Mini Pentest Project: DVWA + Juice Shop

Author: Isha Bachhav

Course / Project: Mini Pentest Project

Date: 30/9/2025

1. Executive Summary

This project performed a controlled penetration test against two intentionally vulnerable web applications: Damn Vulnerable Web Application (DVWA) and OWASP Juice Shop. The objective was to practice common recon, scanning, and exploitation techniques, compare the types and severity of vulnerabilities found in each application, and produce remediation guidance.

Scope: Local lab environment (VMs/containers) hosting DVWA and Juice Shop. All testing executed with explicit permission in an isolated environment.

High-level findings:

DVWA contains classic web vulnerabilities intentionally enabled (SQL Injection, Stored/Reflected XSS, Command Injection, File Upload flaws). These are straightforward to discover and exploit at low security settings.

Juice Shop simulates more modern OWASP Top 10 issues (broken authentication, insecure direct object references, stored XSS, sensitive data exposure). It contains a broader range of chained vulnerabilities and more realistic misconfigurations.

Overall risk summary: Both applications are intentionally vulnerable; Juice Shop presents more realistic business-impact scenarios (auth bypass, data exposure) while DVWA is ideal for learning basic exploitation techniques.

2. Goals & Objectives

Install and configure DVWA and Juice Shop in a lab environment.

Perform reconnaissance and scanning using Nmap, Dirb, Nikto and related tools.

Discover and exploit at least two vulnerabilities in each application.

Compare and contrast the vulnerabilities across the two apps.

Produce a professional report containing technical findings and remediation.

3. Environment & Setup

Hardware/VMs:

Kali Linux (attacker) — VM

DVWA — webapp VM (e.g., Ubuntu + Apache + PHP + MySQL) or provided DVWA Docker container

OWASP Juice Shop — Node.js app, run via Docker or Node

Network: Isolated NAT or host-only network. Assign static IPs (e.g., 10.0.2.15 for Kali, 10.0.2.20 for webapps) to avoid affecting other networks.

Installation summary (commands / hints):

DVWA via Docker:

```
# Pull and run DVWA container
docker pull vulnerables/web-dvwa
docker run --rm -it -p 80:80 vulnerables/web-dvwa
```

Juice Shop via Docker:

```
docker pull bkimminich/juice-shop
docker run --rm -it -p 3000:3000 bkimminich/juice-shop
```

Alternatively install manually (Apache, PHP, MySQL for DVWA; Node.js for Juice Shop). Configure hostnames or update /etc/hosts on attacker VM for convenience.

4. Tools Used

Recon & Scanning: Nmap, Dirb (or Gobuster), Nikto

Proxy / Manual testing: Burp Suite (Community or Professional), OWASP ZAP

Exploitation / Automation: sqlmap, curl, netcat

Enumeration: WhatWeb, Wappalyzer

Documentation: KeepNote/CherryTree or Markdown for notes; screenshotting tools

5. Methodology (STEPS)

Rules of Engagement & Scope confirmation — note environment is lab and explicit permission exists.

Reconnaissance (Passive & Active)

Passive: WhatWeb / Wappalyzer to fingerprint tech stack

Active: Nmap port/service scan, directory brute force (Dirb/Gobuster)

Vulnerability Scanning

Run Nikto and Burp Spider for initial content.

Manual Testing & Exploitation

Use Burp for intercepting requests, injecting payloads

Use sqlmap for SQLi verification and exploitation

Test XSS payloads manually and confirm stored/reflected

Post-Exploitation & Impact Analysis

Reporting & Remediation

6. Commands & Examples (appendix-ready)

Nmap (service discovery & script scan):

```
nmap -sC -sV -p- -T4 10.0.2.20 -oN nmap_fullscan.txt
```

Dirb (directory brute-force):

```
dirb http://10.0.2.20/ /usr/share/wordlists/dirb/common.txt -o dirb_results.txt
```

Nikto (webserver scan):

```
nikto -h http://10.0.2.20 -o nikto_results.txt
```

sqlmap (auto test for SQLi):

```
sqlmap -u "http://10.0.2.20/vulnerabilities/sqli/?id=1&Submit=Submit" --batch --dbs
```

Burp Suite / manual XSS test: Intercept a request and inject <script>alert(1)</script> into form fields to test reflected/stored XSS.

7. Findings: DVWA

Target: DVWA (http://10.0.2.20)

7.1 Vulnerability A — SQL Injection (Low Security Level)

Type: SQL Injection (In-band)

Location: /vulnerabilities/sqli/ parameter id

Evidence / PoC: Using sqlmap returned database names; manual payload 1' OR '1'='1 bypassed intended filter and returned rows.

Impact: Attacker can read database contents, exfiltrate user credentials, and escalate to full data compromise.

Risk: High (if deployed in production)

Remediation: Use parameterized queries (prepared statements), input validation, least privilege DB user.

