Hackers10 Internship Task 1: Web Application Security Testing Report

Business Confidential

R.A.M.P Ranabahu
BICT (Hons)
University of Kelaniya
Date: May 4th, 2025
Track Code: H10

Table of Contents

Confidentiality Statement

Disclaimer

This report presents the results of the web application security assessment conducted as part of an educational internship project for Hackers10. The findings and recommendations reflect the security state of the target website at the time of the assessment and are based on the scope and objectives defined in the internship task list. Due to the time-limited and educational nature of the internship, not all systems or security controls may have been fully evaluated. The assessment focused on identifying specific vulnerabilities and demonstrating practical skills as required by the internship. This report does not constitute a comprehensive security evaluation, and Hackers10 recommends that regular, professional security assessments be conducted to maintain and improve overall security posture.

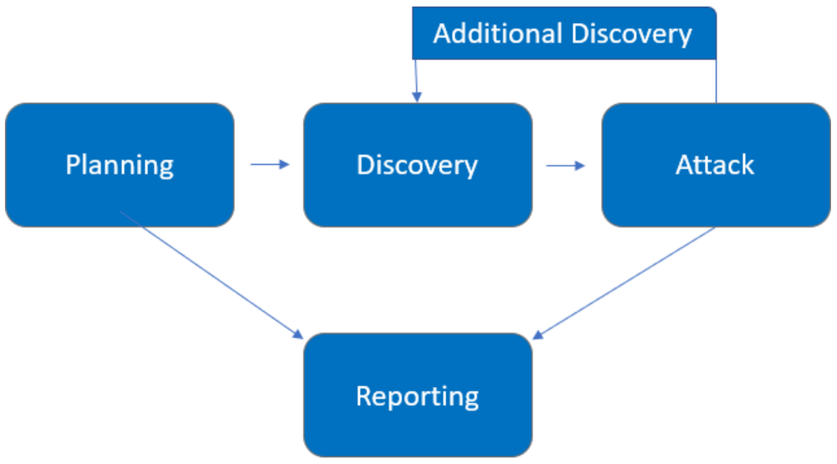
Assessment Overview

Between April 8th, 2025, and May 8th, 2025, a security assessment was conducted as part of the Hackers10 Cyber Security Internship program. The objective was to evaluate the security posture of the assigned vulnerable website through practical penetration testing exercises, following industry-recognized methodologies and the specific tasks outlined in the internship assignment.

The assessment process consisted of the following phases:

- **Planning:** Defined the scope of the assessment based on internship guidelines and established rules of engagement for testing the target website.
- **Discovery:** Conducted reconnaissance activities such as port scanning and directory brute-forcing to identify open ports, exposed directories, and potential vulnerabilities.
- **Attack:** Attempted exploitation of discovered vulnerabilities, including testing for SQL injection, capturing credentials via network interception, and injecting malicious scripts to check for XSS vulnerabilities.
- **Reporting:** Documented all identified vulnerabilities, successful and unsuccessful exploitation attempts, and provided recommendations for remediation, as well as observations on the strengths and weaknesses of the website's security posture.

All testing activities were performed in accordance with ethical hacking standards and were strictly limited to the scope and objectives defined by the Hackers10 internship task list.



Executive Summary

A security assessment was conducted on the website http://testphp.vulnweb.com/ as part of the Hackers10 Cyber Security Internship program, from April 8th, 2025, to May 8th, 2025. The objective was to identify and demonstrate real-world web application vulnerabilities using a series of practical attack techniques defined in the internship task list. This summary outlines the vulnerabilities discovered, the effectiveness of the attacks, and observations regarding the website's security strengths and weaknesses.

Testing Summary

The assessment followed a structured approach, focusing on common web application attack vectors:

- **Port Scanning:** Identified all open ports and running services on the target website, providing insight into the website's exposed attack surface and potential entry points.
- **Directory Brute Forcing:** Enumerated hidden directories and files, uncovering additional resources that could be susceptible to attack.
- **Network Traffic Interception:** Captured login activity and analyzed network traffic to determine if credentials were transmitted in plaintext, revealing potential weaknesses in transport layer security.
- **SQL Injection:** Tested input fields for SQL injection vulnerabilities using manual payloads and automated tools. Successful exploitation could allow unauthorized access to sensitive database information.
- **Cross-Site Scripting (XSS):** Tested input fields for the ability to inject and execute malicious JavaScript payloads, assessing the website's resilience against both stored and reflected XSS attacks.

Key Findings

- Several open ports and services were discovered, some of which may not be necessary for public exposure, increasing the attack surface.
- Hidden directories and files were identified, potentially exposing sensitive resources or administrative interfaces.
- Login credentials were observed in network traffic, indicating a lack of proper encryption and exposing users to credential theft.
- The website was found to be vulnerable to SQL injection, allowing extraction of database information and demonstrating insufficient input validation.
- Successful injection and execution of JavaScript payloads confirmed the presence of XSS vulnerabilities, which could be exploited for session hijacking or defacement.

Conclusion

The assessment revealed multiple critical vulnerabilities, including exposed services, insecure credential transmission, SQL injection, and XSS. These findings highlight the need for improved input validation, stronger encryption practices, and regular security testing. Remediation of these issues is essential to enhance the website's security posture and protect users from potential exploitation.

Note: All testing activities were performed with explicit authorization and strictly within the boundaries of the internship assignment. No denial of service or social engineering attacks were conducted.

Detailed Findings / Vulnerability Details

Below, each vulnerability is documented according to the specific hacking steps completed during the Hackers10 Cyber Security Internship, Task 1: Web Application Security Testing. Each finding is structured to include the title, description, affected asset, steps to reproduce (with tools and commands), evidence, severity and risk rating, business/technical impact, remediation recommendations, and status. This format follows industry best practices and Hackers10's requirements for clear, actionable reporting.

1. Open Ports Discovery

Title & Description:
Open Ports Identified on testphp.vulnweb.com
A port scan was conducted on the target website to identify exposed network services. The scan revealed that two TCP ports are open: port 25 (SMTP) and port 80 (HTTP). The presence of these ports indicates publicly accessible mail and web services.

