Capstone Project: Full VAPT Cycle

1. Simulation: Exploiting DVWA with sqlmap for SQL Injection

Setup:

- Deploy DVWA (local via Docker/XAMPP or Metasploitable 2 VM).
- Log in (default: admin/password).
- Set security level to **Low** via DVWA Security tab.
- Navigate to **SQL Injection** module.

Exploitation Steps with sqlmap:

1. Obtain session cookie (PHPSESSID) from browser after login.
2. Basic command to test and dump database:

   text

   ```
   sqlmap -u "http://<DVWA-IP>/vulnerabilities/sqli/?id=1&Submit=Submit"
   --cookie="PHPSESSID=<your_session>; security=low" --dump --batch
   ```

   - This identifies injection in the id parameter.
   - Enumerates databases (e.g., dvwa), tables (e.g., users), and dumps contents (usernames/hashes like admin:password).
3. For blind SQLi variant:

   text

   ```
   sqlmap -u "http://<DVWA-
   IP>/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --
   cookie="PHPSESSID=<your_session>; security=low" --technique=B --dump
   ```

Outcome: Successful extraction of user credentials demonstrates data exfiltration risk.

Metasploit can chain this (e.g., via auxiliary modules for SQLi), but sqlmap automates effectively here.

2. Detection: Logging OpenVAS Findings

OpenVAS (now Greenbone Vulnerability Management in Kali) scans for vulnerabilities.

Setup and Scan:

- Update Kali: sudo apt update && sudo apt upgrade.
- Install/setup: sudo gvm-setup (creates admin user; note password).
- Access web interface: https://127.0.0.1:9392 (accept self-signed cert).

- Create target (e.g., DVWA IP: 192.168.1.200).
- Launch full scan task.

Sample Findings Log (adapted from typical DVWA scans; XSS/SQLi common):

Timestamp	Target IP	Vulnerability	PTES Phase
2025-08-18 12:00:00	192.168.1.200	Cross-Site Scripting (XSS)	Discovery
2025-08-18 12:15:00	192.168.1.200	SQL Injection	Exploitation
2025-08-18 12:30:00	192.168.1.200	Command Injection	Exploitation
2025-08-18 12:45:00	192.168.1.200	Outdated PHP Version	Discovery

OpenVAS rates severity (High/Medium/Low) with CVSS scores.

3. Remediation: Suggestions and Rescan

- **SQL Injection/XSS:** Implement input sanitization (e.g., PHP's mysqli_real_escape_string or prepared statements/PDOs). Validate/sanitize all user inputs.
- **General Fixes:** Use parameterized queries, enable WAF, update software, set DVWA security to High.
- **Rescan:** After fixes, rerun OpenVAS task. Confirm vulnerabilities drop (e.g., no High risks remain).

4. Reporting: 200-Word PTES Report (Draft for Google Docs)

Penetration Testing Execution Standard (PTES) Technical Report

The penetration test engagement was executed against the target asset 192.168.1.200, simulating an internal threat actor with standard user privileges. The testing lifecycle adhered to the Penetration Testing Execution Standard (PTES), encompassing Intelligence Gathering, Vulnerability Identification, and Exploitation.

Reconnaissance identified the host as a Linux web server running DVWA (Damn Vulnerable Web Application). OpenVAS was utilized to automate vulnerability scanning, revealing multiple critical and high-severity vectors, specifically SQL Injection and Cross-Site Scripting (XSS).

Manual exploitation using SQLMap on the SQL Injection module successfully validated the risk. The tool extracted the underlying database schema and recovered password hashes from the users table, demonstrating full confidentiality breach. Concurrently, the XSS vulnerability was confirmed, allowing arbitrary script execution in the context of a victim's browser session.

These vulnerabilities stem from a lack of input sanitization and the absence of output encoding. Immediate remediation is required to migrate to parameterized queries and implement strict Content Security Policies. The overall security posture of the target is currently critical, leaving the organization exposed to data theft and session hijacking.

Conclusion The application is highly vulnerable at Low security. Remediation and rescanning reduced risks significantly. Prioritize fixes to achieve robust security posture.

Copy this into Google Docs; add screenshots (sqlmap output, OpenVAS reports) and format with headings/tables.

5. Briefing: 100-Word Non-Technical Summary

Our security test on the sample web application revealed serious weaknesses that could allow hackers to access private data without permission. Using automated tools, we found and demonstrated issues like SQL Injection, where bad code let us pull out user details (names and passwords). This is like leaving a door unlocked in your office—anyone could walk in and take files.

The good news: These are fixable with better input checks and code updates. We recommend quick actions to sanitize data and rescan to confirm safety. Overall, the app needs stronger protections to prevent real-world breaches and protect customer information.