Automated Web Pentest Report

Test Environment

Server OS: Kali GNU/Linux Rolling

Python Version: 3.13.3

Scan Date: 2025-06-12 08:04:46 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: 1

Successful exploits: 0

Risk Level: Low

Test Cases Table

#	Payload	Status	Severity
1	[sqlmap-auto]	Fail	Medium

Evidence Details

Severity: Medium

[1] Payload: [sqlmap-auto]

___

__H__

```
 ___ ___[(]_____ ___ ___  {1.9.4#stable}
|_ -| . [,]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org
```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 04:04:44 /2025-06-12/

[04:04:44] [WARNING] using '/tmp/sqlmap_mrumwo80' as the output directory
[04:04:45] [INFO] testing connection to the target URL
[04:04:45] [INFO] checking if the target is protected by some kind of WAF/IPS
[04:04:45] [INFO] testing if the target URL content is stable
[04:04:45] [INFO] target URL content is stable
[04:04:45] [INFO] testing if GET parameter 'id' is dynamic
[04:04:45] [WARNING] GET parameter 'id' does not appear to be dynamic
[04:04:45] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[04:04:45] [INFO] testing for SQL injection on GET parameter 'id'
[04:04:45] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[04:04:45] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[04:04:45] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[04:04:45] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[04:04:45] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[04:04:45] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[04:04:45] [INFO] testing 'Generic inline queries'
[04:04:45] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[04:04:45] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[04:04:45] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[04:04:45] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[04:04:45] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[04:04:45] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[04:04:46] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[04:04:46] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[04:04:46] [WARNING] GET parameter 'id' does not seem to be injectable
[04:04:46] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'

[*] ending @ 04:04:46 /2025-06-12/

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/