Affected Asset/Component:
Server at testphp.vulnweb.com (IP: 44.228.249.3)

Steps to Reproduce:

- The following Nmap command was executed to perform a fast scan of the most common ports:

  ```
  nmap -F testphp.vulnweb.com
  ```

- Nmap reported the following results:

  ```
  25/tcp open smtp
  80/tcp open http
  ```

- Evidence:

  Nmap output:

  - PORT   STATE SERVICE
  - 25/tcp open smtp
  - 80/tcp open http

  - Host IP: 44.228.249.3
  - rDNS: ec2-44-228-249-3.us-west-2.compute.amazonaws.com

Severity Rating:
Medium

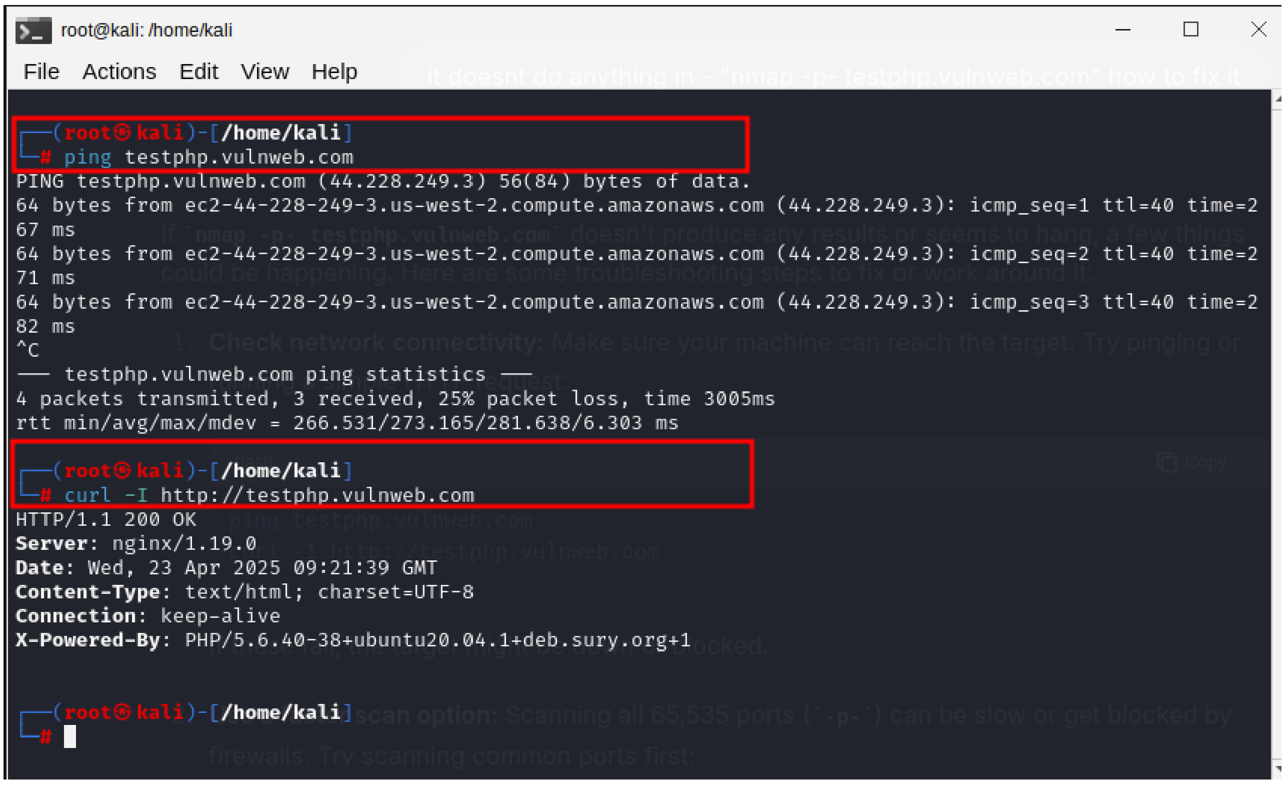
Business/Technical Impact:

- The HTTP service (port 80) is expected for web access, but running SMTP (port 25) on a public-facing server can increase the risk of spam relay abuse or exploitation of mail service vulnerabilities.

- Exposed services may be targeted for further attacks, such as brute-force attempts or exploitation of known vulnerabilities.

Remediation Recommendations:

- Review the necessity of running the SMTP service on the public interface. If not required, restrict access or disable the service.
- Ensure all publicly accessible services are fully patched and configured securely.
- Regularly monitor and audit open ports to minimize the attack surface

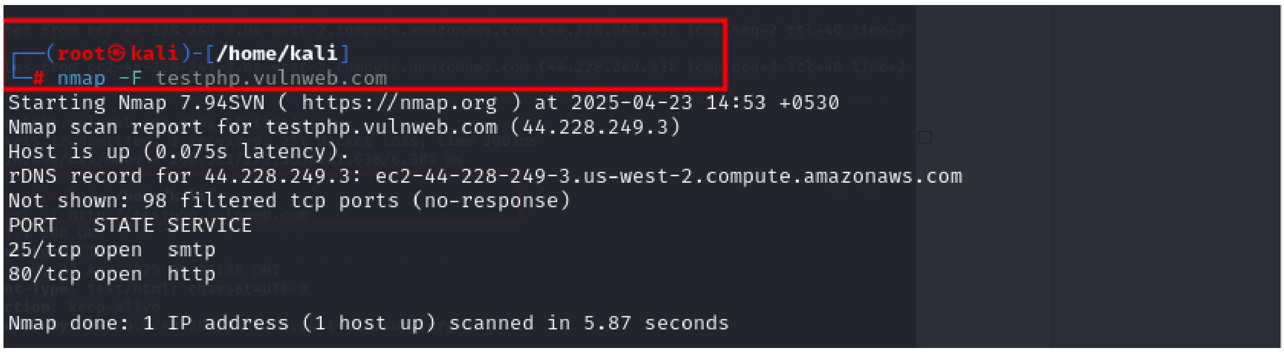


```
┌──(root㉿kali)-[/home/kali]
└─# ping testphp.vulnweb.com
PING testphp.vulnweb.com (44.228.249.3) 56(84) bytes of data.
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3): icmp_seq=1 ttl=40 time=2
67 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3): icmp_seq=2 ttl=40 time=2
71 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3): icmp_seq=3 ttl=40 time=2
82 ms
^C
── testphp.vulnweb.com ping statistics ──
4 packets transmitted, 3 received, 25% packet loss, time 3005ms
rtt min/avg/max/mdev = 266.531/273.165/281.638/6.303 ms

┌──(root㉿kali)-[/home/kali]
└─# curl -I http://testphp.vulnweb.com
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Wed, 23 Apr 2025 09:21:39 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


┌──(root㉿kali)-[/home/kali]
└─# 
```

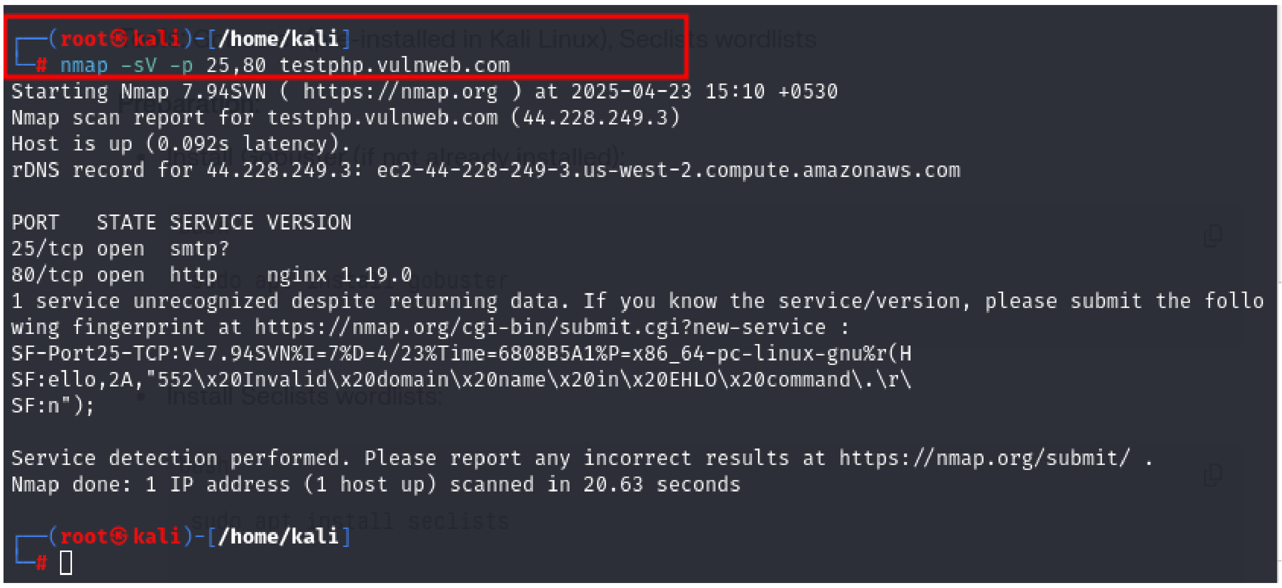
Figure 1:curl



```
┌──(root㉿kali)-[/home/kali]
└─# nmap -F testphp.vulnweb.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-23 14:53 +0530
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.075s latency).
rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com
Not shown: 98 filtered tcp ports (no-response)
PORT   STATE SERVICE
25/tcp open  smtp
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 5.87 seconds
```

Figure 2:port scan



```
┌──(root㉿kali)-[/home/kali]
└─# nmap -sV -p 25,80 testphp.vulnweb.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-23 15:10 +0530
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.092s latency).
rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com

PORT   STATE SERVICE VERSION
25/tcp open  smtp?
80/tcp open  http    nginx 1.19.0
1 service unrecognized despite returning data. If you know the service/version, please submit the follo
wing fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port25-TCP:V=7.94SVN%I=7%D=4/23%Time=6808B5A1%P=x86_64-pc-linux-gnu%r(H
SF:ello,2A,"552\x20Invalid\x20domain\x20name\x20in\x20EHLO\x20command\.\r\
SF:n");

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.63 seconds

┌──(root㉿kali)-[/home/kali]
└─# 
```

Figure 3

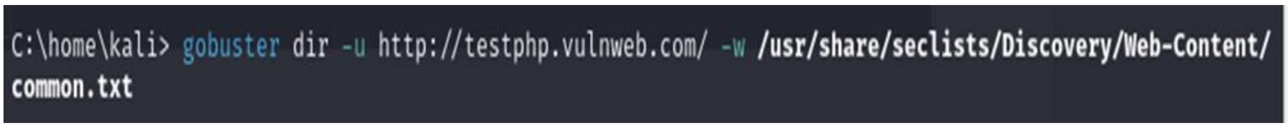
2. Brute Force Directories

Objective:

Discover hidden or sensitive directories and files on the web server http://testphp.vulnweb.com/.

Steps Taken:

- Used Gobuster, a directory brute-forcing tool available on Kali Linux, with a common wordlist to enumerate directories and files.
- Command executed:



```
C:\home\kali> gobuster dir -u http://testphp.vulnweb.com/ -w /usr/share/seclists/Discovery/Web-Content/
common.txt
```

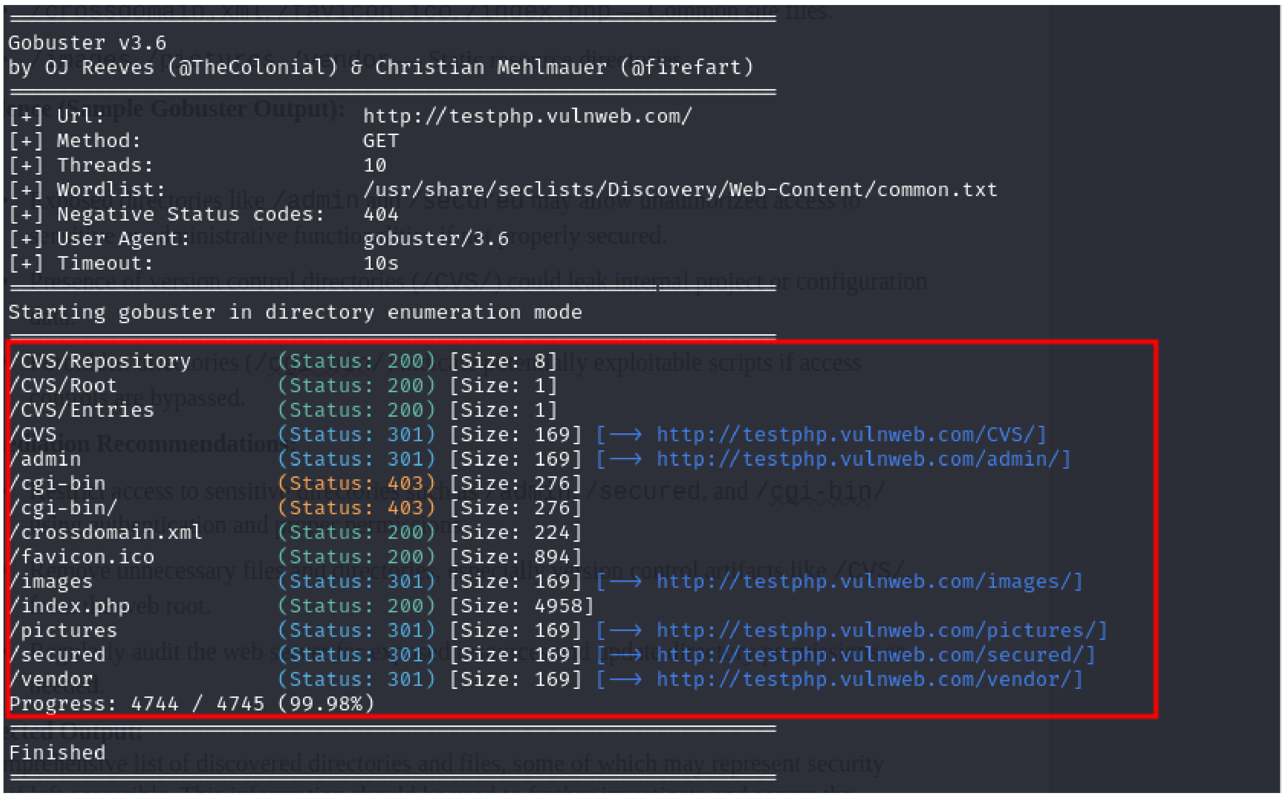
Figure 4:bruteforce

Output Summary:
Gobuster successfully discovered several directories and files that may not be directly linked from the main site, including potential administrative and sensitive resources.

