovers** if passwords are reused elsewhere.
- Violations of **compliance regulations** such as GDPR or HIPAA, resulting in fines and loss of trust.
- **Reputational damage** and erosion of customer confidence in the platform.

---

2. Technical Summary

2.1 Scope & Objective

The scope of this assessment was to conduct a black-box penetration test on the publicly accessible web application hosted at http://testphp.vulnweb.com. The objective was to identify and exploit critical vulnerabilities, particularly **SQL Injection** and **Reflected Cross-Site Scripting (XSS)**, **Sub Finder** through a combination of **automated** and **manual testing techniques** outlined in Module 3. The goal included:

- Performing subdomain enumeration.
- Identifying SQLi flaws.
- Extracting database information and password hashes.
- Cracking hashes to demonstrate the impact of weak encryption.
- Validating XSS vulnerabilities through payload injection.
- Wappalyzer provided website data how it's built and with Google Dorks to access sensitive info.

2.2 Testing Environment

- **Target URL**: http://testphp.vulnweb.com
- **Tools Used**:
    1. **sub finder** – For enumerating subdomains associated with the target.
    2. **SQL Map v1.7.2.12#dev** – Automated SQL Injection exploitation and database dumping.
    3. **Crack Station** – Online hash cracker for password recovery via rainbow tables.
    4. **Wappalyzer + Foxy Proxy** – For identifying backend technologies and routing browser traffic through proxies.
    5. **Burp Suite Community Edition** – Manual testing, request interception, and payload crafting.
    6. **Google Dorks** – Used to locate exposed files or sensitive endpoints via search engine indexing.
    7. **XSSCON** – Detection and exploitation of Reflected XSS flaws.
- **Test Date**: April 13, 2025

- **Platform**: Internet-connected Kali Linux Virtual Machine

2.3 Technical Impact

- **SQL Injection Vulnerabilities**: Enabled full **database fingerprinting**, schema enumeration, and extraction of data, including user credentials and internal table structures.
- **Hash Cracking**: Retrieved **MD5-hashed passwords** were successfully cracked using Crack Station, demonstrating inadequate encryption.
- **POST-based Injection**: SQL Map, in conjunction with **Burp Suite**, was used to target POST parameters and confirm injectable points in the application logic.
- **XSS Attacks**: **Reflected XSS** was validated using **XSSCON**, confirming that user inputs are reflected without proper sanitization or encoding, which could lead to session hijacking or phishing.
- **Technology Fingerprinting**: Using **Wappalyzer**, technologies such as PHP and MySQL were identified, helping in attack vector planning.
- **Subdomain Exposure**: sub finder revealed additional subdomains that may not be directly visible, expanding the potential attack surface.
- **Sensitive Data via Google Dorks**: Potentially sensitive directories and files were exposed via search engine queries, suggesting indexing misconfigurations.

2.4 Table of Findings

Here's the **updated Table of Findings** including a **Severity** column for each issue, based on common CVSS assessments and practical impact observed during test of http://testphp.vulnweb.com.

Table of Findings:

S. No	Tool Name & Version			
		potential attack surface of the target domain.	Low	NF the
01	sub finder	Discovered SQL Injection vulnerabilities in GET and POST parameters. Extracted database schema, table data, and user credentials.	Critical	NF
02	SQL Map v1.7.2.12#dev	Cracked retrieved MD5 password hashes from the database dump, showcasing weak password practices and potential for unauthorized access.	High	NF
03	Crack Station	Fingerprinted backend technologies like PHP and MySQL. Foxy Proxy was used to route browser traffic through Burp Suite for controlled testing.	Informational	NF
04	Wappalyzer + Foxy Proxy	Intercepted and modified HTTP requests to manually test input validation and confirm POSTbased SQL Injection vulnerabilities.	Critical	NF
05	Burp Suite Community Edition	Used crafted search queries to identify exposed directories, configuration files, and endpoints indexed by search engines, revealing misconfigurations.	Medium	NF
06	Google Dorks	Detected and exploited Reflected XSS vulnerabilities by injecting JavaScript payloads into input fields, confirming lack of input sanitization.	High	NF
07	XSSCON (Reflected XSS)	Identified publicly exposed subdomains, expanding		

