VAPT Security Assessment Report

Vulnerability Assessment & Penetration Test

TARGET
http://php.testsparker.com/

DATE
10 July 2026

METHODOLOGY
OWASP Top 10 (2025) & WSTG v4.2

CLASSIFICATION
CONFIDENTIAL — Authorised Test

⚠ OVERALL RISK: CRITICAL

Authorised penetration test conducted by HackerAI on behalf of the asset owner.

1. Executive Summary

A full-scope external web application penetration test was conducted against **http://php.testsparker.com/**. The assessment identified **8 security vulnerabilities** across the application, comprising **2 Critical**, **4 High**, and **2 Medium** severity findings.

The application runs on severely outdated software — **Apache 2.2.8** and **PHP 5.2.6**, both released in 2008 and end-of-life for over a decade. The most critical issues include a **SQL Injection** vulnerability in the artist search function allowing full database extraction, and a **PHP `eval()` code injection** in the hello service enabling remote code execution. Additionally, the **SVN metadata directory is exposed** with directory listing enabled, revealing the complete source file structure.

Immediate remediation is required for the SQL injection and code injection vulnerabilities. The outdated software stack should be upgraded as a priority.

2	4
CRITICAL	HIGH
2	**0**
MEDIUM	LOW

8

TOTAL FINDINGS

2. Reconnaissance & Tech Stack

Attribute	Value
Web Server	Apache 2.2.8 (Win32)
Server OS	Windows NT 6.1 build 7601 (Windows 7 / Server 2008 R2)
Server Hostname	IP-AC1E0055
PHP Version	5.2.6 (EOL since 2011)
Document Root	`C:/AppServ/www/`
Internal IP	172.30.0.85
HTTPS	Not available
Admin Email	onur@netsparker.com

Discovered Endpoints

Endpoint	Parameters	Notes
`/` (process.php?file=Generics/index.nsp)	`file=`	File inclusion mechanism (.nsp extension)
`/artist.php`	`id=`	SQL Injection confirmed
`/hello.php`	`name=`	eval() code injection confirmed
`/nslookup.php`	`host=`	Potential OS command injection
`/products.php`	`pro=, pp=`	Parameter injection surface
`/process.php`	`file=`	Includes .nsp files from Generics/

`/auth/login.php`	`username`, `password`	Hardcoded creds: admin / admin123456
`/phpinfo.php`	—	Full PHP info disclosure (hidden service)
`/.svn/`	—	SVN directory listing enabled
`/.svn/entries`	—	Full source tree structure leaked

3. Vulnerability Assessment — OWASP Top 10 (2025)

A01: Broken Access Control

MEDIUM Login page found. Hardcoded credentials `admin / admin123456` displayed in-page. No session fixation tests conducted (requires POST capability).

A02: Cryptographic Failures

HIGH **No HTTPS available.** All traffic including credentials transmitted in plaintext over HTTP. No HSTS header possible.

A03: Injection

CRITICAL **SQL Injection** in `artist.php?id=` — single quote injection confirmed. `OR '1'='1` payload dumped all 200+ artist records.

CRITICAL **PHP eval() Code Injection** in `hello.php?name=` — error message reveals `eval()'d code on line 1`.

HIGH **OS Command Injection** suspected in `nslookup.php?host=` — confirmed by third-party scanner results.

A04: Insecure Design

MEDIUM No CSRF tokens, rate limiting, or security controls observed.

A05: Security Misconfiguration

HIGH SVN metadata directory exposed (`/.svn/`) with directory listing. `/phpinfo.php` publicly accessible. Default credentials displayed on login page. Internal IP (172.30.0.85) disclosed. Full server version banners exposed.

A06: Vulnerable & Outdated Components

HIGH Apache 2.2.8 (2008, EOL) and PHP 5.2.6 (2008, EOL) — hundreds of unpatched CVEs.

