



Security Assessment Report

Sagar Vishwakarma

Report Dated: 5th Feb
2025

INDEX

No.1	Title	Risk Level
1	Scope of Assessment	
2	Man-in-the-Middle (MitM) Attack	High
3	SQL injection	High
4	CSRF attack	High
5	Server version leakage	High
6	XSS attack	Low

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

► Objective:

- The primary goal of this penetration test is to identify and analyze security weaknesses present in **testphp.vulnweb.com**, a purposefully vulnerable website designed for ethical hacking and cybersecurity training.

► Scope of Assessment:

- The security evaluation will focus on the following areas:

► 1. Web Application Security

- Detecting injection vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), Local File Inclusion (LFI), and Remote File Inclusion (RFI).
- Assessing authentication security against brute-force attacks and credential stuffing attempts.
- Reviewing session handling mechanisms, including risks like **session hijacking and cookie tampering**.

► 2. Server and File System Security

- Identifying **misconfigured directories and files** that could lead to unauthorized access.
- Testing for **access control flaws**, such as **directory traversal attacks and privilege escalation**.
- Examining **network-level vulnerabilities**, including **open ports and exposed services** (if applicable).
- Evaluating **data exposure risks** by analyzing how **sensitive information** (e.g., credentials, database contents) is stored and managed.

► Out-of-Scope Activities:

- Any actions that could **cause downtime or disrupt services**, such as **DDoS (Distributed Denial of Service) attacks** or **resource exhaustion**.
- Exploiting vulnerabilities **beyond the controlled test environment**.
- Conducting **social engineering attacks** or targeting real-world users.

► Testing Approach:

- **Manual Testing:** Conducted using tools like **web browsers, Burp Suite, and command-line utilities**.
- **Automated Scanning:** Limited to **non-intrusive vulnerability scans** to prevent any unintended disruption.
- **Ethical Compliance:** Testing will strictly adhere to ethical guidelines, ensuring **no unauthorized access beyond the predefined environment**.

► Deliverables:

- A **detailed security report** listing discovered vulnerabilities, associated risk levels, and recommended mitigation measures.
- **Screenshots and Proof-of-Concept (PoC) demonstrations** showcasing successful exploit attempts.
- **Actionable security recommendations** to enhance the security posture of the application.

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

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:	<input type="text" value="Done"/>
Credit card number:	<input type="text" value="33321456574"/>
E-Mail:	<input type="text" value="test@test.com"/>
Phone number:	<input type="text" value="0613371337"/>
Address:	<input type="text" value="text"/>
<input type="button" value="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

username=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

username=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:	<input type="text" value="Hello"/>
Credit card number:	<input type="text" value="33321456574"/>
E-Mail:	<input type="text" value="`Hello@Hello.com"/>
Phone number:	<input type="text" value="0613371337"/>
Address:	<div><input type="text" value="text"/></div>
<input type="button" value="update"/>	

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.

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

OR

Username=' or 1=1—

Password=''

Severity : High

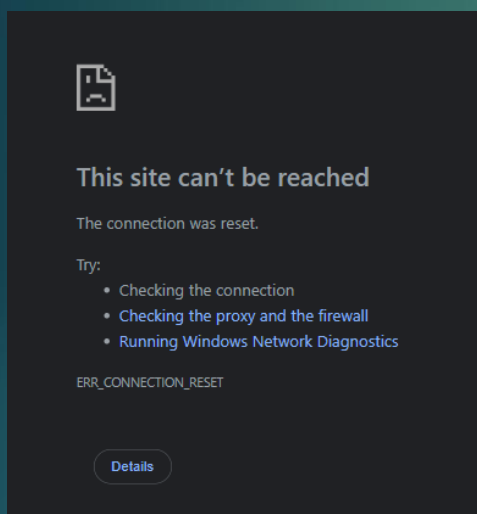
Impact of SQL Injection:

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 <input type="text"/> <input type="button" value="go"/>	If you are already registered please enter your login information below:
Browse categories	Username : <input type="text" value="' OR '1'='1"/>
Browse artists	Password : <input type="password" value="....."/>
Your cart	<input type="button" value="login"/>
Signup	You can also signup here.
Your profile	Signup disabled. Please use the username test and the password test.
Our guestbook	



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.

CSRF Attack

Description: CSRF tricks a user into executing unintended actions on a web application in which they are authenticated

Impact of CSRF:

CSRF can lead to unauthorized actions being performed on behalf of an authenticated user, such as changing account settings, transferring funds, or deleting data. It can compromise user accounts without their knowledge, resulting in security breaches.

