

Ethical Hacking Internship - Technical Report

Intern Name: SARTHAK VIJAY JOSHI

Domain: Ethical Hacking

Batch: 1-1-2025

Company: XYZ Company

Target Application: testphp.vulnweb.com

Introduction

This report documents the penetration testing performed on "testphp.vulnweb.com" as part of XYZ Company's ethical hacking internship. The objective was to identify security vulnerabilities within the application and provide recommendations for mitigation.

Techniques Used

- Information Gathering (Reconnaissance)
- SQL Injection (SQLi)
- Cross-Site Scripting (XSS)

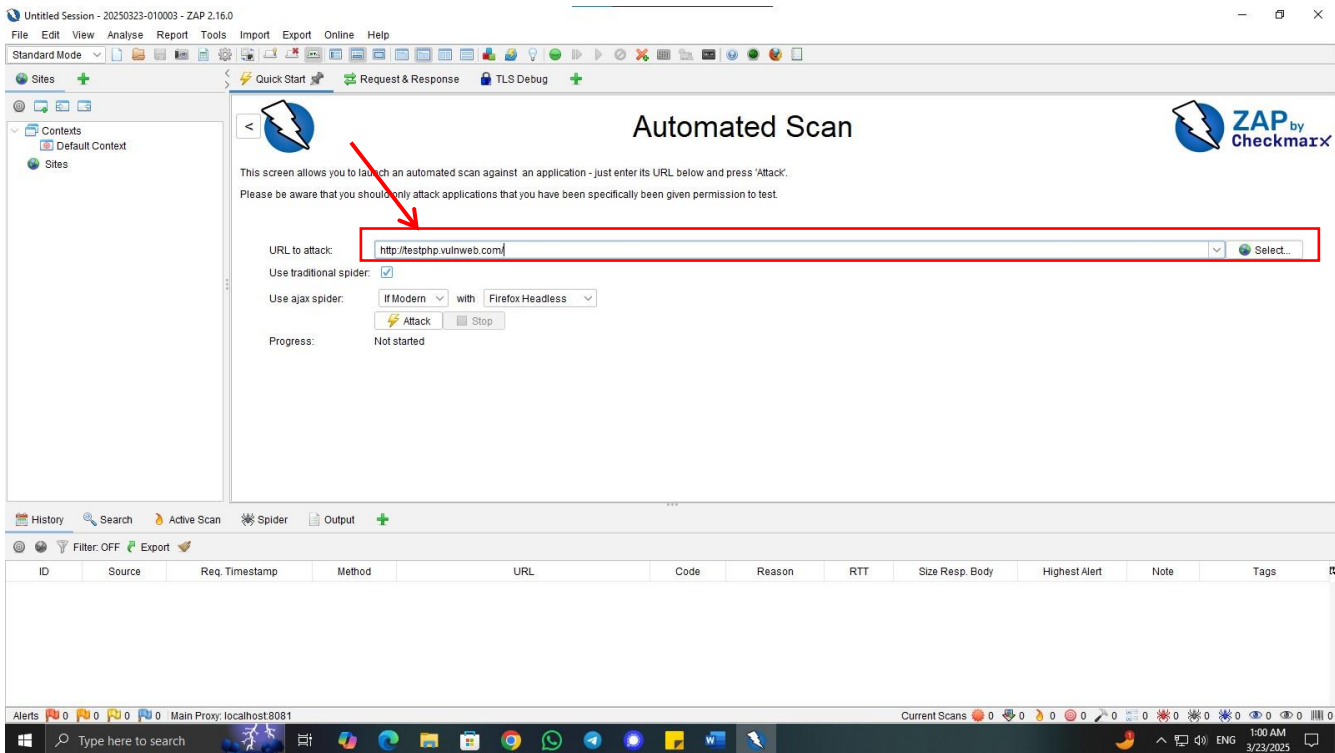
Tools and Frameworks Utilized

- **SQL Map** – (SQL injection vulnerability exploitation)
- **OWASP ZAP**- (automated vulnerability scanning tool)
- **Kali Linux** – (Operating system)
- **Nmap**- (network scanner)

