🛠Manual Penetration Testing Report

testphp.vulnweb.com

Harsh Kumar

Report Dated: 5th Feb 2025

Table of Content

Scope of Assessment for Penetration Testing on testphp.vulnweb.com

Objective:

The primary objective of this penetration test is to evaluate the security vulnerabilities in testphp.vulnweb.com, a deliberately vulnerable website designed for security training and ethical hacking practice.

In-Scope Components:

The assessment will cover the following areas:

1. Web Application Security
Identifying injection vulnerabilities (SQLi, XSS, CSRF, LFI, RFI)
Testing authentication mechanisms (Brute force, credential stuffing)
Evaluating session management (Session hijacking, cookie manipulation)

2. Server and File System Exposure
Checking for misconfigured directories and files
Exploring access control flaws (Directory traversal, privilege escalation)
Network-Level Vulnerabilities
Identifying open ports and exposed services (if applicable)
Data Exposure Risks
Evaluating how sensitive information (e.g., credentials, database content) is handled

3. Out of Scope:
Attacks that can cause downtime or disrupt services (e.g., DDoS, resource exhaustion)
Real-world exploitation of vulnerabilities beyond the test environment
Social engineering attacks or targeting external users

4. Testing Methodology:
Manual Testing: Using web browsers, Burp Suite, and command-line tools
Automated Scans: Limited to non-destructive vulnerability scanning
Ethical Considerations: No unauthorized access beyond the test environment

5. Deliverables:
Detailed Report of identified vulnerabilities, risk levels, and remediation steps
Screenshots & Proof-of-Concepts (PoCs) for successful exploits
Security Recommendations to mitigate identified risks

Directory Traversal

Description:

Directory Traversal (also known as Path Traversal) is a type of security vulnerability that allows an attacker to access files and directories that are stored outside the web document root folder. This can lead to unauthorized access to sensitive files or system configurations, potentially compromising the security of the system or application.

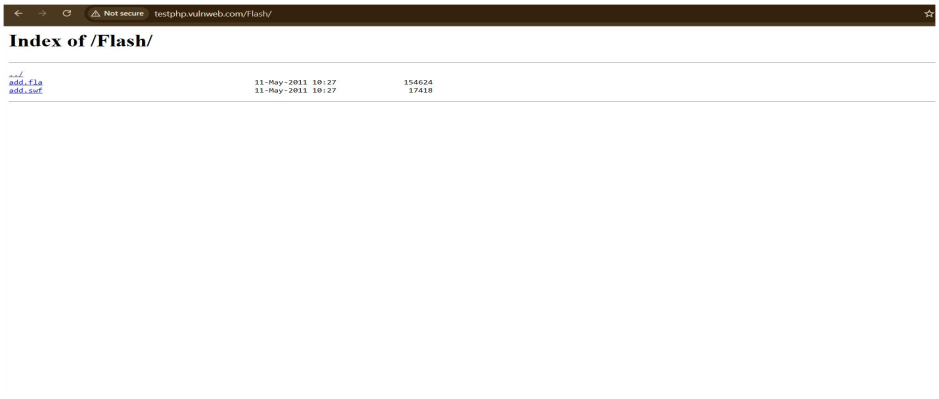
Possible Impacts:

1. Unauthorized Access to Sensitive Files
2. Data Breach
3. Privilege Escalation
4. Denial of Service (DoS)

Severity : HIgh

Steps :

1. Go the testphp site
2. Append Flash in the end of the url
3. Click Enter



Index of /Flash/

```
../
add.fla                                            11-May-2011 10:27              154624
add.swf                                            11-May-2011 10:27               17418
```

MItigation:

1. Reject Malicious Input: Block sequences like ../ or ..\\ and ensure input doesn't allow navigation outside a specified directory.
2. Use APIs that automatically handle path sanitization (e.g., realpath() in PHP, Path.GetFullPath() in .NET) and avoid concatenating user input into file paths.
3. Restrict Directories: Limit file access to specific, secure directories. Use techniques like chroot (on Linux) or sandboxing.

Cross-Site Scripting (XSS)

Description:

Cross-Site Scripting (XSS) is a security vulnerability that allows attackers to inject malicious scripts (usually JavaScript) into web pages viewed by other users. The injected script is executed within the context of the victim's browser, which can lead to unauthorized actions such as stealing session cookies, defacing websites, or redirecting users to malicious sites.