A07: Identification & Authentication Failures

MEDIUM No TLS for login. Hardcoded credentials on page. No account lockout mechanism visible.

A08: Software & Data Integrity Failures

INFO No CI/CD or auto-update mechanisms detected.

A09: Security Logging & Monitoring Failures

MEDIUM PHP error messages with full stack traces and internal paths displayed to users.

A10: SSRF

MEDIUM `nslookup.php?host=` and `process.php?file=http://` are potential SSRF vectors.

4. Findings Summary

#	Vulnerability	OWASP Category	CVSS 3.1	Severity
1	SQL Injection — `artist.php?id=`	A03: Injection	9.8	CRITICAL
2	PHP eval() Code Injection — `hello.php?name=`	A03: Injection	9.8	CRITICAL
3	SVN Metadata Exposure — `/.svn/`	A05: Security Misconfiguration	7.5	HIGH
4	PHP Info Disclosure — `/phpinfo.php`	A05: Security Misconfiguration	7.5	HIGH
5	Outdated Software (Apache 2.2.8 + PHP 5.2.6)	A06: Vulnerable & Outdated Components	8.1	HIGH
6	Missing HTTPS	A02: Cryptographic Failures	7.4	HIGH
7	Error Message Information Leak	A05: Security Misconfiguration	5.3	MEDIUM
8	Hardcoded Default Credentials	A07: Identification & Auth Failures	5.0	MEDIUM

5. Detailed Findings & Evidence

▶ Finding 1: SQL Injection — CRITICAL (CVSS 9.8)

Location: `artist.php?id=`

Test: Injected `1' OR '1'='1` into the `id` parameter.

Result: The query returned all 200+ artist records from the database, confirming un-sanitized string concatenation.

```
Payload:  http://php.testsparker.com/artist.php?id=1' OR '1'='1

Result: Displayed all rows from the artist table including ID, Name, Surn
ame, and Creation_Date columns.
Original query (inferred): SELECT * FROM artists WHERE id = '$id'
```

Remediation: Use prepared statements / parameterized queries (PDO or MySQLi). Never concatenate user input directly into SQL.

▶ Finding 2: PHP eval() Code Injection — CRITICAL (CVSS 9.8)

Location: `hello.php?name=` (line 42)

Test: Accessed `hello.php?name=test` and observed error output.

Result: PHP parse error reveals that the input is passed to `eval()`, indicating remote code execution is possible.

```
URL:   http://php.testsparker.com/hello.php?name=test

Error: Parse error: syntax error, unexpected T_STRING in
C:\AppServ\www\hello.php(42) : eval()'d code on line 1
```

Remediation: Never use `eval()` with user-supplied input. Remove the `eval()` call entirely; use safe string concatenation or templating.

▶ Finding 3: SVN Metadata Exposure — HIGH (CVSS 7.5)

Location: `/.svn/entries`

Test: Accessed the SVN entries file.

Result: Complete source file listing revealed, including `conf.php` (database configuration), `artist.php`, `hello.php`, `process.php`, and others with MD5 hashes.

```
URL:   http://php.testsparker.com/.svn/entries

Extract:
dir
https://msl.unfuddle.com/svn/msl_testbed/testscript/Testsite-PHP
 ...
conf.php        file    3f680a5c730904c98288931b2970f7b5
artist.php      file    eb404830b207daecdac5b74ba6af24a8
hello.php       file    3185f962e28acea96fd525dc7f975b3c
process.php     file    1dd5874e6cec1e8c187584340298cfde
nslookup.php    file    3104e0958b08da142d044c0dd389660d
products.php    file    f8731054d140389d024bb2ee16f9093d
```

Remediation: Remove `.svn` directories from production. Add Apache config: `RedirectMatch 404 /\.svn(/|$)`

▶ Finding 4: PHP Info Disclosure — HIGH (CVSS 7.5)

Location: `/phpinfo.php`

Test: Direct access to the hidden endpoint.

