Penetration Testing Report of VULNWEB

Testphp.vulnweb.com

Prepared By : Naradasu Ravitejaroyal

Contact : naradasuraviteja235@gmail.com

Date : 23/01/2026

Table of Contents

1. Executive Summary

A web application vulnerability assessment and penetration test was conducted on **http://testphp.vulnweb.com**, an intentionally vulnerable application provided by Acunetix for security research and training. The assessment was performed using Kali Linux with standard penetration testing tools and manual verification techniques. The objective was to identify security weaknesses, validate exploitable vulnerabilities, and understand real-world web attack scenarios.

The testing methodology followed industry practices aligned with the OWASP Top 10 framework and included reconnaissance, scanning, enumeration, vulnerability identification, exploitation, and reporting. During reconnaissance, tools such as WhatWeb and Nmap were used to identify server technologies and services. The application was found to run on an Nginx server with a PHP backend, exposing version details and indicating outdated software and information leakage. Automated scanning was carried out using Nikto, Gobuster, and Nuclei, while manual testing and request interception were performed using Burp Suite.

The assessment confirmed several critical and high-risk vulnerabilities. Major findings include SQL Injection enabling authentication bypass and database extraction, and Cross-Site Scripting (stored and reflected) allowing execution of malicious scripts in user browsers. Additional issues include sensitive information disclosure, exposed internal directories, insecure HTTP communication, weak password policy, and missing security headers. All key vulnerabilities were validated through proof-of-concept exploitation.

Overall, the application demonstrates a poor security posture with multiple exploitable weaknesses that could lead to unauthorized access, data theft, and website manipulation if deployed in a production environment. Implementing secure coding practices, strong authentication controls, HTTPS enforcement, proper input validation, and security hardening measures is essential to improve the application's security.

This assessment provided practical experience in vulnerability scanning and exploitation using Kali Linux, reinforcing real-world web security testing methodologies.

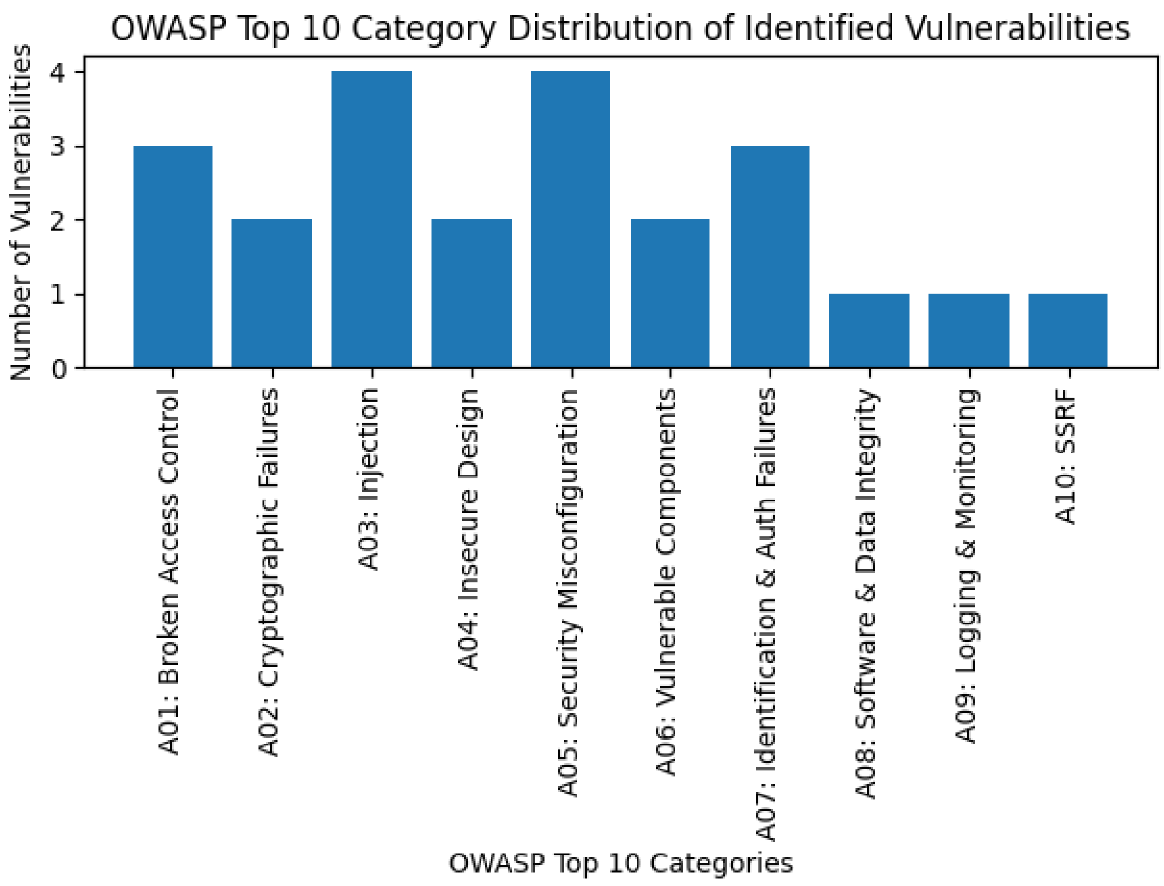
1.1 OWASP category-wise Vulnerability Analysis

To understand the security posture of the target web application, all identified vulnerabilities were mapped to the OWASP Top 10 risk categories. This classification helps in understanding which type of security weaknesses are most dominant in the application.

The analysis was performed based on:

- Automated scan results from WhatWeb, Nmap, Nikto, Gobuster, and Nuclei
- Manual testing using browser payload injection and Burp Suite
- Findings documented in the uploaded penetration testing reports

The following OWASP Top 10 categories were observed:



2. Scope

This project focuses on performing a Web Application Vulnerability Assessment and Penetration Testing on the target website testphp.vulnweb.com. The application is an intentionally vulnerable web platform provided by Acunetix, designed for practicing security testing techniques. The assessment was carried out in a controlled environment using Kali Linux and standard penetration testing tools.

The scope of this project includes identifying common web vulnerabilities, verifying their exploitability, and mapping the findings to OWASP Top 10 risk categories. Both automated scanning and manual testing techniques were applied to ensure accurate results.

2.1 Target Scope

The target of the assessment is:

- Target URL: http://testphp.vulnweb.com
- Application Type: PHP-based Web Application
- Web Server: Nginx

- Database: MySQL (backend)
- Testing Platform: Kali Linux

Included in Scope

- All publicly accessible pages of the web application
- Login and authentication functionalities
- Input forms and URL parameters
- Directory and file enumeration
- Client-side and server-side vulnerability testing

Testing Techniques Used

- Technology fingerprinting (WhatWeb)
- Port and service scanning (Nmap)
- Web vulnerability scanning (Nikto, Nuclei)
- Directory brute-forcing (Gobuster)
- Manual SQL Injection and XSS testing
- Security header and configuration analysis

2.2 Limitations

Although the assessment covered major vulnerability areas, certain limitations were present:

- The target application is an intentionally vulnerable test site, not a real production system.
- No denial-of-service (DoS) or heavy traffic testing was performed to avoid service disruption.
- Source code review was not conducted due to lack of access to backend code.

3. Approach and Methodology

This project followed a structured Web Application Vulnerability Assessment and Penetration Testing (VAPT) methodology. The approach was designed by combining standard penetration testing practices described in the uploaded security assessment reports and the practical testing methods executed using Kali Linux tools. The overall process aligns with OWASP Testing Guide and NIST SP 800-115 standards.

The methodology was divided into sequential phases to ensure systematic coverage of all security aspects of the target application.

3.1 Web Application and Web Services Penetration Testing

Web application and web services penetration testing involves systematically evaluating a web-based system to identify security weaknesses that attackers could exploit. In this project, the target web application testphp.vulnweb.com was tested using both automated vulnerability scanning tools and manual exploitation techniques.

Security Assessment Phases

1. Pre-Engagement

In this phase, the scope and objectives of the assessment were defined. The target application testphp.vulnweb.com was selected for web application security testing, and rules of engagement were established to ensure controlled and ethical testing.

2. Recon (Reconnaissance)

Information about the target environment was collected to understand its technologies and infrastructure. Domain enumeration, connectivity checks, and technology fingerprinting were performed, confirming that the application runs on an Nginx server with a PHP backend. This phase helped identify potential attack surfaces.

3. Threat Modeling

Based on the information gathered, potential attack vectors were analyzed. User input fields, authentication mechanisms, database interactions, file handling features, and exposed directories were identified as risk areas. This phase helped plan targeted security testing activities.

4. Vulnerability Analysis & Exploitation

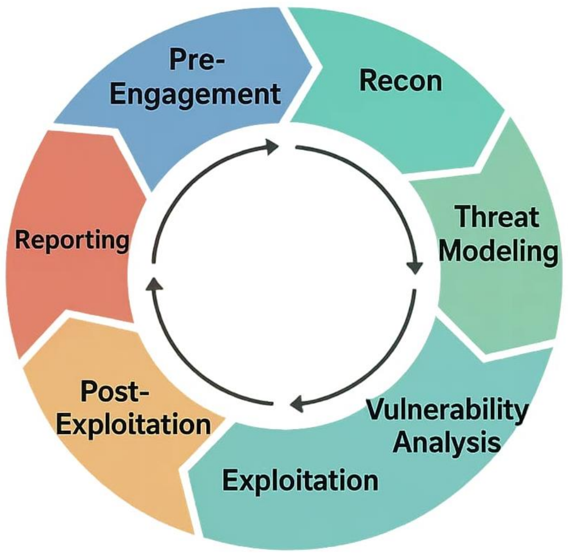
Automated scanning and manual testing were conducted to discover and validate vulnerabilities. Tools such as Burp Suite, SQLMap, and directory scanners were used to identify injection flaws, scripting vulnerabilities, authentication weaknesses, and information disclosure issues. Successful exploitation confirmed the presence of real security risks.

5. Post-Exploitation

After exploitation, the extent and impact of each vulnerability were assessed. Unauthorized access, database extraction, internal file disclosure, and account compromise scenarios were reviewed to understand real-world security implications.

6. Reporting

All identified vulnerabilities, proof-of-concepts, severity ratings, and improvement actions were compiled into a structured vulnerability assessment report to support remediation and strengthen application security.



The seven phases of PTES:
A practitioner-focused lifecycle for ethical hacking

The security assessment of testphp.vulnweb.com identified major vulnerabilities such as SQL Injection, XSS, and Sensitive Data Exposure through automated and manual testing. CVSS analysis showed that most findings are High and Critical risks, highlighting poor security posture. This project demonstrated practical penetration testing techniques and emphasized the importance of secure coding and proper security controls to protect web applications.

3.2 Infrastructure Vulnerabilities Assessment and Pen Testing

Infrastructure vulnerability assessment focuses on identifying weaknesses in the underlying server, network services, and system configurations that support the web application.

In this project, infrastructure-level testing included:

- Port and service scanning using Nmap to identify open ports and running services.
- Server fingerprinting using WhatWeb to detect web server and backend technologies.
- Security misconfiguration checks using Nikto to identify missing headers and version disclosures.
- Directory and file enumeration using Gobuster to detect exposed system folders.

This phase revealed:

- Outdated PHP and Nginx versions
- Server version disclosure
- Missing security headers
- Open directory listing

These findings indicate infrastructure-level misconfigurations that increase the attack surface of the web application.

3.3 Threats

Based on the identified vulnerabilities, the following potential threats were observed:

•SQL Injection Attacks (Critical)

> Attackers can manipulate database queries to bypass authentication and extract or modify sensitive data.

• Cross-Site Scripting (XSS) (High)

> Malicious scripts can be injected into web pages, leading to session hijacking and data theft.

• Sensitive Information Disclosure (High)

> Exposed credential files and open directories allow unauthorized access to confidential data.

• Clickjacking Attacks (Medium)

> Missing X-Frame-Options headers allow attackers to trick users into clicking hidden elements.

• Brute-force Login Attacks (High)

> Weak password policy and unlimited login attempts enable credential guessing.

• Insecure Communication (Medium)

> Use of HTTP instead of HTTPS allows attackers to intercept transmitted data.

• Directory Traversal Attacks (High)

> Improper input validation in file parameters may allow attackers to access restricted system files.

• Local File Inclusion (LFI) (High)

> Attackers can include and execute unauthorized local files from the server.

• Credential Harvesting (Medium)

> Exposed login pages without protection may be used to collect user credentials through phishing or brute force.

• Information Leakage via Error Messages (Low)

> Verbose database or server error messages reveal internal system details useful for attackers.

• Outdated Software Exploitation (Medium)

> Old PHP and Nginx versions may contain known vulnerabilities that attackers can exploit.

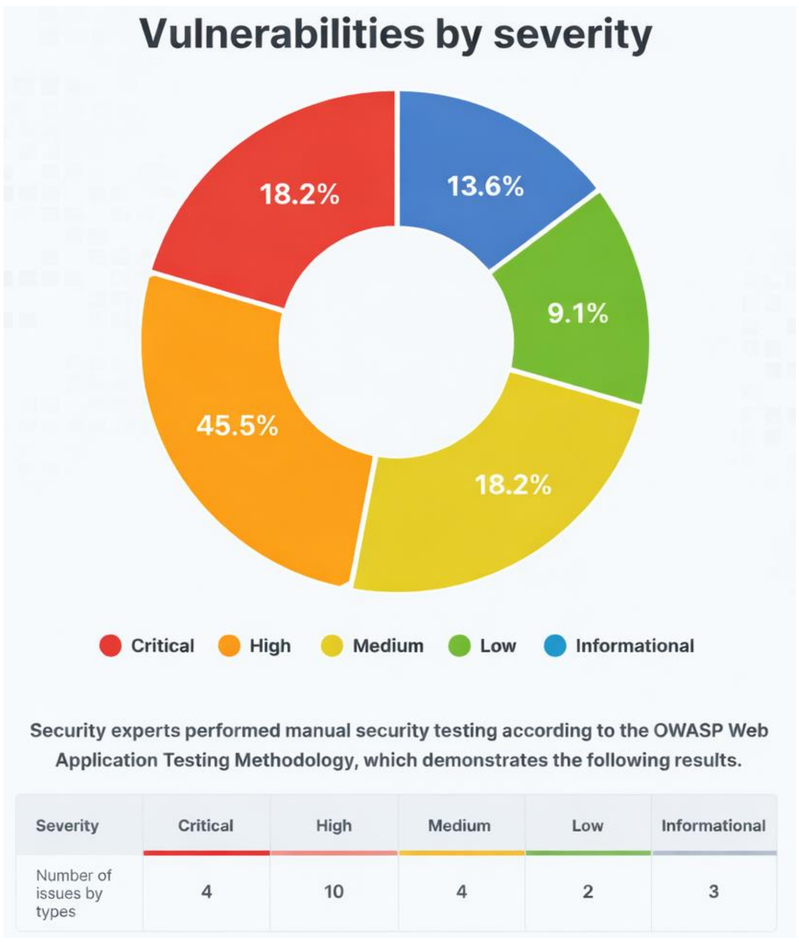
4. Risk Rating

After identifying and validating vulnerabilities in the target web application, each finding was assigned a risk rating to understand its severity and prioritize remediation. The risk rating in this project is based on the Common Vulnerability Scoring System (CVSS v3.1), which provides a numerical score from 0 to 10, where higher values represent greater security risk.

CVSS scoring considers factors such as attack vector, attack complexity, privileges required, user interaction, and the impact on confidentiality, integrity, and availability. Based on these parameters, vulnerabilities were classified into Critical, High, and Medium severity levels.

During the assessment of testphp.vulnweb.com, SQL Injection was identified as the most severe vulnerability with a CVSS score of 9.8 (Critical), as it allows complete compromise of the backend database. Cross-Site Scripting (XSS) vulnerabilities were rated High (7.4) due to their ability to execute malicious scripts in user browsers and hijack sessions. Sensitive Information Disclosure, including exposed credential files and open directories, was rated High (8.2) because it allows unauthorized access to confidential data. Weak authentication mechanisms were also rated High (7.0) since they enable brute-force attacks on login pages.

Findings such as security misconfigurations, missing security headers, clickjacking, directory listing, and outdated PHP and Nginx versions were rated Medium (5.0 – 6.8). These issues increase the application's attack surface and make exploitation easier when combined with other vulnerabilities.



Security experts performed manual security testing according to the OWASP Web Application Testing Methodology, which demonstrates the following results.

Severity	Critical	High	Medium	Low	Informational
Number of issues by types	4	10	4	2	3

Overall, the risk analysis indicates that the application has a poor security posture, with multiple high and critical vulnerabilities that could lead to full system compromise if exploited in a real-world environment. Immediate remediation of high and critical risks is strongly recommended.