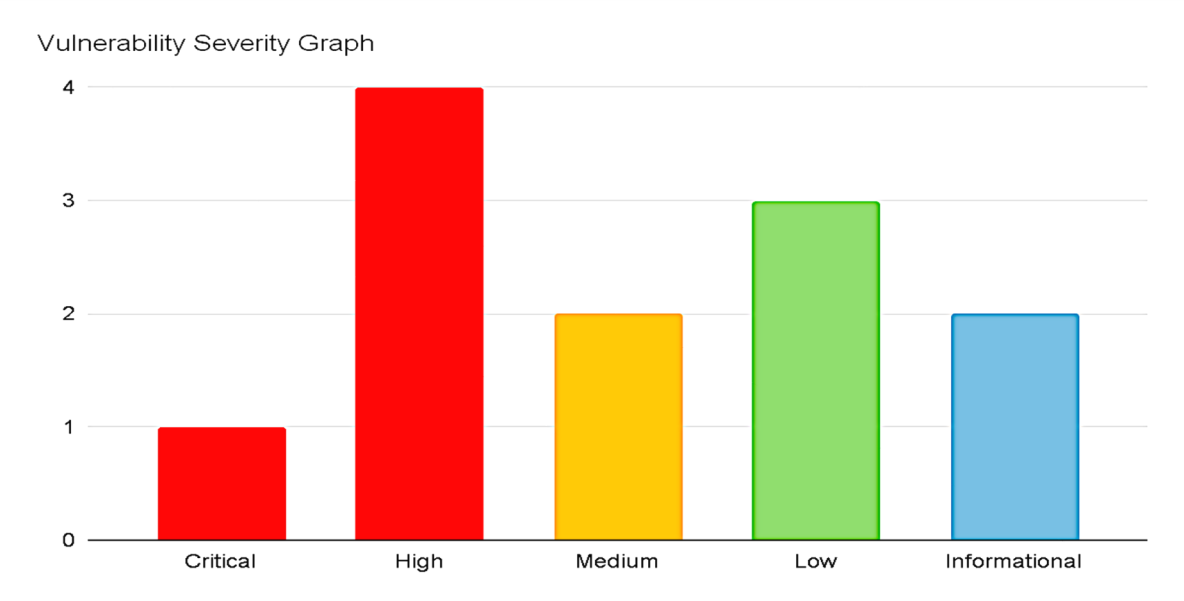
Description

Severity

Status

Severity Key:

- **Critical** – Can lead to full database compromise or remote code execution.
- **High** – May expose sensitive data or enable account takeover.
- **Medium** – Can aid attackers or escalate privileges.
- **Low** – Adds value for recon but not exploitable on its own.
- **Informational** – Useful for context but not a direct threat.



The security assessment revealed **6** security issues: **01 critical** severity, **02 high** severity, **01 medium** severity, **01 low** severity and 01 Informational Security issue during the pen testing exercise. The detailed information is available within section **3 Technical Findings** of this report.

