Title: Capstone Project - Full VAPT Cycle

Non-Technical Summary

This assessment simulated a full penetration test against a controlled lab environment to reveal security weaknesses and recommend fixes. Using automated scanners (OpenVAS) and manual exploitation (DVWA with sqlmap), we identified exploitable input validation issues and cross-site scripting on a web application hosted at 192.168.1.200. Findings were recorded with timestamps and mapped to PTES phases. Vulnerabilities were validated to demonstrate potential impact (for example, data exposure via SQL injection) while avoiding destructive actions. Recommended mitigations focus on secure input handling, configuration hardening, and re-scanning to confirm remediation. This report supports a prioritized remediation plan and future security testing.

PTES-style Report

Scope & Objective: Conduct a penetration test against the lab web application (192.168.1.200) to identify and verify vulnerabilities aligned with the PTES phases: pre-engagement, intelligence, threat modeling, vulnerability analysis, exploitation, post-exploitation, and reporting.

Tools used: OpenVAS (vulnerability discovery), sqlmap (automated SQLi exploitation), Metasploit (post-exploitation when applicable), and manual verification techniques.

Findings & Impact: The engagement confirmed an input validation failure resulting in SQL injection and multiple cross-site scripting findings. Successful SQL injection allowed retrieval of backend data (demonstrated read-only extraction of database entries) which could lead to data confidentiality loss. XSS f indings permit injection of scripts that could hijack user sessions or perform actions in user context. These vulnerabilities correlate to medium-to-high risk depending on data sensitivity and exposure.

Exploitation Summary: Exploitation was non-destructive and focused on proof-of-concept data extraction using sqlmap against vulnerable parameters in DVWA. No lateral movement outside the scoped target was performed. All steps were logged with timestamps and relevant evidence (screenshots, sqlmap output, OpenVAS reports). Recommendations: Prioritize input sanitization and prepared statements, implement contextual output encoding for web responses, enforce least privilege on database accounts, apply a secure development lifecycle for new code, and re-scan after remediation. Additionally, enable central logging and IDS/IPS alerts for anomalous query patterns.

Logged OpenVAS Findings

Timestamp	Target IP	Vulnerability	PTES Phase
2025-08-18 12:00:00	192.168.1.200	XSS	Exploitation

DVWA / sqlmap Exploitation Steps (safe, proof-of-concept)
Prerequisites:

- DVWA running and reachable (example: http://192.168.1.200/)
- DVWA security set to *low* (for test-lab only)
- Valid session cookie (login to DVWA from browser to capture PHPSESSID)

workflow and commands (run from Kali):

1. Capture a vulnerable URL in the browser, e.g.:
   http://192.168.1.200/vulnerabilities/sqli/?id=1&Submit=Submit

2. Run sqlmap with the session cookie and ask it to enumerate the DB (proof-of-concept, non-destructive):
   *sqlmap -u "http://192.168.1.200/vulnerabilities/sqli/?id=1&Submit=Submit" *
   *--cookie="PHPSESSID=<your_session_id>; security=low" *
   --batch --level=3 --risk=2 --threads=2 –dbs

3. If databases are listed and you want to dump a specific table for PoC (read-only):
   sqlmap -u "<same-url>" --cookie="..." -D dvwa --tables
   sqlmap -u "<same-url>" --cookie="..." -D dvwa -T users --dump --stop=100

4. Save outputs
   Do **not** run --os-shell or --file-write in a lab that forbids such actions unless explicitly allowed by scope.
   Notes on ethics & scope: Limit to information retrieval required for demonstration; avoid destructive payloads. Document every command, timestamp, and output for inclusion in the report.

Quick Metasploit / Post-exploitation Guidance

- Use extracted credentials (if any) to test authentication where allowed. Example: use msfconsole only if a service is known vulnerable and within scope.
- Keep exploitation non-destructive. Record evidence rather than altering system state.

Remediation Checklist

- Use parameterized queries / prepared statements for all DB access.
- Validate input server-side and client-side; apply strict allow-lists for expected input.
- Apply contextual output encoding for HTML/JS contexts to prevent XSS.
- Enforce least privilege on database accounts; avoid using admin DB accounts for web apps.
- Deploy a Web Application Firewall and set rules to block common SQLi/XSS payloads.
- Re-run OpenVAS and sqlmap after fixes; maintain a vulnerability tracker.