1.used automated tools for identifying vulnerability

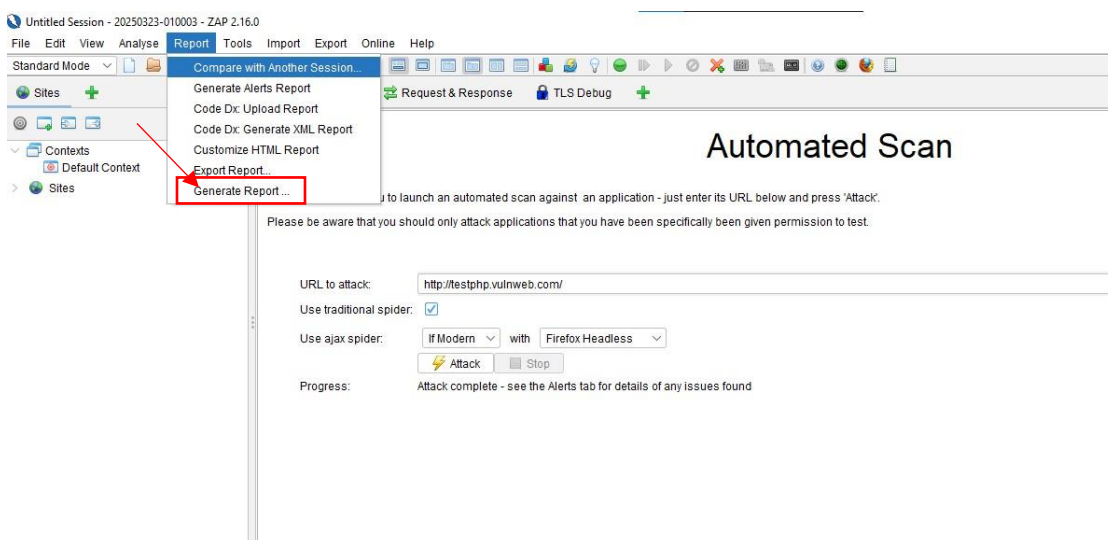
Step1:

open zap proxy to scan the website to identify the vulnerability



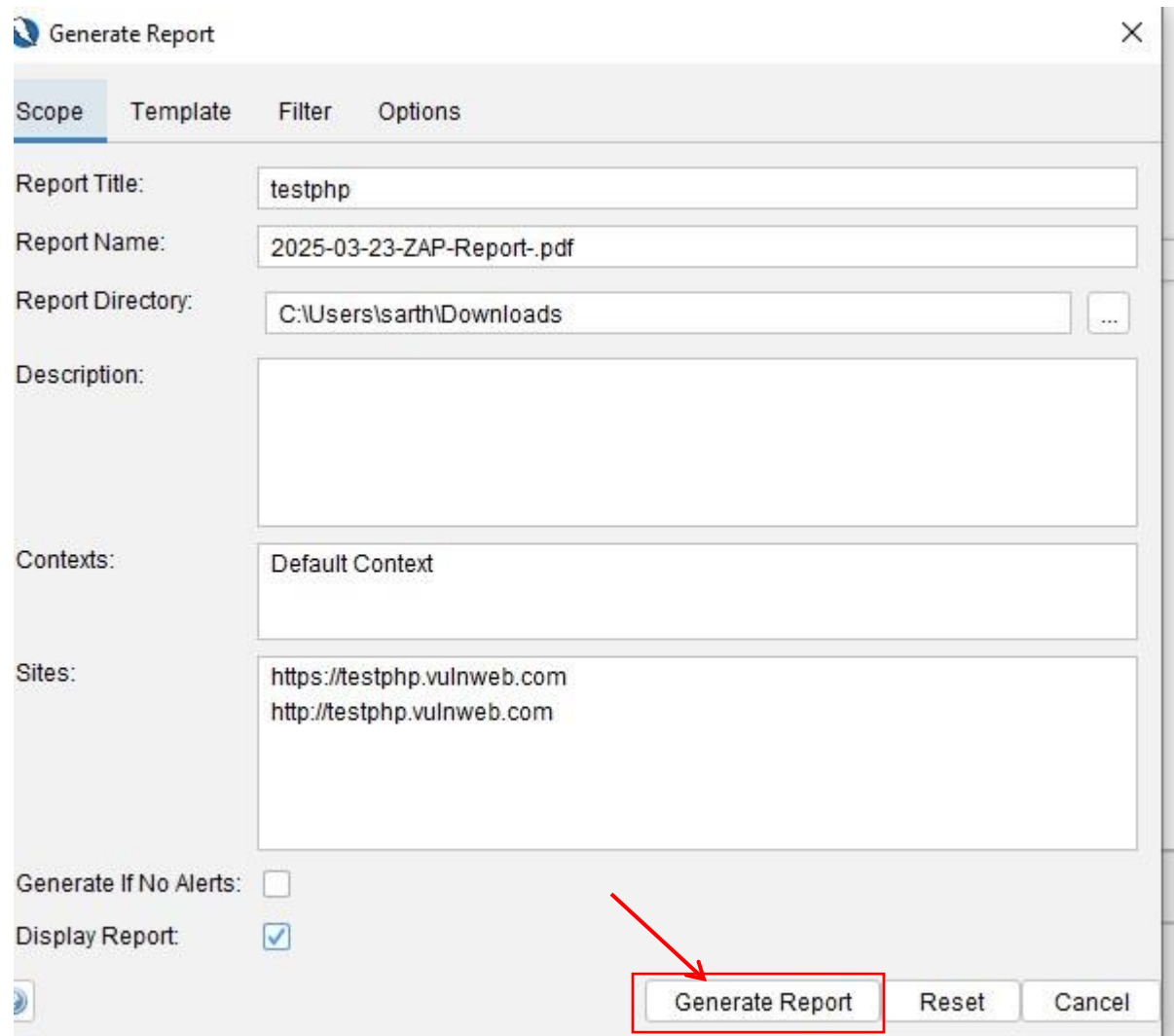
Step2:

After scanning the full website is done go to report and generate the report



Step 3:

After clicking generate report fill report title, report name, report location description then clicks generate report



Generate Report [X]

Scope | Template | Filter | Options

Report Title: testphp

Report Name: 2025-03-23-ZAP-Report-.pdf

Report Directory: C:\Users\sarth\Downloads ...

Description:

Contexts: Default Context

Sites: https://testphp.vulnweb.com
http://testphp.vulnweb.com

Generate If No Alerts: ☐

Display Report: ☒

Generate Report Reset Cancel

Step4:

Open the report and focus on the high-risk vulnerabilities

The screenshot shows a PDF report titled "2025-03-23-ZAP-Report-.pdf" opened in a web browser. The report is generated by ZAP by Checkmarx testphp. It includes the following information:

- Sites:** <https://testphp.vulnweb.com> <http://testphp.vulnweb.com>
- Generated on:** Sun, 23 Mar 2025 00:54:20
- ZAP Version:** 2.16.0
- ZAP by:** Checkmarx

Summary of Alerts

Risk Level	Number of Alerts
High	9
Medium	7
Low	6
Informational	13

A red arrow points to the "High" risk level row in the summary table.

Alerts

Name	Risk Level	Number of Instances
Advanced SQL Injection - AND boolean-based blind - WHERE or HAVING clause	High	3
Advanced SQL Injection - MySQL >= 5.0 boolean-based blind - Parameter replace	High	1
Advanced SQL Injection - MySQL >= 5.0.12 AND time-based blind (SELECT)	High	9
Advanced SQL Injection - MySQL UNION query	High	1

The Windows taskbar at the bottom shows the time as 2:01 AM on 3/23/2025.

Step 5:

Check the high-risk vulnerability click first xss high-risk vulnerability

The screenshot shows a PDF report titled "2025-03-23-ZAP-Report.pdf" open in a browser. The report displays a table of alerts under the heading "Alerts". The table has three columns: "Name", "Risk Level", and "Number of Instances". The "Cross Site Scripting (Reflected)" alert is highlighted with a red box, and a red arrow points to it. The table lists various vulnerabilities, including SQL Injection, Cross-Domain Misconfiguration, and Cross Site Scripting.

Name	Risk Level	Number of Instances
Advanced SQL Injection - AND boolean-based blind - WHERE or HAVING clause	High	3
Advanced SQL Injection - MySQL >= 5.0 boolean-based blind - Parameter replace	High	1
Advanced SQL Injection - MySQL >= 5.0.12 AND time-based blind (SELECT)	High	9
Advanced SQL Injection - MySQL UNION query (NULL - 1 to 10 columns)	High	5
Cross Site Scripting (Reflected)	High	19
Cross-Domain Misconfiguration - Adobe - Read	High	1
SQL Injection	High	1
SQL Injection - MySQL	High	14
Source Code Disclosure - File Inclusion	High	4
Absence of Anti-CSRF Tokens	Medium	40
Anti-CSRF Tokens Check	Medium	4
Backup File Disclosure	Medium	14
Content Security Policy (CSP) Header Not Set	Medium	48
HTTP Only Site	Medium	2
Missing Anti-clickjacking Header	Medium	44
XSLT Injection	Medium	2
In Page Banner Information Leak	Low	3
Insufficient Site Isolation Against Spectre Vulnerability	Low	156