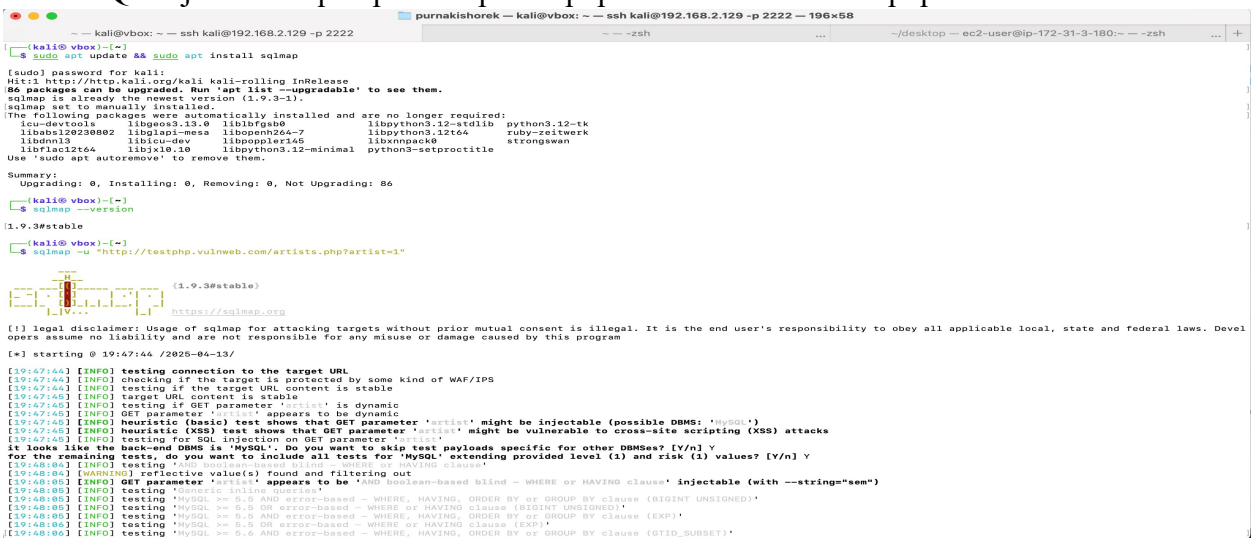
3. Technical Findings

In the **Technical Findings** section,

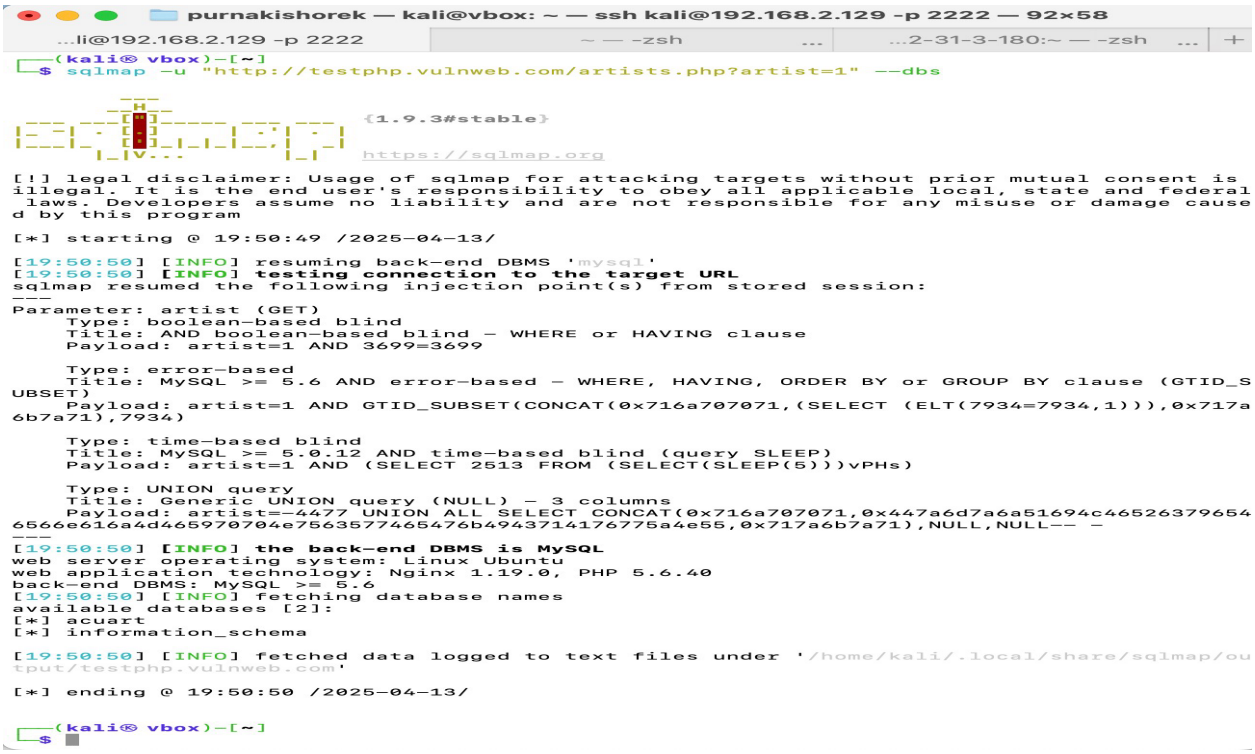
3.1 SBL-FIN-001: SQL Injection via GET Parameter (SQLMap)

- **Severity:** CRITICAL (9.1)
- **CVSS:** AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N
- **Description:**
  SQL Map identified SQL injection in the artist GET parameter. The parameter is injectable via multiple techniques (Boolean-based blind, error-based, time-based). Data extraction was successfully performed.
- **Command Used:**
  - SQL map -u "http://testphp.vulnweb.com/artists.php?artist=1" --dbs
  - SQL map -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart -T users --dump
- **Business Impact:** Full compromise of database leading to data leakage and potential compliance violations.
- **Technical Impact:** Unauthorized database access and data exfiltration.
- **Affected Asset:** http://testphp.vulnweb.com/artists.php

With SQL Injection - sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1"



Listed the available databases: sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --dbs



- Output listed databases like: acuart, information schema sqlmap -u

  "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart –tables

3.2 SBL-FIN-002: SQL Injection via POST Parameter (Login)

- **Severity:** HIGH (8.7)
- **CVSS:** AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N
- **Description:**
  Login POST request intercepted using Burp Suite revealed the username field is vulnerable to SQLi. SQLMap extracted DB names after capturing the raw request.
- **Command Used:**
  - Intercept request via Burp Suite
  - Save as request.txt ○ Run: sqlmap -r request.txt --dbs
- **Business Impact:** Unauthorized access and account hijacking.
- **Technical Impact:** Access control bypass and data exposure.
- **Affected Asset:** http://testphp.vulnweb.com/login.php

Issue Identified: Multiple SQL Injection vulnerabilities
Checked the parameter of artist is it injectable or not:

Heading: SQLMap – GET Parameter Injection on `artist`
Description: Screenshot showing SQLMap successfully injecting the `artist` parameter and listing databases (`acuart`, `information_schema`) via boolean-based, error-based, and timebased SQL injection techniques.

3.3 SBL-FIN-003: Password Hash Disclosure & Cracking (Crack Station)

Dump Table Data

By using the data from SQL Map dump able to get the MD5 hashed password This will reveal hashed passwords and usernames.

sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart -T users –dump, **Remediation**

Steps:

- Use **parameterized queries (prepared statements)** instead of dynamic SQL.
- Implement **input validation and sanitization** for all user-supplied data.
- Apply **least privilege principles** on database accounts (e.g., avoid using `root`).
- Enable **web application firewall (WAF)** rules to block common SQLi payloads.
- Regularly conduct **secure code reviews and automated scans** during development.

