2026

Web Application Security Assessment Report

CONFIDENTIAL – SAMPLE REPORT

ILIAS BELHARDA

Table des matières

Introduction

This report presents the results of a web application security assessment conducted on the target website as part of a controlled and non-intrusive testing engagement.

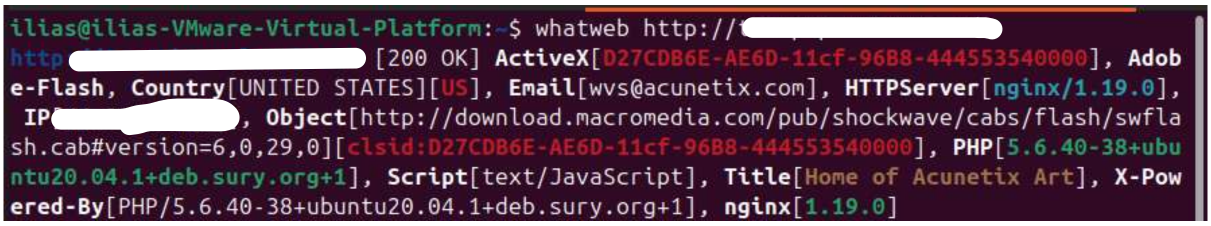
The objective of this assessment was to identify potential security vulnerabilities, misconfigurations, and weaknesses that could expose the application to exploitation. The testing methodology followed a black-box approach, meaning the assessment was performed without prior knowledge of the application's internal architecture.

The evaluation included:

- Network exposure analysis
- SSL/TLS configuration review
- Web server misconfiguration assessment
- Automated vulnerability scanning aligned with OWASP Top 10

All testing activities were conducted using industry-standard tools including Nmap, Nikto, SSLScan, and OWASP ZAP.

Recon Phase



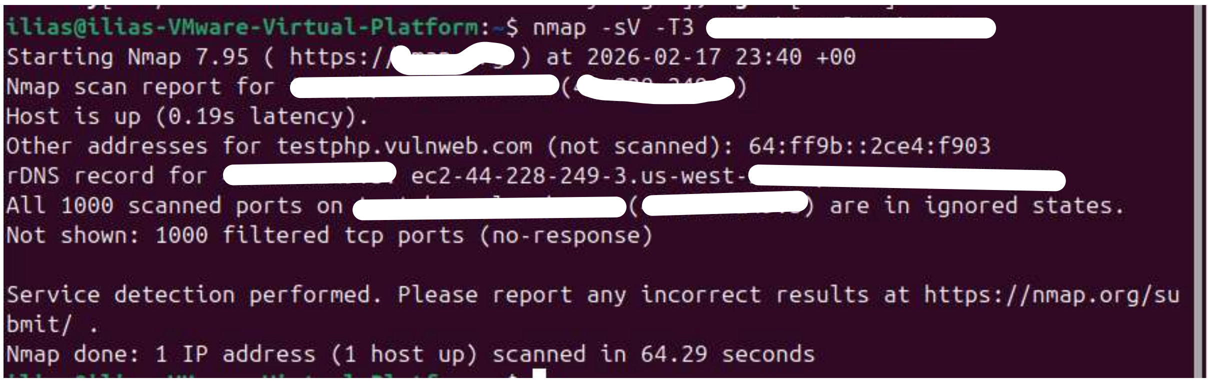
```
ilias@ilias-VMware-Virtual-Platform:~$ whatweb http://[redacted]
http[redacted] [200 OK] ActiveX[D27CDB6E-AE6D-11cf-96B8-444553540000], Adob
e-Flash, Country[UNITED STATES][US], Email[wvs@acunetix.com], HTTPServer[nginx/1.19.0],
 IP[redacted], Object[http://download.macromedia.com/pub/shockwave/cabs/flash/swfla
sh.cab#version=6,0,29,0][clsid:D27CDB6E-AE6D-11cf-96B8-444553540000], PHP[5.6.40-38+ubu
ntu20.04.1+deb.sury.org+1], Script[text/JavaScript], Title[Home of Acunetix Art], X-Pow
ered-By[PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1], nginx[1.19.0]
```

Figure 1Technology Identification

The target website was analyzed to identify the underlying technologies. The website is running on Apache HTTP Server and uses PHP-based architecture. No explicit CMS was detected.