Severity : High

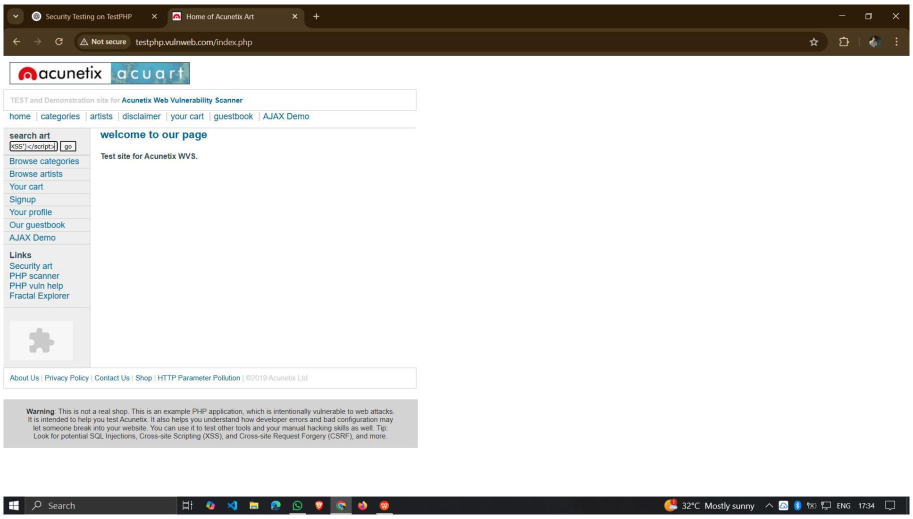
Impact of XSS:

1. Session Hijacking
2. Account Takeover
3. Data Theft
4. Malware Distribution
5. Defacement

Steps:

1. Find an Input Field Go to: http://testphp.vulnweb.com/comment.php
Inject a Script : <script>alert('XSS')</script>
2. If an alert box appears, the site is vulnerable.

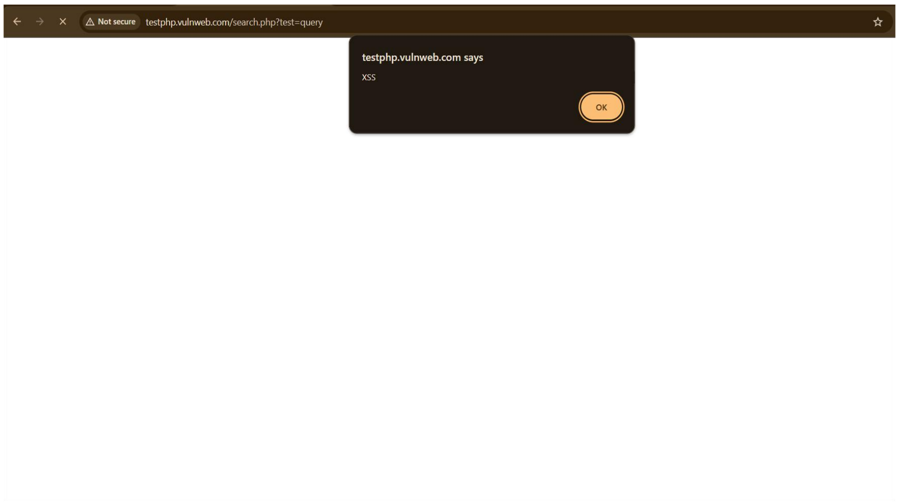
Screenshots:


Security Testing on TestPHP
Home of Acunetix Art
Not secure
testphp.vulnweb.com/index.php
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
XSS')</script>
go
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
Search
32°C Mostly sunny
ENG
17:34

welcome to our page

Test site for Acunetix WVS.

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.


Not secure testphp.vulnweb.com/search.php?test=query

testphp.vulnweb.com says

XSS

Mitigation:

1. Sanitize all user inputs to remove harmful characters.
2. Implement a CSP header to restrict where scripts can be loaded from and prevent inline JavaScript execution.
3. Set cookies with HttpOnly to prevent JavaScript access.

SQL injection

Description:

SQL Injection occurs when an attacker manipulates a website’s database queries by inserting malicious SQL statements in input fields. This can lead to unauthorized access to data, modification of databases, or even complete database control.

