PROJECT ABSTRACT

Network Port Scanner Desktop App

A desktop network port scanner for lab and home networks: configurable targets and port ranges, live scan progress with open/closed/filtered states, banner grabbing for service detection, and a plain-language security-findings report — built-to-order with source code, report, PPT and viva kit.

01. Title

Network Port Scanner Desktop App

02. Introduction / Background

Every computer on a network has 65,535 TCP ports, and the ones left open determine the machine's attack surface: an exposed Telnet port, a database listening on the LAN, a forgotten web admin panel. Network administrators map this surface with port scanners, and computer-networks courses teach the underlying TCP handshake — but students rarely get to watch it happen. This project builds a desktop port scanner as both a practical auditing tool and a teaching instrument: point it at the lab network, watch SYN packets go out and SYN-ACK/RST replies come back in a live log, see open ports resolved to services via banner grabbing, and get the results as a security-findings report written in plain language instead of scanner jargon.

03. Problem Statement

Professional scanners are powerful but intimidating for students, and their output assumes networking expertise. A final-year project needs an original implementation whose scanning logic — TCP connect vs SYN, open vs closed vs filtered — the student can explain packet by packet, with an ethics guardrail: the tool is explicitly scoped to networks the user owns or has permission to test.

04. Objectives

- Implement TCP connect scanning over configurable targets (IP, range, CIDR) and port ranges.
- Classify each port as open, closed or filtered from handshake responses.
- Perform banner grabbing on open ports to identify services and versions.
- Show live scan progress with a streaming packet-level log.
- Generate a security-findings report grading issues (high/medium/low) in plain language.
- Provide scan profiles (quick, full, stealth SYN, UDP) and export to HTML/CSV/JSON.
- Display an ethics notice scoping use to owned or authorized networks.

05. Proposed Methodology

The scanner opens TCP connections to each target port with a configurable timeout: a completed handshake means open, a RST reply means closed, and no reply within the timeout means filtered (likely firewalled). Host discovery runs first via ping sweep so dead hosts are skipped. Banner grabbing connects to open ports and reads the service's self-identification

(e.g. SSH or HTTP server strings). Findings are produced by a rule set — cleartext protocols (Telnet, FTP) flagged high, database ports on the LAN flagged medium — each with a remediation sentence. Scans are multithreaded with a worker pool so a /24 subnet completes in minutes rather than hours.

06. System Architecture / Working

Three layers: (1) Scan engine — target parser (IP/range/CIDR), worker pool, handshake classifier, banner grabber; (2) Live UI — progress bar, results table updating in real time, streaming log; (3) Reporting — per-host port tables, graded findings, export module. The working flow is: enter target and profile → start scan → watch live results → open a host for details → read the findings → export the report.

07. Hardware / Software Requirements

- Windows/macOS/Linux desktop or laptop on a LAN — no special hardware
- Python with socket/threading or an Electron + native module backend
- Raw-socket SYN scanning needs admin/root; TCP-connect mode works unprivileged
- Only for networks the user owns or has written permission to test

08. Expected Outcomes

A working desktop app that scans the lab/home network, classifies ports correctly against the documented handshake logic, identifies common services via banners, and produces a graded findings report. Scan-time figures are design targets for typical lab hardware; classification logic is deterministic and demonstrated in the report.

09. Applications

- Auditing home and lab network exposure
- Computer-networks coursework on TCP and scanning
- Pre-deployment port audits for student projects
- Security-awareness demonstrations

10. Future Scope

- OS fingerprinting from TCP/IP stack quirks
- Vulnerability-database cross-reference for banner versions
- Scheduled scans with change alerts
- Topology map of discovered hosts