Risk: Informational

Network & Service Scan



```
ilias@ilias-VMware-Virtual-Platform:~$ nmap -sV -T3 [redacted]
Starting Nmap 7.95 ( https:/[redacted] ) at 2026-02-17 23:40 +00
Nmap scan report for [redacted]([redacted])
Host is up (0.19s latency).
Other addresses for testphp.vulnweb.com (not scanned): 64:ff9b::2ce4:f903
rDNS record for [redacted] ec2-44-228-249-3.us-west-[redacted]
All 1000 scanned ports on [redacted]([redacted]) are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)

Service detection performed. Please report any incorrect results at https://nmap.org/su
bmit/ .
Nmap done: 1 IP address (1 host up) scanned in 64.29 seconds
```

Figure 2 Open Ports & Services

Finding:

All scanned ports were filtered (no-response).

Description:

A service version detection scan was performed using Nmap. The results indicate that all 1000 commonly scanned TCP ports are filtered. This suggests the presence of a firewall or security group restricting direct network access.

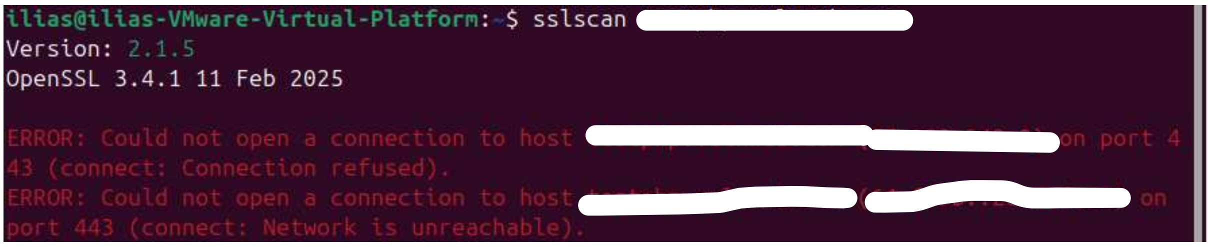
Risk Level:

Low (Positive Security Control)

Security Observation:

The target appears to be hosted on Amazon Web Services (AWS), and firewall filtering is effectively limiting external attack surface exposure.

SSL Analysis



```
ilias@ilias-VMware-Virtual-Platform:~$ sslscan [redacted]
Version: 2.1.5
OpenSSL 3.4.1 11 Feb 2025

ERROR: Could not open a connection to host [redacted][redacted]on port 4
43 (connect: Connection refused).
ERROR: Could not open a connection to host [redacted][redacted] on
port 443 (connect: Network is unreachable).
```

Figure 3 SSL/TLS Configuration Assessment

Finding:

HTTPS service not available (Port 443 closed)

Description:

An SSL/TLS scan was attempted on port 443. The connection was refused, indicating that HTTPS is not enabled on the target server.

Risk Level:

Medium

Impact:

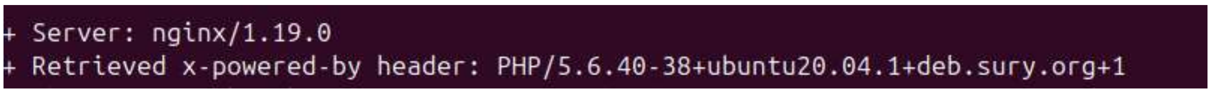
Without HTTPS encryption, data transmitted between users and the server may be intercepted or modified by attackers (Man-in-the-Middle attacks).

Recommendation:

Configure HTTPS using a valid SSL/TLS certificate (e.g., Let's Encrypt) and redirect all HTTP traffic to HTTPS.

Web Vulnerability Scan

Server Information Disclosure



```
+ Server: nginx/1.19.0
+ Retrieved x-powered-by header: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
```

Finding:

Server version and PHP version disclosed.

Risk Level:

Medium

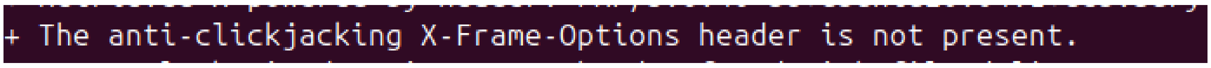
Impact:

Exposing server and PHP versions allows attackers to identify known vulnerabilities associated with outdated software versions.

Recommendation:

Disable server version disclosure in Nginx configuration and remove X-Powered-By header.

Missing Security Header



```
+ The anti-clickjacking X-Frame-Options header is not present.
```

Finding:

Missing X-Frame-Options header.