5. Summary of Findings

The vulnerability assessment and penetration testing conducted on http://testphp.vulnweb.com identified multiple security weaknesses that significantly impact the confidentiality, integrity, and security of the application. A total of 9 vulnerabilities were discovered, including 2 critical, 5 high, and 2 medium severity issues. No low-severity vulnerabilities were identified.

The assessment identified critical vulnerabilities, including a SQL Injection flaw in the login functionality that enables authentication bypass and potential full access to the backend database. Additionally, sensitive credentials were exposed in publicly accessible files, allowing unauthorized users to compromise valid accounts without authentication.

Several high- and medium-severity issues were also observed, such as the lack of brute-force protection, missing access controls on restricted endpoints, disclosure of sensitive webpage content, and the absence of HTTPS encryption and HTTP security headers. These weaknesses increase the risk of account compromise, privilege escalation, information leakage, and exposure to Man-in-the-Middle and clickjacking attacks, collectively weakening the application's overall security posture.

Overall, the findings indicate that the application is vulnerable to basic reconnaissance and exploitation techniques and can be compromised with minimal effort. Immediate remediation of critical and high-severity vulnerabilities is strongly recommended to reduce the risk of unauthorized access, data breaches, and system compromise.

6. Application fingerprinting

Application fingerprinting is the process of identifying the underlying technologies, frameworks, server software, programming languages, and configuration details used by a web application. This phase is performed during the reconnaissance stage of penetration testing to understand the target's technology stack and detect potential attack surfaces. By revealing software versions and backend components, fingerprinting helps security testers identify outdated or misconfigured services that may contain known vulnerabilities.

6.1 whatweb

In this project, application fingerprinting was performed on the target web application http://testphp.vulnweb.com using the WhatWeb tool in Kali Linux. WhatWeb analyzes HTTP response headers, webpage content, and embedded scripts to detect technologies running behind the target.

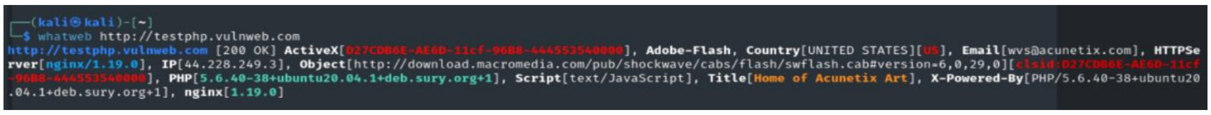
Command used :

- whatweb http://testphp.vulnweb.com

Explanation of Command

- whatweb – Launches the WhatWeb fingerprinting tool.
- http://testphp.vulnweb.com – Specifies the target URL to be scanned.

This command sends requests to the web server and inspects returned data to identify server type, backend language, scripting technologies, and additional components.



```
┌──(kali㉿kali)-[~]
└─$ whatweb http://testphp.vulnweb.com
http://testphp.vulnweb.com [200 OK] ActiveX[D27CDB6E-AE6D-11cf-96B8-444553540000], Adobe-Flash, Country[UNITED STATES][US], Email[wvs@acunetix.com], HTTPSe
rver[nginx/1.19.0], IP[44.228.249.3], Object[http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,29,0][clsid:D27CDB6E-AE6D-11cf
-96B8-444553540000], PHP[5.6.40-38+ubuntu20.04.1+deb.sury.org+1], Script[text/JavaScript], Title[Home of Acunetix Art], X-Powered-By[PHP/5.6.40-38+ubuntu20
.04.1+deb.sury.org+1], nginx[1.19.0]
```

Fingerprinting Results :

From the WhatWeb scan output, the following information was identified:

- HTTP Status: 200 OK (Target reachable)
- Web Server: Nginx 1.19.0
- Backend Language: PHP 5.6.40
- Operating System: Ubuntu 20.04
- IP Address: 44.228.249.3
- X-Powered-By Header: PHP/5.6.40
- Scripting Support: JavaScript detected
- Title of Webpage: "Home of Acunetix Art"
- Additional Components: Adobe Flash and ActiveX objects detected

Analysis of Results :

The fingerprinting phase confirmed that the application is hosted on an Nginx web server with a PHP backend running on Ubuntu. The scan also revealed that both the web server and backend language disclose their exact version numbers through response headers. Exposing such version information is considered information leakage, as it allows attackers to search for publicly available exploits targeting specific software versions.

Additionally, the detection of JavaScript and dynamic page content indicates that the application processes user inputs at both client and server sides. This aligns with later findings of input-based vulnerabilities such as SQL Injection and Cross-Site Scripting. The presence of outdated backend software (PHP 5.6.40) further increases the likelihood of known exploitable vulnerabilities.

Security Impact :

Exposed technology and version details enable attackers to:

- Identify outdated software components
- Search for known exploits in vulnerability databases
- Plan targeted attacks based on the technology stack

6.2. Nmap

In this project, Nmap was used to perform service and version detection on the target web application http://testphp.vulnweb.com.

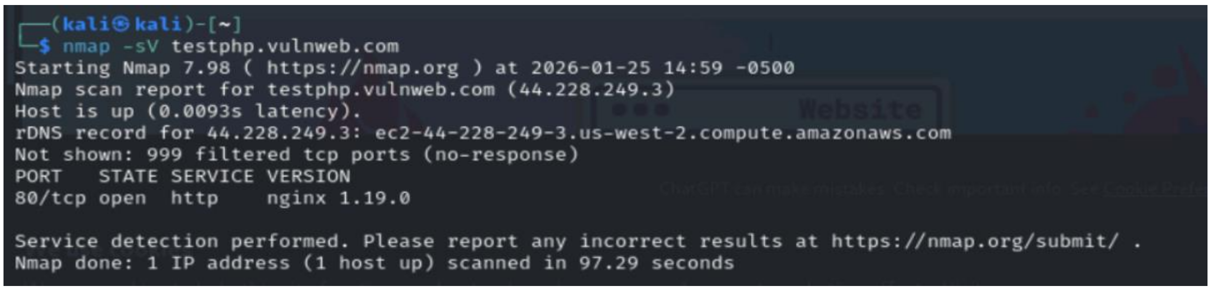
Command used :

- ➢ nmap -sV testphp.vulnweb.com

Explanation of Command :

- nmap – Launches the Nmap scanning tool
- -sV – Enables service and version detection
- testphp.vulnweb.com – Specifies the target domain to scan

This command scans the target host to identify open ports and determines the service type and version running on each open port.



```
┌──(kali㉿kali)-[~]
└─$ nmap -sV testphp.vulnweb.com
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-25 14:59 -0500
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.0093s 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

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 97.29 seconds
```

Fingerprinting Results :

The Nmap scan revealed the following:

- Target IP Address: 44.228.249.3
- Host Status: Host is up and reachable
- Open Port: 80/tcp
- Running Service: HTTP
- Detected Service Version: Nginx 1.19.0
- DNS Record: Hosted on Amazon AWS infrastructure

Analysis of Results :

The Nmap fingerprinting results confirm that the web application is publicly accessible through port 80 (HTTP) and is running on an Nginx web server version 1.19.0. Identifying the exact service version indicates information disclosure, as attackers can search for publicly known vulnerabilities related to that specific server version. This information helps in defining the network attack surface and guides further vulnerability scanning and exploitation steps.

Security Impact :

Exposing service and version details allows attackers to:

- Identify active network services
- Detect outdated or vulnerable software
- Perform targeted vulnerability exploitation

Therefore, Nmap service detection plays an essential role in mapping infrastructure and application exposure.

7.Detailed Findings

This section outlines the confirmed vulnerabilities identified during penetration testing of testphp.vulnweb.com. Each finding includes its risk level, CVSS score, affected URL, technical description, proof-of-concept steps, observed results, and recommended remediation measures. The findings are based on validated automated scans and manual testing conducted using Kali Linux tools.

7.1 Vulnerability: Server Banner Disclosure

Risk Rating	Informational
CVSS Score	**2.1**
Affected URL	http://testphp.vulnweb.com/

Risk Description

The web server discloses backend technology details in HTTP response headers. The response headers reveal information such as:

- Server: nginx
- X-Powered-By: PHP version

Exposing server and software version details provides useful reconnaissance information to attackers. This may allow them to identify known vulnerabilities associated with specific technologies and versions.

Business Impact

Disclosure of backend technology stack

Assists attackers in fingerprinting the target environment

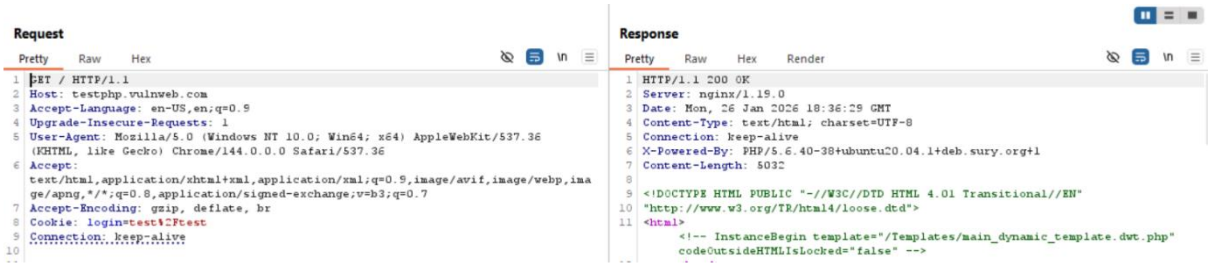
Increases risk of targeted exploitation using known version-specific attacks

Steps to Reproduce

1. Open Burp Suite or browser developer tools.
2. Navigate to: http://testphp.vulnweb.com/
3. View the HTTP response headers.
4. Observe the presence of:
    - Server header
    - X-Powered-By header

This confirms server banner disclosure.

Output Finding



```
GET / HTTP/1.1
Host: testphp.vulnweb.com
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/144.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Cookie: login=test%2Ftest
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Mon, 26 Jan 2026 18:36:29 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: 5032

<!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" -->
```

HTTP response headers contained:

- Server: nginx
- X-Powered-By: PHP/5.6.40

Recommendations

- Remove or obscure server version information from HTTP headers.
- Disable the X-Powered-By header.
- Configure the web server to return generic banner responses.
- Perform regular banner and fingerprinting reviews.

7.2 Vulnerability: Missing Security Headers

Risk Rating	Informational
CVSS Score	**2.6**
Affected URL	http://testphp.vulnweb.com/

Risk Description

The application does not implement important **HTTP security headers** that provide browser-side security hardening.

During testing, HTTP responses were observed to lack standard security headers such as:

- Content-Security-Policy
- X-Content-Type-Options
- Strict-Transport-Security
- Referrer-Policy
- X-Frame-Options

The absence of these headers weakens client-side protections and may increase the impact of potential browser-based attacks such as Cross-Site Scripting (XSS), MIME-type sniffing, clickjacking, and unnecessary referrer information exposure.

Although this does not directly result in an immediate exploitable vulnerability, it represents a **security hardening improvement**.

Business Impact

Reduced browser-side security protection

Increased exposure to client-side injection attacks

Lack of enforced HTTPS communication policies

Potential referrer information leakage

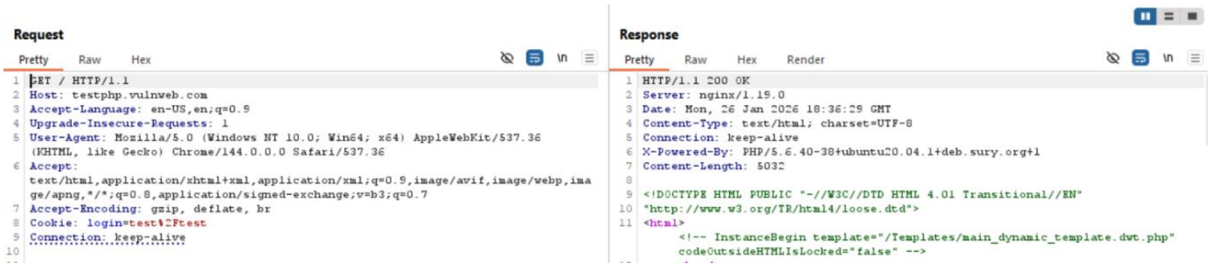
Steps to Reproduce

1. Open Burp Suite or browser developer tools.
2. Navigate to: http://testphp.vulnweb.com/
3. View the HTTP response headers.

4. Observe that security headers such as Content-Security-Policy, X-Content-Type-Options, Strict-Transport-Security, Referrer-Policy, and X-Frame-Options are not present.

This confirms missing security header protections.

Output Finding



```
GET / HTTP/1.1
Host: testphp.vulnweb.com
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/144.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Cookie: login=test%2Ftest
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Mon, 26 Jan 2026 18:36:29 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: 5032

<!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" -->
```

Burp Suite captured HTTP responses where standard security headers were absent.

If present, would look like

```
HTTP/1.1 200 OK

Server: nginx/1.19.0

X-Powered-By: PHP/5.6.40

Content-Type: text/html; charset=UTF-8

Connection: keep-alive


Content-Security-Policy: default-src 'self'

X-Content-Type-Options: nosniff

Strict-Transport-Security: max-age=31536000; includeSubDomains

Referrer-Policy: same-origin

X-Frame-Options: DENY
```

Recommendations

- Implement appropriate HTTP security headers to improve browser-side security.
- Define Content-Security-Policy to restrict script execution sources.
- Enable X-Content-Type-Options to prevent MIME-type sniffing.
- Apply Strict-Transport-Security to enforce HTTPS usage.
- Configure Referrer-Policy to control referrer information sharing.

7.3 Finding: Reconnaissance & OSINT Enumeration

Risk Rating	Informational
CVSS Score	1.9
Affected Target	testphp.vulnweb.com

Risk Description

Passive reconnaissance and Open-Source Intelligence (OSINT) enumeration were performed against the target domain. Publicly available infrastructure information was successfully gathered, including domain registration details, DNS records, IP resolution data, and subdomain structure.

WHOIS enumeration revealed domain registration and hosting information.
DNS enumeration using standard lookup techniques revealed active DNS records and resolved IP addresses.
Subdomain enumeration identified additional domain-related information under the parent domain.

Although this information is publicly available by design, it provides useful reconnaissance data that may assist attackers in profiling the target environment and planning further attacks.

Business Impact

Disclosure of domain ownership and hosting details

Exposure of DNS and IP infrastructure information

Identification of domain structure and subdomains

Assists attackers in reconnaissance and attack planning

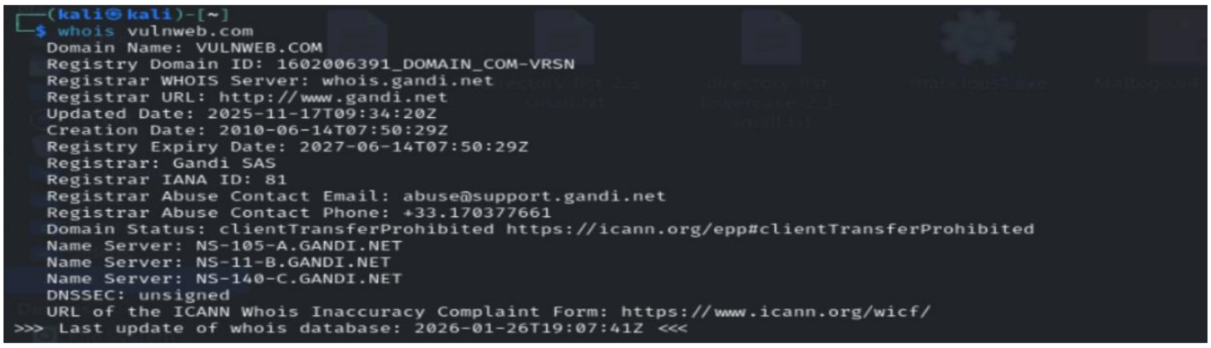
Steps to Reproduce

1. Perform WHOIS lookup to obtain domain registration details.
2. Perform DNS enumeration to identify DNS records and IP resolution.
3. Perform subdomain enumeration to identify related subdomains.

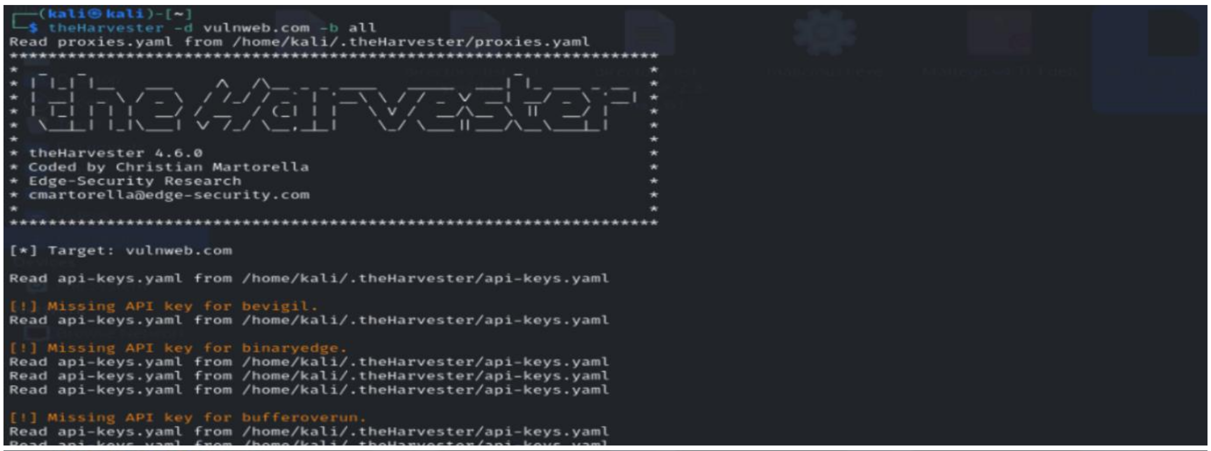
This confirms that infrastructure-related information is publicly discoverable.