Payload:

Username: ' OR '1'='1

Password: ' OR '1'='1

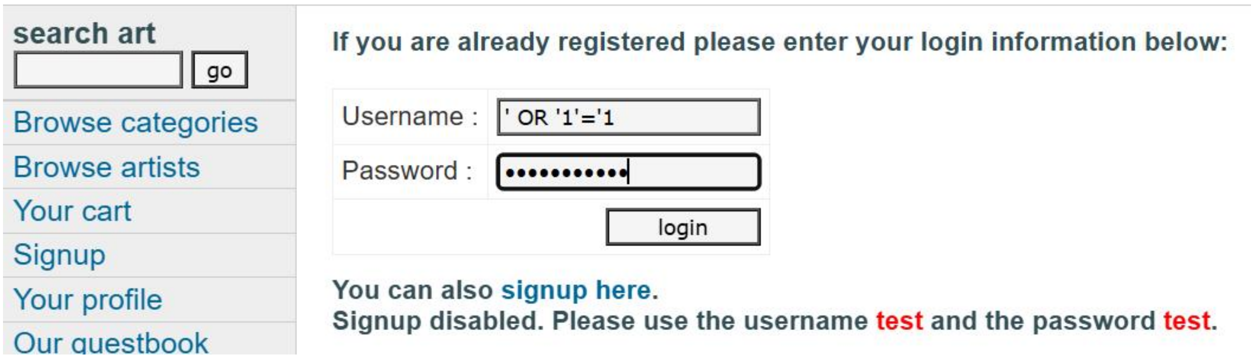
Severity: High

Impact of SQL Injection:

SQL injection can lead to unauthorized data access, manipulation, or deletion, causing severe privacy breaches and system compromise. It can also result in website defacement, downtime, and significant reputation damage.

STEPS

1. Open the test website's login page.
2. Enter ' OR '1'='1 in both Username and Password fields.
3. Click the Login button.
4. If login is successful, the site is vulnerable to SQL Injection.
5. If not, try admin' -- in Username and leave Password blank.
6. Click Login and observe if it bypasses authentication.
7. For blind SQL injection, enter ' AND 1=1 -- in Username and leave Password blank.
8. If successful, repeat with ' AND 1=2 --; failure indicates vulnerability.
9. Document results and responses for each attempt.
10. Use the findings to understand and mitigate SQL injection risks.


search art

go

Browse categories

Browse artists

Your cart

Signup

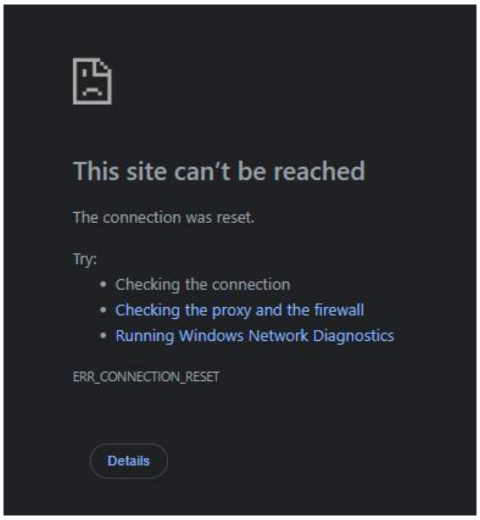
Your profile

Our guestbook

If you are already registered please enter your login information below:

Username :	' OR '1'='1
Password :	•••••••••••
	login

You can also signup here.
Signup disabled. Please use the username test and the password test.


This site can’t be reached

The connection was reset.

Try:

- Checking the connection
- Checking the proxy and the firewall
- Running Windows Network Diagnostics

ERR_CONNECTION_RESET

Details

Reference:
https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

Mitigation:
Use prepared statements and input validation to prevent malicious code execution. Limit database access, deploy Web Application Firewalls (WAFs), and perform regular security audits to identify and fix vulnerabilities.

Man-in-the-Middle (MitM) Attack

Absence of Anti-CSRF Tokens

Description:

No Anti-CSRF tokens were found in the form. CSRF attacks exploit the trust a site has in a user, causing unintended actions without their knowledge. It differs from XSS, which exploits user trust in a site. Other names for CSRF include XSRF, one-click attack, and session riding.