Risk Level:

Low

Impact:

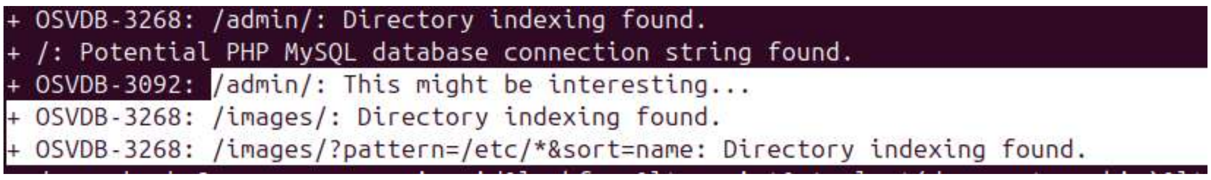
This may allow clickjacking attacks where malicious websites embed the target site within an iframe.

Recommendation:

Add the following header in server configuration:

X-Frame-Options: SAMEORIGIN

Directory Indexing Enabled



```
+ OSVDB-3268: /admin/: Directory indexing found.
+ /: Potential PHP MySQL database connection string found.
+ OSVDB-3092: /admin/: This might be interesting...
+ OSVDB-3268: /images/: Directory indexing found.
+ OSVDB-3268: /images/?pattern=/etc/*&sort=name: Directory indexing found.
```

Finding:

Directory listing enabled in multiple locations.

Risk Level:

Medium

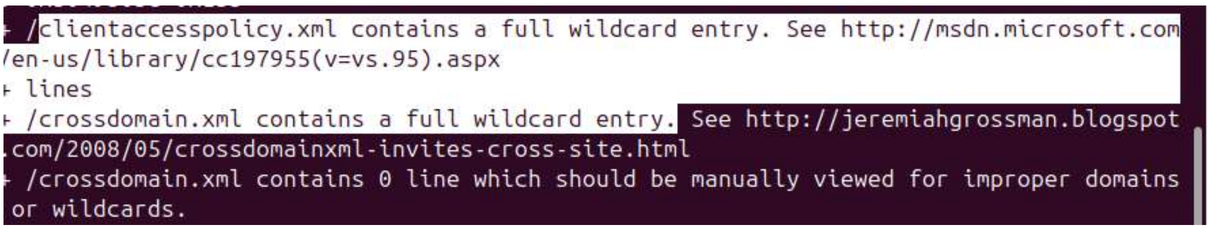
Impact:

Attackers may enumerate files, discover sensitive resources, and gather intelligence for further attacks.

Recommendation:

Disable directory indexing in the web server configuration.

Crossdomain & Clientaccesspolicy Misconfiguration



```
+ /clientaccesspolicy.xml contains a full wildcard entry. See http://msdn.microsoft.com
/en-us/library/cc197955(v=vs.95).aspx
+ lines
+ /crossdomain.xml contains a full wildcard entry. See http://jeremiahgrossman.blogspot
.com/2008/05/crossdomainxml-invites-cross-site.html
+ /crossdomain.xml contains 0 line which should be manually viewed for improper domains
or wildcards.
```

Finding:

Overly permissive cross-domain policy.

Risk Level:

Medium

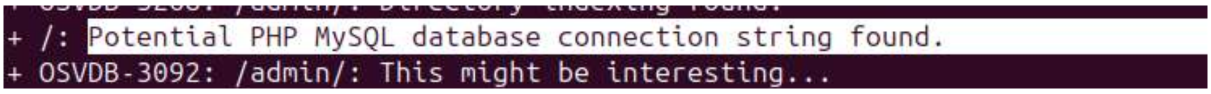
Impact:

May allow unauthorized cross-domain data access.

Recommendation:

Restrict allowed domains instead of using wildcard (*).

Potential Database Connection Strings



```
+ /: Potential PHP MySQL database connection string found.
+ OSVDB-3092: /admin/: This might be interesting...
```

Finding:

Potential database connection string exposure.

Risk Level:

High (in real-world scenario)

Impact:

If connection strings are exposed, attackers could attempt database exploitation.

Recommendation:

Ensure sensitive configuration data is not exposed in accessible files.

Finding	**Risk**	**Status**
Server Version Disclosure	Medium	Open
Missing Security Headers	Low	Open
Directory Indexing	Medium	Open
Crossdomain Misconfiguration	Medium	Open
Potential DB String Exposure	High	Open

Passive Scan (OWASP ZAP)

An automated active scan was performed using OWASP ZAP to detect common web application vulnerabilities aligned with OWASP Top 10.

OWASP ZAP – Automated Scan Results

The screenshot displays the results of an active vulnerability scan performed using OWASP ZAP against the target application (testphp.vulnweb.com).