Output Finding

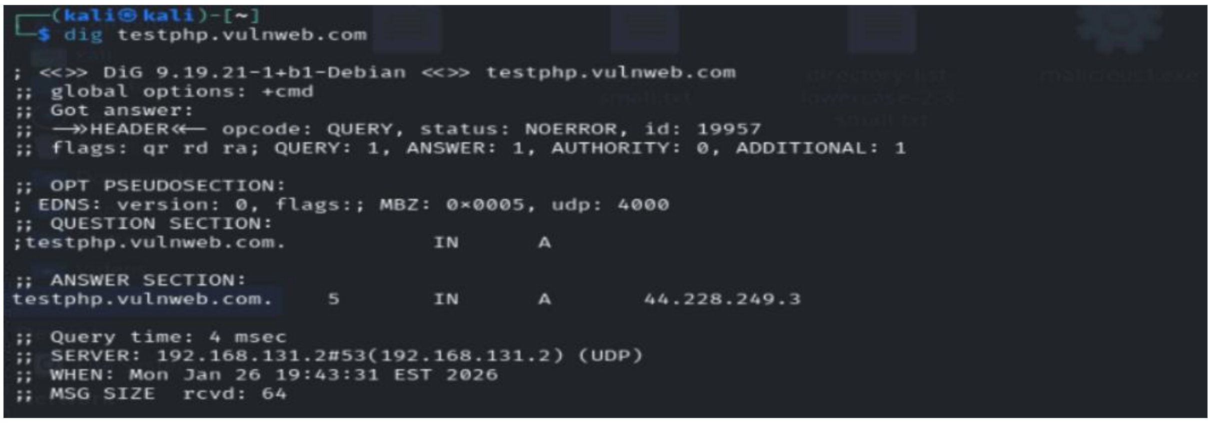
WHOIS lookup returned domain registration information.



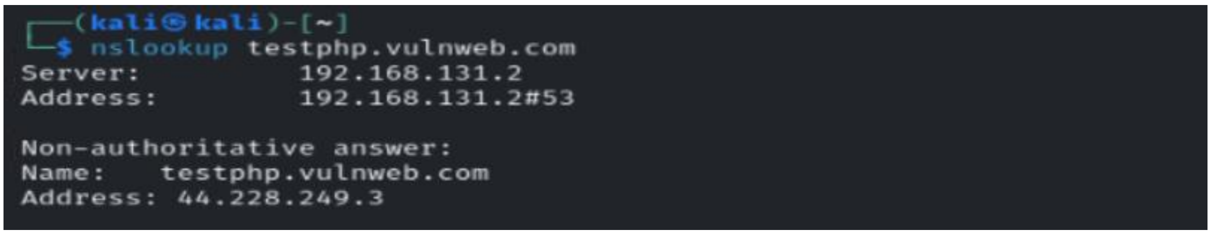
```
┌──(kali㉿kali)-[~]
└─$ whois vulnweb.com
   Domain Name: VULNWEB.COM
   Registry Domain ID: 1602006391_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.gandi.net
   Registrar URL: http://www.gandi.net
   Updated Date: 2025-11-17T09:34:20Z
   Creation Date: 2010-06-14T07:50:29Z
   Registry Expiry Date: 2027-06-14T07:50:29Z
   Registrar: Gandi SAS
   Registrar IANA ID: 81
   Registrar Abuse Contact Email: abuse@support.gandi.net
   Registrar Abuse Contact Phone: +33.170377661
   Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
   Name Server: NS-105-A.GANDI.NET
   Name Server: NS-11-B.GANDI.NET
   Name Server: NS-140-C.GANDI.NET
   DNSSEC: unsigned
   URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2026-01-26T19:07:41Z <<<
```



```
┌──(kali㉿kali)-[~]
└─$ theHarvester -d vulnweb.com -b all
Read proxies.yaml from /home/kali/.theHarvester/proxies.yaml
*******************************************************************
*  _   _                                            _             *
* | |_| |__   ___    /\  /\__ _ _ ____   _____  ___| |_ ___ _ __  *
* | __|  _ \ / _ \  / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__| *
* | |_| | | |  __/ / __  / (_| | |   \ V /  __/\__ \ ||  __/ |    *
*  \__|_| |_|\___| \/ /_/ \__,_|_|    \_/ \___||___/\__\___|_|    *
*                                                                 *
* theHarvester 4.6.0                                              *
* Coded by Christian Martorella                                   *
* Edge-Security Research                                          *
* cmartorella@edge-security.com                                   *
*                                                                 *
*******************************************************************

[*] Target: vulnweb.com

Read api-keys.yaml from /home/kali/.theHarvester/api-keys.yaml

[!] Missing API key for bevigil.
Read api-keys.yaml from /home/kali/.theHarvester/api-keys.yaml

[!] Missing API key for binaryedge.
Read api-keys.yaml from /home/kali/.theHarvester/api-keys.yaml
Read api-keys.yaml from /home/kali/.theHarvester/api-keys.yaml
Read api-keys.yaml from /home/kali/.theHarvester/api-keys.yaml

[!] Missing API key for bufferoverun.
Read api-keys.yaml from /home/kali/.theHarvester/api-keys.yaml
```



```
┌──(kali㉿kali)-[~]
└─$ dig testphp.vulnweb.com

; <<>> DiG 9.19.21-1+b1-Debian <<>> testphp.vulnweb.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19957
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; MBZ: 0×0005, udp: 4000
;; QUESTION SECTION:
;testphp.vulnweb.com.           IN      A

;; ANSWER SECTION:
testphp.vulnweb.com.    5       IN      A       44.228.249.3

;; Query time: 4 msec
;; SERVER: 192.168.131.2#53(192.168.131.2) (UDP)
;; WHEN: Mon Jan 26 19:43:31 EST 2026
;; MSG SIZE  rcvd: 64
```



```
┌──(kali㉿kali)-[~]
└─$ nslookup testphp.vulnweb.com
Server:         192.168.131.2
Address:        192.168.131.2#53

Non-authoritative answer:
Name:   testphp.vulnweb.com
Address: 44.228.249.3
```

DNS enumeration revealed active DNS records and IP resolution details. Subdomain enumeration returned domain-related discovery results.

Recommendations

- Monitor publicly available OSINT sources for unintended information exposure.
- Ensure no sensitive internal subdomains are publicly accessible.

7.4 Vulnerability: SQL Injection

Risk Rating	Critical
CVSS Score	**9.8**
Affected URL	http://testphp.vulnweb.com/login.php http://testphp.vulnweb.com/artists.php http://testphp.vulnweb.com/product.php

Risk Description

The application is vulnerable to SQL Injection due to improper validation of user input. Attackers can inject malicious SQL queries through input fields and URL parameters, causing unintended execution of database commands. This vulnerability allows database manipulation, data extraction, and authentication bypass without valid credentials.

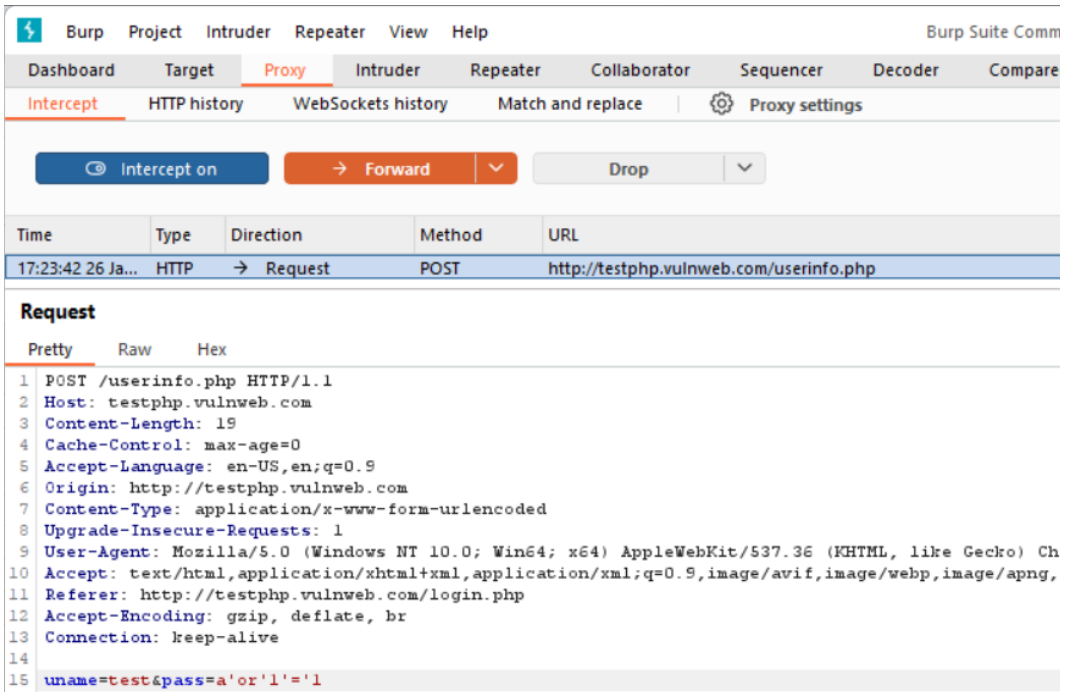
Business Impact

- Unauthorized access to user accounts
- Bypass of login authentication mechanism
- Exposure of sensitive database records
- Possible modification or deletion of data
- Complete compromise of backend database

Steps to Reproduce

1. Navigate to the login page: http://testphp.vulnweb.com/login.php
2. Enter the following payload in the **username** field:
3. ' OR '1'='1
4. Enter any value in the password field.
5. Submit the request or intercept using **Burp Suite**.
6. Observe that login is successful without valid credentials.
7. Similar payloads in URL parameters (e.g., product.php?id=1') return database errors or altered responses.

Output Finding



Time	Type	Direction	Method	URL
17:23:42 26 Ja...	HTTP	→ Request	POST	http://testphp.vulnweb.com/userinfo.php

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Content-Length: 19
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://testphp.vulnweb.com
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
Referer: http://testphp.vulnweb.com/login.php
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

uname=test&pass=a'or'1'='1
```

- Burp Suite captured the injected SQL payload in request parameters.
- The server returned a successful login response.
- Restricted pages became accessible without valid credentials.
- Error-based and boolean-based SQL responses were observed in product and artist pages.

Recommendations

- Use **parameterized queries (prepared statements)**.
- Apply **server-side input validation and sanitization**.
- Avoid dynamic query construction.
- Suppress detailed database error messages.
- Use least-privileged database accounts.
- Conduct regular security testing.

7.5 Vulnerability: Reflected Cross-Site Scripting (XSS) in Search Bar

Risk Rating	Critical
CVSS Score	**9.0**
Affected URL	http://testphp.vulnweb.com/

Risk Description

The application is vulnerable to **Reflected Cross-Site Scripting (XSS)** in the search bar.

User-supplied input in the search field is reflected back into the webpage without proper validation or encoding. An attacker can inject malicious JavaScript code into the search parameter. When another user loads the crafted link, the injected script executes in the victim's browser.

This allows attackers to steal session cookies, capture login credentials, redirect users to malicious websites, or manipulate webpage content.

Business Impact

Execution of malicious scripts in victim's browser

Theft of session cookies and login credentials

Redirection to phishing or malicious websites

Website defacement

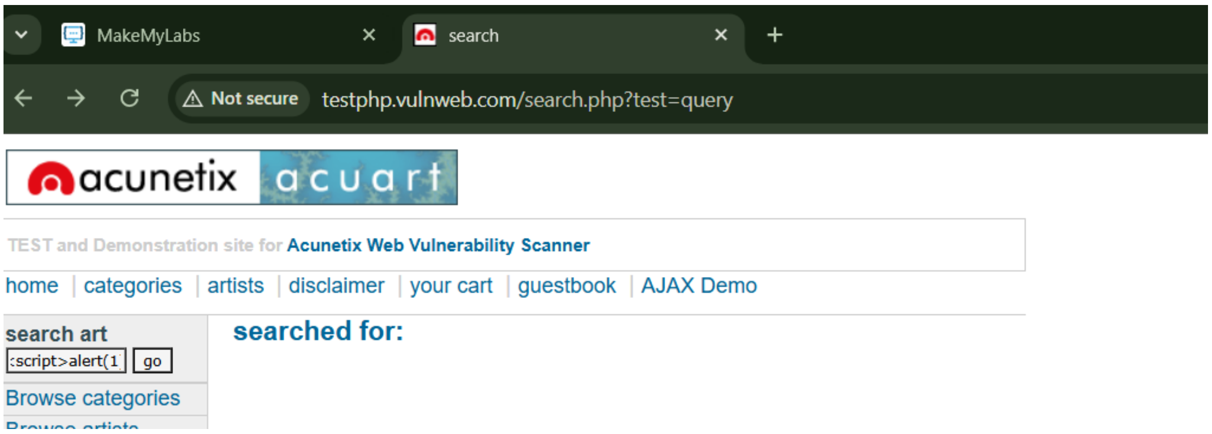
Loss of user trust and reputational damage

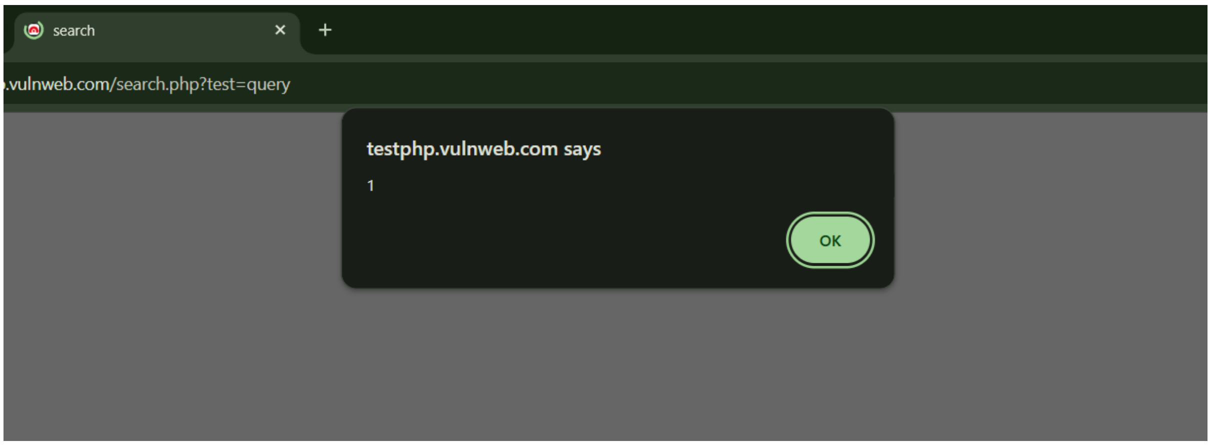
Steps to Reproduce

1. Navigate to the website: http://testphp.vulnweb.com/
2. Locate the search bar.
3. Enter the following payload into the search field: <img src=s onerror=alert(1)>
4. Click the search button.
5. Observe that a JavaScript alert box appears in the browser.

This confirms that user input is reflected without sanitization, triggering script execution.

Output Finding


MakeMyLabs
search
Not secure
testphp.vulnweb.com/search.php?test=query
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
:script>alert(1
go
searched for:
Browse categories



testphp.vulnweb.com says

1

Injected payload: <img src=s onerror=alert(1)>

Result: A browser alert box pops up displaying 1, confirming successful execution of injected JavaScript code.

Recommendations

- Implement strict input validation and filtering on all user-supplied data.
- Apply proper output encoding before reflecting user input in HTML content.
- Use Content Security Policy (CSP) to restrict script execution.
- Deploy a Web Application Firewall (WAF) to block malicious XSS payloads.
- Perform regular security testing for XSS vulnerabilities.

7.6 Vulnerability: Stored Cross-Site Scripting (Stored XSS)

Risk Rating	Critical
CVSS Score	**9.2**
Affected URL	http://testphp.vulnweb.com/guestbook.php

Risk Description

The application is vulnerable to **Stored Cross-Site Scripting (XSS)** in the Guestbook page.

User-supplied input submitted through the comment form is stored in the backend database and later displayed to other users without proper validation or output encoding. As a result, malicious script content can be permanently stored and automatically executed in the browsers of users who visit the page.

This vulnerability allows attackers to steal session cookies, hijack user accounts, redirect victims to malicious websites, or manipulate webpage content.

Business Impact

Permanent execution of malicious scripts in user browsers

Theft of session cookies and authentication tokens

Unauthorized account access

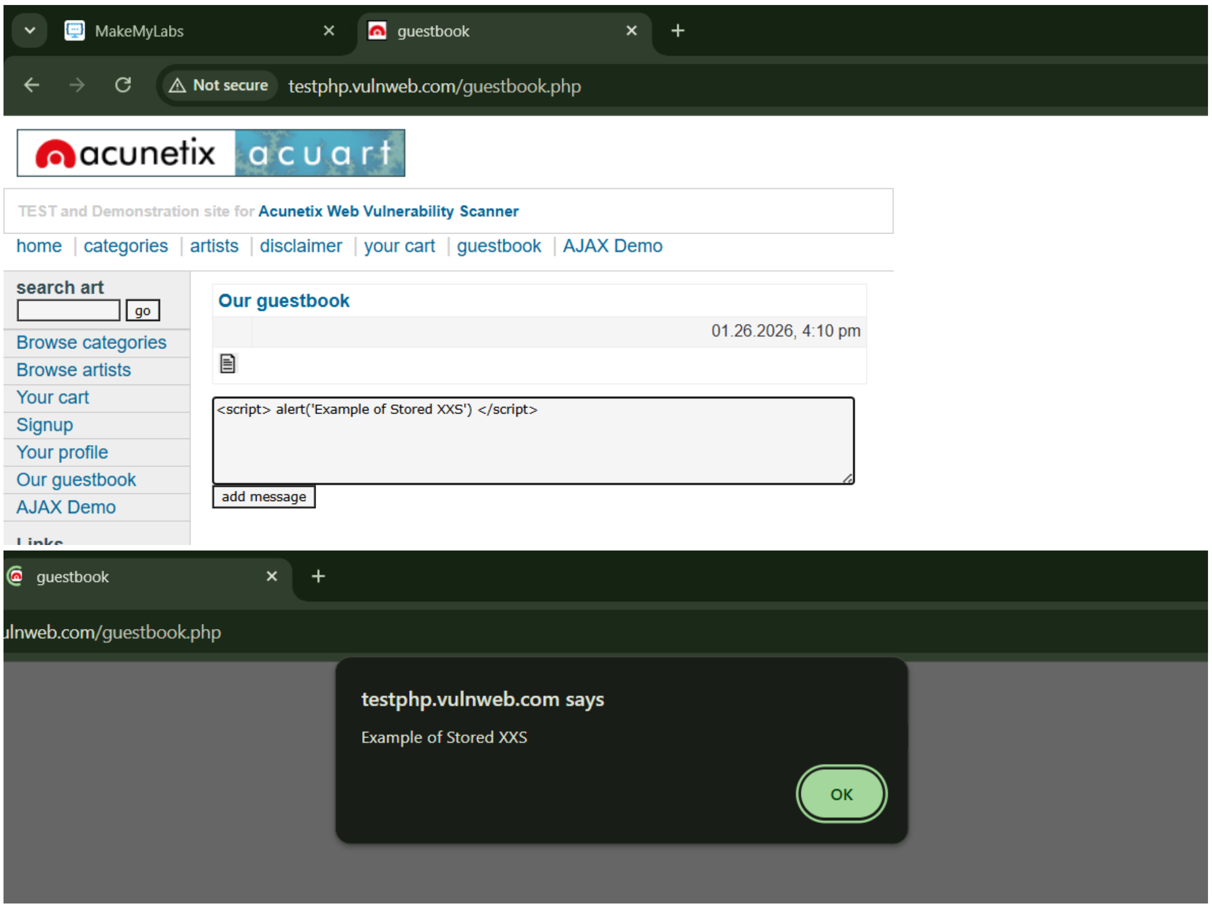
Website defacement

Loss of user trust and reputational damage

Steps to Reproduce

1. Navigate to: http://testphp.vulnweb.com/guestbook.php
2. Enter a name and a comment containing a script-based test input in the comment field.
3. Submit the comment form.
4. Reload the guestbook page.
5. Observe that the stored input executes automatically when the page loads.
6. This confirms the presence of a Stored XSS vulnerability.

Output Finding