Step 6:

After clicking the vulnerability (Cross site scripting) show the possible paths to find the XSS vulnerability

The screenshot shows a PDF report titled "2025-03-23-ZAP-Report.pdf". The report details a Cross-Site Scripting (XSS) vulnerability. A red arrow points to the URL field in the attack details table, which contains a malicious payload.

WASC Id	19
Plugin Id	90019
High	Cross Site Scripting (Reflected)
Description	<p>Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML, JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology.</p> <p>When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise.</p> <p>There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based.</p> <p>Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash.</p> <p>Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site link (e.g. an attacker site or a malicious link sent via email, just simply view the web page containing the code).</p>
URL	http://testphp.vulnweb.com/artists.php?artist=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E
Method	GET
Attack	<script>alert(1);</script>
Evidence	<script>alert(1);</script>
Other Info	
URL	http://testphp.vulnweb.com/http?pp=%22+onMouseOver%3D%22alert%281%29%3B
Method	GET
Attack	"onMouseOver="alert(1);
Evidence	"onMouseOver="alert(1);

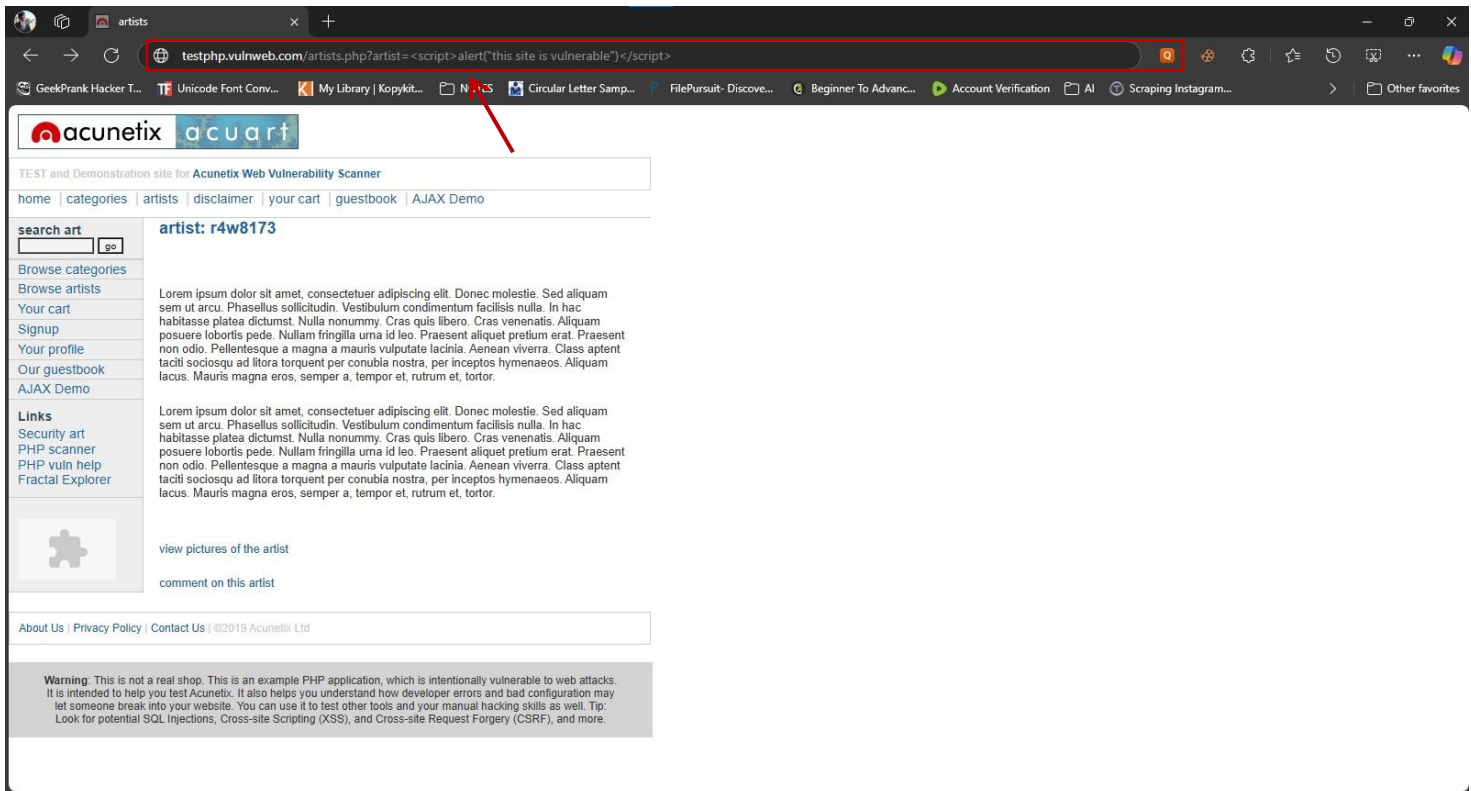
Step 7:

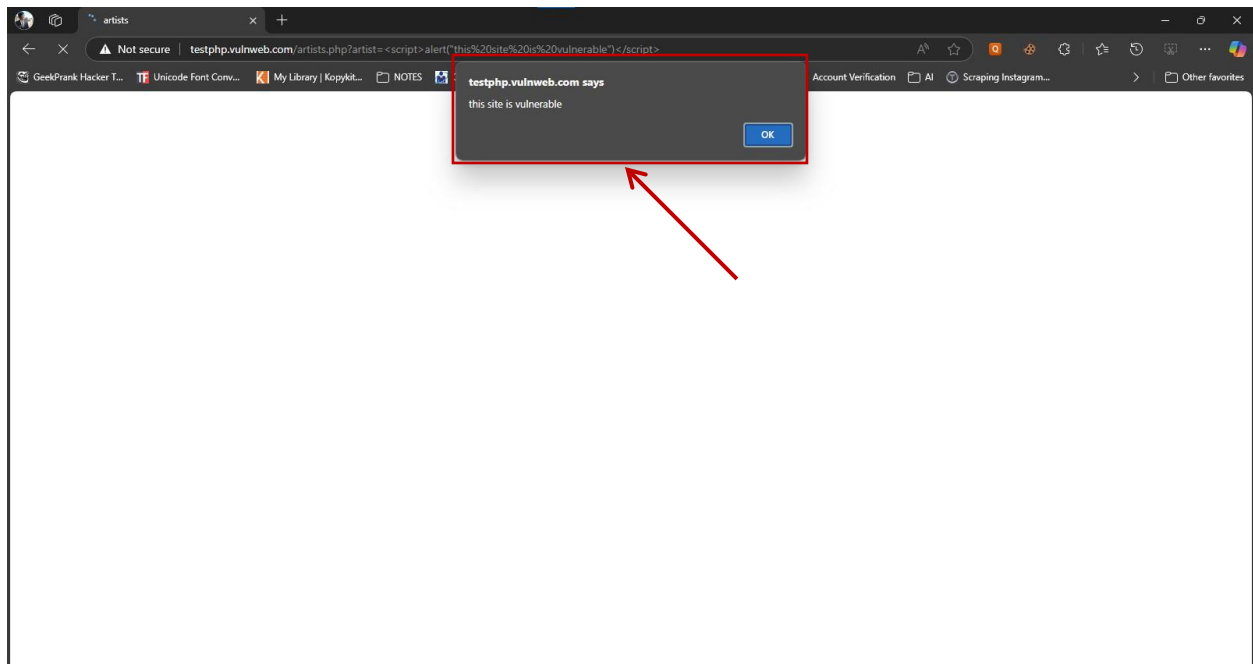
Check the path , vulnerable Parameter and attack payload

URL	http://testphp.vulnweb.com/artists.php?artist=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E
Method	GET
Attack	<script>alert(1);</script>
Evidence	<script>alert(1);</script>
Other Info	

Step 8:

open the link and insert the payload to the vulnerable parameter and check the response if payload run successfully the website is vulnerable with XSS vulnerability