The Alerts panel shows multiple detected security issues categorized by severity. Notably:

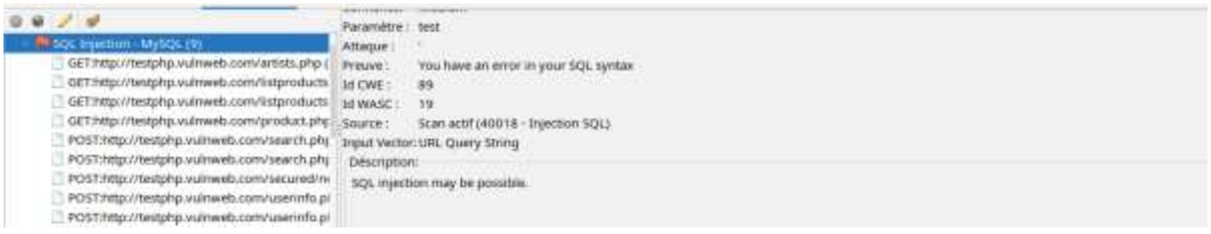
- **SQL Injection (MySQL)** vulnerabilities were identified across multiple endpoints, indicating improper input validation and potential database exposure.
- **Absence of Anti-CSRF Tokens** was detected in HTML forms, suggesting that the application lacks protection against Cross-Site Request Forgery attacks.
- Several **missing security headers** were identified, including Content Security Policy (CSP) and anti-clickjacking headers.
- **Server information disclosure** was observed via HTTP response headers (e.g., Nginx and PHP version exposure).

The right panel shows detailed alert information including:

- Parameter affected
- CWE classification
- Risk description
- Evidence from HTTP responses
- Recommended remediation guidance

This confirms the presence of multiple security weaknesses aligned with OWASP Top 10 categories.

Finding: SQL Injection (MySQL)



```
SQL Injection - MySQL (9)
  GET:http://testphp.vulnweb.com/artists.php (
  GET:http://testphp.vulnweb.com/listproducts
  GET:http://testphp.vulnweb.com/listproducts
  GET:http://testphp.vulnweb.com/product.php
  POST:http://testphp.vulnweb.com/search.php
  POST:http://testphp.vulnweb.com/search.php
  POST:http://testphp.vulnweb.com/secured/n
  POST:http://testphp.vulnweb.com/userinfo.p
  POST:http://testphp.vulnweb.com/userinfo.p
```

```
Paramètre :   test
Attaque :     '
Preuve :      You have an error in your SQL syntax
Id CWE :      89
Id WASC :     19
Source :      Scan actif (40018 - Injection SQL)
Input Vector: URL Query String
Description:
 SQL injection may be possible.
```

Risk Level:

High

CWE:

CWE-89 (SQL Injection)

Description:

The application appears vulnerable to SQL Injection attacks. The scanner identified error-based SQL responses indicating that user input is not properly sanitized before being processed by the database.

Evidence:

ZAP detected the following response:

"You have an error in your SQL syntax"

Affected endpoints include:

- /artists.php
- /listproducts
- /product.php
- /search.php
- /userinfo.php

(Screenshot this alert panel)

Impact (Very Important)

SQL Injection vulnerabilities may allow attackers to:

• Extract sensitive database information
• Bypass authentication
• Modify or delete records
• Gain administrative access

In real-world scenarios, this can lead to full database compromise.

Recommendation

• Use prepared statements (parameterized queries)
• Implement input validation

• Apply least privilege database accounts
• Avoid exposing SQL error messages

Example secure PHP practice:
Use PDO prepared statements instead of raw SQL concatenation.

Conclusion

The security assessment identified several vulnerabilities of varying severity levels within the target application.

Most notably, a high-risk SQL Injection vulnerability was detected, which could potentially allow attackers to access or manipulate backend database systems. Additionally, multiple medium-risk issues were identified, including the absence of CSRF protection mechanisms and insecure server configurations.

While no critical remote code execution vulnerabilities were observed during this non-intrusive engagement, the presence of injection flaws and configuration weaknesses significantly increases the attack surface of the application.

It is strongly recommended that remediation efforts prioritize:

1. Mitigation of SQL Injection vulnerabilities through parameterized queries
2. Implementation of Anti-CSRF protections
3. Hardening of HTTP security headers
4. Removal of unnecessary server information disclosure

Addressing these findings will significantly improve the overall security posture and reduce the risk of exploitation.