Key Findings:

- `/admin` (Status: 301) — Possible administrative interface.
- `/secured` (Status: 301) — May contain protected resources.
- `/CVS/`, `/CVS/Repository`, `/CVS/Root`, `/CVS/Entries` (Status: 200/301) — Version control files and directories, which could expose internal information.
- `/cgi-bin/` (Status: 403) — Directory exists but access is forbidden; may contain scripts.
- `/crossdomain.xml`, `/favicon.ico`, `/index.php` — Common site files.
- `/images`, `/pictures`, `/vendor` — Static resource directories.

Evidence (Sample Gobuster Output):



```
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://testphp.vulnweb.com/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/seclists/Discovery/Web-Content/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/CVS/Repository      (Status: 200) [Size: 8]
/CVS/Root            (Status: 200) [Size: 1]
/CVS/Entries         (Status: 200) [Size: 1]
/CVS                 (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/CVS/]
/admin               (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/admin/]
/cgi-bin             (Status: 403) [Size: 276]
/cgi-bin/            (Status: 403) [Size: 276]
/crossdomain.xml     (Status: 200) [Size: 224]
/favicon.ico         (Status: 200) [Size: 894]
/images              (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/images/]
/index.php           (Status: 200) [Size: 4958]
/pictures            (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/pictures/]
/secured             (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/secured/]
/vendor              (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/vendor/]
Progress: 4744 / 4745 (99.98%)
===============================================================
Finished
===============================================================
```

Figure 5

- Exposed directories like `/admin` and `/secured` may allow unauthorized access to sensitive or administrative functionalities if not properly secured.
- Presence of version control directories (`/CVS/`) could leak internal project or configuration data.
- Forbidden directories (`/cgi-bin/`) indicate potentially exploitable scripts if access controls are bypassed.

Remediation Recommendations:

- Restrict access to sensitive directories such as `/admin`, `/secured`, and `/cgi-bin/` using authentication and proper permissions.
- Remove unnecessary files and directories, especially version control artifacts like `/CVS/`, from the web root.

Expected Output:
A comprehensive list of discovered directories and files, some of which may represent security risks if left accessible. This information should be used to further investigate and secure the application.

3. Cleartext Credential Transmission (Wireshark Interception)

Credentials Transmitted in Plaintext Over HTTP
During the login process on http://testphp.vulnweb.com/, user credentials are sent over an unencrypted HTTP connection, making them vulnerable to interception by attackers monitoring network traffic.

Affected Asset/Component:
Login functionality of http://testphp.vulnweb.com/ (specifically, POST requests to `/userinfo.php`)

Steps to Reproduce:

- Started Wireshark on the Kali Linux system and began capturing traffic on the active network interface.
- Navigated to the login page and submitted the credentials:
    - Username: `test`
    - Password: `test`
- Stopped the Wireshark capture after logging in.

- Applied the display filter `http.request.method == "POST"` in Wireshark to locate the login request.
- Examined the packet details and observed the credentials in plaintext within the HTTP POST data.

Evidence:

- Wireshark capture shows a POST request to `/userinfo.php` with the following form data:
    - `uname=test`
    - `pass=test`
- Credentials are clearly visible in the packet details (see attached screenshot).

Severity Rating:
High

Business/Technical Impact:

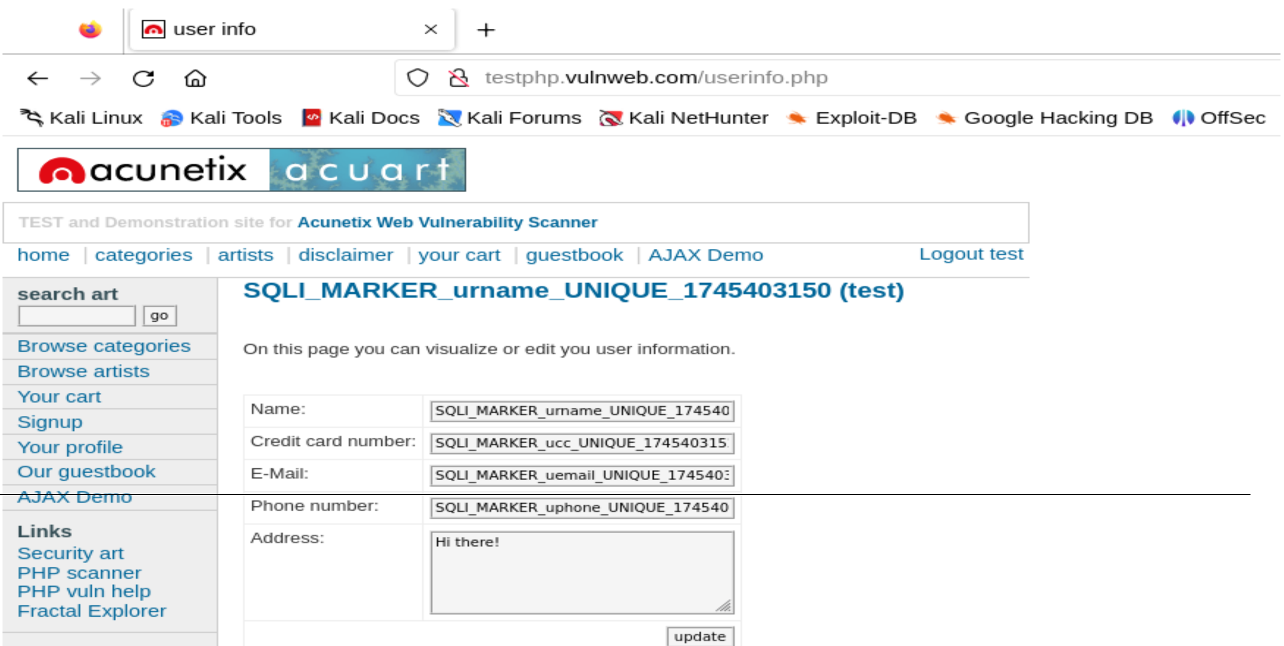
- Any attacker with access to the network can easily intercept user credentials, leading to unauthorized account access and potential data breaches.
- This vulnerability exposes all users of the website to credential theft and further exploitation.

Remediation Recommendations:

- Implement HTTPS (SSL/TLS) for all pages, especially those handling sensitive data like login credentials.
- Redirect all HTTP traffic to HTTPS to ensure data is always encrypted in transit.
- Educate users to avoid submitting credentials on websites that do not use HTTPS.

