VAPT Report — Damn Vulnerable Web Application (DVWA)

Author: Amit Mondal
Project Type: Vulnerability Assessment & Penetration Testing
Environment: Kali Purple VM (VirtualBox)
Date: December 2025

1. Executive Summary

This report presents a complete **Vulnerability Assessment & Penetration Testing (VAPT)** conducted on the Damn Vulnerable Web Application (DVWA).
The objective of this assessment was to identify, exploit, and document security flaws through both manual and automated techniques, following standard penetration testing methodology.

The assessment confirms the presence of multiple **critical vulnerabilities**, including SQL Injection, Command Injection, Cross-Site Scripting, and Unrestricted File Upload, along with several misconfigurations.

This project demonstrates hands-on penetration testing skills and the ability to document findings in a professional format.

2. Scope of Assessment

In-Scope Target:

- DVWA installed on **Kali Purple VM**
- Local environment only (no external systems)

Tools Used:

- Burp Suite
- Nmap
- Nikto
- Firefox (Proxy with Burp)
- Apache2, MariaDB, PHP

Vulnerabilities Tested:

- SQL Injection
- Command Injection
- Stored & Reflected XSS
- File Upload Exploitation

- Server misconfigurations

3. Methodology

The assessment followed the standard **OWASP Penetration Testing Approach**:

3.1 Reconnaissance

- Identified open ports and services using Nmap
- Observed Apache configuration and DVWA setup

3.2 Scanning

- Nmap service scan for version detection
- Nikto scan for insecure headers, directories, and server exposures

3.3 Manual Vulnerability Testing

- SQL Injection on GET parameters
- OS Command Injection via ping function
- Stored & Reflected Cross-Site Scripting
- Unrestricted File Upload bypass

3.4 Post-Exploitation

- Extracted database details
- Executed system-level commands
- Validated persistence of XSS payloads

3.5 Reporting

- Findings documented with description, payloads, impact, and remediation

4. Key Findings Summary

Vulnerability	Severity	Description
SQL Injection	Critical	Database extraction, authentication bypass
Command Injection	Critical	OS command execution as www-data
Stored XSS	High	Persistent JS execution affecting all users
Reflected XSS	High	Immediate script execution via input fields
File Upload Vulnerability	High	Arbitrary PHP file upload → RCE risk
Misconfigurations	Medium	Missing headers, directory indexing, exposed files

5. Detailed Findings

5.1 SQL Injection — Critical

Location: `/vulnerabilities/sqli/?id=`

Payloads Used

```
1' OR '1'='1
1' UNION SELECT 1,2 -- -
1' UNION SELECT user(),2 -- -
1' UNION SELECT database(),2 -- -
1' UNION SELECT version(),2 -- -
1' UNION SELECT @@hostname,2 -- -
```

Impact

- Access to database user
- Extracted DB name, version, and hostname
- Bypassed authentication logic

Remediation

- Use parameterized SQL queries
- Strict input validation
- Disable detailed SQL error messages

5.2 Command Injection — Critical

Location: `/vulnerabilities/exec/`

Payloads Used

```
127.0.0.1; whoami
127.0.0.1; ls -la
```

Impact

- Executed system commands as user **www-data**
- Directory structure and system information exposed

Remediation

- Remove unsafe functions (system, exec, shell_exec)
- Use strict server-side validation
- Implement allowlists for acceptable parameters

5.3 Stored Cross-Site Scripting (XSS) — High

Payload:

```
<script>alert('Stored XSS')</script>
```

Impact

- Persistent JavaScript code execution
- Potential for session hijacking or phishing

Remediation

- Escape output using `htmlspecialchars()`
- Filter and sanitize all inputs
- Add Content Security Policy (CSP)

5.4 Reflected XSS — High

Payload:

```
<script>alert('Hello')</script>
```

Impact

- Immediate JS execution
- Attacker can inject malicious code dynamically

Remediation

- Encode dynamic outputs
- Implement server-side sanitization

5.5 File Upload Vulnerability — High

Observed Behavior:
DVWA allowed uploading a malicious `.php` file and displayed its stored location.

Impact

- Potential remote code execution
- Attacker can run arbitrary server-side scripts

Remediation

- Verify MIME type and file content
- Block executable file uploads
- Disable PHP execution inside upload directory
- Rename uploads to random hashes

5.6 Server Misconfigurations — Medium

Findings

- Missing critical security headers
- Directory listing enabled
- Exposure of sensitive folders (`/tests/`, `/docs/`, `/config/`)

Remediation

- Implement `X-Frame-Options`, `X-Content-Type-Options`, `CSP`
- Disable directory listing
- Restrict access to internal folders

6. CVSS Scoring Summary

Vulnerability	Score	Severity
SQL Injection	9.8	Critical
Command Injection	9.8	Critical
File Upload	8.6	High
Stored/Reflected XSS	7.5	High
Misconfiguration	5.0	Medium

7. Recommendations (High-Level)

Security Controls

- Implement input validation & strict sanitization

- Use parameterized queries everywhere
- Enforce strong Content Security Policy (CSP)
- Prevent file execution in upload directories
- Add security headers to reduce attack surface
- Restrict directory access

System Hardening

- Disable unnecessary PHP functions
- Keep server packages updated
- Restrict database permissions

8. Conclusion

This assessment successfully identified and exploited multiple high-severity vulnerabilities inside DVWA.
The project demonstrates hands-on capability in:

- Manual and automated pentesting
- Vulnerability exploitation
- Web security misconfiguration detection
- Report writing and documentation

This VAPT engagement aligns with real-world penetration testing workflows and strengthens practical cybersecurity skills.

9. Author

Amit Mondal
Cybersecurity Enthusiast • Ethical Hacker • Web Application Penetration Tester