POC

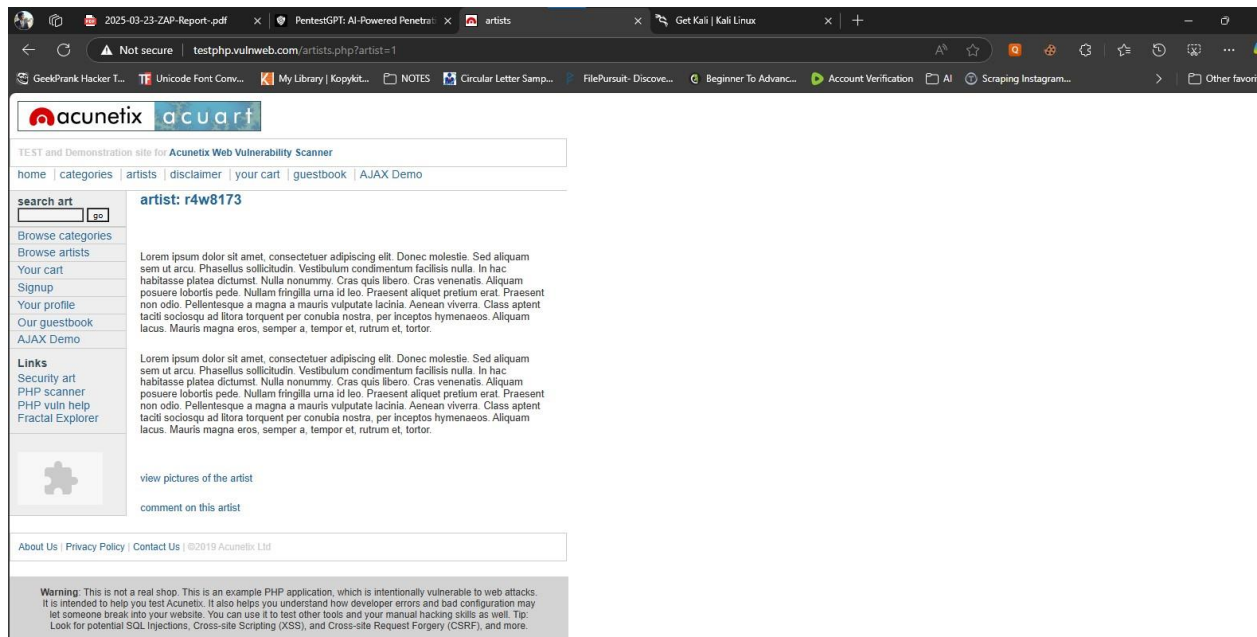
1. Executive Summary

This report details a Cross-Site Scripting (XSS) vulnerability discovered in the <http://testphp.vulnweb.com/> web application. The vulnerability allows an attacker to inject malicious scripts into web pages viewed by other users. This can lead to session hijacking, defacement, or other malicious activities.

2. Vulnerability Details 2.1 Vulnerability Type

Cross-Site Scripting (XSS)

