Network Scanner Using Python

By Inlighn Tech

Objective:

The objective of this project is to create a Python-based network scanner that detects active devices in a given IP range. This project will help students understand network scanning, ARP requests, multi-threading, and socket programming in Python.

Project Overview:

Network scanning is essential for cybersecurity professionals to monitor active devices in a network. This project builds a network scanner using Python's **Scapy** library to send ARP requests and retrieve IP addresses, MAC addresses, and hostnames of connected devices. The tool supports multi-threading to speed up scanning in large networks.

How the Project Works:

1. **User Input:** The user provides a CIDR-based network address (e.g., `192.168.1.0/24`).
2. **IP Address Extraction:** The script extracts all valid host IPs from the provided subnet.
3. **ARP Request:** Each IP address is scanned using an ARP (Address Resolution Protocol) request.
4. **MAC Address Retrieval:** If the device is active, its MAC address is collected.
5. **Hostname Resolution:** The scanner attempts to fetch the hostname using reverse DNS lookup.
6. **Multi-threading:** The script uses multiple threads to scan multiple devices in parallel.
7. **Results Display:** The discovered devices are displayed in a tabular format with their IP address, MAC address, and hostname.

Key Concepts Covered:

- ✔ **Networking Basics:** Understanding ARP and network scanning.
- ✔ **Python Networking:** Using **Scapy** for network communication.

- ✔ **Multi-threading:** Implementing parallel execution to speed up scanning.
- ✔ **Socket Programming:** Resolving hostnames from IP addresses.
- ✔ **Queue Management:** Using Python's `Queue` module for thread-safe result storage.

Step-by-Step Implementation:

1. Install required dependencies (`scapy`).
2. Accept network input from the user in CIDR format.
3. Generate IP addresses from the subnet.
4. Use ARP requests to identify active hosts.
5. Retrieve MAC addresses and attempt hostname resolution.
6. Implement multi-threading to scan multiple devices simultaneously.
7. Display the scan results in a structured format.

Expected Outcomes:

- By completing this project, students will:
- Gain hands-on experience in **network scanning techniques**.
- Learn how to use **Scapy for network packet manipulation**.
- Understand **ARP-based scanning and device discovery**.
- Implement **multi-threading for efficient network scans**.
- Develop a **basic cybersecurity tool for real-world applications**.

Next Steps:

Students should implement their own version of the network scanner using the outlined concepts. A video lecture will be provided later to demonstrate the correct implementation and solution. This project serves as a foundational step for network security and penetration testing tasks in Python.

For further enhancements, students can:

- **Add OS Detection: Extend the scanner to identify the operating systems of discovered devices.**
- **Implement a GUI: Develop a graphical interface for better usability and visualization of scan results.**
- **Scan Open Ports: Integrate port scanning functionality to detect active services on discovered devices.**