```
<script> alert('Example of Stored XXS') </script>
```

testphp.vulnweb.com says

Example of Stored XXS

A script-based test input submitted through the guestbook form was stored in the database.

Upon reloading the page, the script executed automatically in the browser.

Recommendations

- Validate and sanitize all user-supplied input before storing it.
- Apply proper output encoding when displaying user content.
- Implement Content Security Policy (CSP) to restrict script execution.
- Use a Web Application Firewall (WAF) to filter malicious XSS payloads.
- Perform regular security testing for XSS vulnerabilities.

7.7 Vulnerability: Sensitive Information Disclosure

Risk Rating	Critical
CVSS Score	**9.1**
Affected URL	http://testphp.vulnweb.com/login.php

Risk Description

The application exposes valid user login credentials openly within the system. The username **"test"** and password **"test"** are visible and can be discovered by any user accessing the signup or login-related pages.

This results in **Sensitive Information Disclosure**, where authentication credentials are revealed without access restrictions. Any attacker can use these credentials to log in and gain unauthorized access to the application.

Business Impact

Unauthorized access to user accounts

Credential leakage in plaintext form

Bypass of authentication mechanisms

Potential privilege escalation

Loss of confidentiality and integrity of user data

Full compromise of affected user accounts

Steps to Reproduce

Navigate to the login page: http://testphp.vulnweb.com/login.php

Observe that default user credentials are publicly known or visible through related signup/login functionality.

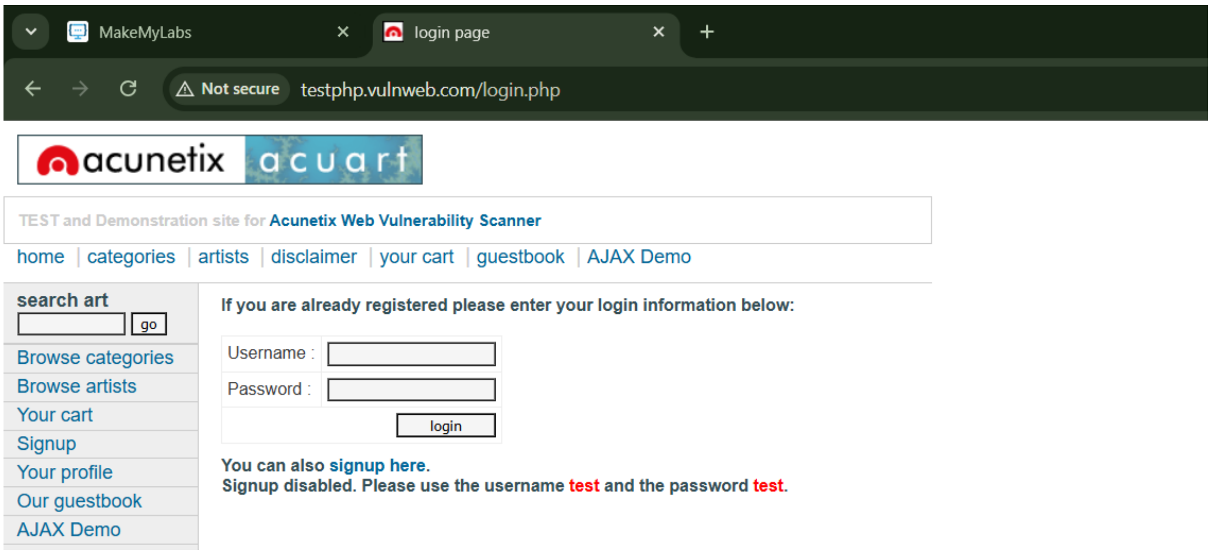
Use the exposed credentials:

Username:test
Password: test

Log in successfully without legitimate authorization.

This confirms exposure of sensitive authentication data.

Output Finding



TEST and Demonstration site for Acunetix Web Vulnerability Scanner

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.

Valid login credentials were found to be openly available:

Username: test

Password: test

Using these credentials on the login page results in successful authentication.

Recommendations

- Remove hardcoded or publicly exposed credentials from the application.
- Force all existing users to reset passwords.
- Never display credentials in any webpage or client-side code.
- Store passwords using strong hashing algorithms.
- Implement proper access control for authentication data.
- Perform regular security audits to detect information leakage.

7.8 Vulnerability: Blind SQL Injection

Risk Rating	High
CVSS Score	**7.8**
Affected URL	http://testphp.vulnweb.com/product.php?pic=1

Risk Description

The application is vulnerable to **Blind SQL Injection**. During testing, the application did not display detailed SQL error messages; however, database queries were still influenced by user-supplied input. By analyzing differences in server responses and response time behavior, it was confirmed that backend SQL queries were being executed based on injected conditions.

Both **Boolean-based Blind SQL Injection** and **Time-based Blind SQL Injection** techniques were identified. This allows an attacker to extract database information through inference, even when direct error messages are not shown.

Business Impact

Unauthorized extraction of database data through inference techniques

Exposure of sensitive application records

Possible database manipulation

Loss of confidentiality and data integrity

Potential full backend database compromise

Steps to Reproduce

1. Navigate to: http://testphp.vulnweb.com/product.php?pic=1
2. Intercept the request using Burp Suite.
3. Observe the normal server response as a baseline.
4. During testing, conditional query inputs were supplied to the parameter.
5. Response behavior was analyzed:

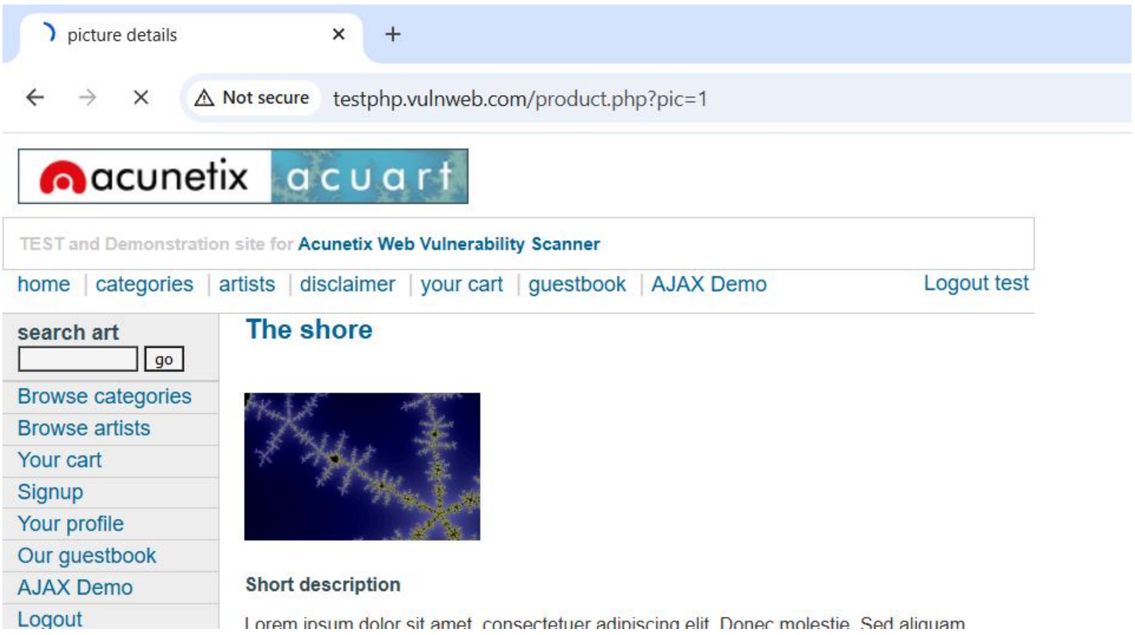
   Page content changed for different logical conditions → Boolean-based behavior

   Response delay observed for specific conditions → Time-based behavior

6. Automated testing confirmed successful database enumeration using inference techniques.

This verifies Blind SQL Injection.

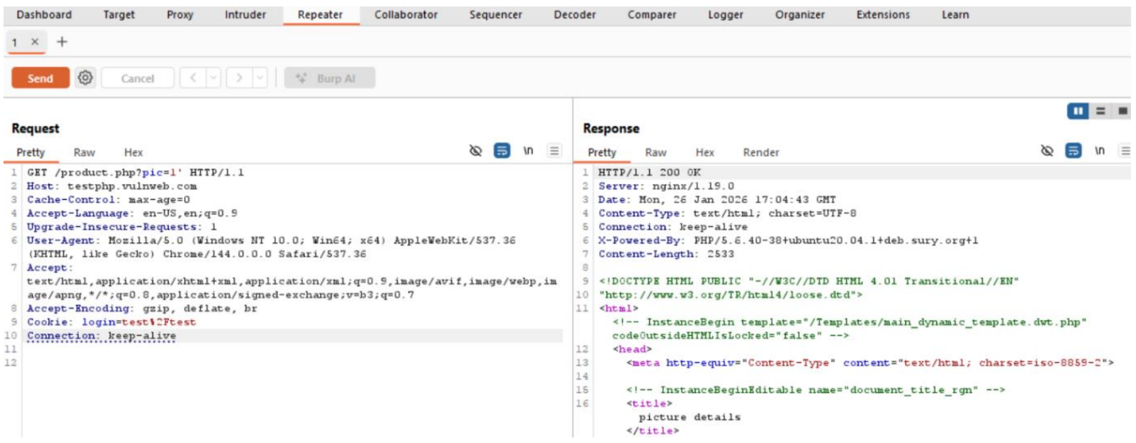
Output Finding


picture details
Not secure
testphp.vulnweb.com/product.php?pic=1
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

The shore

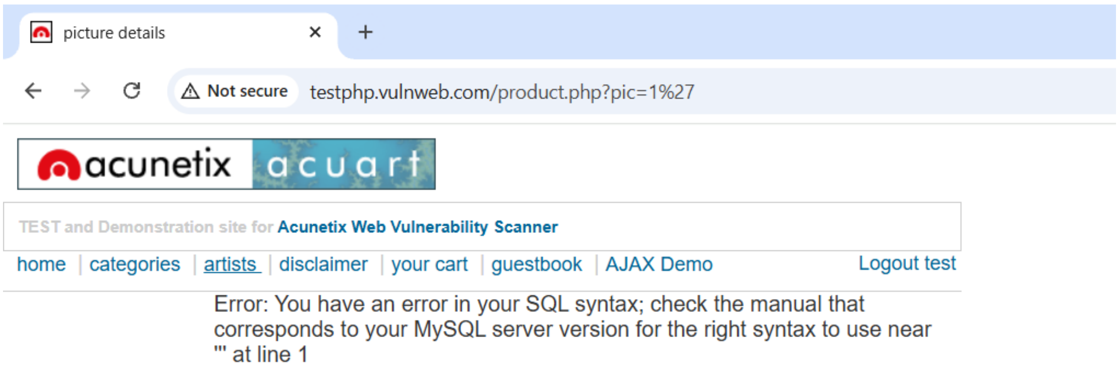
Short description

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



```
GET /product.php?pic=1' HTTP/1.1
Host: testphp.vulnweb.com
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/144.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,im
age/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Cookie: login=test%2Ftest
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Mon, 26 Jan 2026 17:04:43 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: 2533

