Assignment no 6

DVWA — Installation, Testing

Target: DVWA on local Kali VM (http://127.0.0.1/DVWA/)

Prepared for: Skill Horizon / ZERODAYVAULT Author: Date:

Overview

This document contains everything you need for Assignment 6 — DVWA: step-by-step installation on Kali, how to change security levels, manual vulnerability exercises (with safe payloads), light automated confirmation (Nikto, WhatWeb, Dirb, Nuclei), evidence-capture instructions, and a ready-to-submit vulnerability report template. Use only on your local VM.

1. Install & start DVWA on Kali

Open a terminal and run:

```
sudo apt update
sudo apt install dvwa -y
sudo dvwa-start
```

Then open in browser: http://127.0.0.1/DVWA/.

If DB not set up: log in with admin / password, go to Setup → Create / Reset Database. You should see Database setup completed.

Manual install (if package missing):

```
sudo apt install git apache2 php php-mysqli php-xml php-gd libapache2-mod-php -y
cd /var/www/html
sudo git clone https://github.com/digininja/DVWA.git dvwa
sudo chown -R www-data:www-data dvwa
# copy config: sudo cp dvwa/config/config.inc.php.dist dvwa/config/config.inc.php
# edit DB settings in dvwa/config/config.inc.php if necessary
sudo systemctl restart apache2
```

2. DVWA Security Levels (Low → Medium → High)

Log in as admin / password.

Navigate to DVWA Security. Set Low. Perform the tests below and capture evidence.

Change to Medium. Re-run the same tests and note differences.

Change to High. Re-run and document which tests are mitigated.

Capture screenshots for each test at each security level (Low / Medium / High).

3. Manual vulnerability exercises (safe payloads + how to capture evidence)

Create folder structure before testing:

mkdir -p ~/dvwa-report/{screenshots,requests,scans}

A. SQL Injection (sqli)

Where: /DVWA/vulnerabilities/sqli/ (GET parameter id) Low payload (proof): 1' OR '1'='1 How to run & save evidence:

curl -s "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1'%20OR%20'1'='1&Submit=Submit" > ~/dvwa-report/requests/sqli_result.html

Capture: screenshot of browser showing results; save curl output above. Notes: On Medium/High, query may be sanitized — document differences. Remediation: Use prepared statements/parameterized queries.

B. Cross-Site Scripting (XSS)

Reflected XSS (where): /DVWA/vulnerabilities/xss_r/ Test payload (Low): <script>alert(1)</script> Safe alternative (less intrusive): <script>alert(1)</script> for visualization in source Capture: screenshot with alert (or page showing reflected input) and request (via browser devtools or Burp/ZAP). Save as ~/dvwa-report/requests/xss_reflected.txt. Remediation: Contextual output encoding, CSP headers, input validation.

Stored XSS (where): /DVWA/vulnerabilities/xss_s/ — submit payload as a comment/message; then view page where stored content is rendered. Capture screenshot.

c. Command Injection

Where: /DVWA/vulnerabilities/exec/ (or Command Injection module) Payload (Low): 127.0.0.1; echo DVWA-CMD-TEST How to verify safely: Use a benign echo as above. Save page output to ~/dvwa-report/requests/cmd_injection.txt. Remediation: Avoid executing shell commands with user input; whitelist inputs; escape properly.

D. File Upload / Unrestricted File Upload

Where: /DVWA/vulnerabilities/upload/ Test: Upload a benign test.txt and, only on your local VM, a test.php containing <?php phpinfo(); ?> to see if PHP executes (this is allowed only in lab). Access verification: note the returned upload path and attempt to access it: http://127.0.0.1/DVWA/hackable/uploads/<filename> — save screenshot. Remediation: Validate file type & content, store outside webroot, rename files, set restrictive permissions.

E. Insecure Direct Object Reference (IDOR)

Where: modules where resource IDs are in URLs or parameters (e.g., file inclusion or view pages). Test: Change id= values in URL to attempt to access other users' data. Save screenshots and request logs. Remediation: Enforce server-side authorization; use indirect references.

F. Security Misconfigurations & Insecure Headers

What to check: Missing X-Frame-Options, X-XSS-Protection, Content-Security-Policy, Strict-Transport-Security, Server banners, directory listing. Command to capture headers:

curl -I http://127.0.0.1/DVWA/ 2>&1 | tee ~/dvwa-report/requests/headers.txt

Remediation: Add security headers in Apache config and hide server banner.