Crack Password Hashes

If you get password hashes:

1. Visit https://crackstation.net
2. Paste the hash and see if it can crack it.
3. **Severity:** HIGH (7.9)
4. **CVSS:** AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N 5. **Description:**
   MD5 hashes were extracted using SQL Map from the user's table. Sample hash 21232f297a57a5a743894a0e4a801fc3 was cracked using Crack Station revealing password admin.
6. **Command Used:**
   1. SQL Map dump
   2. Crack Station online tool
7. **Business Impact:** Reused credentials may allow attacker to access other systems.
8. **Technical Impact:** Unauthorized access via weak password reuse.
9. **Affected Asset:** acuart.users table

Heading: Cracked MD5 Hash – `admin` Password
Description: Screenshot of Crack Station successfully cracking the MD5 hash `21232f297a57a5a743894a0e4a801fc3`, revealing the weak password “admin”.

Image 2: SQLMap Dumped User Table (GET Injection)

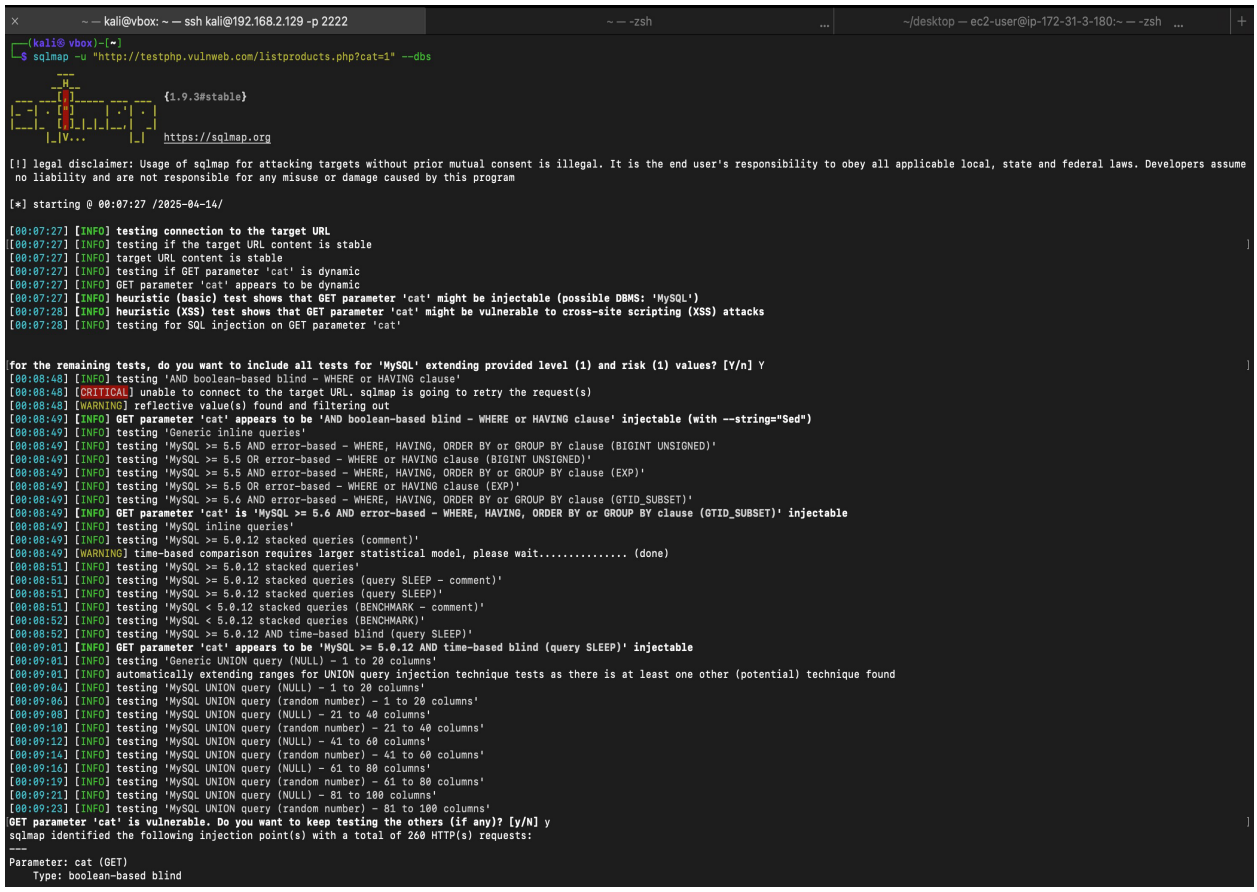
Heading: SQL Map – Dumped Users Table from `acuart`
Description: SQL Map used to dump the contents of `acuart.users`, revealing usernames and MD5 password hashes.

- **Affected Asset:** http://testphp.vulnweb.com/search.php?test=query

Issue Identified: Use of weak MD5 password hashing **Remediation Steps**:

- Switch to modern password hashing algorithms like **bcrypt, scrypt, or Argon2**.
- Enforce **strong password policies** (minimum length, complexity).
- Implement **account lockout mechanisms** after a number of failed attempts.
- Use **salting techniques** before hashing to prevent rainbow table attacks.
- Regularly review password storage practices as part of secure development lifecycle