<!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" -->
    <title>
      picture details
    </title>
```


picture details
Not secure
testphp.vulnweb.com/product.php?pic=1%27
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
Logout test

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line 1

Automated SQL injection testing reported:

- **Boolean-based Blind SQL Injection**
- **Time-based Blind SQL Injection**

Backend DBMS identified as **MySQL ≥ 5.6**. Database and table enumeration were successfully completed through inference methods.

Recommendations

- Use prepared statements and parameterized queries.
- Avoid dynamic SQL query construction.
- Sanitize and validate all user inputs.
- Suppress detailed SQL error messages.
- Apply least-privilege permissions to database accounts.
- Perform regular security testing for SQL Injection.

7.9 Vulnerability: Brute Force Attack on Login Page

Risk Rating	High
CVSS Score	**8.2**
Affected URL	http://testphp.vulnweb.com/login.php

Risk Description

The login page of the application is vulnerable to a **Brute Force attack**. The application does not implement any mechanism to limit repeated login attempts.

An attacker can systematically try multiple username and password combinations through automated tools. A successful login attempt returns a **200 OK** response, confirming valid credentials. This allows unauthorized access to user accounts.

Business Impact

Unauthorized access to user accounts

Credential compromise through password guessing

Possible privilege escalation

Loss of confidentiality and integrity of user data

Potential full account takeover

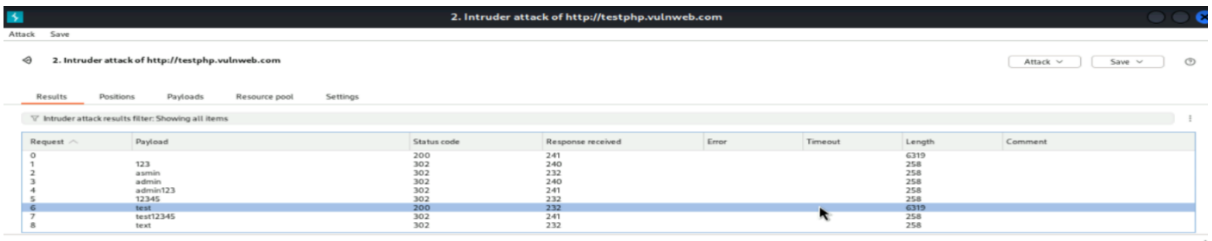
Steps to Reproduce

1. Navigate to the login page: http://testphp.vulnweb.com/login.php

2. Enter a valid username.
3. Use an automated tool (e.g., Burp Suite Intruder) to send multiple password attempts.
4. Observe that the application allows unlimited login attempts without restriction.
5. Identify successful login attempts by **HTTP 200 OK** responses.

This confirms that the login page is vulnerable to brute force attacks.

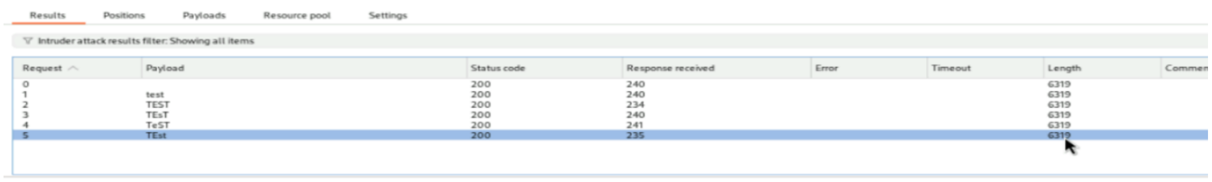
Output Finding



2. Intruder attack of http://testphp.vulnweb.com
Attack
Save
2. Intruder attack of http://testphp.vulnweb.com
Attack
Save
Results
Positions
Payloads
Resource pool
Settings
Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	241			6319	
1	123	302	240			258	
2	asmin	302	232			258	
3	admin	302	240			258	
4	admin123	302	241			258	
5	12345	302	232			258	
6	test	200	232			6319	
7	test12345	302	241			258	
8	text	302	232			258	

Trying a password bruteforce attack on the login page (200 response indicates successful)



Request	Payload	Status code	Response received	Error	Timeout	Length	Commen
0		200	240			6319	
1	test	200	240			6319	
2	TEST	200	234			6319	
3	TEsT	200	240			6319	
4	TeST	200	241			6319	
5	TEst	200	235			6319	

List of successful passwords

Multiple login attempts were sent to the login endpoint.

Successful attempts returned **HTTP 200 OK** responses.

A list of valid passwords was identified through repeated requests.

Recommendations

- Implement a limit on login attempts per IP address.
- Track failed login attempts per user or IP.
- Enforce temporary account lockout after multiple failures.
- Introduce CAPTCHA after several failed attempts.
- Display generic error messages for failed logins.
- Store passwords using strong hashing algorithms.
- Perform continuous monitoring for abnormal login behavior.

7.10 Vulnerability: Local File Inclusion (Path Traversal)

Risk Rating	High
CVSS Score	**7.5**
Affected URL	http://testphp.vulnweb.com/showimage.php?file=

Risk Description

The application is vulnerable to **Local File Inclusion (LFI)** due to improper validation of the file parameter in showimage.php.

An attacker can manipulate the file parameter using directory traversal sequences to access sensitive files from the server's filesystem. By injecting traversal payloads, internal system files can be included and their contents disclosed in the server response.

This confirms that user input is directly used in file-handling functions without proper sanitization.

Business Impact

Disclosure of sensitive system files

Exposure of server configuration and operating system details

Information leakage useful for further exploitation

Potential escalation to Remote Code Execution

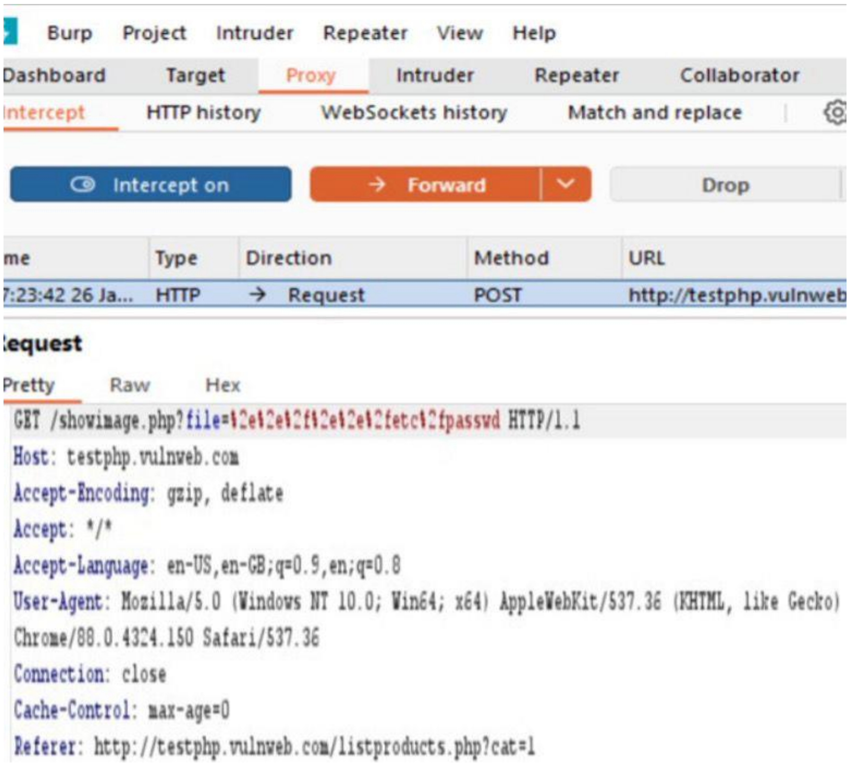
Overall compromise of application security

Steps to Reproduce

1. Navigate to the vulnerable endpoint: http://testphp.vulnweb.com/showimage.php?file=
2. Intercept the request using Burp Suite Repeater.
3. Modify the file parameter with a directory traversal payload pointing to a system file:
   - /showimage.php?file=../../../../../../../../../proc/version
4. Send the request.

Observe that the response returns the contents of the /proc/version file.

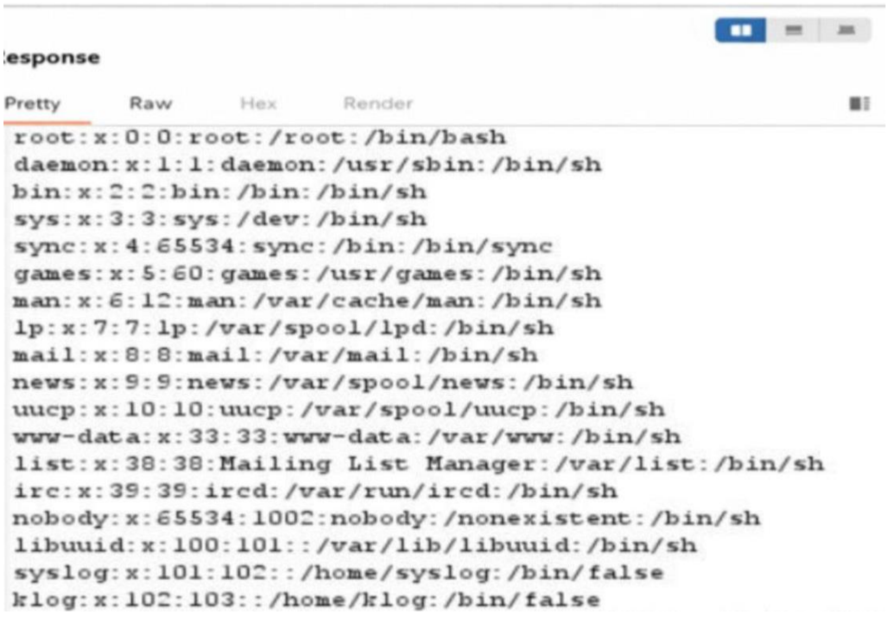
Output Finding



```
me              Type   Direction    Method   URL
7:23:42 26 Ja...  HTTP   → Request    POST     http://testphp.vulnweb
```

Request

```
GET /showimage.php?file=%2e%2e%2f%2e%2e%2fetc%2fpasswd HTTP/1.1
Host: testphp.vulnweb.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/88.0.4324.150 Safari/537.36
Connection: close
Cache-Control: max-age=0
Referer: http://testphp.vulnweb.com/listproducts.php?cat=1
```



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
nobody:x:65534:1002:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:102::/home/syslog:/bin/false
klog:x:102:103::/home/klog:/bin/false
```

Response:

HTTP/1.1 200 OK

Content-Type: image/jpeg

Linux version 5.4.0-1030-aws (buildd@lcy01-amd64-028)

(gcc version 9.3.0) #31-Ubuntu SMP Fri Nov 13 11:40:37 UTC 2020

The response confirms successful retrieval of system file content, proving Local File Inclusion.

Recommendations

- Validate and sanitize file input parameters strictly.
- Allow only whitelisted filenames or file identifiers.
- Block directory traversal patterns (../, %2e%2e%2f).
- Store application files outside the web root.
- Restrict file system permissions.
- Perform regular security testing for file inclusion flaws.

7.11 Vulnerability: Sensitive Information Disclosure (Exposed Admin Directory)

Risk Rating	High
CVSS Score	**8.4**
Affected URL	http://testphp.vulnweb.com/admin/

Risk Description

The application exposes the **admin directory** publicly without access restrictions. Navigating to the /admin path reveals directory contents that contain sensitive organizational and system-related information.

This results in **Sensitive Information Disclosure**, allowing an attacker to gather internal details that may be leveraged for further attacks such as authentication bypass, privilege escalation, or targeted exploitation.

Business Impact

Disclosure of sensitive administrative information

Exposure of internal system or configuration files

Assistance to attackers in reconnaissance activities

Increased risk of further targeted attacks

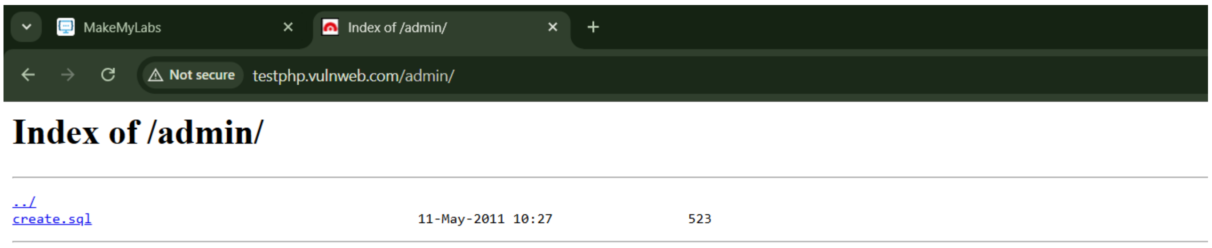
Loss of confidentiality and system security

Steps to Reproduce

1. Open a web browser.
2. Navigate to: http://testphp.vulnweb.com/admin/
3. Observe that the directory contents are displayed without authentication.
4. Review the listed files and information exposed.

This confirms that sensitive administrative resources are publicly accessible.

Output Finding



Index of /admin/

```
../
create.sql                                        11-May-2011 10:27                 523
```

Accessing the /admin directory displays internal files and information without requiring authentication.

Recommendations

- Restrict access to the admin directory using authentication controls.
- Disable directory listing on the web server.
- Move sensitive files outside the web root directory.
- Apply role-based access control for administrative resources.
- Perform regular scans for exposed directories.

7.12 Vulnerability: Exposed Login Credentials (Sensitive Information Disclosure)

Risk Rating	High
CVSS Score	**8.0**
Affected URL	http://testphp.vulnweb.com/pictures/credentials.txt

Risk Description

The application exposes a publicly accessible text file containing valid user login credentials. The file credentials.txt is stored inside a web-accessible directory and can be retrieved directly without authentication.

An attacker can obtain these credentials and reuse them to log in to restricted areas of the application, leading to unauthorized account access.

Business Impact

Unauthorized access to user accounts

Credential leakage in plaintext format

Bypass of authentication controls

Potential privilege escalation

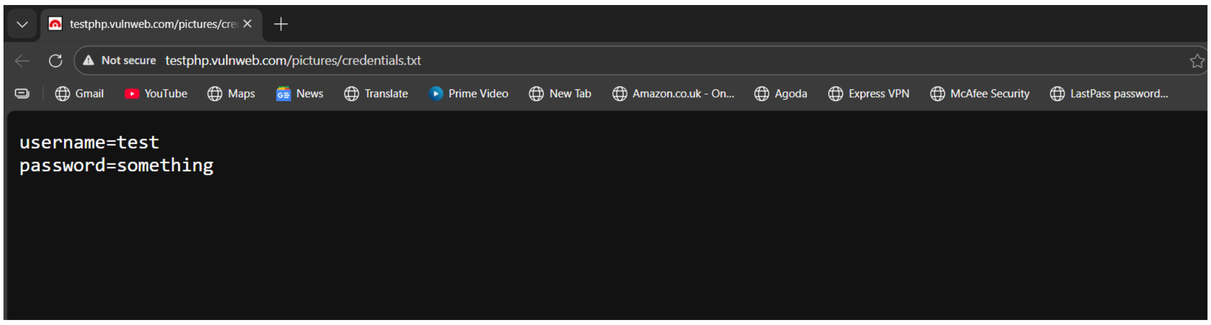
Loss of confidentiality and integrity of user data

Steps to Reproduce

1. Navigate to the following URL: http://testphp.vulnweb.com/pictures/credentials.txt
2. Observe that the file is accessible without authentication.
3. View the exposed plaintext credentials.
4. Use the obtained credentials to log in at: http://testphp.vulnweb.com/login.php

Observe successful login without legitimate authorization.

Output Finding



```
username=test
password=something
```

Request:

GET /pictures/credentials.txt HTTP/1.1

Host: testphp.vulnweb.com

Response:

HTTP/1.1 200 OK

Content-Type: text/plain

Username: test

Password: something

The response confirms exposure of valid login credentials.

Recommendations

- Remove sensitive credential files from publicly accessible directories.
- Never store passwords in plaintext format.
- Implement strict access control on sensitive resources.
- Rotate compromised credentials immediately.
- Perform regular scans for exposed files and directories.

7.13 Vulnerability: Sensitive Information Disclosure (Exposed CVS Directory)

Risk Rating	High
CVSS Score	**8.3**
Affected URL	http://testphp.vulnweb.com/CVS/

Risk Description

The application exposes the **CVS directory** publicly without access restrictions. Navigating to the /CVS path reveals directory contents containing **log history and internal version control data**, which may include sensitive organizational and system-related information.

This leads to **Sensitive Information Disclosure**, allowing attackers to gather internal project structure, file history, and configuration details that can be used for further targeted attacks.

Business Impact

Exposure of internal development and log files

Disclosure of project structure and file history

Assistance in attacker reconnaissance

Increased risk of further exploitation

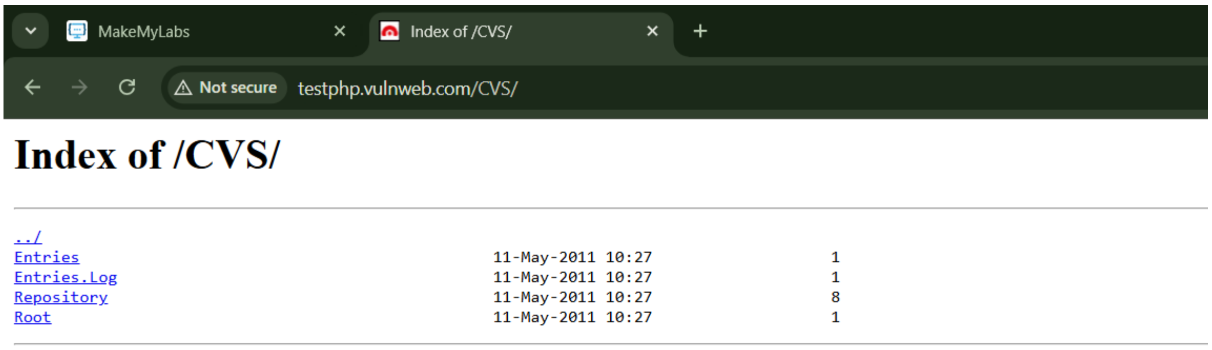
Loss of confidentiality of organizational data

Steps to Reproduce

1. Open a web browser.
2. Navigate to: http://testphp.vulnweb.com/CVS/
3. Observe that the directory contents are displayed without authentication.
4. Review the exposed files and log information.