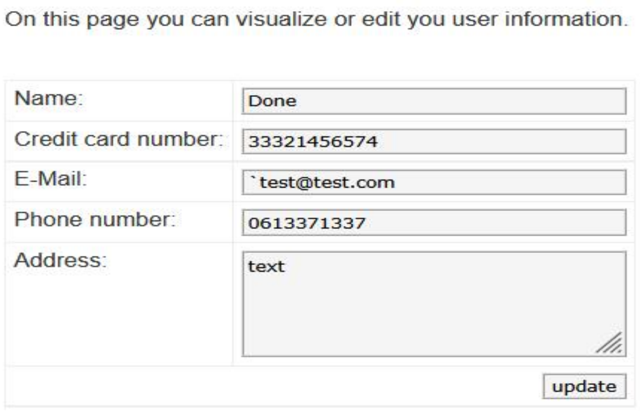
Possible Impacts:

Unauthorized Actions
Session Hijacking
Loss of Trust
Privilege Escalation

Severity: High

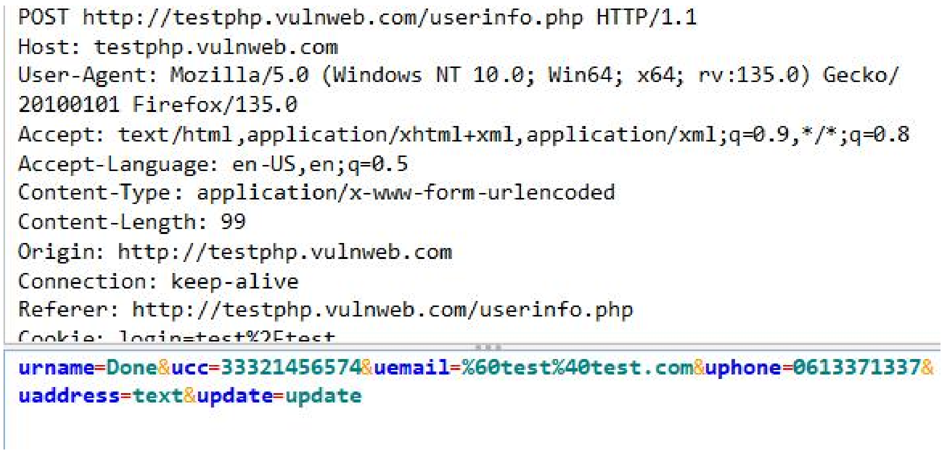
Steps to reproduce

Step1: After Login we will be landing to userInfo page.
POST: http://testphp.vulnweb.com/userinfo.php
Step2: After filling the form click to update button and data will be intercepted.
Step3: Intercepting the request in middle (before server) and changing the data of userinfo form.
Step4: Sending response to client with updated data.

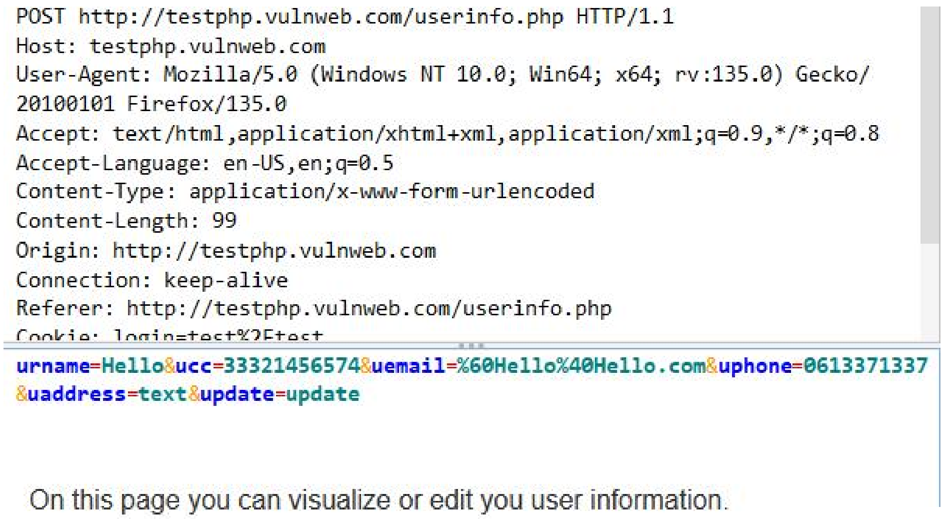


On this page you can visualize or edit you user information.