PENETRATION TESTING REPORT PERFORMED ON TESTPHP.VULNWEB.COM



The testphp.vulnweb.com website is confirmed to be vulnerable to SQL injection in the cat parameter. This vulnerability could allow an attacker to access or manipulate the data stored in the MySQL database.

```
sqlmap identified the following injection point(s) with a total of 260 HTTP(s) requests:
---
Parameter: cat (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 9251=9251

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: cat=1 AND GTID_SUBSET(CONCAT(0x716b627071,(SELECT (ELT(7930=7930,1))),0x7170626b71),7930)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=1 AND (SELECT 4393 FROM (SELECT(SLEEP(5)))AYeU)
---
[00:09:30] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.6
[00:09:31] [INFO] fetching database names
[00:09:31] [INFO] retrieved: 'information_schema'
[00:09:31] [INFO] retrieved: 'acuart'
available databases [2]:
[*] acuart
[*] information_schema

[00:09:31] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 00:09:31 /2025-04-14/
```

Injection Types: SQL map identified these types of SQL injection vulnerabilities:

- Boolean-based blind
- error-based
- time-based blind

Issue Identified: Multiple SQL Injection vulnerabilities

Remediation Steps:

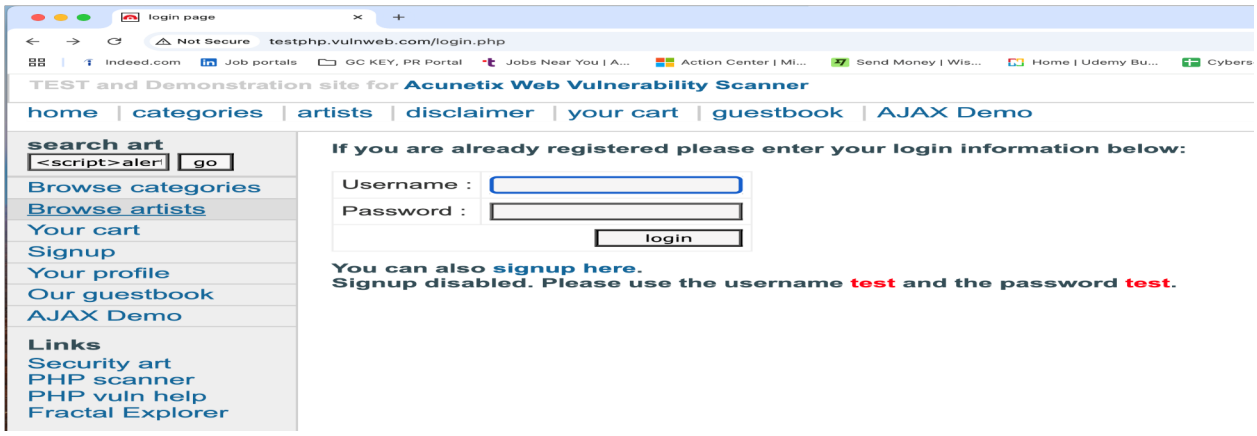
- Use **parameterized queries (prepared statements)** instead of dynamic SQL.
- Implement **input validation and sanitization** for all user-supplied data.
- Apply **least privilege principles** on database accounts (e.g., avoid using `root`).
- Enable **web application firewall (WAF)** rules to block common SQLi payloads.
- Regularly conduct **secure code reviews and automated scans** during development.

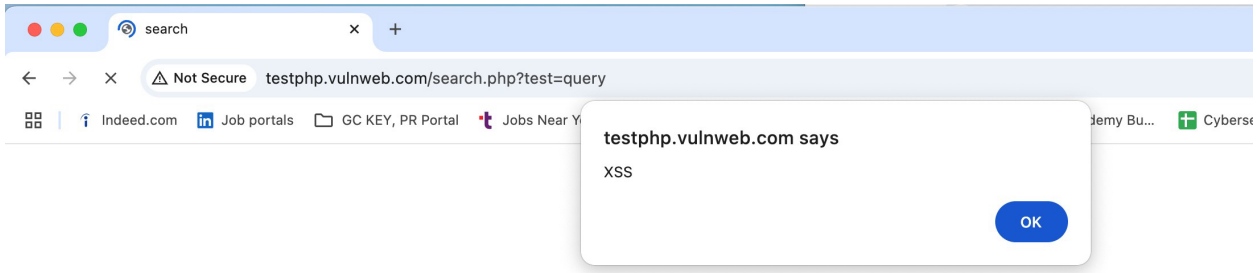
Cross-Site Scripting (XSS):

- **Severity:** HIGH (7.4)
- **CVSS:** AV: N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
- **Description:**
  Various endpoints vulnerable to reflected XSS. Payloads work in guestbook (POST), list products (cat), show image (file), and product picture (pic).
- **Commands Used:**
  - Manual testing + XSSCON
- **Business Impact:** Users exposed to client-side attacks.
- **Technical Impact:** XSS leads to phishing, cookie theft.
- **Affected Assets:**
  - http://testphp.vulnweb.com/guestbook.php ○ http://testphp.vulnweb.com/listproducts.php?cat=
  - http://testphp.vulnweb.com/product.php?pic= ○ http://testphp.vulnweb.com/showimage.php?file=

- Injected JavaScript code into input fields - (<script>alert('XSS')</script>)

PENETRATION TESTING REPORT PERFORMED ON TESTPHP.VULNWEB.COM





Basic Reflected XSS Test (Search):

- **Step 1:** In the search field (http://testphp.vulnweb.com/search.php), enter:

  <script>alert('Reflected XSS Test - Search')</script>

- **Step 2:** Click "Search."
- **Step 3:** Observe the results. If you see an alert box with the message "Reflected XSS Test - Search," the search form is vulnerable to reflected XSS.