This confirms that sensitive version control data is publicly accessible.

Output Finding



Index of /CVS/

```
../
Entries                                            11-May-2011 10:27                   1
Entries.Log                                        11-May-2011 10:27                   1
Repository                                         11-May-2011 10:27                   8
Root                                               11-May-2011 10:27                   1
```

Accessing the /CVS directory displays internal log and repository-related files without requiring authentication.

Recommendations

- Disable directory listing on the web server.
- Restrict access to version control directories.
- Remove CVS and repository metadata from the web root.
- Implement proper access control rules.
- Perform regular scans for exposed sensitive directories.

7.14 Vulnerability: Sensitive Information Disclosure (Exposed .idea Directory)

Risk Rating	High
CVSS Score	**7.6**
Affected URL	http://testphp.vulnweb.com/.idea/

Risk Description

The application exposes the **.idea directory** publicly without access restrictions.

The .idea folder is generated by JetBrains IDEs (such as IntelliJ or PhpStorm) and contains **internal project configuration and workspace files**. These files reveal information about the project structure, development environment, file paths, and version control configuration.

Exposing such internal development metadata allows attackers to gather reconnaissance information that can be used for targeted attacks against the application.

This results in **Sensitive Information Disclosure**.

Business Impact

Disclosure of internal project structure and configuration

Exposure of development environment details

Assists attackers in mapping backend architecture

Increases risk of further targeted exploitation

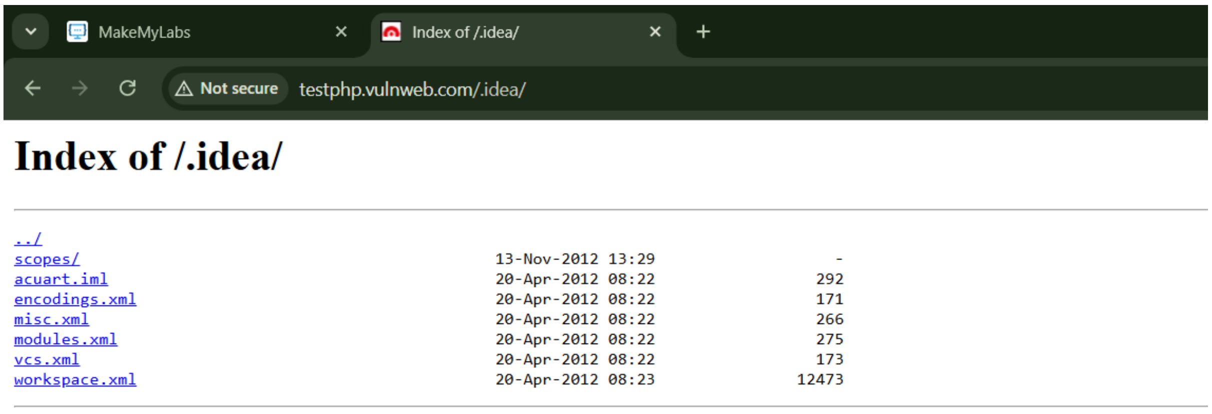
Loss of confidentiality of internal system information

Steps to Reproduce

1. Open a web browser.
2. Navigate to: http://testphp.vulnweb.com/.idea/
3. Observe that the directory listing is displayed publicly.
4. Notice internal configuration files such as:
   - modules.xml
   - misc.xml
   - workspace.xml
   - vcs.xml

This confirms unrestricted access to internal IDE configuration data.

Proof of Concept / Output Finding



Index of /.idea/

```
../
scopes/                                            13-Nov-2012 13:29                   -
acuart.iml                                         20-Apr-2012 08:22                 292
encodings.xml                                      20-Apr-2012 08:22                 171
misc.xml                                           20-Apr-2012 08:22                 266
modules.xml                                        20-Apr-2012 08:22                 275
vcs.xml                                            20-Apr-2012 08:22                 173
workspace.xml                                      20-Apr-2012 08:23               12473
```

The .idea directory was accessible without authentication and displayed internal project configuration files.

Recommendations

- Disable directory listing on the web server.
- Restrict access to internal configuration directories.
- Remove IDE and development configuration files from the web root.
- Implement proper access control rules.
- Perform regular scans for exposed sensitive directories.

7.15 Vulnerability: Sensitive Information Disclosure via Local File Inclusion

Risk Rating	High
CVSS Score	**8.6**
Affected URL	http://testphp.vulnweb.com/showimage.php?file=

Risk Description

The application is vulnerable to **Local File Inclusion (LFI)** due to improper validation of the file parameter in showimage.php.

During testing, the parameter was found to accept user-controlled file paths. By injecting path traversal patterns through automated testing, the application returned contents of sensitive system files. This confirms that internal server files can be accessed without authentication.

This vulnerability results in **Sensitive Information Disclosure**, allowing attackers to read system configuration files and potentially extract user credentials or application secrets.

Business Impact

Disclosure of sensitive system files

Exposure of user account and configuration data

Assistance to attackers in reconnaissance

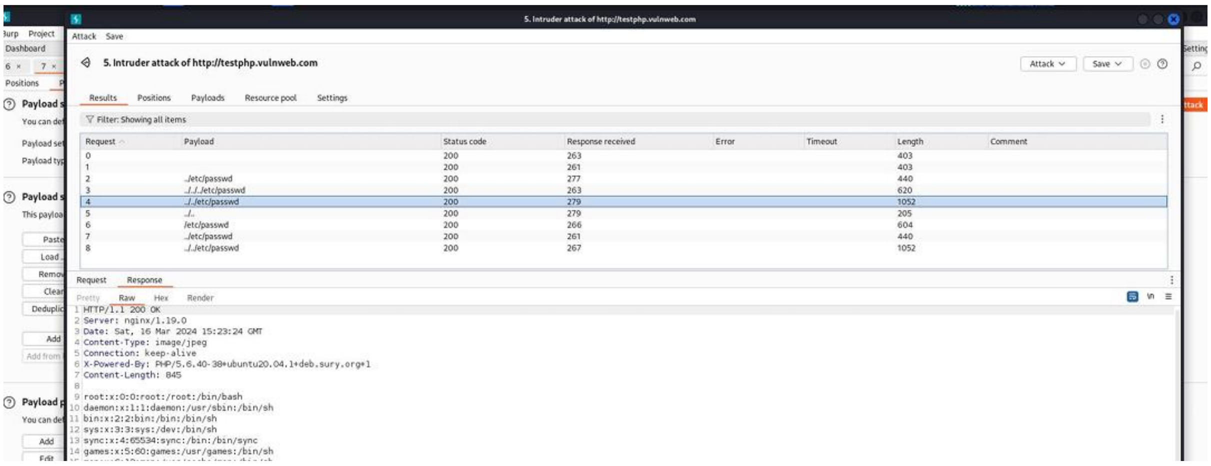
Potential escalation to further system compromise

Loss of confidentiality of backend infrastructure

Steps to Reproduce

1. Navigate to the vulnerable endpoint: http://testphp.vulnweb.com/showimage.php?file=
2. Intercept the request using Burp Suite.
3. Configure Burp Intruder to inject a path traversal wordlist into the file parameter.
4. Send multiple requests with modified file path inputs.
5. Observe that certain requests return HTTP 200 responses with increased response length.
6. Inspect the response content and confirm that sensitive system file data is returned.

Output Finding



5. Intruder attack of http://testphp.vulnweb.com
Attack
Save
Results
Positions
Payloads
Resource pool
Settings
Filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	263			403	
1		200	261			403	
2	../etc/passwd	200	277			440	
3	../../../etc/passwd	200	263			620	
4	../../etc/passwd	200	279			1052	
5	../..	200	279			205	
6	/etc/passwd	200	266			604	
7	../etc/passwd	200	261			440	
8	../../etc/passwd	200	267			1052	

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Sat, 16 Mar 2024 15:23:24 GMT
Content-Type: image/jpeg
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: 845

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
```

Burp Suite Intruder successfully retrieved contents of internal system files.

One of the responses contained: root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:/usr/sbin:/usr/sbin/nologin

...

This confirms successful Local File Inclusion and disclosure of sensitive system information.

Recommendations

- Validate and sanitize all file path inputs strictly.
- Allow only whitelisted filenames for file access.
- Block directory traversal patterns in user input.
- Disable direct file path handling from user-supplied parameters.
- Restrict web server file permissions.
- Perform routine security testing for file inclusion vulnerabilities.

7.16 Vulnerability: Weak Password Policy

Risk Rating	High
CVSS Score	**8.0**
Affected URL	http://testphp.vulnweb.com/login.php

Risk Description

The application follows a **weak password policy**. During testing, it was observed that the login system accepts simple and predictable passwords without enforcing minimum complexity requirements.

Lack of strong password enforcement allows attackers to easily guess or brute-force user passwords, leading to unauthorized account access.

Business Impact

Increased risk of account compromise

Unauthorized access through password guessing

Higher success rate of brute-force attacks

Loss of confidentiality of user accounts

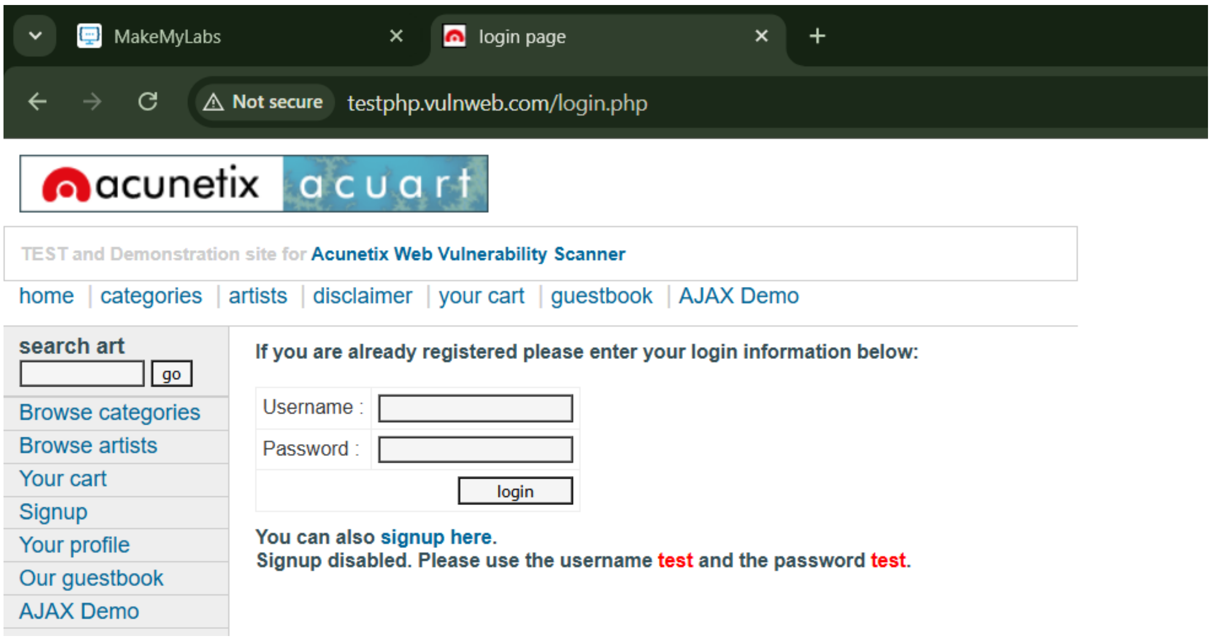
Potential privilege escalation

Steps to Reproduce

1. Navigate to: http://testphp.vulnweb.com/login.php
2. Attempt to log in using simple credentials such as:
   Username:test
   Password: test
3. Observe that the application accepts weak passwords without enforcing complexity rules.

This confirms the absence of a strong password policy.

Output Finding


MakeMyLabs
login page
Not secure testphp.vulnweb.com/login.php

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

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

search art
go

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

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

Username :	
Password :	
	login

You can also signup here.
Signup disabled. Please use the username test and the password test.

Weak credentials such as:

Username: test

Password: test

were accepted successfully by the login system, confirming weak password enforcement.

Recommendations

- Enforce a minimum password length of at least 8 characters.
- Require use of uppercase, lowercase, numbers, and special characters.
- Implement a password strength meter for real-time feedback.
- Store passwords securely using strong hashing algorithms (e.g., bcrypt).
- Introduce Multi-Factor Authentication (MFA) for additional security.
- Perform periodic password policy audits.

7.17 Vulnerability: Outdated PHP Version Disclosure

Risk Rating	High
CVSS Score	**7.4**
Affected URL	http://testphp.vulnweb.com/secured/phpinfo.php

Risk Description

The application exposes a phpinfo() page publicly. This page reveals detailed system and configuration information, including the PHP version running on the server.

From the exposed page, the PHP version was identified as: PHP Version 5.1.6

PHP 5.1.6 is outdated and no longer supported, containing multiple known security vulnerabilities. Public disclosure of software versions helps attackers perform targeted exploitation using known vulnerabilities for that specific version.

This results in Sensitive Information Disclosure and Use of Outdated Components, increasing the risk of system compromise.

Business Impact

Disclosure of internal system and configuration details

Exposure of outdated software versions with known vulnerabilities

Assists attackers in crafting targeted exploits

Potential full system compromise

Loss of confidentiality and system security

Steps to Reproduce

1. Open a web browser.
2. Navigate to: http://testphp.vulnweb.com/secured/phpinfo.php
3. Observe that the phpinfo page is publicly accessible.
4. Note that the page displays: PHP Version 5.1.6

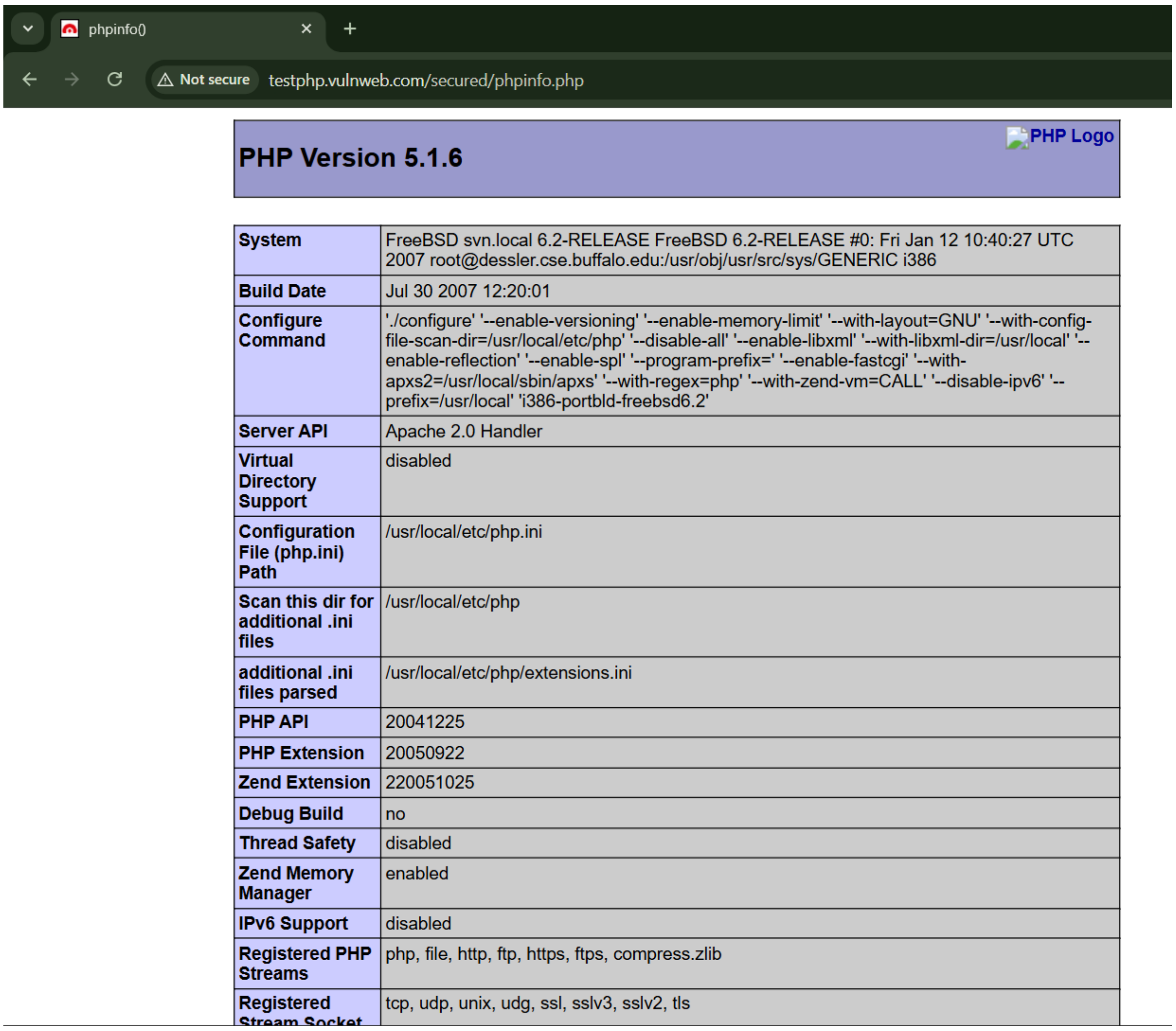
This confirms disclosure of outdated PHP version and system configuration details.

Output Finding

The phpinfo page was accessed without authentication.

The following information was exposed: PHP Version 5.1.6

Along with full system and server configuration details.



phpinfo()

Not secure testphp.vulnweb.com/secured/phpinfo.php

PHP Version 5.1.6

System	FreeBSD svn.local 6.2-RELEASE FreeBSD 6.2-RELEASE #0: Fri Jan 12 10:40:27 UTC 2007 root@dessler.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC i386
Build Date	Jul 30 2007 12:20:01
Configure Command	'./configure' '--enable-versioning' '--enable-memory-limit' '--with-layout=GNU' '--with-config-file-scan-dir=/usr/local/etc/php' '--disable-all' '--enable-libxml' '--with-libxml-dir=/usr/local' '--enable-reflection' '--enable-spl' '--program-prefix=' '--enable-fastcgi' '--with-apxs2=/usr/local/sbin/apxs' '--with-regex=php' '--with-zend-vm=CALL' '--disable-ipv6' '--prefix=/usr/local' 'i386-portbld-freebsd6.2'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php.ini
Scan this dir for additional .ini files	/usr/local/etc/php
additional .ini files parsed	/usr/local/etc/php/extensions.ini
PHP API	20041225
PHP Extension	20050922
Zend Extension	220051025
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	disabled
Registered PHP Streams	php, file, http, ftp, https, ftps, compress.zlib
Registered Stream Socket	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls

Recommendations

- Disable or remove publicly accessible phpinfo pages in production environments.
- Upgrade PHP to a supported and secure latest stable version.
- Restrict access to diagnostic pages using authentication.
- Avoid exposing detailed system configuration information.
- Perform regular patch management and software updates.

7.18 Vulnerability: Clickjacking

Risk Rating	Medium
CVSS Score	**6.5**
Affected URL	http://testphp.vulnweb.com/

Risk Description

The target website is vulnerable to **Clickjacking**. The webpage can be embedded inside an external <iframe> without restriction. This allows an attacker to trick users into clicking hidden or overlaid elements, potentially leading to unauthorized actions such as credential submission, unwanted purchases, or redirection to malicious content.

The absence of the **X-Frame-Options** security header confirms that the application does not prevent framing by external sites.

Business Impact

Users can be tricked into performing unintended actions

Possible credential theft through UI redressing

Unauthorized transactions or form submissions

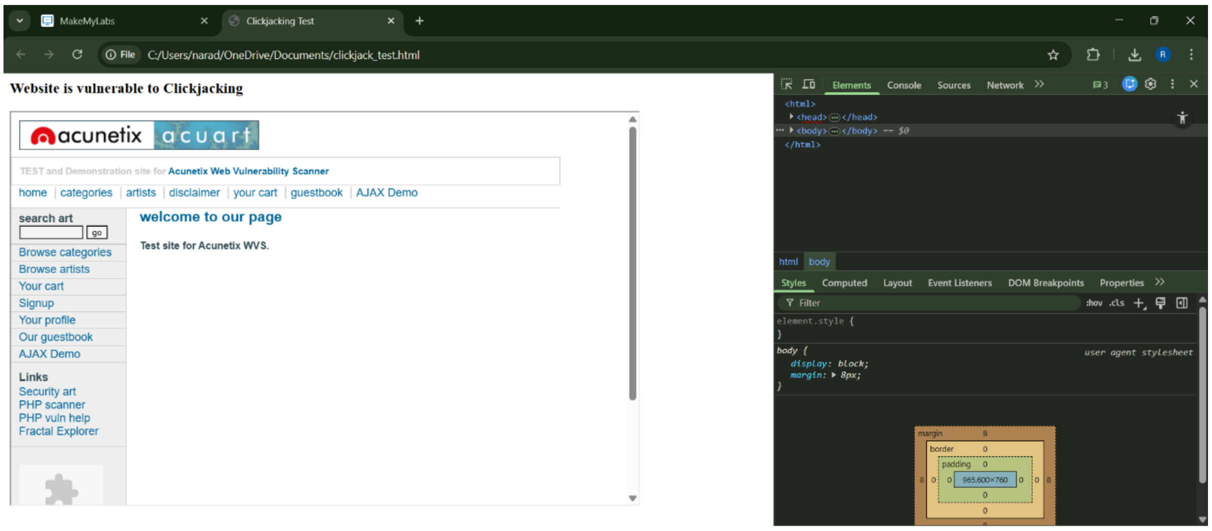
Loss of user trust and reputational damage

Steps to Reproduce

1. Create a local HTML file containing an iframe:
   - ➢ <iframe src="http://testphp.vulnweb.com/" width="800" height="500"></iframe>
2. Open the file in a browser.
3. Observe that the target webpage loads successfully inside the iframe.

This confirms that the page can be framed by external sites.

Output Finding



Website is vulnerable to Clickjacking

welcome to our page

Test site for Acunetix WVS.

