# Penetration Testing Tool

Mini-Nmap + Mini-SQLMap + Safe PoC

## 1. Abstract

This project develops a Python-based vulnerability scanner that crawls a target web application, enumerates endpoints and parameters, performs lightweight network discovery (open ports and banner grabbing), tests for common web vulnerabilities (SQL Injection, Cross-Site Scripting, CSRF, missing security headers), and — only in a controlled lab environment — can run one-shot proof-of-concept (PoC) demonstrations to show impact. The tool produces a professional report (PDF/HTML) summarizing findings, evidence and remediation suggestions. All testing is intended for isolated lab targets (DVWA, OWASP Juice Shop) and follows ethical guidelines.

## 2. Scope & Objectives

Scope:

• Build an integrated tool that performs discovery, detection and optional lab-only PoC demonstrations.
• Run tests only on controlled lab targets (DVWA, Juice Shop) inside VirtualBox/VMware snapshots.
• Produce an automated report with evidence and CVSS-like severity scoring.

Objectives:

1. Implement an end-to-end workflow: Recon → Scan → Detect → (Optional) PoC → Report.
2. Implement basic port scanning, web crawling, SQLi and XSS detection logic using original code.
3. Automate report generation and present clear remediation guidance.

## 3. Key Features

• Discovery: Automatic crawling and form/parameter extraction.
• Port & Banner Scanning: TCP connect() checks for common ports and banner grabs (mini-Nmap behavior).
• Passive Checks: Security header inspection, cookie flags, common path detection.
• Active Detection: Conservative, non-destructive tests for SQLi (error/boolean) and reflected/stored XSS.
• Exploit PoC (LAB-ONLY): Single-shot demonstrations (e.g., harmless XSS alert) enabled only with explicit confirmation and allowlist.
• Reporting: Generates PDF/HTML with findings, evidence, CVSS-like severity, and

remediation steps.

## 4. Architecture & Modules

Modules:
• Controller (CLI/Local UI): Configures target, mode (scan/test/exploit) and reporting.
• Crawler: Finds URLs, forms, parameters (requests + BeautifulSoup; Selenium optional).
• Port Scanner: Probes configured common ports and captures banners (socket-based).
• Passive Analyzer: Checks headers, TLS basics, and robots.txt.
• Detection Modules: SQLi detector, XSS detector, CSRF check, common paths checker.
• Exploit/PoC Module: Lab-only, single-shot PoCs with strict checks.
• Reporter: PDF/HTML generator and raw JSON export for reproducibility.

## 5. Detection Methodology

• SQL Injection: Use error-based strings and boolean-difference checks (non-destructive) to flag likely injection points.
• XSS: Inject unique marker tokens and detect unescaped reflection in the response or DOM (Selenium optional for JS-heavy pages).
• CSRF: Detect state-changing forms without anti-CSRF tokens.
• Header Checks: Look for missing HSTS, CSP, X-Frame-Options, X-Content-Type-Options and insecure cookie attributes.
• Port/Banner: Probe common ports and parse minimal banner text to infer services (do not assert CVEs without manual verification).

## 6. Tools Used

Development & Runtime:
• Python 3 (primary language)
• requests, BeautifulSoup4 (crawling and HTTP interactions)
• socket, concurrent.futures (port scanning and concurrency)
• reportlab or fpdf (PDF report generation)
• selenium (optional, for DOM-based XSS detection)

Testing & Lab:
• Kali Linux (attacker VM) — development/testing environment
• DVWA, OWASP Juice Shop, Metasploitable2 (vulnerable targets for safe testing)

Reference Tools (for learning & validation only):
• Nmap, Burp Suite, SQLMap, Nikto (we study their behavior but do not depend on them for

core scanner functions)

## 7. Team Responsibilities

• Member 1 — Core & Integration: Controller, config, allowlist, module integration, CI.
• Member 2 — Crawler & Web Detection: Crawler, form parsing, XSS detection logic.
• Member 3 — Network & Port Scanning: Port scanner, banner grabbing, passive checks.
• Member 4 — PoC & Reporting: Safe PoC routines, report generator, CVSS scoring and documentation.

## 8. Timeline (8 weeks)

Week 1 — Requirements, lab setup and snapshots.
Week 2 — Implement crawler and discovery module.
Week 3 — Implement port scanner and banner grabbing.
Week 4 — Implement passive header checks and common paths.
Week 5 — Implement SQLi and XSS detection logic.
Week 6 — Implement exploit/PoC module with allowlist and confirm flows.
Week 7 — Build report generator and integrate modules.
Week 8 — Testing, documentation, and final demo preparation.