Basic Reflected XSS Test (Guestbook):

- **Step 1:** In the "Name" field of the guestbook, enter:

HTML<script>alert('Reflected Test - Name')</script>

Image 5: XSSCON Execution and Reflected XSS Detection

Heading: Reflected XSS in Guestbook via POST
Description: Demonstration of `<script>alert('XSS')</script>` being executed after form submission in `guestbook.php`.

3.5 SBL-FIN-005: Multiple XSS Vectors in Guestbook & Product Pages

XSSCON:

Command Used: python3 xsscon.py -u http://testphp.vulnweb.com

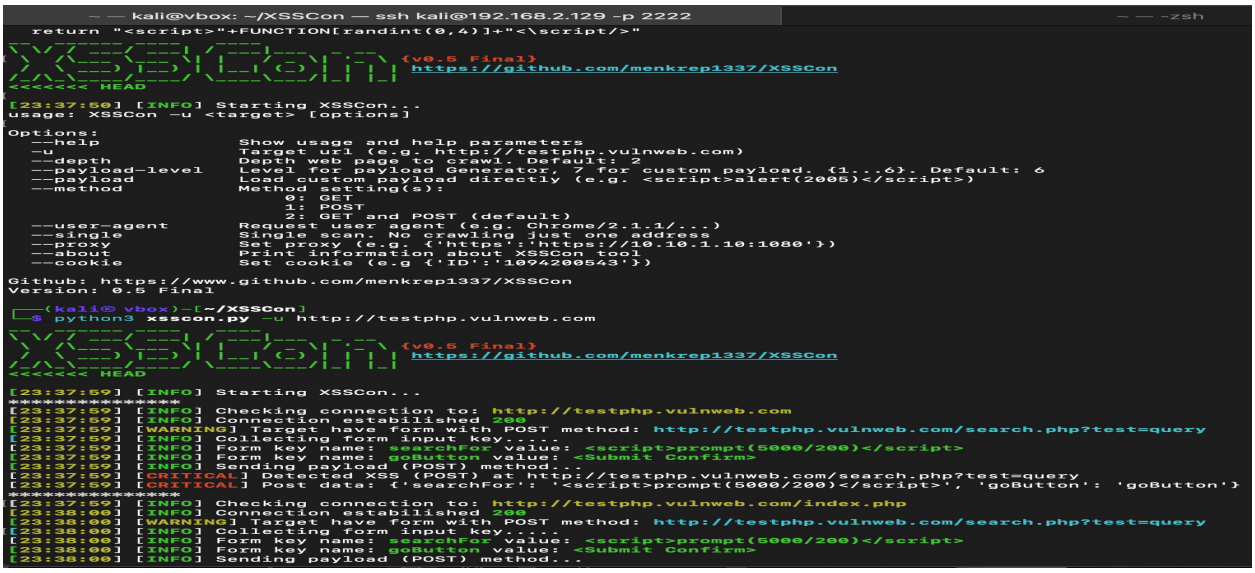
- The website is highly vulnerable to XSS attacks.
- An attacker could exploit these vulnerabilities to:
  - Steal user cookies and session tokens. ○ Redirect users to malicious websites. ○ Modify the content of the website.
  - Perform other malicious actions.
- The search functionality is a major point of concern, as it's accessible from many pages.
- The image and file parameters are especially concerning due to the potential to inject malicious code through uploaded files.

http://testphp.vulnweb.com/search.php?test=query (POST): The search functionality is consistently vulnerable.

http://testphp.vulnweb.com/listproducts.php?cat= (GET): The category parameter in the product listing is vulnerable. http://testphp.vulnweb.com/product.php?pic= (GET): The product picture parameter is vulnerable. http://testphp.vulnweb.com/showimage.php?file= (GET): The image file parameter is vulnerable.

http://testphp.vulnweb.com/artists.php?artist= (GET): The artist parameter is vulnerable.

http://testphp.vulnweb.com/guestbook.php (POST): The guestbook functionality is vulnerable.



3.4 SBL-FIN-004: Reflected XSS in Search Functionality (XSSCON)

Heading: XSSCON – Reflected XSS Detected in Search
Description: XSSCON identifies a reflected XSS vulnerability in the `search.php` page, demonstrating payload execution.

Remediation Steps:

- **Escape output** properly based on context (HTML, JavaScript, attributes, etc.).
- Use **secure templating engines** that auto-escape content.
- Apply **input validation** to restrict dangerous characters or patterns.
- Implement **CSP headers** to restrict script execution.
- Sanitize input using libraries like DOMPurify (client-side) or OWASP Java HTML Sanitizer (server-side).