```
<html>
  <head> … </head>
  <body> … </body> == $0
</html>
```

```
element.style {
}
body {                              user agent stylesheet
    display: block;
    margin: 8px;
}
```

The target website rendered successfully inside an external iframe without restriction.

Recommendations

- Implement the X-Frame-Options HTTP header:
- X-Frame-Options: DENY
- Or use Content Security Policy:
- Content-Security-Policy: frame-ancestors 'none';
- These headers prevent the page from being embedded in iframes.

7.19 Vulnerability: Insecure Connection (HTTP instead of HTTPS)

Risk Rating	Medium
CVSS Score	**6.0**
Affected URL	http://testphp.vulnweb.com/

Risk Description

The application communicates over **HTTP** instead of **HTTPS**. This means all data exchanged between the user and the server is transmitted in plaintext without encryption. An attacker monitoring the network can intercept and read sensitive information such as login credentials, session cookies, and personal data.

The absence of SSL/TLS encryption exposes users to man-in-the-middle and session hijacking attacks.

Business Impact

Exposure of sensitive user data in transit

Risk of credential and session cookie theft

Possibility of man-in-the-middle attacks

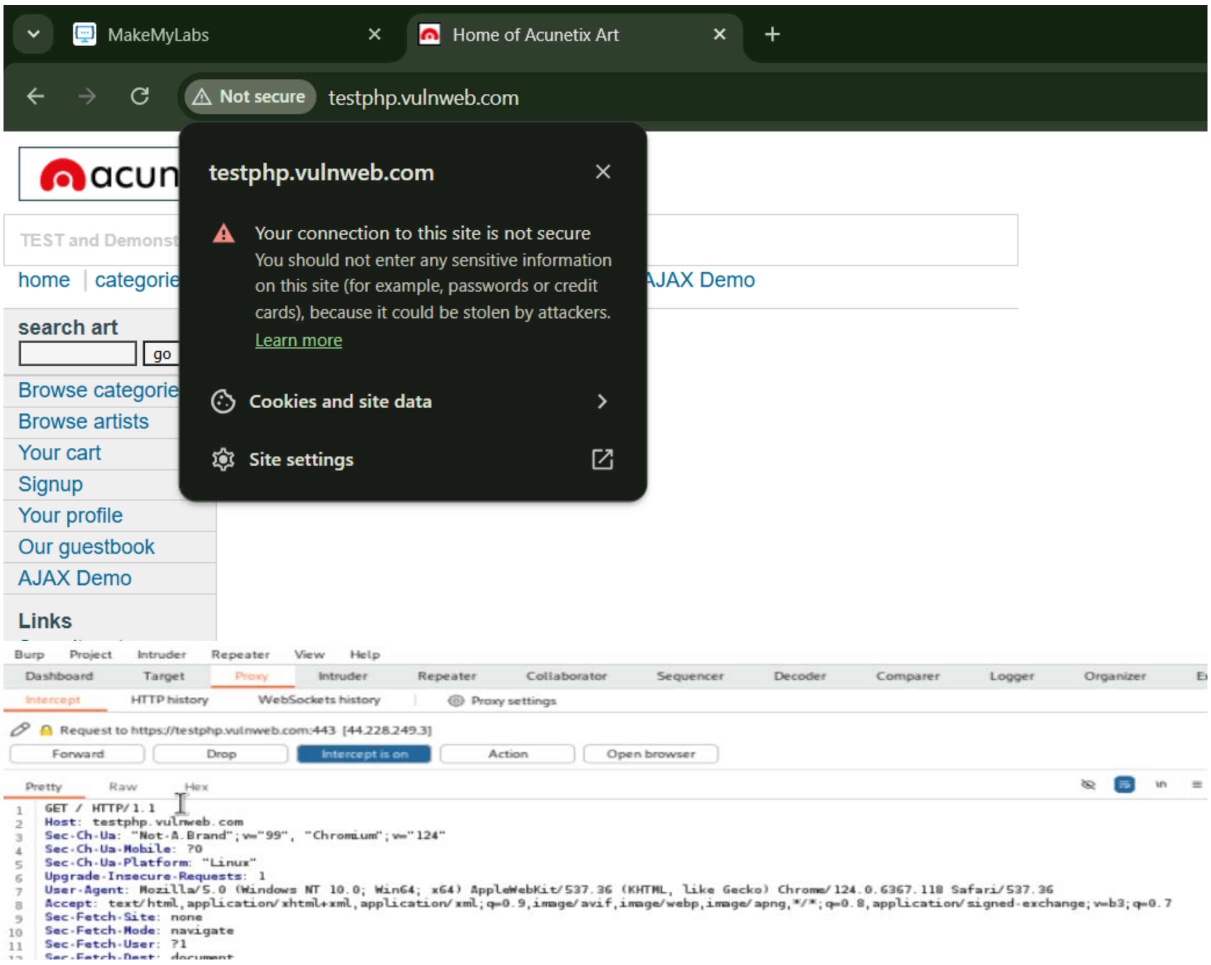
Loss of user trust and reputational damage

Steps to Reproduce

1. Open a web browser.
2. Navigate to: http://testphp.vulnweb.com/
3. Observe that the website loads using **HTTP** and the browser displays a **"Not Secure"** indicator in the address bar.

This confirms that the application does not enforce encrypted HTTPS communication.

Output Finding


MakeMyLabs
Home of Acunetix Art
Not secure
testphp.vulnweb.com
home | categorie
AJAX Demo
search art
go
Browse categorie
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex

testphp.vulnweb.com

Your connection to this site is not secure

You should not enter any sensitive information on this site (for example, passwords or credit cards), because it could be stolen by attackers. Learn more

Cookies and site data

Site settings

Request to https://testphp.vulnweb.com:443 [44.228.249.3]

```
GET / HTTP/1.1
Host: testphp.vulnweb.com
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
```

The website loads successfully over **HTTP** instead of HTTPS.

The browser marks the connection as **Not Secure**, confirming data is transmitted without encryption.

Recommendations

- Obtain a valid SSL/TLS certificate from a trusted Certificate Authority.
- Install and configure SSL on the web server.
- Redirect all HTTP traffic to HTTPS.
- Update all internal links to use HTTPS URLs.
- Regularly test SSL configuration for proper encryption.

7.20 Vulnerability: Missing Multi-Factor Authentication (MFA)

Risk Rating	Medium
CVSS Score	**6.8**
Affected URL	http://testphp.vulnweb.com/login.php

Risk Description

The application does not implement **Multi-Factor Authentication (MFA)** on the login page.

Authentication relies only on username and password. If credentials are guessed, brute-forced, or leaked, an attacker can log in successfully without any additional verification. The absence of MFA significantly increases the risk of unauthorized account access.

Business Impact

Single-factor authentication increases risk of account takeover

Compromised credentials lead directly to unauthorized access

No additional barrier against brute-force or credential-stuffing attacks

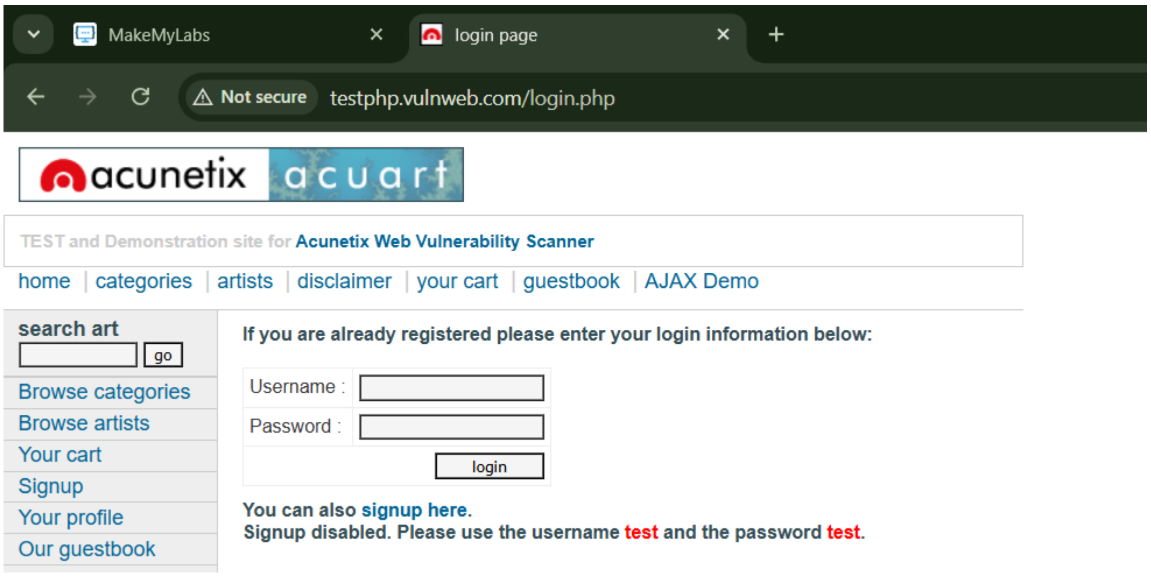
Loss of confidentiality and integrity of user accounts

Steps to Reproduce

1. Navigate to: http://testphp.vulnweb.com/login.php
2. Enter valid username and password.
3. Observe that login is successful without any secondary verification step.

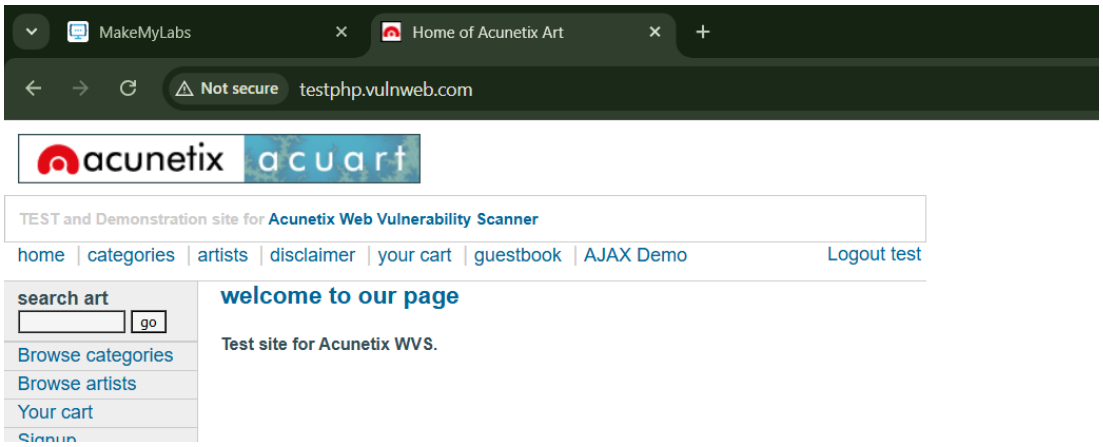
This confirms absence of MFA protection.

Output Finding


MakeMyLabs
login page
Not secure
testphp.vulnweb.com/login.php
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
Username :
Password :
login

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

You can also signup here.
Signup disabled. Please use the username test and the password test.


MakeMyLabs

Home of Acunetix Art

Not secure testphp.vulnweb.com

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

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

search art

Browse categories

Browse artists

Your cart

welcome to our page

Test site for Acunetix WVS.

Login is completed successfully using only username and password.

No OTP, email verification, or mobile confirmation is required.

Recommendations

- Implement Multi-Factor Authentication for all user logins.
- Use One-Time Passwords (OTP) via email or SMS.
- Consider authenticator apps (Google Authenticator, Microsoft Authenticator).
- Require MFA for all administrative accounts.

7.21 Vulnerability: Sensitive Information Disclosure & Use of Deprecated Flash Technology

Risk Rating	Medium
CVSS Score	**6.5**
Affected URL	http://testphp.vulnweb.com/Flash/

Risk Description

The application exposes the **/Flash/** directory publicly without access restrictions. Directory listing is enabled, allowing any user to browse and download internal Flash project files. The directory contains:

- add.fla (Flash source project file)
- add.swf (Compiled Flash file)

Flash source (.fla) files may contain embedded scripts, hardcoded URLs, or internal development logic. Public exposure of such files reveals internal client-side resources and application structure, resulting in **Sensitive Information Disclosure**.

Additionally, the application uses **Adobe Flash (.swf)** content. Flash is a **deprecated client-side technology** that is no longer supported by modern browsers due to historical security vulnerabilities. Continued use of Flash-based components may expose users to client-side security risks.

Business Impact

Exposure of internal client-side development files

Disclosure of embedded scripts or hardcoded endpoints

Assists attackers in application reconnaissance

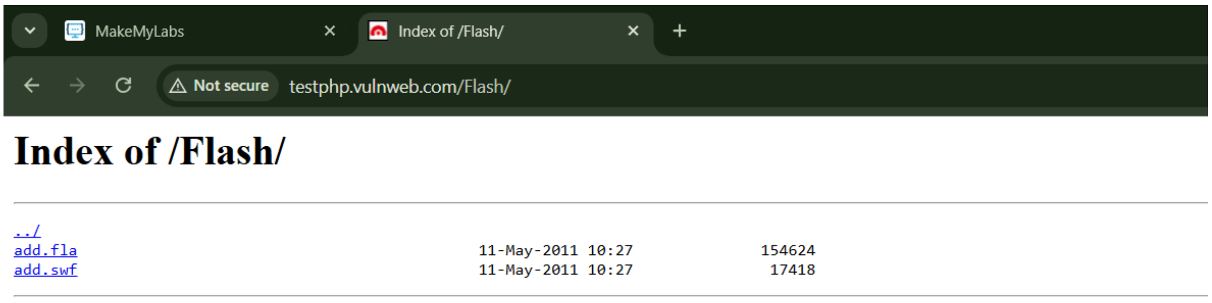
Use of deprecated Flash technology increases client-side security risk

Loss of confidentiality of internal resources

Steps to Reproduce

1. Open a web browser.
2. Navigate to: http://testphp.vulnweb.com/Flash/
3. Observe that directory listing is enabled.
4. Notice that internal Flash files (add.fla, add.swf) are publicly accessible.
5. This confirms unrestricted access to internal development artifacts and usage of deprecated Flash content.

Output Finding



Index of /Flash/