Screenshot Reference:

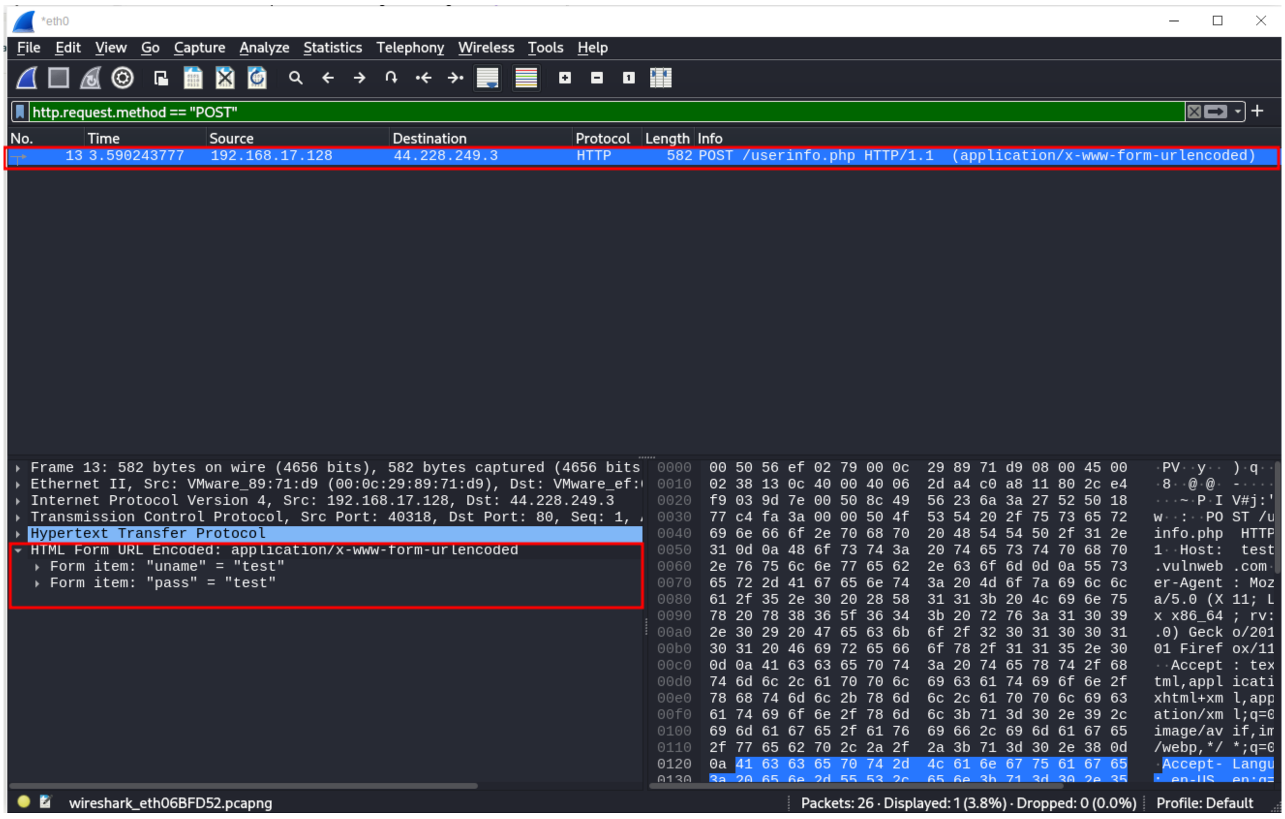
- The attached Wireshark screenshot clearly displays the intercepted credentials in plaintext during the login process.



SQLI_MARKER_urname_UNIQUE_1745403150 (test)

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

Name:	SQLI_MARKER_urname_UNIQUE_174540
Credit card number:	SQLI_MARKER_ucc_UNIQUE_174540315
E-Mail:	SQLI_MARKER_uemail_UNIQUE_174540:
Phone number:	SQLI_MARKER_uphone_UNIQUE_174540
Address:	Hi there!


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
http.request.method == "POST"

No.    Time          Source            Destination     Protocol  Length  Info
    13 3.590243777   192.168.17.128    44.228.249.3    HTTP         582  POST /userinfo.php HTTP/1.1  (application/x-www-form-urlencoded)
```

```
› Frame 13: 582 bytes on wire (4656 bits), 582 bytes captured (4656 bits
› Ethernet II, Src: VMware_89:71:d9 (00:0c:29:89:71:d9), Dst: VMware_ef:
› Internet Protocol Version 4, Src: 192.168.17.128, Dst: 44.228.249.3
› Transmission Control Protocol, Src Port: 40318, Dst Port: 80, Seq: 1,
› Hypertext Transfer Protocol
⌄ HTML Form URL Encoded: application/x-www-form-urlencoded
  › Form item: "uname" = "test"
  › Form item: "pass" = "test"
```

```
0000  00 50 56 ef 02 79 00 0c  29 89 71 d9 08 00 45 00   ·PV··y·· )·q···
0010  02 38 13 0c 40 00 40 06  2d a4 c0 a8 11 80 2c e4   ·8··@·@· -·····
0020  f9 03 9d 7e 00 50 8c 49  56 23 6a 3a 27 52 50 18   ···~·P·I V#j:'
0030  77 c4 fa 3a 00 00 50 4f  53 54 20 2f 75 73 65 72   w··:··PO ST /u
0040  69 6e 66 6f 2e 70 68 70  20 48 54 54 50 2f 31 2e   info.php  HTTP
0050  31 0d 0a 48 6f 73 74 3a  20 74 65 73 74 70 68 70   1··Host:  test
0060  2e 76 75 6c 6e 77 65 62  2e 63 6f 6d 0d 0a 55 73   .vulnweb .com·
0070  65 72 2d 41 67 65 6e 74  3a 20 4d 6f 7a 69 6c 6c   er-Agent : Moz
0080  61 2f 35 2e 30 20 28 58  31 31 3b 20 4c 69 6e 75   a/5.0 (X 11; L
0090  78 20 78 38 36 5f 36 34  3b 20 72 76 3a 31 30 39   x x86_64 ; rv:
00a0  2e 30 29 20 47 65 63 6b  6f 2f 32 30 31 30 30 31   .0) Geck o/201
00b0  30 31 20 46 69 72 65 66  6f 78 2f 31 31 35 2e 30   01 Firef ox/11
00c0  0d 0a 41 63 63 65 70 74  3a 20 74 65 78 74 2f 68   ··Accept : tex
00d0  74 6d 6c 2c 61 70 70 6c  69 63 61 74 69 6f 6e 2f   tml,appl icati
00e0  78 68 74 6d 6c 2b 78 6d  6c 2c 61 70 70 6c 69 63   xhtml+xm l,app
00f0  61 74 69 6f 6e 2f 78 6d  6c 3b 71 3d 30 2e 39 2c   ation/xm l;q=0
0100  69 6d 61 67 65 2f 61 76  69 66 2c 69 6d 61 67 65   image/av if,im
0110  2f 77 65 62 70 2c 2a 2f  2a 3b 71 3d 30 2e 38 0d   /webp,*/ *;q=0
0120  0a 41 63 63 65 70 74 2d  4c 61 6e 67 75 61 67 65   ·Accept- Langu
0130  3a 20 65 6e 2d 55 53 2c  65 6e 3b 71 3d 30 2e 35   : en-US, en;q=
```