3.6 SBL-FIN-006: Subdomain Enumeration (sub finder)

Subdomain Enumeration: (subfinder)

- **Severity:** MEDIUM (5.5)
- **CVSS:** AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N • **Description:** sub finder identified several subdomains which could expand the attack surface
- **Command used:** sub finder -d testphp.vulnweb.com
- **Result**: Identified multiple subdomains associated with the target domain, potentially increasing the attack surface.
- **Business Impact:** Information gathering may lead to further exploitation.
- **Technical Impact:** Discovery of unprotected subdomains.
- **Affected Asset:** *.testphp.vulnweb.com

```
┌──(kali㉿vbox)-[~/test]
└─$ subfinder -d vulnweb.com -o subdomains.txt

                         __     _____           __
   _____  __  __/ /_   / __(_)___  ____/ /__  _____
  / ___/ / / / / __ \/ /_/ / __ \/ __  / _ \/ ___/
 (__  ) /_/ / / /_/ / __/ / / / / /_/ /  __/ /
/____/\__,_/_.___/_/ /_/_/ /_/ /_/\__,_/\___/_/

                projectdiscovery.io

[INF] Current subfinder version v2.6.0 (outdated)
[INF] Loading provider config from /home/kali/.config/subfinder/provider-config.yaml
[INF] Enumerating subdomains for vulnweb.com
test.php.vulnweb.com
rest.vulnweb.com
testhtml5.vulnweb.com
edu-rost.ruwww.vulnweb.com
viruswall.vulnweb.com
restasp.vulnweb.com
testaspx.vulnweb.com
testaspnet.vulnweb.com
www.testasp.vulnweb.com
antivirus1.vulnweb.com
edu-rost.rutestasp.vulnweb.com
httptestaspnet.vulnweb.com
www.test.php.vulnweb.com
tetphp.vulnweb.com
www.vulnweb.com
testap.vulnweb.com
testapsnet.vulnweb.com
ttestphp.vulnweb.com
www.testphp.vulnweb.com
testphp.vulnweb.com
virus.vulnweb.com
www.virus.vulnweb.com
test.vulnweb.com
5cwww.vulnweb.com
testasp.vulnweb.com
blogger.com.vulnweb.com
odincovo.vulnweb.com
testaps.vulnweb.com
[INF] Found 28 subdomains for vulnweb.com in 5 seconds 618 milliseconds
```

3.7 SBL-FIN-007: sub finder enumeration

Above Image showing by using sub finder – Discovered Subdomains
Description: Terminal output from sub finder showing discovered subdomains under `testphp.vulnweb.com`, indicating expanded attack surface.

Remediation Steps:

- Audit and decommission unused or legacy subdomains.

- Apply web application firewalls (WAF) to restrict access to sensitive or internal subdomains.
- Implement proper DNS hygiene by removing stale records.
- Monitor new subdomains using tools like Security Trails or DNS monitoring services.

Wappalyzer: SBL-FIN-005: Outdated Software Exposure — PHP & Nginx

- **Severity:** HIGH (7.2)
- **CVSS:** AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N
- **Description:**
  Wappalyzer fingerprinted tech stack: PHP 5.6.40, Nginx. Known exploits exist for both.

- **Links:**
  - PHP 5.6.40 DoS Exploit
  - PHP CVE List
- **Business Impact:** Could allow targeted DoS or remote execution.
- **Technical Impact:** System compromise from known exploits.
- **Affected Asset:** Entire web stack

Findings:

- Exploit for Nginx observed: denial of service - https://www.exploitdb.com/exploits/50973
- Exploit for PHP 5.6.40 – Memory corruption, Denial of service
  https://www.cvedetails.com/version/1334572/PHP-PHP-5.6.40.html

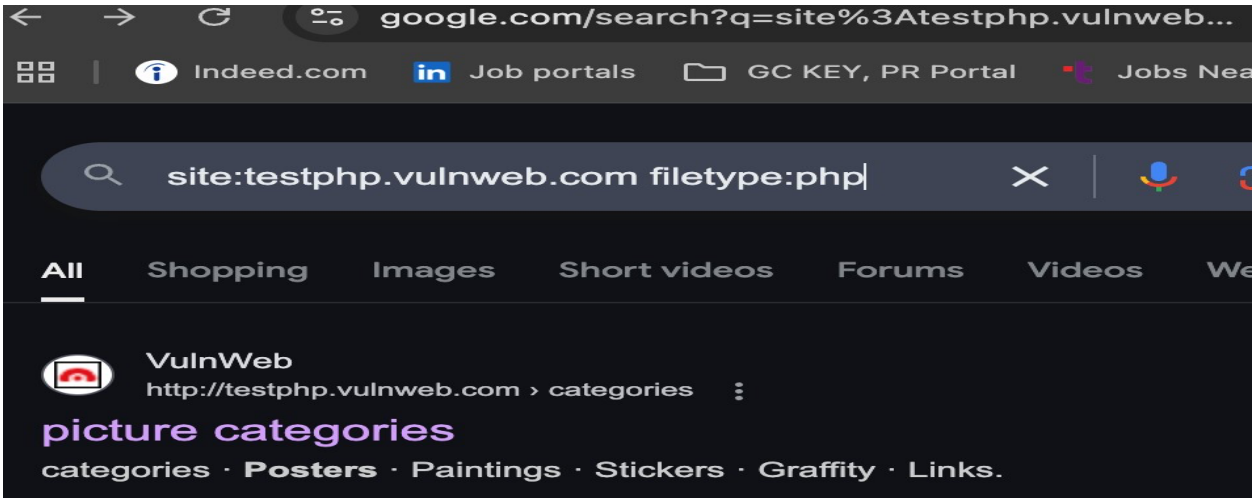
Wappalyzer (Outdated Software Detection)

