FUTURE_CS_01

Web Application Security Testing

Tester Name: Pratik Sayankar

Intern Domain :- Cyber Security Intern

Tools Used: OWASP ZAP 2.14.0

Attacking machine : Kali Linux

Target :- http://testphp.vulnweb.com

Date :- 12-Aug-2025

1. Executive Summary

A penetration test was conducted on http://testphp.vulnweb.com using OWASP ZAP to simulate a real-world attack. The goal was to identify common vulnerabilities, understand their impact, and recommend mitigation measures.

Key Findings:
- Multiple instances of SQL Injection vulnerabilities.
- Reflected XSS present in search and input forms.

Overall Risk Rating: HIGH (Due to exploitable authentication bypass and SQLi).

2. Methodology

The test followed the OWASP Web Security Testing Guide (WSTG):
1. Reconnaissance & Mapping - Enumerated endpoints with OWASP ZAP Spider.
2. Automated Vulnerability Scanning - Active Scan against identified parameters.
3. Manual Validation - Verified findings by crafting custom payloads.
4. Proof of Concept (PoC) - Captured evidence and impact scenarios.

Scope:
- Base URL: http:http://testphp.vulnweb.com

3. Detailed Vulnerability Analysis

3.1 SQL Injection (High)

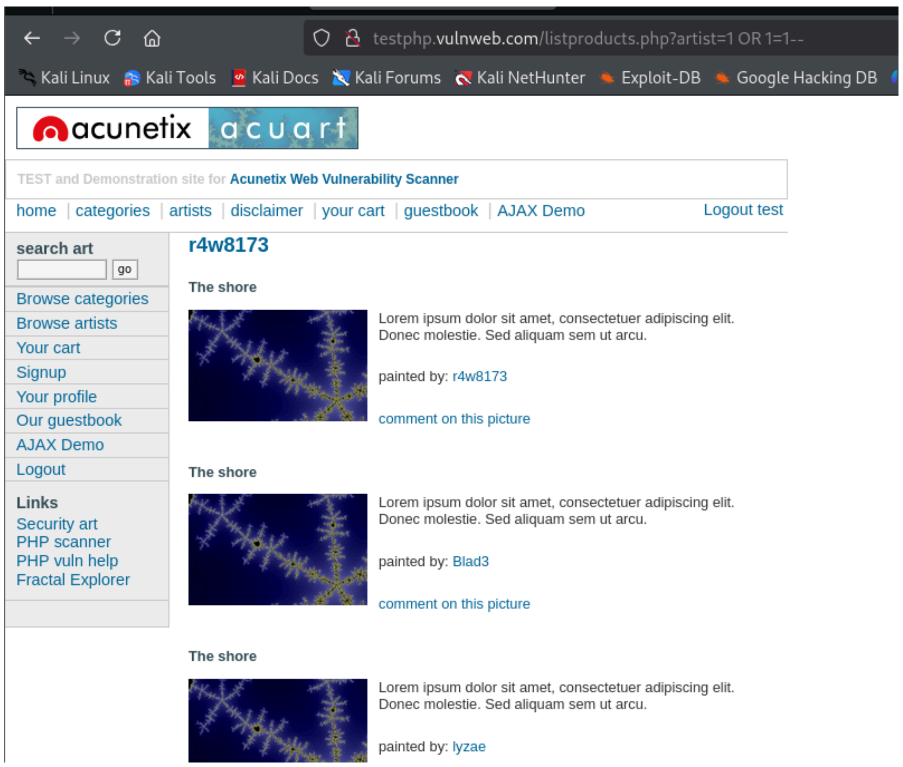
Description: Input fields in the SQLi module fail to sanitize user input in url.

Impact: Attackers can extract or modify database contents.

Proof of Concept: http://testphp.vulnweb.com/listproducts.php?artist=1 OR 1=1--

Evidence: OWASP ZAP flagged 'SQL Injection' on parameter id in GET requests.

Mitigation: Use prepared statements and parameterized queries and validate user inputs.


testphp.vulnweb.com/listproducts.php?artist=1 OR 1=1--

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

acunetix acuart

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo Logout test

search art
go

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

r4w8173

The shore

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu.

painted by: r4w8173

comment on this picture

The shore

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu.

painted by: Blad3

comment on this picture

The shore

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu.

painted by: lyzae

Manual injection in URL to show all the artists from database.