G. Sensitive Information in Source/JS

Action: View page source and external JS files. Save source:

curl -s http://127.0.0.1/DVWA/ > ~/dvwa-report/requests/homepage_source.html

Search for keys, endpoints, or commented-out credentials. Capture screenshots. Remediation: Never store secrets in client-side code.

H. Session Management Weaknesses

What to check: Cookie flags (HttpOnly, Secure, SameSite), session ID regeneration on login, session timeout. Capture cookies: Use browser devtools or curl to list Set-Cookie.

curl -I http://127.0.0.1/DVWA/ 2>&1 | tee ~/dvwa-report/requests/cookies.txt

Remediation: Set HttpOnly, Secure, SameSite flags; regenerate session IDs on login; set reasonable timeouts.

4. Light automation (run on localhost DVWA)

Run these commands from Kali and save outputs in ~/dvwa-report/scans:

```
mkdir -p ~/dvwa-report/scans && cd ~/dvwa-report/scans
whatweb -a 3 http://127.0.0.1/DVWA/ 2>&1 | tee whatweb.txt
nikto -h http://127.0.0.1/DVWA -output nikto.txt 2>&1 | tee nikto_live.txt
dirb http://127.0.0.1/DVWA/ /usr/share/wordlists/dirb/common.txt -o dirb.txt
# or gobuster:
# gobuster dir -u http://127.0.0.1/DVWA/ -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -o gobuster.txt
nuclei -u http://127.0.0.1/DVWA/ -o nuclei.txt 2>&1 | tee nuclei_live.txt
```

Save the outputs and include them in Appendix.

5. Evidence capture checklist (for each finding)

Exact URL and DVWA Security level (Low/Medium/High).

Payload used.

Screenshot showing the payload and the result.

Raw HTTP request/response (curl, Burp, or ZAP export).

Tool output that flagged the issue.

Short reproduction steps.

6. Vulnerability report template (DVWA-focused)

Copy this into a Markdown or Word file, fill in evidence paths, then export to PDF.

```
# DVWA Security Assessment Report
**Target:** DVWA on local Kali VM (http://127.0.0.1/DVWA/)
**Assessor:** <Your Name>
**Dates:** <start - end>
```

Executive Summary
Briefly summarize top findings and overall risk posture.

Scope
- DVWA installed locally on Kali VM. All testing performed locally.

Methodology
- Manual testing across DVWA modules at security levels Low/Medium/High.
- Light automated confirmation scans: WhatWeb, Nikto, Dirb, Nuclei.

Findings
Finding 1 — SQL Injection (Reflected) — Confirmed (Low)
- **Location:** `/DVWA/vulnerabilities/sqli/`
- **DVWA level:** Low
- **Severity:** High (if in production)
- **Evidence:** `~/dvwa-report/requests/sqli_result.html`, screenshot `~/dvwa-report/screenshots/sqli_low.png`
- **Reproduction:** `id=1' OR '1'='1`
- **Impact:** Data disclosure, authentication bypass
- **Remediation:** Use parameterized queries, input validation, least-privilege DB user.

Finding 2 — Reflected XSS — Confirmed (Low)
- **Location:** `/DVWA/vulnerabilities/xss_r/`
- **Payload:** `<script>alert(1)</script>`
- **Evidence:** screenshot `~/dvwa-report/screenshots/xss_reflected_low.png`
- **Remediation:** Output encoding, CSP, input validation.

... add other findings similarly ...

Remediation Roadmap
1. Immediate: Patch SQLi and Command Injection issues.
2. Short-term: Implement security headers and file upload restrictions.
3. Long-term: Secure SDLC, code reviews, automated CI tests.

Appendix
- Commands used (include the commands from this doc).

- Raw scan outputs: `~/dvwa-report/scans/`
- Screenshots and request logs: `~/dvwa-report/screenshots/`, `~/dvwa-report/requests/`.
7. Sample findings (example text you can copy)

SQL Injection (Reflected) — Confirmed (Low)

Evidence: HTTP response returned multiple records for id=1' OR '1'='1 (saved as sqli_result.html).

Impact: Attackers can enumerate or modify database content.

Fix: Use prepared statements, parameterized queries.

Missing Security Headers — Info/Low

Evidence: curl -I shows no X-Frame-Options, CSP, or HSTS.

Fix: Add headers in apache2.conf or site config.

8. How to export to PDF (easy ways)

Copy the report section into a Word document and Save As → PDF.

Or save the Markdown and use a converter like pandoc: