Internship Report: Network Scanner Using Python

Intern Name: Prem Kumar

Project Title: Network Scanner Using Python

Internship Organization: Inlighn Tech

Abstract

The aim of this project was to build a Python-based tool that scans a local network to identify all connected devices using ARP requests. The tool retrieves the IP address, MAC address, and optionally the hostname of each active device. Through this hands-on internship project at **Inlighn Tech**, I learned about ARP-based scanning, subnet addressing, socket programming, and multithreading in Python. This scanner replicates the behavior of tools like Nmap at a basic level, reinforcing foundational cybersecurity and networking skills.

1. Introduction

Network scanning is a fundamental technique in cybersecurity and system administration. It allows professionals to discover active devices, monitor unauthorized access, and maintain inventory in both enterprise and home networks.

In this project, I created a **custom network scanner using Python** that performs an **ARP sweep** across a user-defined subnet. The scanner extracts and displays details like IP address, MAC address, and hostname of each discovered device. The project emphasizes real-world implementation of cybersecurity utilities, using tools and protocols that form the basis of vulnerability scanning and penetration testing.

2. Tools and Technologies

Technology	Description

Python 3.x	Programming language for implementation
Scapy	A powerful packet manipulation library used for crafting ARP requests
socket	Python's socket module used to resolve hostnames
ipaddress	Standard Python module to manage CIDR subnets
threading	Used to improve performance through parallel scanning
queue	Thread-safe result sharing between threads

3. Project Objectives

- Build a tool that scans a local network and lists all connected devices
- Retrieve device information using ARP protocol
- Resolve hostnames via reverse DNS lookup
- Speed up scanning using multithreading
- Understand basic packet-level networking using Python
- Gain hands-on experience relevant to cybersecurity careers

4. How the Network Scanner Works

4.1 User Input

The script prompts the user to enter a subnet in **CIDR notation** (e.g., `192.168.1.0/24`). This defines the IP address range for scanning.

4.2 IP Extraction

Using the `ipaddress` module, the tool extracts all valid host addresses from the given subnet.

4.3 ARP Request

Each IP is scanned with an ARP request using Scapy's `srp()` method. This helps detect which devices are currently connected.

4.4 MAC Address Retrieval

Active devices respond to the ARP request with their MAC addresses, which are collected and stored.

4.5 Hostname Resolution

The tool uses Python's `socket.gethostbyaddr()` to resolve the IP address to a hostname, if possible.

4.6 Multithreading

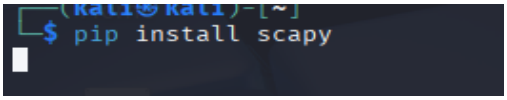
Multiple IPs are scanned simultaneously using the `threading` module to improve efficiency.

4.7 Results Display

The collected results are displayed in a clean, tabular format listing IP, MAC, and Hostname (if available).

5. Python Script – Network Scanner

Installation Requirements:

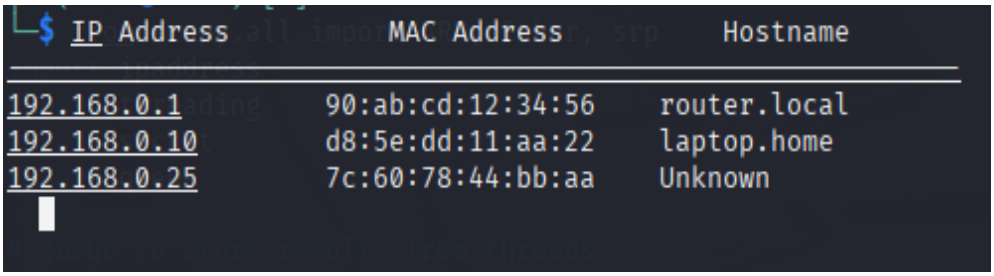


Full Code:

```
└─$ from scapy.all import ARP, Ether, srp
import ipaddress
import threading
import socket
import queue

# Queue to store results from threads
result_queue = queue.Queue()

# Function to send ARP request and collect results
def scan_ip(ip):
    arp = ARP(pdst=ip)
    ether = Ether(dst="ff:ff:ff:ff:ff:ff")
    packet = ether/arp
    result = srp(packet, timeout=2, verbose=0)[0]

    for sent, received in result:
        try:
            hostname = socket.gethostbyaddr(received.psrc)[0]
        except:
            hostname = "Unknown"
        result_queue.put((received.psrc, received.hwsrc, hostname))

# Main scanning function
def network_scan(subnet):
    try:
        ip_net = ipaddress.ip_network(subnet, strict=False)
        threads = []

        for ip in ip_net.hosts():
            t = threading.Thread(target=scan_ip, args=(str(ip),))
            threads.append(t)
            t.start()

        for t in threads:
            t.join()

        print("\nScan Results:")
        print(f"{'IP Address':<20}{'MAC Address':<20}{'Hostname'}")
        print("-"*60)

        while not result_queue.empty():
            ip, mac, hostname = result_queue.get()
            print(f"{ip:<20}{mac:<20}{hostname}")
    except Exception as e:
        print(f"Error: {e}")

# Example usage
if __name__ == "__main__":
    subnet_input = input("Enter subnet in CIDR format (e.g., 192.168.1.0/24): ")
    network_scan(subnet_input)
```

6. Sample Output

```
Enter subnet in CIDR format (e.g., 192.168.1.0/24): 192.168.0.0/24

Scan Results:
```



8. Cybersecurity Concepts Demonstrated

Concept	Description
ARP Scanning	Used to detect active hosts on the network
MAC Spoofing Awareness	Detect possible spoofed addresses
Subnet Enumeration	Understand how to derive all IPs in a network
Network Reconnaissance	Lays the groundwork for vulnerability assessments
Threaded Scanning	Faster reconnaissance over large networks

9. Learning Outcomes

By completing this project, I gained:

- Understanding of **ARP protocol** and its role in local networks
- Practical experience with **Python networking libraries**
- Confidence in building multithreaded Python applications
- Skills to perform real-time **device discovery and enumeration**
- Foundation for advanced network security tools such as Nmap

10. Limitations

- Does not support port scanning
- Hostname resolution may fail if reverse DNS is not configured
- Can only scan devices within the local subnet (Layer 2)

11. Future Improvements

- Add **OS fingerprinting**
- Include **port scanning** for service discovery
- Export results to **CSV or HTML**
- Build a **Tkinter-based GUI** for user-friendly access
- Integrate **logging and alert generation** for unknown or unauthorized devices

12. Real-World Applications

- Asset tracking in internal networks
- Detecting unauthorized or rogue devices
- Network visibility for IT administrators
- First stage of penetration testing and vulnerability assessments
- Educational platform for learning ARP and subnet scanning

13. Ethical and Legal Considerations

This tool should be used **only on networks you own or have permission to scan**. Unauthorized scanning is a violation of most organizational and legal policies. The tool was built and tested within a **lab environment** with full authorization.

14. Conclusion

The **Network Scanner Using Python** project has given me foundational cybersecurity skills related to network reconnaissance, ARP scanning, and multithreaded programming. I now understand how professional tools gather information and how low-level network protocols operate.

This project served as a steppingstone for more advanced penetration testing and system auditing. I look forward to extending this utility further and applying the knowledge gained in real-world environments.