Issue Identified: Outdated PHP (5.6.40), Nginx versions **Remediation Steps**:

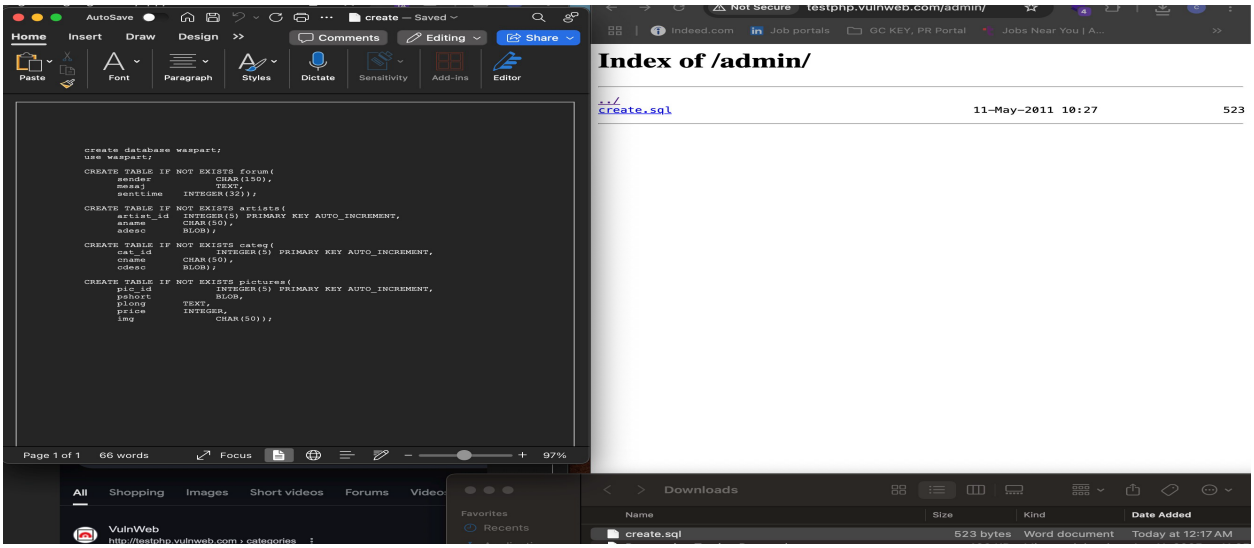
- **Upgrade PHP** to a currently supported version (e.g., PHP 8.1 or later).
- Update **Nginx** to the latest stable version.
- Apply security patches and follow **vendor advisories** for all technologies in use.
- Implement **patch management processes** for server software and frameworks.

Google Dorks:

- **Description:** Using Google Dorks (site:testphp.vulnweb.com filetype:php), admin panel located at /admin/. Often an entry point for brute-force or privilege escalation.
- **Business Impact:** Unauthorized access to sensitive backend controls.
- **Technical Impact:** Potential privilege escalation and configuration tampering.
- **Affected Asset:** http://testphp.vulnweb.com/admin/
- Google Dorks Tried with testphp.vulnweb.com – (site:testphp.vulnweb.com filetype:php)



(http://testphp.vulnweb.com/admin/) admin panels was revealed which are often targets for attackers.



3.8 SBL-FIN-008: Exposed Admin Panel (Google Dorking)

Heading: Admin Panel Discovered – Google Dorking
Description: Screenshot showing `/admin/` panel exposed through Google search (`site:testphp.vulnweb.com filetype:php`), revealing access to a sensitive endpoint.

Remediation Steps:

- Use **robots.txt** to disallow indexing of sensitive paths (though not a security control).
- Implement **authentication and IP whitelisting** for admin interfaces.
- Remove unnecessary admin panels or move them behind a VPN/firewall.
- Monitor search engine results regularly for exposed endpoints.

4. Appendix

4.1 Tools List

4.1 Tools List

S No.	Tool Name & Version	Description
01	**Subdomain enumeration (Sub finder):**	Discovers subdomains associated with a target domain
02	SQL Map 1.7.2.12#dev	Automated SQLi scanner and database dumper
03	Crack Station	Online hash cracker using rainbow tables
04	Wappalyzer	For routing requests through Burp
05	Burp Suite Community Edition + Foxy Proxy	Proxy to intercept and analyze web traffic
06	Google Dorks	Able to search for a specific term in the URL
07	XSS CON with XSS (Reflected XSS) – V 5.0	Code Injection attack, where attacker injects malicious code within a legal website.

4.2 Limitations

- No authentication bypass testing attempted.
- Application resets frequently; inconsistent session state.
- Testphp.vulnweb.com is a training environment — limited real-world impact.
- No brute-force or DoS tests performed.