3.2 DOM Based XSS (High)

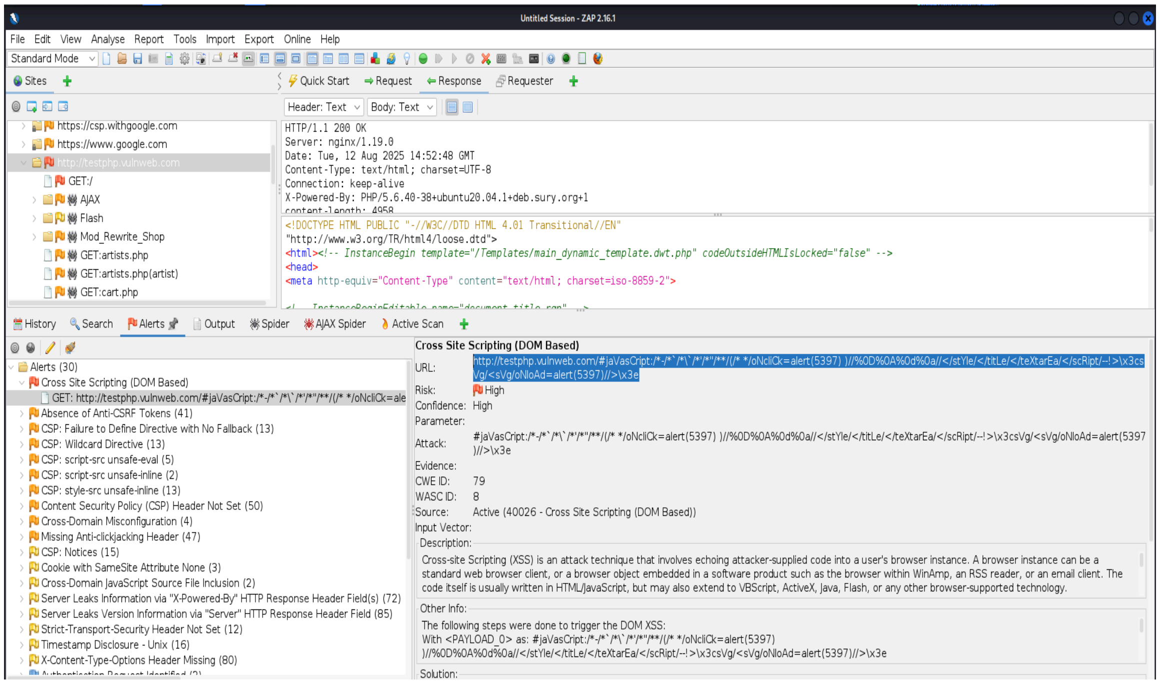
Description: Search input reflects unsanitized user input directly into HTML output.

Impact: Allows execution of arbitrary JavaScript in victim browsers.

Proof of Concept: http://testphp.vulnweb.com/#jaVasCript:/*-/*`/*\`/*'/*"/**/(/*
*/oNcliCk=alert(5397)
)//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=a
lert(5397)//>\x3e

Evidence: Payload executed successfully in browser.

Mitigation: Encode all output, validate input, implement CSP.



```
https://csp.withgoogle.com
https://www.google.com
http://testphp.vulnweb.com
    GET:/
    AJAX
    Flash
    Mod_Rewrite_Shop
    GET:artists.php
    GET:artists.php(artist)
    GET:cart.php
```

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Tue, 12 Aug 2025 14:52:48 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
content-length: 4958
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><!-- InstanceBegin template="/Templates/main_dynamic_template.dwt.php" codeOutsideHTMLIsLocked="false" -->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

<!-- InstanceBeginEditable name="document_title_rgn" -->
```

```
Alerts (30)
  Cross Site Scripting (DOM Based)
    GET: http://testphp.vulnweb.com/#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=ale
  Absence of Anti-CSRF Tokens (41)
  CSP: Failure to Define Directive with No Fallback (13)
  CSP: Wildcard Directive (13)
  CSP: script-src unsafe-eval (5)
  CSP: script-src unsafe-inline (2)
  CSP: style-src unsafe-inline (13)
  Content Security Policy (CSP) Header Not Set (50)
  Cross-Domain Misconfiguration (4)
  Missing Anti-clickjacking Header (47)
  CSP: Notices (15)
  Cookie with SameSite Attribute None (3)
  Cross-Domain JavaScript Source File Inclusion (2)
  Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (72)
  Server Leaks Version Information via "Server" HTTP Response Header Field (85)
  Strict-Transport-Security Header Not Set (12)
  Timestamp Disclosure - Unix (16)
  X-Content-Type-Options Header Missing (80)
  Authentication Request Identified (2)
```

Cross Site Scripting (DOM Based)

```
URL:         http://testphp.vulnweb.com/#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397) )//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3cs
             Vg/<sVg/oNloAd=alert(5397)//>\x3e
Risk:        High
Confidence:  High
Parameter:
Attack:      #jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397) )//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397
             )//>\x3e
Evidence:
CWE ID:      79
WASC ID:     8
Source:      Active (40026 - Cross Site Scripting (DOM Based))
Input Vector:
Description:
  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.
Other Info:
  The following steps were done to trigger the DOM XSS:
  With <PAYLOAD_0> as: #jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397)
  )//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Solution:
```

4. Recommendations

- Enforce input validation and output encoding.
- Implement prepared statements for all database queries.
- Introduce CSRF protection mechanisms.
- Limit error message verbosity.
- Remove or obfuscate server version headers.