2.2 Affected Component



2.3 Vulnerability Description

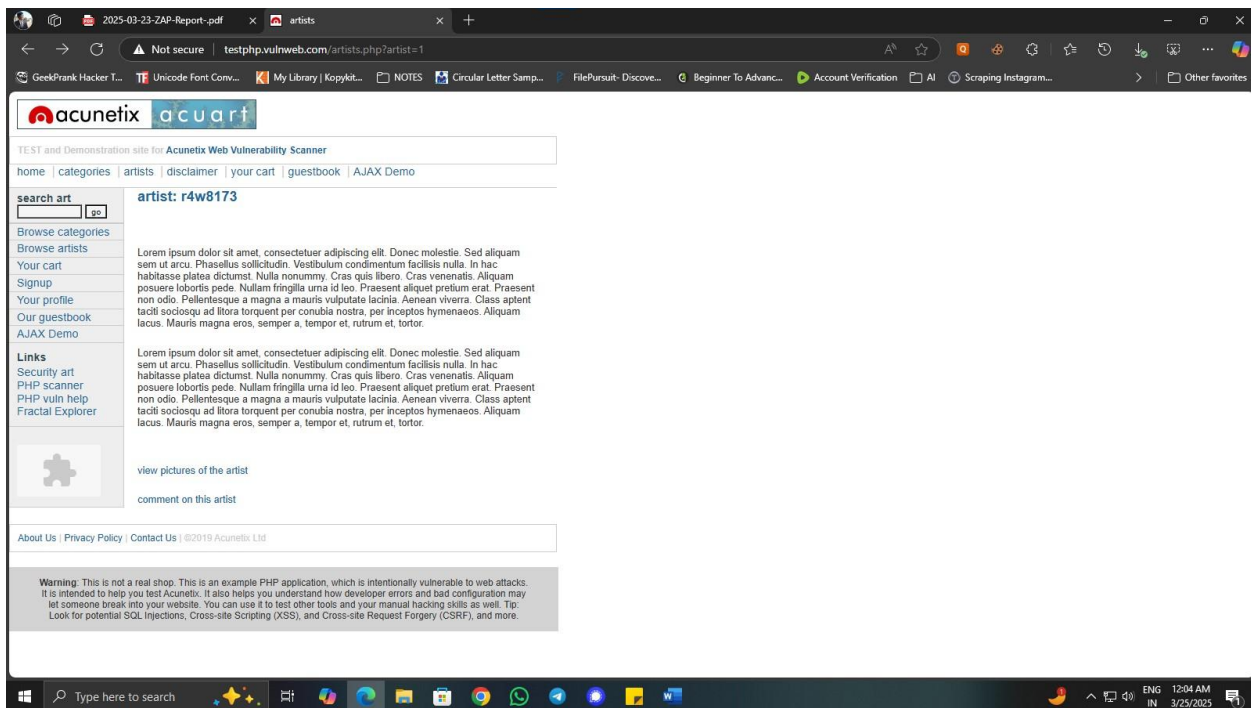
The application does not properly sanitize user input, allowing an attacker to inject malicious scripts into web pages. These scripts can be executed in the context of other users' browsers, leading to potential security risks.