```
wireshark_eth06BFD52.pcapng    Packets: 26 · Displayed: 1 (3.8%) · Dropped: 0 (0.0%)    Profile: Default
```

Figure 7:Wireshark_Cap

4.SQL Injection Vulnerability Assessment

SQL Injection in Product Listing (`cat` Parameter)
A critical SQL injection vulnerability was identified in the `cat` parameter of the `listproducts.php` page on http://testphp.vulnweb.com/. Automated testing with sqlmap confirmed that this parameter is susceptible to multiple SQL injection techniques, allowing an attacker to enumerate databases, extract sensitive information, and potentially compromise the entire backend database.

Affected Component:

- URL: `http://testphp.vulnweb.com/listproducts.php?cat=1`
- Parameter: `cat` (GET)

Steps to Reproduce:

1. **Initial SQL Injection Detection:**
   Ran sqlmap to test the `cat` parameter:

```
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --batch
```

- sqlmap detected the following injection types:
  - Boolean-based blind
  - Error-based
  - Time-based blind
  - UNION query-based
- **Database Enumeration:**
  Enumerated available databases:

```
    sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbs --batch
```

- Discovered databases:
  - `acuart`
  - `information_schema`
- **Table and Column Enumeration:**
  Listed tables in the `acuart` database:

```
  sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart --tables –batch
```

- Tables found: `artists`, `carts`, `categ`, `featured`, `guestbook`, `pictures`, `products`, `users`

Listed columns in the `users` table:

```
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart -T users --columns --batch
```

- Columns found: `name`, `address`, `cart`, `cc`, `email`, `pass`, `phone`, `uname`
- **Data Extraction:**
  Extracted user data from the `users` table:

```
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart -T users --dump --batch
```

- Example data extracted:

cc	cart	pass	email	phone	uname	name	address
1234-5678-2300-9000	2d0789623d3e62323fca31c4b51cdde8	test	email@email.com	2323345	test	John Smith	21 street - cyber TPDAY

Evidence:

- sqlmap output confirmed multiple SQL injection vectors and successful data extraction from the backend MySQL database.
- Sensitive information such as usernames, hashed passwords, email addresses, phone numbers, and credit card numbers were retrieved from the `users` table.

Severity Rating:
Critical

Business/Technical Impact:

- Attackers can extract, modify, or delete sensitive data from the backend database.
- Exposure of user credentials and personal information, including credit card numbers, could lead to identity theft, financial fraud, and regulatory penalties.
- Full compromise of the application's data integrity and confidentiality.

Remediation Recommendations:

- Implement parameterized queries or prepared statements for all database interactions.
- Validate and sanitize all user-supplied input.
- Employ web application firewalls (WAF) to detect and block SQL injection attempts.
- Regularly test and audit the application for injection vulnerabilities.

Conclusion:
The presence of a critical SQL injection vulnerability in a core application parameter demonstrates a severe weakness in input handling and backend security. Immediate remediation and a thorough security review are strongly recommended.

5.Cross-Site Scripting (XSS) Vulnerability Assessment

Stored and Reflected XSS in User Input Fields
Testing on http://testphp.vulnweb.com revealed that multiple input fields—including the guestbook comment section and search box—are vulnerable to Cross-Site Scripting (XSS) attacks. By injecting malicious JavaScript payloads, it was possible to execute arbitrary scripts in the browser of any user viewing the affected pages, confirming both stored and reflected XSS vulnerabilities.

Affected Components:

- Guestbook comment/message fields (`/guestbook.php`)
- Search box and other user-input fields

Steps to Reproduce:

1. **Stored XSS (Guestbook):**
    - Navigated to the guestbook page.
    - Submitted the following payload in the "Message" field:

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

- After submission, reloaded the guestbook page and observed a JavaScript alert pop-up, confirming the script was stored and executed for all users viewing the page.
- **Reflected XSS (Search Box):**
- Entered the following payload in the website’s search box:

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

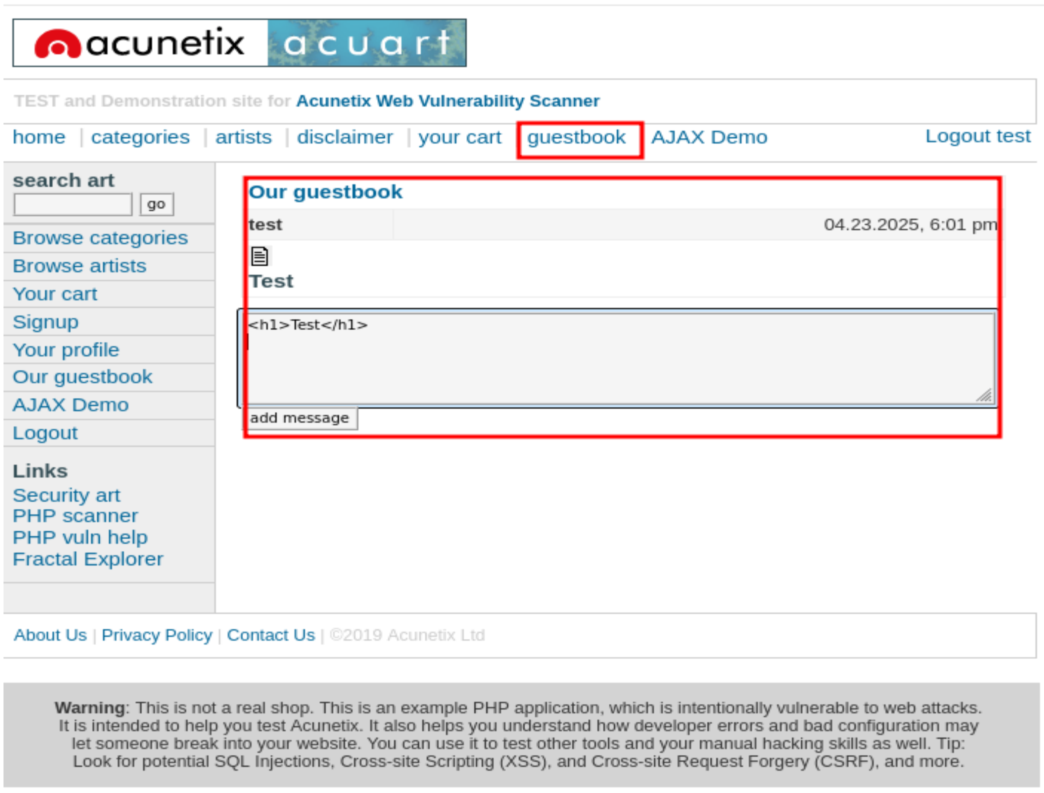
- Upon submitting the search, the alert box appeared immediately, indicating that the input was reflected and executed without proper sanitization.

Example Payloads Used:

Payload	Type	Description
`<script>alert('XSS')</script>`	Basic	Pops up an alert box
`<img src=x onerror=alert('XSS')>`	Evasive	Pops alert if image fails to load
`<svg/onload=alert('XSS')>`	Evasive	Pops alert on SVG load
`<a onmouseover="alert('XSS')">Test</a>`	Event-based	Pops alert on mouse hover

Evidence:

- Screenshots of alert pop-ups triggered by the injected payloads in both the guestbook and search functionalities.
- The payload appears unfiltered in the page source, confirming lack of input validation and output encoding.


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

Our guestbook

test   04.23.2025, 6:01 pm

Test

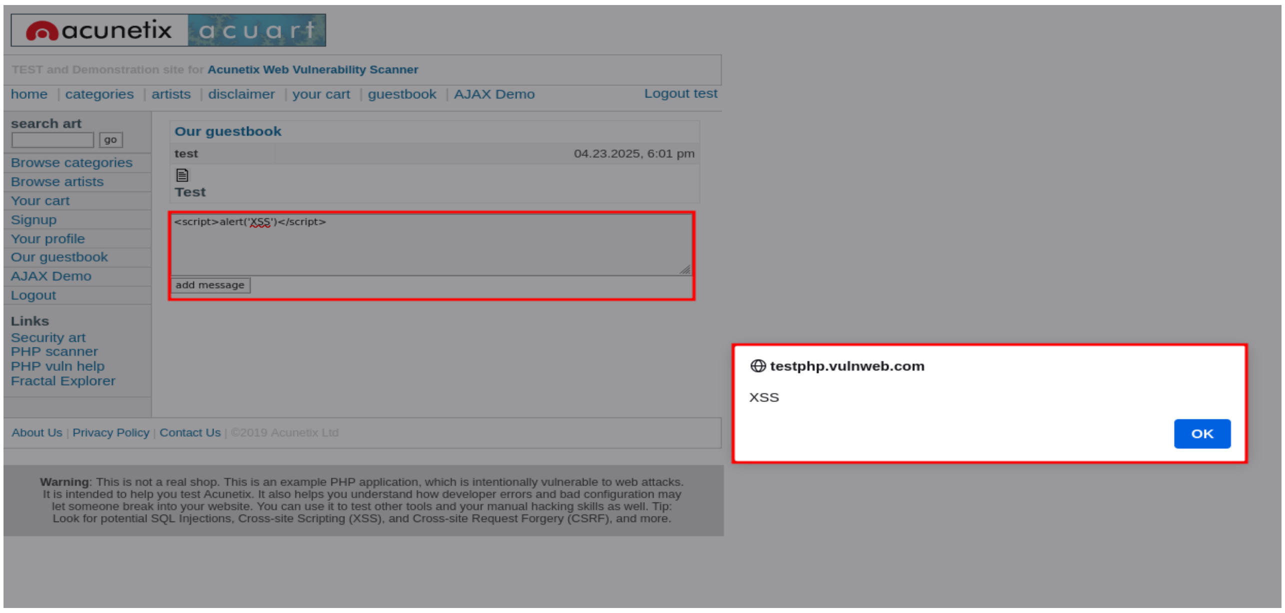
`<h1>Test</h1>`

add message

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.

Figure 8:guestbook_check


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
add message
About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd
OK

Our guestbook

test 04.23.2025, 6:01 pm

Test

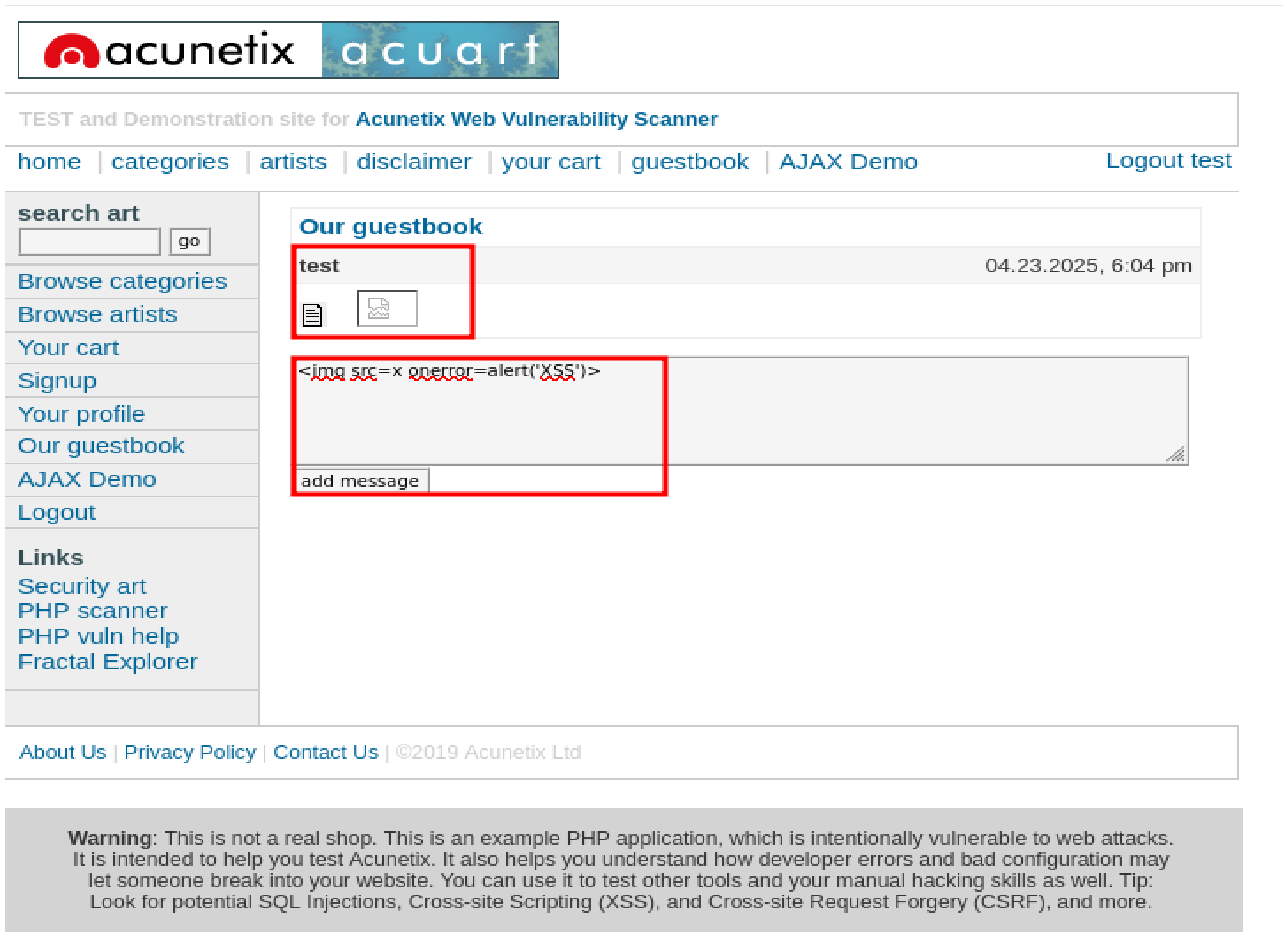
```
<script>alert('XSS')</script>
```

testphp.vulnweb.com

XSS

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:alert_xss


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
add message
About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

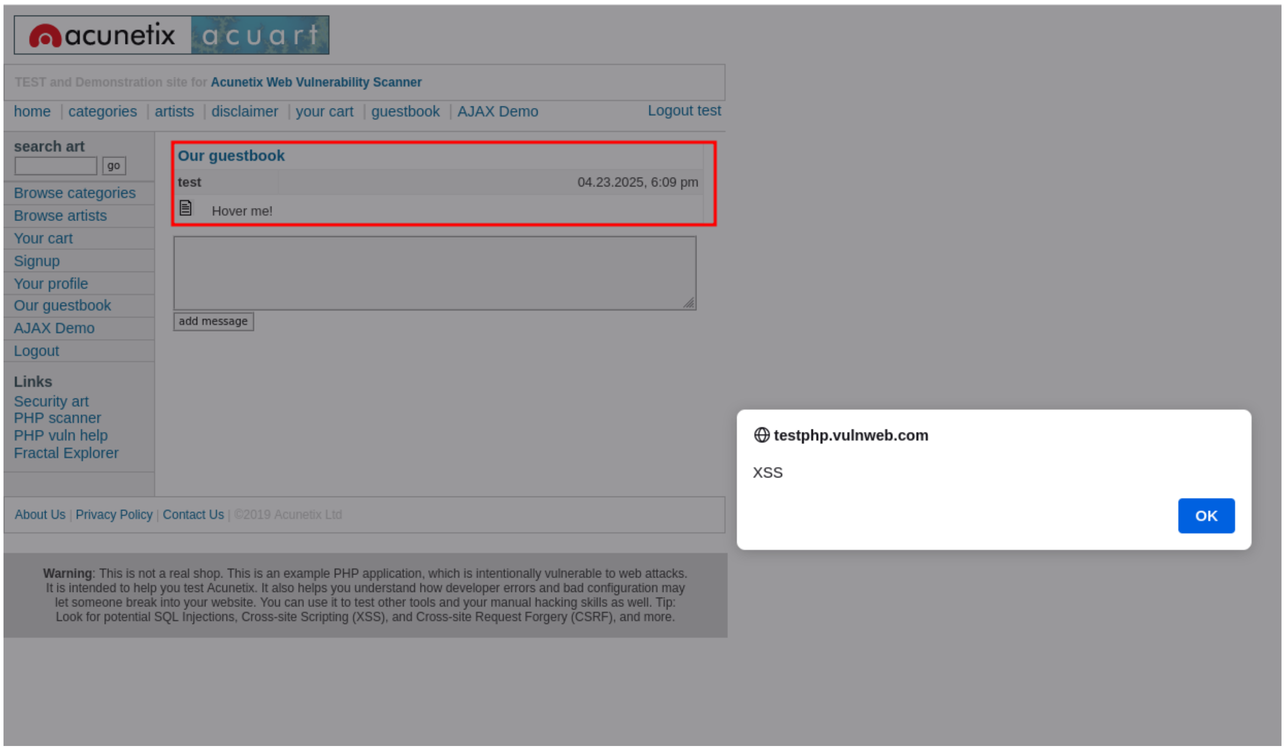
Our guestbook

test 04.23.2025, 6:04 pm