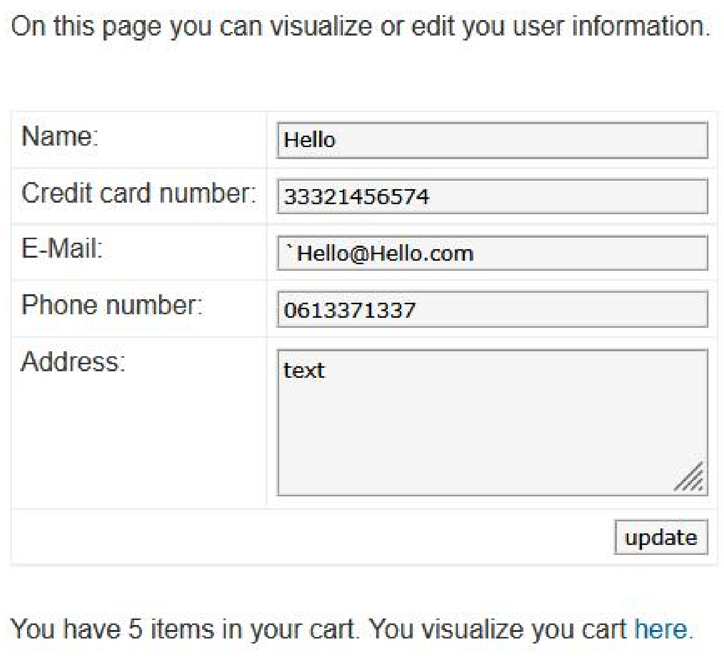
Name:	Done
Credit card number:	33321456574
E-Mail:	`test@test.com
Phone number:	0613371337
Address:	text
	update



```
POST http://testphp.vulnweb.com/userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/
20100101 Firefox/135.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 99
Origin: http://testphp.vulnweb.com
Connection: keep-alive
Referer: http://testphp.vulnweb.com/userinfo.php
Cookie: login=test%2Ftest
urname=Done&ucc=33321456574&uemail=%60test%40test.com&uphone=0613371337&
uaddress=text&update=update
```



```
POST http://testphp.vulnweb.com/userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/
20100101 Firefox/135.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 99
Origin: http://testphp.vulnweb.com
Connection: keep-alive
Referer: http://testphp.vulnweb.com/userinfo.php
Cookie: login=test%2Ftest
urname=Hello&ucc=33321456574&uemail=%60Hello%40Hello.com&uphone=0613371337
&uaddress=text&update=update
```



On this page you can visualize or edit you user information.

Name:	Hello
Credit card number:	33321456574
E-Mail:	`Hello@Hello.com
Phone number:	0613371337
Address:	text

You have 5 items in your cart. You visualize you cart here.

Solutions and Mitigation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

CSRF Attack

Description:

XSS allows an attacker to inject malicious JavaScript into a webpage, which then executes in a victim’s browser. This can be used to steal cookies, redirect users, or modify the page content.

Payload:

```
var form = document.createElement('form');
form.method = 'POST';
form.action = 'http://<your-test-site>/guestbook';  // Replace with your actual site URL

var input = document.createElement('input');
input.type = 'hidden';
input.name = 'message';  // Replace 'message' with the actual field name if different
input.value = 'CSRF attack successful!';

form.appendChild(input);
document.body.appendChild(form);
form.submit();
```

Impact of CSRF:

XSS can lead to stolen cookies, session hijacking, and unauthorized actions on behalf of the user. It may also redirect users to malicious sites or modify web page content, damaging user trust and system integrity.

Severity: High

Steps:

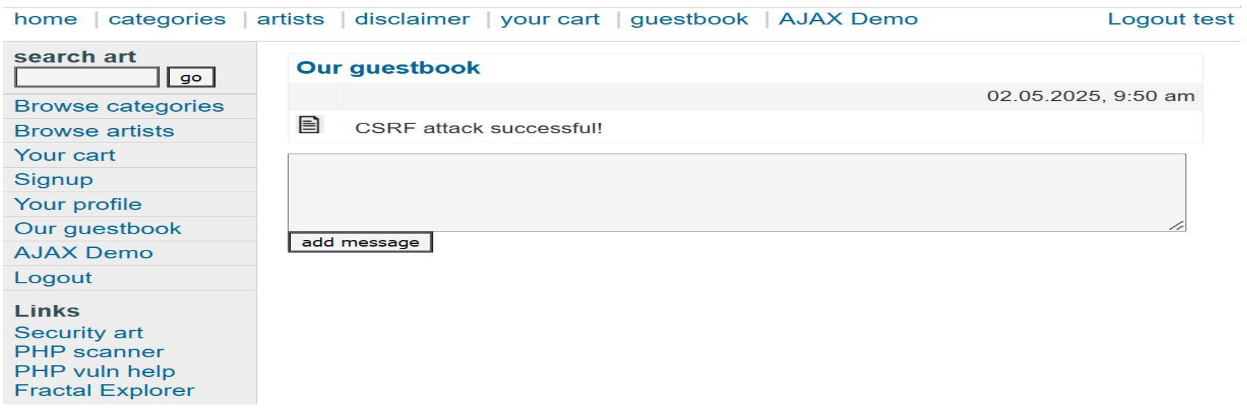
1.Open a New Tab in the Same Browser

Stay logged into the target website (your test site) in one tab.

2.Create a Malicious HTML Form Locally: Open the Developer Tools in your browser (F12 or right-click and select Inspect).

3. Go to the Console tab.: Inject the Malicious Form via JavaScript:

Paste this payload code in the console to simulate a CSRF attack.



Our guestbook

02.05.2025, 9:50 am

CSRF attack successful!

Mitigation:

Use anti-CSRF tokens to validate requests and ensure they are intentional. Implement same-site cookie attributes and require re-authentication for sensitive actions to protect against CSRF attacks.

Server Version Leakage

Description:

Server Version Leakage refers to the unintended exposure of a server's version information, which can occur through error messages, HTTP headers, or other means.

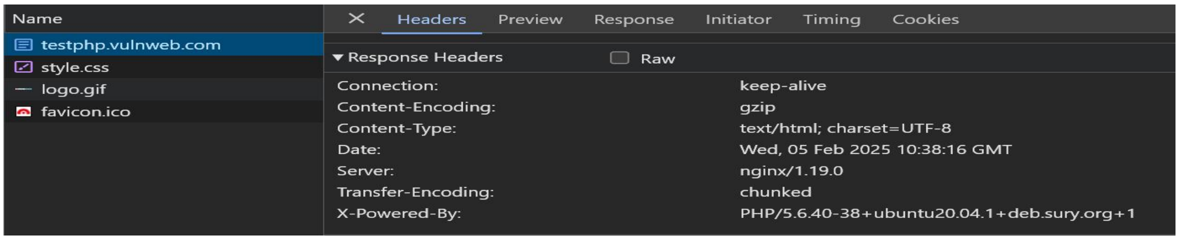
Impact:

This leakage can allow attackers to identify vulnerabilities specific to that server version, increasing the risk of exploitation. It can lead to unauthorized access, data breaches, or server compromise, affecting the security and integrity of the system.

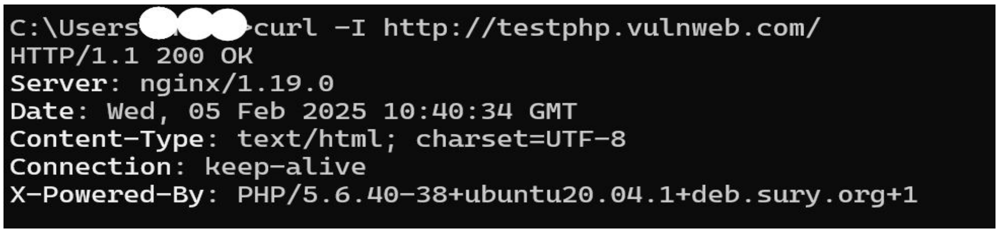
Severity: High

Steps to Reproduce:

1. Open http://testphp.vulnweb.com/ in your browser.
2. Right-click > Inspect > Network tab > Refresh the page.
4. Click on the first request > Scroll to Response Headers > Check for the Server header.
4. Run curl -I http://testphp.vulnweb.com/ in the terminal to view HTTP headers.
5. Look for the Server header in the curl output, e.g., Server: Apache/2.4.7 (Ubuntu).
6. Run nmap -sV testphp.vulnweb.com to detect service versions.
7. Check the Nmap output for exposed server version details.



```
Connection:             keep-alive
Content-Encoding:       gzip
Content-Type:           text/html; charset=UTF-8
Date:                   Wed, 05 Feb 2025 10:38:16 GMT
Server:                 nginx/1.19.0
Transfer-Encoding:      chunked
X-Powered-By:           PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
```



```
C:\Users[redacted]>curl -I http://testphp.vulnweb.com/
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Wed, 05 Feb 2025 10:40:34 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
```

Mitigation:

To mitigate server version leakage, disable detailed error messages, remove or obscure version information in HTTP headers, and implement security best practices such as server hardening. Regularly update server software to patch known vulnerabilities and minimize exposure.

XSS Attack

Description:

XSS allows an attacker to inject malicious JavaScript into a webpage, which then executes in a victim’s browser. This can be used to steal cookies, redirect users, or modify the page content.

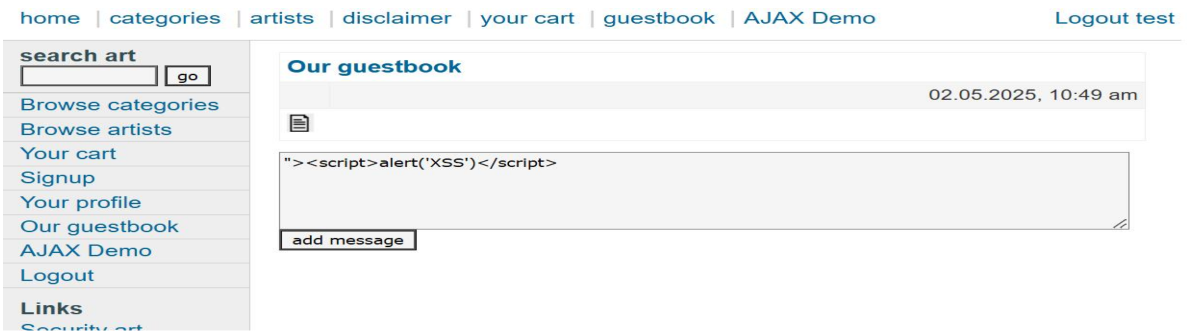
Impact of XSS:

XSS can lead to stolen cookies, session hijacking, and unauthorized actions on behalf of the user. It may also redirect users to malicious sites or modify web page content, damaging user trust and system integrity.

Severity: High

Steps to Reproduce:

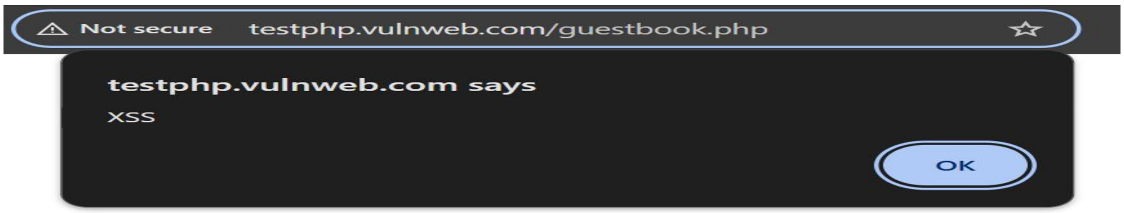
1. Go to http://testphp.vulnweb.com/guestbook.php.
2. Locate the textbox and Add Message button.
3. Enter payload ""><script>alert('XSS')</script> in the textbox.
4. Click the Add Message button.
5. Check if an alert box pops up with 'XSS'.
6. Try alternate payload <img src="x" onerror="alert('XSS')">.
7. Inspect page source (Ctrl+U) or open Developer Tools (F12).
8. Check if payload appears inside the value attribute of an <input> tag.
9. Try breaking the attribute with " autofocus onfocus=alert('XSS') x=".
10. Verify if the alert triggers after page reload or focus.



Our guestbook

02.05.2025, 10:49 am

```
"><script>alert('XSS')</script>
```


Not secure testphp.vulnweb.com/guestbook.php

testphp.vulnweb.com says

XSS

Mitigation:

Sanitize and validate user inputs to prevent script injection. Use Content Security Policy (CSP) and HttpOnly flags for cookies to restrict script access, and implement proper output encoding to safely display data.

Conclusion

Manual penetration testing is a crucial process for identifying vulnerabilities that automated tools may miss, offering a more realistic simulation of potential attacks. It allows for in-depth exploration of systems, providing tailored and accurate assessments by leveraging human intelligence to detect complex issues. While it is time-consuming and requires skilled experts, manual testing is highly effective in uncovering subtle vulnerabilities and should complement automated testing for a comprehensive security evaluation.