7.2 Vulnerability B — Stored XSS

Type: Cross-site Scripting (Stored)

Location: /vulnerabilities/xss_s/ comment field

Evidence / PoC: Injected <script>alert('XSS')</script> was stored and executed in victim browser when viewing comments.

Impact: Session theft, account takeover, phishing.

Remediation: Contextual output encoding, Content-Security-Policy, sanitize user input server-side.

7.3 Vulnerability C — Command Injection (Low)

Type: OS Command Injection

Location: /vulnerabilities/exec/ IP ping function

PoC: Input 127.0.0.1; cat /etc/passwd returned system file contents.

Impact: Remote command execution, complete server compromise.

Remediation: Avoid shelling out with unsanitized user input; use safe libraries and validate inputs. Apply principle of least privilege.

8. Findings: Juice Shop (sample findings)

Target: OWASP Juice Shop (http://10.0.2.21:3000)

8.1 Vulnerability A — Broken Authentication (Account Enumeration / Weak Passwords)

Type: Authentication/Identity

Location: Login & Password Reset flows

PoC: Password reset/email flows reveal whether account exists; weak default password accounts allow login.

Impact: Account takeover, data theft.

Remediation: Do not reveal detailed error messages, add rate-limiting, enforce strong passwords, use multi-factor auth.

8.2 Vulnerability B — Stored XSS in product reviews

Type: Stored XSS

Location: Product review/comment feature

PoC: Submitting <img src=x onerror=alert(1)> in a review triggered JS in other users' browsers.

Impact: Same as DVWA XSS but may affect more user interactions and persist.

Remediation: Sanitize and encode output, use server-side frameworks that auto-escape, implement CSP.

8.3 Vulnerability C — Sensitive Data Exposure

Type: Sensitive Data Exposure (e.g., secrets in source, info leakage)

Location: Source files or API endpoints (in a realistic app this would be credentials, tokens)

PoC: Juice Shop intentionally stores some challenge secrets in accessible resources; attackers can find secrets via directory enumeration.

Impact: Compromise of back-end services, token theft.

Remediation: Remove secrets from source, use environment variables, secure storage, rotate secrets.

9. Comparison & Analysis

Attack Surface: DVWA focuses on classic input-based vulnerabilities; Juice Shop contains a broader range including auth flaws and business-logic issues.

Exploitability: Many DVWA vulnerabilities are trivially exploitable at Low security settings; Juice Shop requires chaining or deeper logic understanding for high-impact results.

Real-world relevance: Juice Shop models more real-world scenarios (user accounts, e-commerce flows). DVWA is best for learning basics.

10. Risk Rating (Sample)

Use CVSS-style risk ratings for each finding. Example:

SQLi (DVWA) — CVSS: 9.8 (Critical)

Stored XSS (DVWA/Juice Shop) — CVSS: 6.1–7.5 (High/Medium) depending on context

Command Injection — CVSS: 10 (Critical)

Broken Auth — CVSS: 9.0 (Critical)

11. Remediation Recommendations (General)

Input Validation & Parameterized Queries: Use prepared statements and ORMs.

Output Encoding: Escape user input in HTML, JavaScript, URL contexts.

Authentication & Session Management: Enforce strong password policies, MFA, secure cookie flags (HttpOnly, Secure), and session timeouts.

Least Privilege: App should run with minimal OS and DB privileges.

Secure Config & Secrets Management: Store secrets outside source, use environment variables and secret managers.

Logging & Monitoring: Implement application logging and monitor for abnormal activity.

Dependency Management: Keep libraries up-to-date; perform SCA.

Content Security Policy (CSP): Mitigate XSS impacts.

12. Evidence & Artifacts

Include screenshots, intercepted requests, nmap/nikto outputs, sqlmap outputs, Burp logs. (Attach files if submitting electronically.)

13. Ethical Considerations & RoE

Testing was confined to an isolated lab with explicit permission.

No destructive operations performed against production systems.

All sensitive data discovered is treated confidentially and only included in the report for remediation.

14. Appendices

A — Recommended Commands & Examples (copyable)

(Repeat of section 6 with ready-to-run commands; include Burp & sqlmap usage examples.)

B — Grading Checklist (points to help get full marks)

Clear Objective & Scope — state target IPs, time, and permission.

Environment Setup — show commands used to deploy DVWA and Juice Shop.

Recon & Scanning — include Nmap, Dirb, Nikto outputs and interpretation.

At least two vulnerabilities exploited per app — show PoC, payloads, and screenshots.

Impact Assessment — CVSS or similar ratings and realistic impact explanation.

Remediation — clear, actionable fixes per finding.

Professional Presentation — executive summary, organized sections, references.

C — Useful References

OWASP Testing Guide

OWASP Juice Shop project documentation

DVWA documentation

Burp Suite, sqlmap and Nikto docs