```
<img src=x onerror=alert('XSS')>
```

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 10:xss_img



Our guestbook

test 04.23.2025, 6:09 pm

Hover me!

testphp.vulnweb.com

XSS

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.

Figure 11:xss_hover

Severity Rating:
High

Business/Technical Impact:

- Attackers can execute arbitrary JavaScript in the browsers of site visitors, leading to session hijacking, credential theft, phishing, defacement, or further attacks.
- Stored XSS impacts all users viewing the affected page, amplifying the risk.

Remediation Recommendations:

- Implement robust input validation and output encoding for all user-supplied data.
- Sanitize inputs on both client and server sides.
- Employ Content Security Policy (CSP) headers to restrict script execution.
- Regularly test and audit the application for XSS vulnerabilities.

Conclusion:
The presence of both stored and reflected XSS vulnerabilities exposes users and the application to significant security risks. Immediate remediation is necessary to protect user data and maintain the integrity of the website.

Conclusion:

This penetration test of http://testphp.vulnweb.com identified several critical vulnerabilities, including SQL injection and cross-site scripting (XSS), which could allow attackers to compromise sensitive data and execute malicious code in users' browsers. The SQL injection vulnerability in the `cat` parameter of the `listproducts.php` page enabled full database enumeration and data extraction, as demonstrated by the use of sqlmap. Additionally, multiple input fields were found to be susceptible to stored and reflected XSS attacks, confirming a lack of input validation and output encoding.

The findings highlight the urgent need for remediation to protect user data, maintain application integrity, and comply with security best practices.

Recommendations:

- **Remediate Critical Vulnerabilities:**
  - Implement parameterized queries and prepared statements throughout the application to prevent SQL injection.
  - Sanitize and validate all user inputs on both client and server sides.
  - Employ output encoding and Content Security Policy (CSP) headers to mitigate XSS.
  - Remove sensitive information from error messages and HTTP headers.
- **Enforce Secure Communication:**
  - Use HTTPS for all data transmission, especially for login and sensitive operations.
  - Set secure cookie attributes (`Secure`, `HttpOnly`) to prevent session hijacking.
- **Regular Security Assessments:**
  - Schedule periodic penetration tests and code reviews to proactively identify and address vulnerabilities.
  - Keep all software components and dependencies up to date.
- **User Awareness & Incident Response:**
  - Educate users about secure password practices and phishing risks.
  - Develop and test an incident response plan for potential breaches.

Next Steps:

- Address the critical vulnerabilities identified in this report as a priority.

- Conduct a retest after remediation to confirm all issues have been resolved.
- Integrate secure development practices and regular security testing into the SDLC.

Appendices:

- **A. Tool Output and Logs:**
    - Attach relevant sqlmap logs, screenshots, and evidence of successful exploitation as supporting documentation.
- **B. Risk Rating Tables:**
    - Provide CVSS scores or a summary of risk ratings for each finding.
- **C. Methodology:**
    - Briefly describe the testing methodology, tools used (e.g., Kali Linux, sqlmap, Gobuster, Wireshark), and scope of assessment.
- **D. Glossary and References:**
    - Define technical terms and provide references to remediation best practices or official documentation.

Final Note:
This report is intended to provide actionable insights for improving the security posture of the tested application. Prompt remediation and ongoing vigilance are essential to reduce risk and protect both users and organizational assets.