**Simple Advanced Network Scanner**

**Course**: Information Security  
**Project**: Python‑Based Network Scanner with ARP Discovery, Port Scanning & Reporting  
**Submitted by**: *Ammar*

**📑 Table of Contents**

1. [Introduction](#introduction)
2. [Problem Statement](#problem-statement)
3. [Objectives](#objectives)
4. [Technology Stack](#technology-stack)
5. [System Design & Architecture](#system-design--architecture)
6. [Implementation Details](#implementation-details)
   * 6.1 ARP Discovery
   * 6.2 Multi‑threaded TCP Port Scanning
   * 6.3 OS Hinting via TTL
   * 6.4 Progress Reporting
   * 6.5 Export & HTML Reporting
7. [Usage & Demonstration](#usage--demonstration)
8. [Results & Sample Report](#results--sample-report)
9. [Conclusion](#conclusion)
10. [Future Enhancements](#future-enhancements)
11. [References](#references)

**1. Introduction**

Network reconnaissance is a foundational step in both penetration testing and network administration. This project presents a **Python‑based scanner** that combines ARP host discovery with fast, multi‑threaded TCP port scanning, enriched by OS “hinting” and automated report generation (CSV, JSON, and HTML).

**2. Problem Statement**

Many simple scanners either focus only on ARP discovery or only on port scanning, and rarely provide a polished report. Administrators and security testers need a **single, easy‑to‑use tool** that:

* Quickly finds live hosts on a LAN
* Scans arbitrary port ranges concurrently
* Offers clues about operating systems
* Outputs human‑readable, shareable reports

**3. Objectives**

* **Automate** ARP‑based host discovery on a given IP/range.
* **Scan** specified TCP ports on each discovered host using threads.
* **Infer** a basic OS hint from the TTL field of ARP replies.
* **Display** real‑time progress via progress bars.
* **Export** results to CSV, JSON, and generate a clean HTML report via Jinja2 templates.

**4. Technology Stack**

| **Component** | **Purpose** |
| --- | --- |
| **Python 3** | Core programming language |
| **Scapy** | Low‑level network packet crafting/ARP |
| **socket** | TCP port connection checks |
| **threading & queue** | Concurrency for port scanning |
| **tqdm** | Progress bars |
| **Jinja2** | Templating for HTML report |
| **SQLite** | *(Optional)* lightweight logging storage |

**5. System Design & Architecture**

plaintext

CopyEdit

┌──────────────────────────────┐

│ User invokes CLI │

│ (target + ports + exports) │

└──────────────┬───────────────┘

│

▼

┌───────────────┐

│ ARP Scan │───► Discover live hosts (IP, MAC, TTL)

└───────────────┘

│

▼

┌───────────────┐

│ Port Scanner │───► Multi‑thread: test each port on each host

└───────────────┘

│

▼

┌───────────────┐

│ Collect Data │───► Build list of {ip, mac, os\_hint, open\_ports}

└───────────────┘

│

┌─────────┼─────────┐

▼ ▼ ▼

┌───────┐ ┌────────┐ ┌────────┐

│ CSV │ │ JSON │ │ HTML │

│ Export│ │ Export │ │ Report │

└───────┘ └────────┘ └────────┘

**6. Implementation Details**

**6.1 ARP Discovery**

* Uses **Scapy’s** ARP and Ether layers to broadcast on the LAN.
* Captures replies to identify live hosts.
* Extracts TTL from replies to hint at OS: TTL > 128 ⇒ likely Windows; otherwise Linux/Unix.

**6.2 Multi‑threaded TCP Port Scanning**

* Uses Python’s socket.connect\_ex() in threads.
* A Queue holds the list of ports; worker threads pop ports, attempt connection, and record successes.
* Default thread count = 50 (configurable).

**6.3 OS Hinting via TTL**

* The TTL field in the ARP reply often reflects the sender’s default TTL.
* Provides a coarse “OS hint” without full fingerprinting.

**6.4 Progress Reporting**

* **tqdm** tracks progress over the list of live hosts during port scanning.
* Keeps the user informed of scan status, improving usability.

**6.5 Export & HTML Reporting**

* **CSV**: columnar export of IP, MAC, OS hint, comma‑separated open ports.
* **JSON**: structured export with full detail.
* **HTML**: Jinja2 template renders a styled table with scan time, all hosts, and port lists.

**7. Usage & Demonstration**

1. **Install dependencies**

bash

CopyEdit

pip install scapy tqdm jinja2

1. **Run a full scan**

bash

CopyEdit

python scanner.py \

-t 192.168.1.0/24 \

-p 22,80,443,3389 \

--csv myscan.csv \

--json myscan.json \

--html report.html

1. **Inspect outputs**
   * myscan.csv opened in Excel or text editor
   * myscan.json for programmatic use
   * report.html in your browser

**8. Results & Sample Report**

**Sample HTML Report Screenshot**

| **IP Address** | **MAC Address** | **OS Hint** | **Open Ports** |
| --- | --- | --- | --- |
| 192.168.1.10 | 00:11:22:33:44:55 | Linux/Unix | 22, 80, 443 |
| 192.168.1.15 | 66:77:88:99:AA:BB | Windows | 3389 |
| … | … | … | … |

*CSV, JSON, and HTML files are included in the submission package.*

**9. Conclusion**

This project delivers a compact, **all‑in‑one** network reconnaissance tool suitable for administrators and pentesters. By combining ARP discovery, threaded port scanning, OS hinting, and rich reporting, it provides actionable visibility into local networks with minimal setup.

**10. Future Enhancements**

* **ICMP/UDP scanning** for full protocol coverage.
* **Service banner grabbing** to identify application versions.
* **GUI front‑end** using Tkinter or Electron.
* **Continuous monitoring mode** with scheduling and change alerts.
* **Integration** with threat intelligence feeds.

**11. References**

1. The Scapy Documentation — [https://scapy.net](https://scapy.net" \t "_new)
2. Python socket Library — [https://docs.python.org/3/library/socket.html](https://docs.python.org/3/library/socket.html" \t "_new)
3. tqdm Progress Bar — [https://github.com/tqdm/tqdm](https://github.com/tqdm/tqdm" \t "_new)
4. Jinja2 Templating — https://jinja.palletsprojects.com