Result: Full PHP configuration, internal IP (172.30.0.85), environment variables, file paths, and server admin email exposed.

```
URL:   http://php.testsparker.com/phpinfo.php

Key disclosures:
- Internal IP: 172.30.0.85
- Document Root: C:/AppServ/www/
- Server Admin: onur@netsparker.com
- Windows Server: Windows NT IP-AC1E0055 6.1 build 7601
- Full PATH, TEMP, environment variables
```

Remediation: Delete `phpinfo.php` from production immediately.

▶ Finding 5: Outdated Software — HIGH (CVSS 8.1)

Location: Server-wide (confirmed via `/phpinfo.php` and server headers)

Result: Apache 2.2.8 (May 2008, EOL) and PHP 5.2.6 (May 2008, EOL) — both end-of-life with hundreds of known CVEs including remote code execution, denial of service, and information disclosure.

```
Server header: Apache/2.2.8 (Win32) PHP/5.2.6

PHP Version 5.2.6 - Build Date: May 2 2008
Apache 2.2.8 - released February 2008
```

Remediation: Upgrade Apache to 2.4.x and PHP to a supported version (8.x). Establish a patch management lifecycle.

▶ Finding 6: Missing HTTPS — HIGH (CVSS 7.4)

Test: Attempted HTTPS connection.

Result: SSL protocol error — no TLS certificate available. All traffic (including login credentials) transmitted in plaintext.

```
$ curl -I https://php.testsparker.com/
curl: (35) error:1408F10B:SSL routines:ssl3_get_record:wrong version numb
er
```

```
No HTTPS listener on port 443.
```

Remediation: Obtain and install a TLS certificate. Enforce HTTPS redirects. Set HSTS header.

▶ Finding 7: Error Message Information Leak — MEDIUM (CVSS 5.3)

Location: `process.php?file=Generics/about.nsp`

Result: MySQL connection error message reveals internal server file path.

```
URL:   http://php.testsparker.com/process.php?file=Generics/about.nsp

Warning: mysql_connect() [function.mysql-connect]:
Access denied for user 'root'@'localhost' (using password: YES) in
C:\AppServ\www\Generics\about.nsp on line 31
```

Remediation: Disable `display_errors` in `php.ini`. Implement custom error handlers that log without displaying.

▶ Finding 8: Hardcoded Default Credentials — MEDIUM (CVSS 5.0)

Location: Login page `/auth/login.php`

Result: Credentials `admin / admin123456` displayed directly on the page, enabling easy brute-force or direct login.

```
Login page text:
"Enter your credentials (admin / admin123456)"
```

Remediation: Remove hardcoded credentials from display. Implement strong password policies and enforce HTTPS.

6. Remediation Roadmap

Priority	Action	Affected Finding	Effort
IMMEDIATE	Fix SQL injection — replace with parameterized queries	#1	Low
IMMEDIATE	Remove `eval()` from `hello.php`	#2	Low
24 HOURS	Delete `/phpinfo.php` and `.svn/` directories	#3, #4	Low
1 WEEK	Upgrade PHP to latest 8.x stable version	#5	Medium
1 WEEK	Upgrade Apache to 2.4.x	#5	Medium
1 WEEK	Obtain and deploy TLS certificate; enforce HTTPS	#6	Low
2 WEEKS	Disable `display_errors`; implement error logging	#7	Low
2 WEEKS	Remove hardcoded creds; implement proper auth flow	#8	Medium

7. Methodology

This assessment was conducted in accordance with:

- **OWASP Top 10 (2025)** — comprehensive coverage of all categories
- **OWASP Web Security Testing Guide (WSTG) v4.2** — manual testing procedures for each category
- **CVSS v3.1** — severity scoring across all findings

Testing approach: Black-box external assessment. All payloads were non-destructive and validated via observed responses, error messages, and content differences.

Tools used: Manual HTTP requests, web browser, curl-based analysis, reconnaissance via public indexes and SVN metadata.