```
../
add.fla                                            11-May-2011 10:27              154624
add.swf                                            11-May-2011 10:27               17418
```

The /Flash/ directory was accessible without authentication and displayed internal Flash project files.

Recommendations

- Disable directory listing on the web server.
- Remove Flash source files from the web root.
- Restrict access to internal resource directories.
- Replace Flash-based content with modern secure technologies such as HTML5.
- Perform regular scans for exposed sensitive directories and outdated client-side components.

7.22 Vulnerability: Cross-Domain Referrer Leakage

Risk Rating	Low
CVSS Score	**4.3**
Affected URL	http://testphp.vulnweb.com/listproducts.php

Risk Description

The application includes links to external third-party domains on pages that are loaded using URLs containing query parameters. When a user's browser requests resources from these external domains, it automatically includes the **Referer HTTP header** containing the full originating URL.

This behavior causes **Cross-Domain Referrer Leakage**, potentially exposing internal application URLs and any sensitive query string parameters to external domains that are not under the application's control.

Business Impact

Disclosure of internal application URLs to third-party domains

Potential leakage of sensitive query parameters

Assists attackers in reconnaissance

Privacy risk for users

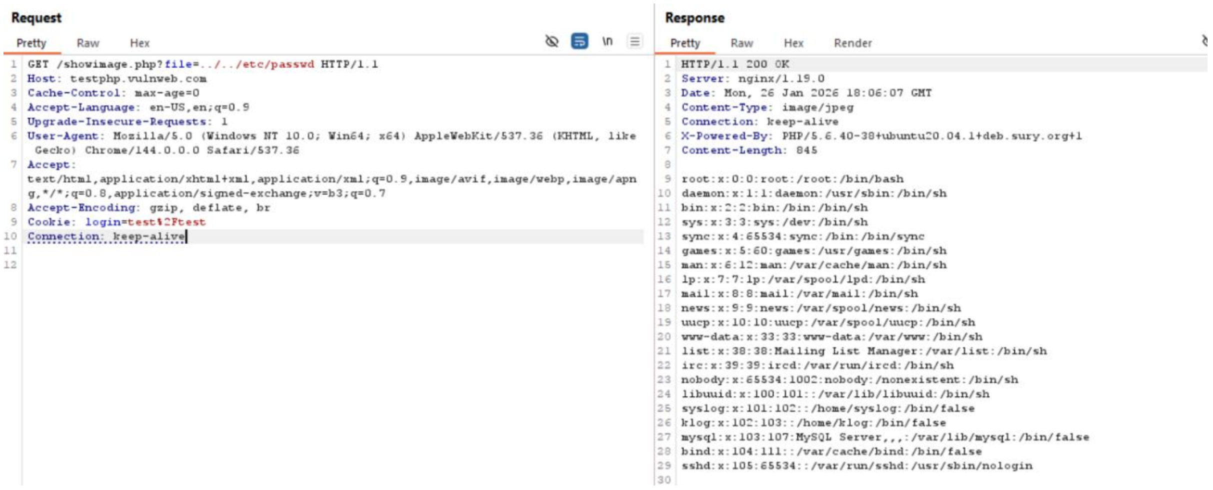
Steps to Reproduce

1. Navigate to: http://testphp.vulnweb.com/showimage.php
2. Intercept the request in Burp Suite.

3. Observe that the page contains links to external domains.
4. When the browser loads these external resources, the request includes a Referer header:

   Referer: http://testphp.vulnweb.com/listproducts.php
5. This confirms cross-domain referrer leakage.

Output Finding



```
GET /showimage.php?file=../../etc/passwd HTTP/1.1
Host: testphp.vulnweb.com
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/144.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Cookie: login=test%2Ftest
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Mon, 26 Jan 2026 18:06:07 GMT
Content-Type: image/jpeg
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: 845

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
nobody:x:65534:1002:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:102::/home/syslog:/bin/false
klog:x:102:103::/home/klog:/bin/false
mysql:x:103:107:MySQL Server,,,:/var/lib/mysql:/bin/false
bind:x:104:111::/var/cache/bind:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
```

Burp Suite captured external domain requests containing the Referer header with the internal application URL.

Recommendations

• Avoid transmitting sensitive information in URL query strings.

• Prevent leakage of internal URLs to external third-party domains.

• Limit or remove unnecessary external links on sensitive application pages.

• Implement proper referrer control policies to restrict information sharing.

• Conduct regular security reviews to identify unintended data exposure.

7.23 Vulnerability: Insecure Cookie Configuration (Missing Secure and HttpOnly Flags)

Risk Rating	Low
CVSS Score	**5.3**
Affected URL	http://testphp.vulnweb.com/login.php http://testphp.vulnweb.com/userinfo.php

Risk Description

The application sets session cookies without the **Secure** and **HttpOnly** attributes.

- The **Secure** flag ensures cookies are transmitted only over HTTPS connections. Its absence allows cookies to be sent over unencrypted HTTP.
- The **HttpOnly** flag prevents client-side scripts from accessing cookies. Its absence increases the risk of cookie theft through XSS attacks.

This insecure cookie configuration increases the risk of **session hijacking** and **unauthorized account access**.

Business Impact

Possible interception of session cookies over unsecured networks

Increased risk of session hijacking

Potential unauthorized access to user accounts

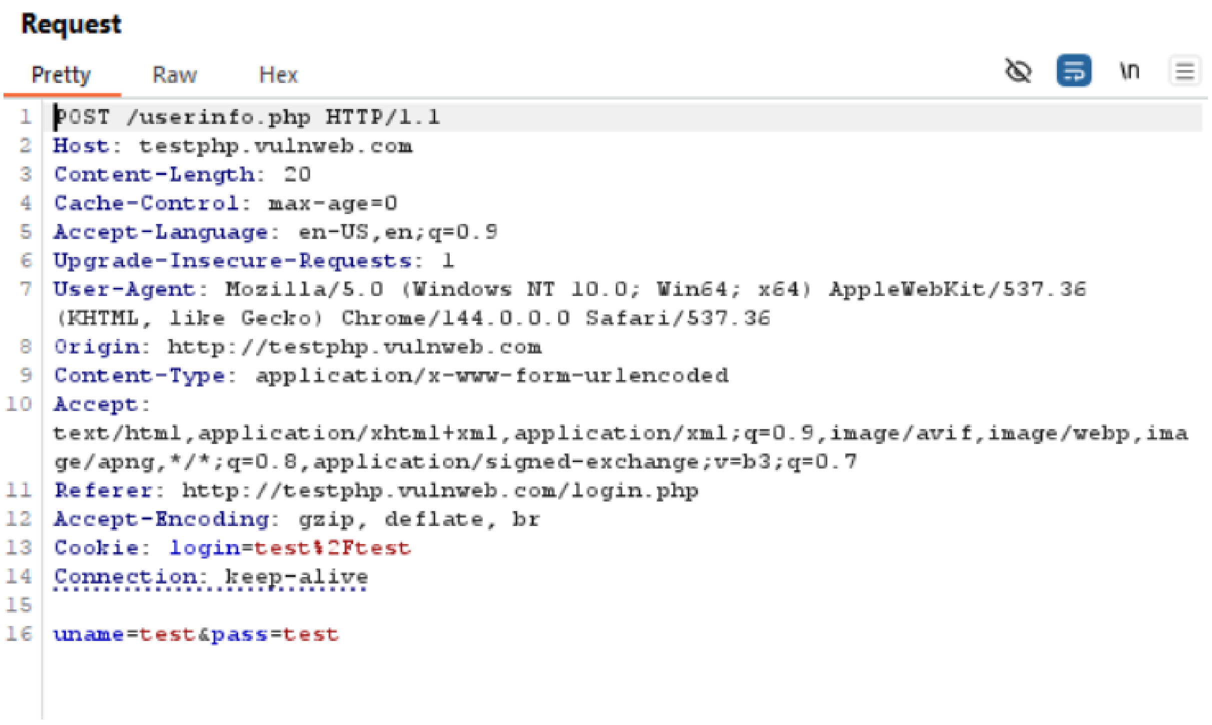
Reduced overall session security

Steps to Reproduce

1. Navigate to: http://testphp.vulnweb.com/login.php
2. Log in using valid credentials (test / test).
3. Intercept the authenticated request in Burp Suite.
4. Observe the request headers containing:
5. Cookie: login=test%2Ftest
6. Inspect the corresponding server response.
7. Notice that the **Set-Cookie** header does not include:
   - Secure
   - HttpOnly

This confirms insecure cookie configuration.

Output Finding



```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Content-Length: 20
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/144.0.0.0 Safari/537.36
Origin: http://testphp.vulnweb.com
Content-Type: application/x-www-form-urlencoded
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://testphp.vulnweb.com/login.php
Accept-Encoding: gzip, deflate, br
Cookie: login=test%2Ftest
Connection: keep-alive

uname=test&pass=test
```

Burp Suite captured session cookies without Secure and HttpOnly attributes.

Recommendations

- Enable the **Secure** flag to ensure cookies are sent only over HTTPS.
- Enable the **HttpOnly** flag to prevent JavaScript access to cookies.
- Enforce HTTPS across the entire application.
- Implement proper session management best practices.

REFERENCES

https://owasp.org/www-project-top-ten/

https://portswigger.net/web-security

https://owasp.org/www-community/attacks/SQL_Injection

https://owasp.org/www-community/attacks/xss/

https://www.acunetix.com/blog/articles/

8. CONCLUSION

The security assessment of **testphp.vulnweb.com** was conducted to identify vulnerabilities affecting the confidentiality, integrity, and availability of the application. The assessment revealed multiple critical, high, medium, and informational security weaknesses, including SQL Injection, Cross-Site Scripting (XSS), Path Traversal, Brute Force susceptibility, sensitive information disclosures, and missing security controls.

Exploitation of these vulnerabilities demonstrates that an attacker could gain unauthorized access, extract or manipulate backend database content, compromise user accounts, disclose internal application resources, and perform client-side attacks against users of the application. Additionally, several security misconfigurations and missing hardening measures were identified, which increase the overall attack surface of the application.

Although testphp.vulnweb.com is an intentionally vulnerable demonstration environment, the findings reflect real-world security risks that could lead to severe compromise if present in a production system. Implementing the recommended remediation measures — including secure coding practices, input validation, proper authentication controls, access restrictions, and security header configurations — will significantly improve the security posture of the application.

This assessment successfully highlights the importance of regular security testing, proactive vulnerability management, and adherence to industry best practices to protect modern web applications from evolving threats.

9. IMPROVEMENTS

Critical

- Replace dynamic SQL queries with parameterized queries.
- Sanitize and encode all user inputs before displaying or storing.
- Remove exposed usernames and passwords from application pages.

High

- Suppress database error messages to prevent blind SQL inference.
- Add login attempt limits to stop brute-force attacks.
- Restrict file access in showimage.php to prevent path traversal.
- Remove exposed credential, CVS, and IDE directories from web root.
- Enforce strong password policy.
- Upgrade PHP version and hide version disclosure.

Medium

- Prevent pages from loading inside iframes (stop clickjacking).
- Enforce HTTPS for all pages.
- Add multi-factor authentication for login.
- Remove deprecated Flash content and restrict Flash directory access.

Low

- Prevent referrer leakage to external domains.
- Set Secure and HttpOnly flags on session cookies.

Informational

- Hide server and technology banners from HTTP headers.
- Add standard security headers for browser protection.
- Periodically review publicly available OSINT information.

10. APPENDIX

Appendix Summary

- Provides supporting information on the completeness of the security assessment.
- Summarizes testing effectiveness and methodology used.
- Demonstrates coverage of major web application security risks.
- Confirms alignment with industry-standard security assessment practices.
- Includes qualitative observations of testing process and results.

Assessment Coverage Overview

- Manual vulnerability testing using browser-based techniques.
- Proxy-based request interception and manipulation using Burp Suite.
- Automated vulnerability scanning and directory enumeration.
- Database exploitation and injection testing.
- Authentication and session management testing.
- File handling and path traversal testing.
- Security configuration and header analysis.

10.1 Quality of Assessment Findings

Qualitative Observations

- Critical vulnerabilities such as SQL Injection and XSS were successfully identified and exploited.
- High-risk findings exposed weaknesses in authentication, file handling, and access control.
- Medium-risk findings revealed missing security controls and deprecated technologies.
- Low and informational findings highlighted security hardening gaps.
- Findings reflect real-world web application security risks.
- Testing approach was systematic and comprehensive.

Assessment Quality Summary Table

Assessment Area	**Testing Approach Used**	**Outcome**
Reconnaissance & OSINT	WHOIS, DNS, Subdomain Enumeration	Infrastructure information identified
Input Validation	Manual Payload Testing	Injection and XSS vulnerabilities found
Authentication Testing	Brute-force & Login Analysis	Account takeover risks confirmed
File Handling	Path Traversal & LFI Testing	Internal file disclosure confirmed
Access Control	Directory Enumeration	Exposed internal directories found
Security Configuration	Header & Cookie Analysis	Missing security controls identified
Information Disclosure	Manual & Automated Review	Sensitive data exposure confirmed
Reporting	Documentation & Risk Mapping	Complete vulnerability report prepared

11. OWASP Top 10 Comparison

OWASP-Category (2025/2026)	**Status**	**Mapped Findings from Assessment**
A01: Broken Access Control	✕ Fail	Exposed /admin, /CVS, /.idea directories accessible without authentication
A02:Cryptographic Failures	✕ Fail	Application uses HTTP instead of HTTPS, cookies missing Secure flag
A03: Injection	✕ Fail	SQL Injection, Blind SQL Injection, Local File Inclusion / Path Traversal
A04: Insecure Design	✕ Fail	Weak password policy, Missing Multi-Factor Authentication
A05:Security Misconfiguration	✕ Fail	Directory listing enabled, Missing security headers, Insecure cookie configuration
A06: Vulnerable & Outdated Components	✕ Fail	Outdated PHP version, Deprecated Flash technology
A07: Identification & Authentication Failures	✕ Fail	Login bypass via SQL Injection, Brute-force login, Exposed credentials
A08: Software & Data Integrity Failures	✕ Fail	Exposed CVS repository and IDE configuration files revealing internal project integrity data
A09: Security Logging & Monitoring Failures	✕ Fail	Unlimited brute-force attempts allowed with no detection or blocking mechanisms
A10:Server-Side Request Forgery (SSRF)	✕ Fail	File inclusion endpoint (showimage.php) accepts user-controlled file paths, enabling server-side resource access behaviour