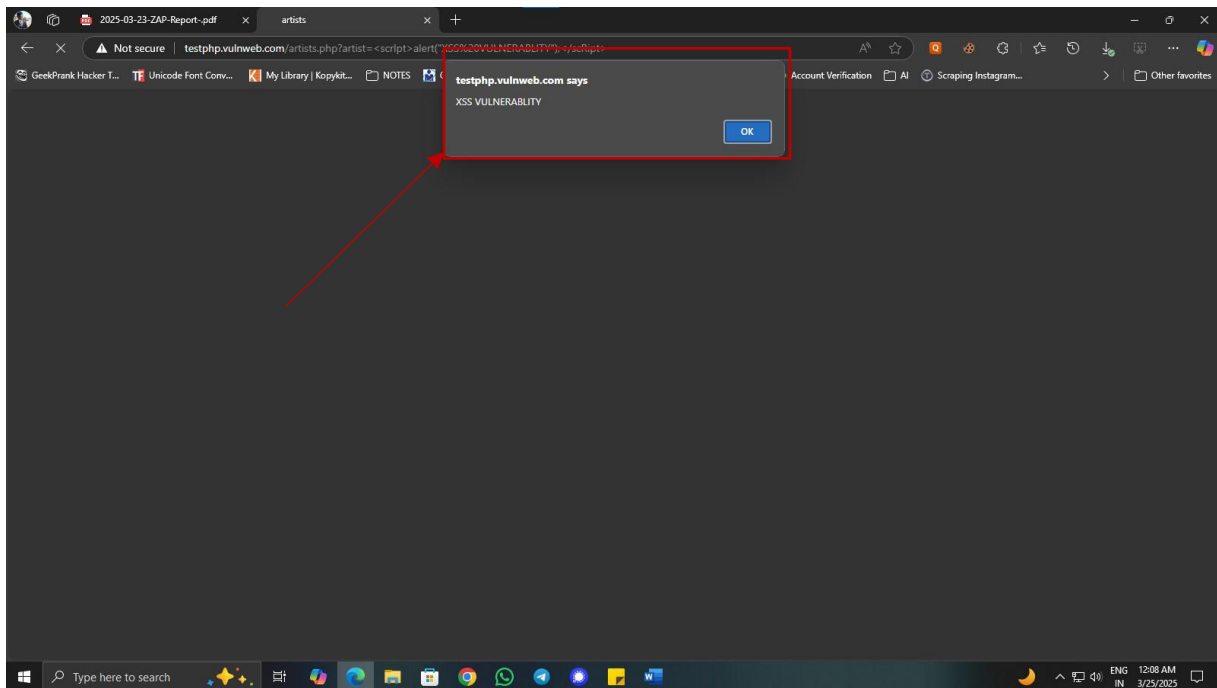
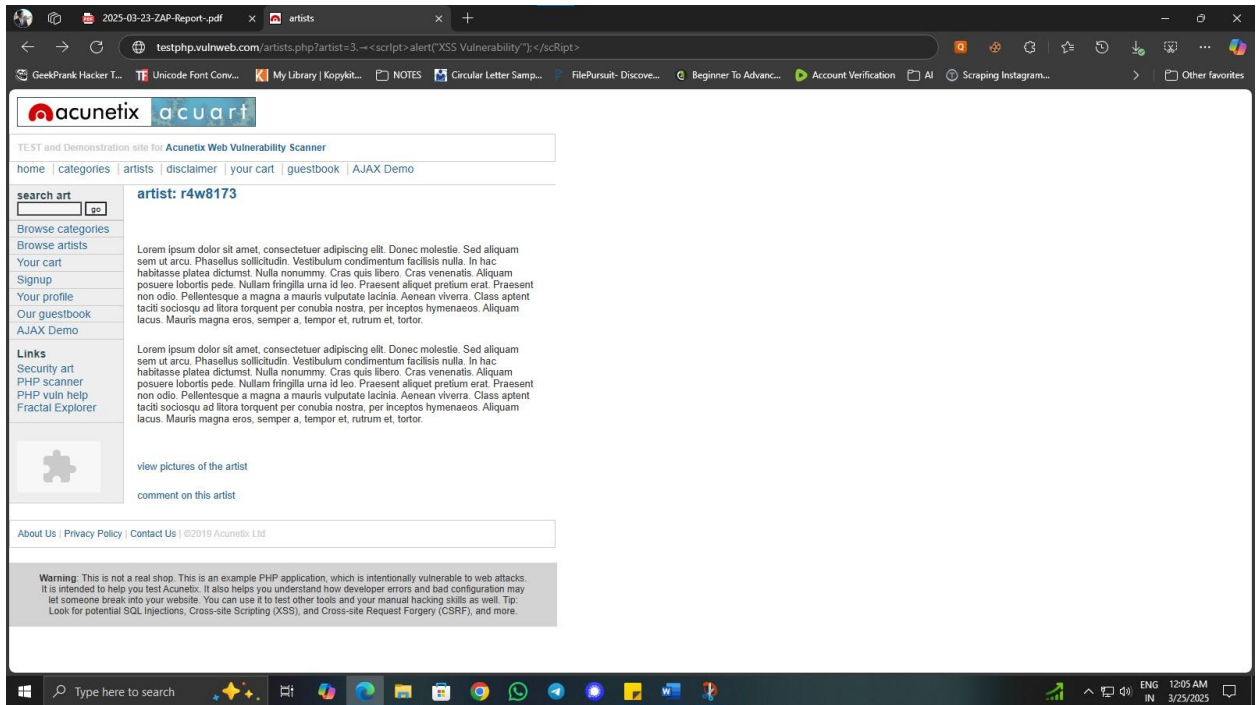
3. Proof of Concept

3.1 Steps to Reproduce

1. Open the web application and go to `http://testphp.vulnweb.com/artists.php?artist=1`
2. Enter the following payload into the URL parameter
3. `<script>alert("XSSVulnerability") ;</script>`
4. The injected script will execute, displaying an alert box with the message "XSS Vulnerability".

3.3 Screenshots:





4. Impact

4.1 Potential Risks

- Session Hijacking: An attacker can steal session cookies and impersonate users.
- Data Theft: Sensitive information can be exfiltrated from the user's browser.
- Defacement: The attacker can modify the appearance of the web page.
- Malware Distribution: Malicious scripts can be used to distribute malware to users.

4. Recommendations

4.1 Immediate Actions

- Temporary Mitigation: Implement input validation and output encoding to prevent script injection.
- Monitoring: Monitor for any suspicious activities related to the vulnerable component.

4.2 Long-Term Solutions

- Input Sanitization: Ensure all user inputs are properly sanitized and validated.
- Output Encoding: Encode all output to prevent scripts from being executed.
- Security Training: Provide training for developers on secure coding practices.
- Regular Audits: Conduct regular security audits and penetration testing.

5. Conclusion

The identified XSS vulnerability poses a significant risk to the security of the web application and its users. Immediate action is required to mitigate the risk and implement long-term solutions to prevent similar vulnerabilities in the future.