Metasploitable 2 (DVWA) — Web Application Security Test Report

Scope / target: Metasploitable VM running Damn Vulnerable Web Application (DVWA) —
web app vulnerabilities discovered during testing: SQL Injection, Reflected XSS and OS Command Injection
Test date: 22/10/2025
Tester: VIJAY S R

Executive summary

I tested DVWA on the Metasploitable 2 lab and confirmed three issues typical for an intentionally vulnerable training app:

- SQL Injection (CWE-89) — allows data disclosure / authentication bypass.
- Reflected Cross-Site Scripting (CWE-79) — can run attacker JavaScript in victim browsers.
- OS Command Injection (CWE-78) — remote command execution on the host via web input.

All web-app issues fall under OWASP Top 10:2021 → A03:2021 — Injection (OWASP groups SQLi, command injection and XSS into Injection).
References: OWASP Top 10 and DVWA project info.

Test environment / tools used

Target: Metasploitable-2 VM with DVWA

tools & commands used (examples):

Browser + intercept (Burpsuite) for payload injection.

Browser for XSS testing.

Browser for PoC on OS command injection.

Findings (detailed)

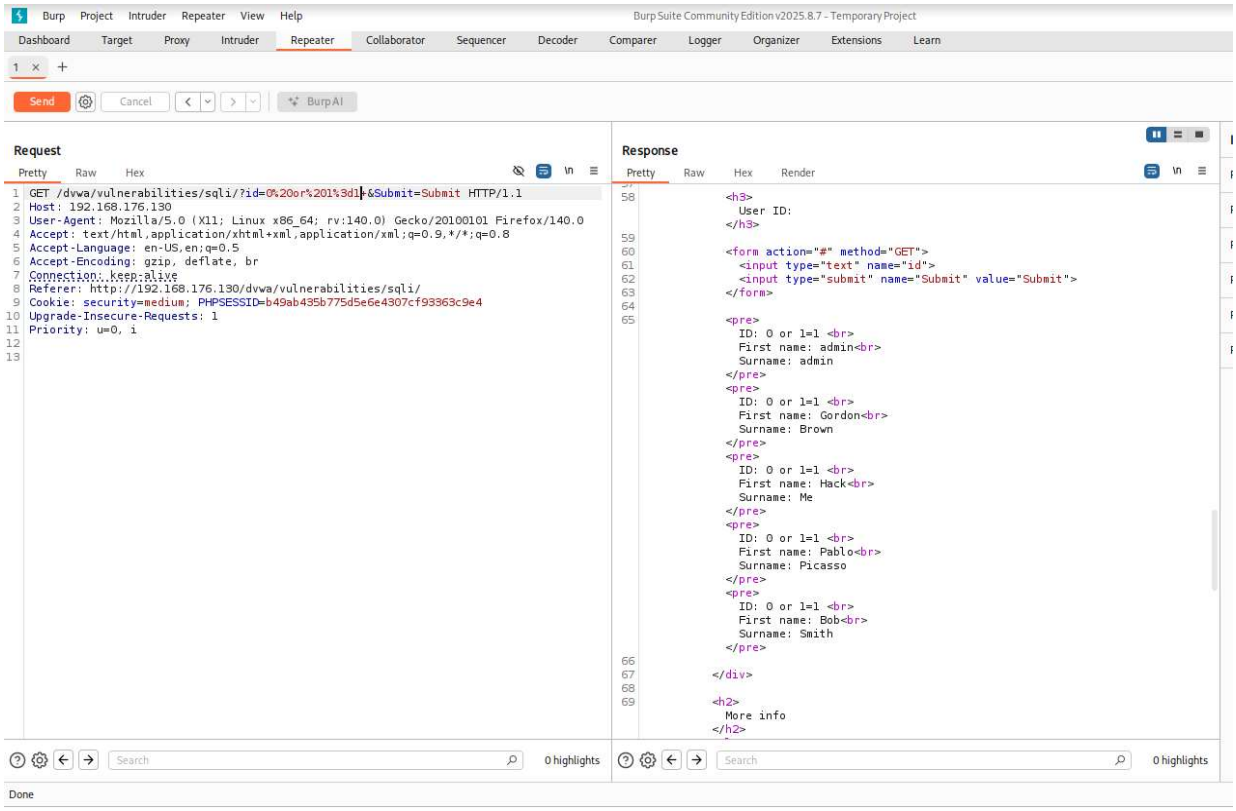
- For each finding I give: short description, CWE, OWASP mapping, CVSS v3.1 base score & vector (explained), PoC / evidence summary, and remediation.

1) SQL Injection — DVWA (CWE-89)

OWASP mapping: A03:2021 — Injection.
Impact: Confidentiality / integrity loss — data disclosure, login bypass, DB tampering.
CVSS v3.1 (recommended base): 9.8 — CRITICAL
Vector string: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Why that score: remote over network (AV:N), low attack complexity (AC:L), no privileges (PR:N), no user interaction (UI:N), full confidentiality/integrity/availability impact (C:H/I:H/A:H). See FIRST CVSS guidance.

Proof / PoC (example):
intercept the request using burpsuit before submit payload.
On DVWA SQL Injection page, submit payload in userid field: 0 or 1=1

Successful result on both burpsuit and Browser: application returns users first name and surname
(evidence: changed response, dumped users information).



Exploitability: High — easy to reproduce in DVWA (intentionally vulnerable).

Remediation:
High-level fixes (short)

1. Use parameterized queries / prepared statements.

2. Use least-privilege DB accounts. (no DROP/DDL from web app)

3. Validate & normalize input — allowlist where possible.

4. Avoid dynamic SQL building; if needed use strong quoting APIs.

5. Centralize DB access & sanitize error messages.

6. Enable query logging + alerts for anomalous queries.

References: OWASP Injection guidance; CVSS calculator.

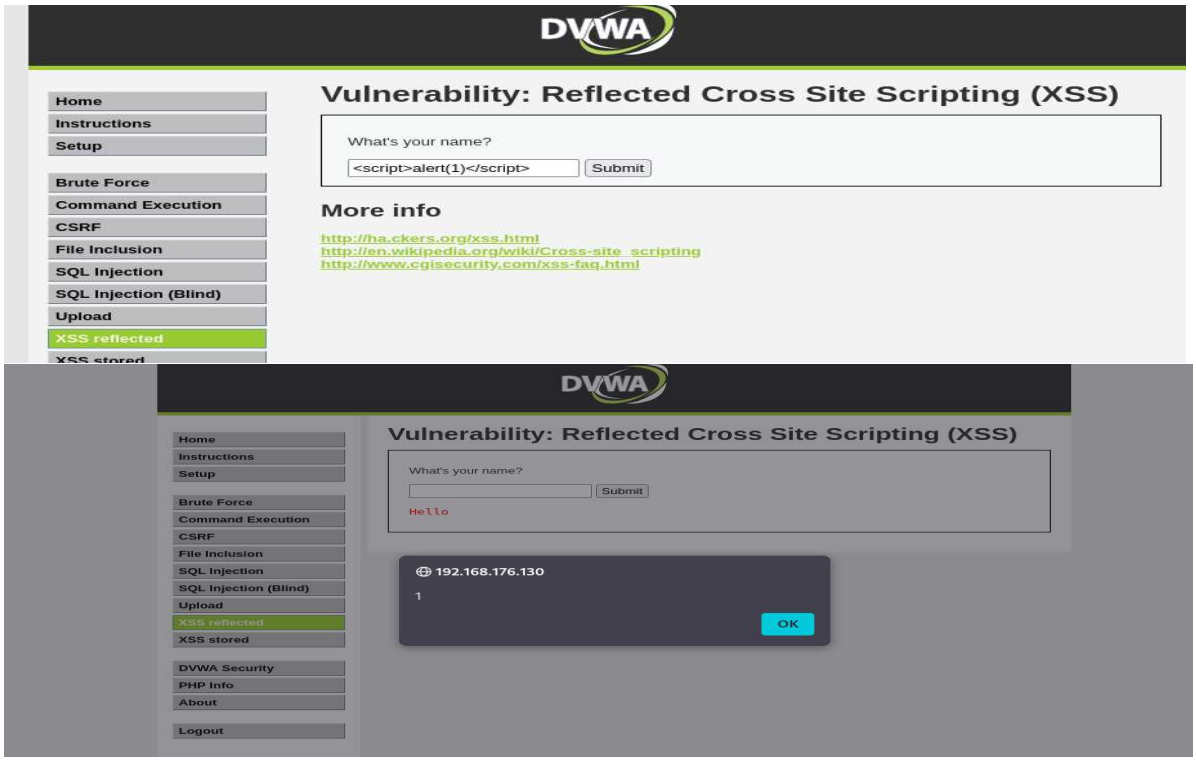
2) Reflected Cross-Site Scripting (XSS) — DVWA (CWE-79)

OWASP mapping: A03:2021 — Injection (XSS is included under Injection in 2021 list).
Impact: Client-side script execution → session theft, phishing, CSRF bypass.
CVSS v3.1 (recommended base): 6.1 — MEDIUM
Vector string (example): CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N
Why that score: remote network (AV:N); attack requires victim to click (UI:R); limited confidentiality/integrity impact vs. full host compromise. See OWASP XSS description.

PoC (example):

DVWA Reflected XSS,Submit payload in the box field:
<script>alert(1)</script>

Successful result on Browser: application returns a popup window and shows 1



Remediation:

1. Output-encode by context (HTML body, attribute, JS, URL, CSS).

2. Use templating engines / frameworks that auto-escape.

3. Implement Content Security Policy (CSP) as defense-in-depth.

4. Set cookies with HttpOnly and Secure flags.

5. Validate input, but encoding on output is primary.

References: OWASP XSS documentation.

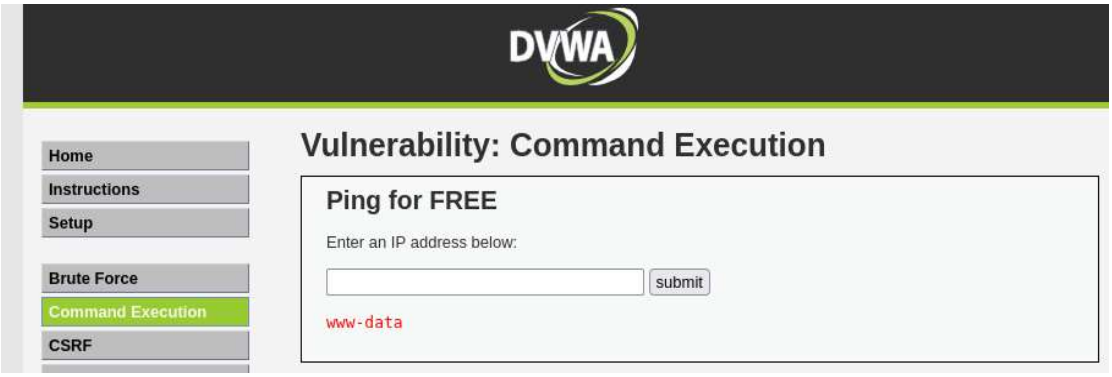
3) OS Command Injection (CWE-78) — DVWA

OWASP mapping: A03:2021 — Injection.
Impact: Remote command execution on hosting system → complete compromise of app host, pivoting.CVSS v3.1 (recommended base): 9.8 — CRITICAL
Vector string: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Why that score: network reachable, low complexity, no privileges needed, no user interaction, full impacts. CVE mapping: generic CWE-78 (many specific CVEs exist for individual products). See NVD CWE mapping.

PoC (example):

On command execution page, submit payload in the box field: ;whoami

Successful result on Browser: application returns device's user information



Remediation:

High-level fixes (short)

1. Do not call shell commands with user input. Use language-native APIs.

2. If external commands required, avoid passing user input to shell — use safe argument APIs (proc_open with argument arrays, exec without shell?)

3. Use strict allowlists for any arguments.

4. Run services with least privilege & process isolation.

5.Audit & log command invocations.

References: NIST mapping of CWE-78; FIRST CVSS guidance.

vulnerability	**OWASP Top10**	**CWE**	**CVSS v3.1(base)**	**Risk**
SQL injection	A03:Injection	CWE-89	9.8	Critical
XSS(Reflected)	A03:Injection	CWE-79	6.1	Medium
OS Command Injection	A03:Injection	CWE-78	9.8	Critical

Cross-cutting controls & process recommendations

1. Retest after fixes — run targeted tests (sqlmap, XSS payloads, "command injection" strings) and verify no execution/leak.

2. Code review & automated SAST in CI pipeline for these CWE patterns.

3. Implement RASP or EDR to detect suspicious command executions at runtime.

4. WAF tuned rules can reduce risk during remediation.

5. Logging & alerting for suspicious inputs (e.g., queries with ' OR 1=1 patterns, <script> strings).

6. Training for devs on secure coding for injection classes (OWASP materials).

7. Least privilege for DB + OS + service accounts, network segmentation (isolate dev/instructional VMs).