05 FULL VAPT CYCLE

EXECUTIVE SUMMARY

During a controlled security assessment of the DVWA instance at 192.168.116.132, we identified multiple web application vulnerabilities that could allow attackers to read or alter sensitive application data. The most critical issue is SQL Injection in the id parameter of /dvwa/vulnerabilities/sqli/, which allowed database enumeration and controlled data extraction using sqlmap. A reflected Cross-Site Scripting (XSS) was also observed, enabling an attacker to execute scripts in the browser of another user. Additionally, an outdated PHP runtime was flagged by OpenVAS, increasing overall risk. Short-term mitigations (WAF, configuration hardening) and code fixes (parameterized queries, contextual output encoding) are recommended immediately; re-scan after fixes to validate remediation.

SCOPE & RULES OF ENGAGEMENT

- **Scope:** http://192.168.116.132/dvwa.
- **In-scope assets:** DVWA web application and underlying web stack on 192.168.116.132 only.
- **Out-of-scope:** Any external networks or hosts outside lab.
- **Authorization:** Testing performed only in lab environment with explicit permission.
- **Testing windows & impact:** Non-destructive where possible; intentional data extraction performed on lab DB for proof-of-concept only.

METHODOLOGY & TOOLS

1. Reconnaissance — identify hosts, service URLs.
2. Vulnerability discovery — OpenVAS (GVM) scans + manual verification.
3. Exploitation — use sqlmap to confirm and exploit SQLi; manual tests for XSS.
4. Evidence collection — save outputs, screenshots, scan reports.
5. Remediation & reporting.

Tools used:

- Kali Linux
- sqlmap (for SQL injection verification and DB extraction)

- Greenbone Vulnerability Manager / OpenVAS (scanning & reporting)
- Browser dev tools (manual verification, cookies)
- Bash (timestamped logging and evidence collection)

FINDINGS

Finding 1 — SQL Injection (High)

Location: http://192.168.116.132/dvwa/vulnerabilities/sqli/?id=<value>&Submit=Submit

Description: The id parameter is injectable. Using automated exploitation (sqlmap) we confirmed the ability to enumerate databases and extract table contents.

Impact: Full data disclosure from accessible DB tables; potential for admin account compromise if credentials exist in DB. In a production app, SQLi can lead to total data breach and pivoting to other systems.

Reproduction (commands & summary):

1. Manual indicator: visiting .../sqli/?id=1' produced query differences / error behavior.
2. Confirm with sqlmap (authenticated DVWA session cookie used in this test):

Result: usernames and password hashes from dvwa.users retrieved.

Recommendation:

- Immediately implement parameterized queries / prepared statements for all DB access.
- Use least-privilege DB accounts (no superuser credentials).
- Validate and whitelist expected input types on server side.
- Employ query-layer protections and a WAF as short-term mitigation.
- After fixes, re-run sqlmap and OpenVAS to validate.

Finding 2 — Reflected XSS (Medium)

Location: /dvwa/vulnerabilities/xss_r/ (reflected input shows without contextual escaping)

Description: Input submitted to a page is reflected unescaped, allowing script injection.

Impact: Execution of JavaScript in victim browsers — session theft, CSRF escalation, content spoofing.

Recommendation:

- Contextual output encoding (HTML-escape values shown in HTML content, JS-encode where injected into scripts, attribute-encode for HTML attributes).

- Use framework-provided escaping functions and CSP headers (Content-Security-Policy) to reduce impact.
- Mark user-controlled inputs as untrusted and sanitize on server side.

Finding 3 — Outdated PHP / Server Configuration (High)

Location: Web server stack (scanned by OpenVAS).

Description: OpenVAS reported an outdated PHP version and weak HTTP headers/configuration that increase attack surface.

Impact: May allow exploitation via known CVEs in older PHP versions or third-party modules.

Evidence & reproduction: Key items: PHP version X.Y.Z flagged, missing secure cookie attributes, and directory indexing check.

Recommendation:

- Patch/update PHP to a supported version and apply vendor hotfixes.
- Harden HTTP headers: set X-Content-Type-Options: nosniff, X-Frame-Options: DENY, Referrer-Policy, and Strict-Transport-Security if HTTPS used.
- Disable unnecessary modules and directory listings.

RISK PRIORITIZATION & REMEDIATION PLAN

Immediate (0–7 days):

- Fix SQLi by replacing dynamic SQL with prepared statements and parameterized ORM calls. Rotate DB credentials.
- Apply short-term WAF rules to block common SQLi/XSS payloads.
- Fix PHP version if known critical CVE exists (apply vendor patch).

Short-term (1–4 weeks):

- Implement output encoding across the app. Add CSP header.
- Harden server HTTP headers and cookie flags (HttpOnly, Secure, SameSite).
- Perform code review to find other injection/XSS issues.

Medium-term (1–3 months):

- Add automated CI checks for insecure patterns (e.g., raw SQL concatenation).
- Schedule regular OpenVAS scans and periodic manual pentests.
- Implement logging & alerting for anomalous queries / web requests.

EVIDENCE & ARTIFACTS

All evidence stored in project folder evidence/:

- evidence/sqlmap-output.txt — sqlmap dump & logs (DB enumeration & table dumps).
- evidence/xss-screenshot.png — proof-of-issue screenshot for XSS.
- evidence/openvas-report.pdf — full OpenVAS scan export.
- evidence/findings.csv — CSV log of findings with timestamps.

CONCLUSION

This identified classic web application weaknesses (SQL Injection, XSS) and configuration issues (outdated PHP). The SQL Injection is high-risk and should be remediated as priority by applying parameterized queries and least-privilege DB access. XSS should be mitigated by output encoding and CSP. After implementing recommended changes, re-scan and perform targeted manual verification to ensure fixes are effective.

200-WORD PTES-STYLE SUMMARY

This assessment followed the PTES methodology to perform a focused web application penetration test against DVWA hosted at 192.168.1.200. The engagement included reconnaissance, automated scanning with OpenVAS, and exploitation of application-level vulnerabilities using sqlmap. Scans identified multiple issues, the most critical being SQL Injection and reflected Cross-Site Scripting. SQL Injection was confirmed using sqlmap against the /dvwa/vulnerabilities/sqli/ parameter, allowing enumeration and controlled extraction of database entries. OpenVAS flagged additional issues including outdated PHP components and weak HTTP configuration. Impact ranges from unauthorized data disclosure to potential remote code execution if combined with other flaws. Recommended remediation begins with immediate input validation and parameterized queries in the application, removal of inline SQL, and least-privilege database account usage. Implementing contextual output encoding, a strict Content Security Policy, and updating the software stack are medium-term steps. Post-remediation, repeat automated scans and targeted manual verification to validate fixes. Evidence, command logs, and scan exports are attached. Overall, vulnerabilities found are typical of insecure development settings and can be mitigated through prioritized code fixes, configuration hardening, and an ongoing vulnerability management process.

100-WORD NON-TECHNICAL BRIEFING NON-TECHNICAL SUMMARY

During a controlled security test of the web application (DVWA) at 192.168.116.132, we found vulnerabilities that allow attackers to read or manipulate site data. The most serious issue is a SQL injection that could expose user information if exploited. We also found a vulnerability that could let attackers insert scripts into web pages (XSS). We recommend immediate code fixes (stop building SQL queries from raw user input), update the server software, and re-run scans after fixes. Short-term mitigation like a web application firewall can reduce risk while permanent fixes are implemented.