Sevrity : high

Steps:

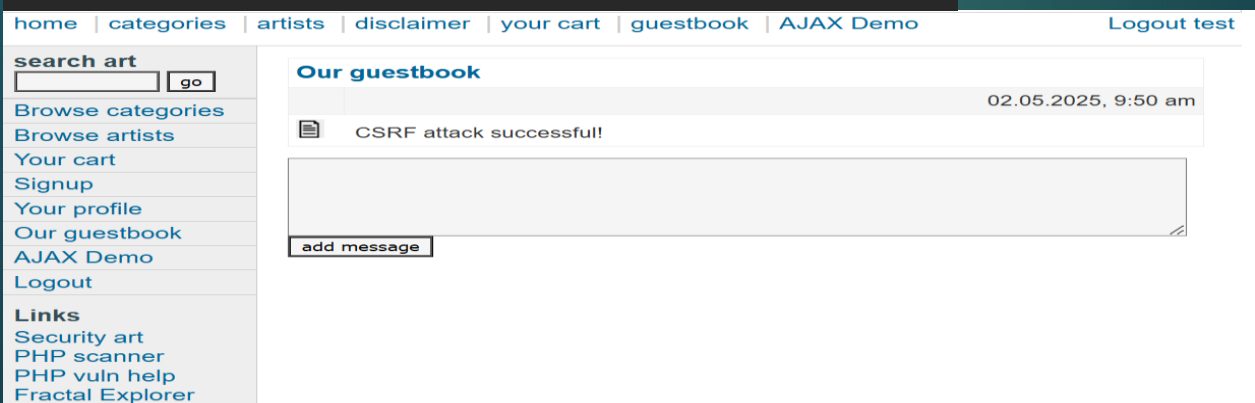
1.Open a New Tab in the Same Browser

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.

```
var form = document.createElement('form');
form.method = 'POST';
form.action = 'http://testphp.vulnweb.com/guestbook';
var input = document.createElement('input');
input.type = 'hidden';
input.name = 'message';
input.value = 'CSRF attack successful!';
form.appendChild(input);
document.body.appendChild(form);
form.submit();
```



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.

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.

Name	✕	Headers	Preview	Response	Initiator	Timing	Cookies
testphp.vulnweb.com							
style.css							
logo.gif							
favicon.ico							
		▼ Response Headers		<input type="checkbox"/> Raw			
		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\●●●●● 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:

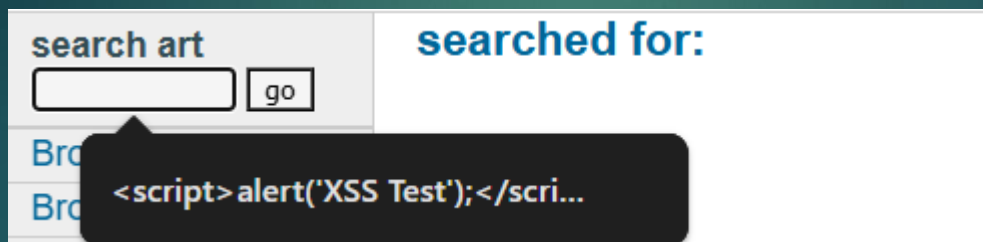
- ▶ XSS allows an attacker to inject malicious JavaScript into a webpage, which then executes in a victim's browser. This can be used to steal cookies, redirect users, or modify the page content.

Impact of XSS:

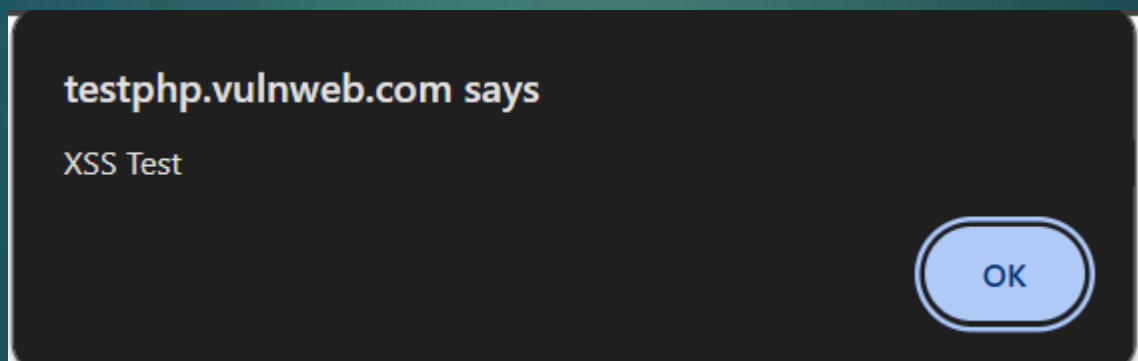
- ▶ XSS can lead to stolen cookies, session hijacking, and unauthorized actions on behalf of the user. It may also redirect users to malicious sites or modify web page content, damaging user trust and system integrity.

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 Test')</script>` in the textbox.
4. Click the Add Message button.



The screenshot shows a web interface with a search bar labeled "search art" containing a text input and a "go" button. To the right is a section labeled "searched for:". Below the search bar, there is a message input field with a tooltip showing the payload `<script>alert('XSS Test');</scri...`. Below the input field, there are two buttons labeled "Bro" and "Add Message".



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.