Prerequisites for the Network Scanner Project

By: **Inlighn Tech**

Overview

The Network Scanner project requires students to develop a Python script that scans a local network to identify active devices, retrieving their IP addresses, MAC addresses, and hostnames. The solution uses the scapy library to send ARP requests, socket for hostname resolution, threading for concurrent scanning, and ipaddress to parse network ranges, with results stored in a Queue and displayed in a formatted table. To write this script independently, students must master several key concepts and skills from the curriculum. These prerequisites, paired with practical Python code examples, ensure students understand how network scanning works and can implement the script effectively.

Prerequisite Knowledge and Skills

1. **ARP Fundamentals – Address Resolution Protocol**
   - **What to Know**:

- ARP resolves IP addresses to MAC addresses in a LAN by broadcasting requests and receiving replies.
- A network scanner uses ARP to discover devices by querying all IPs in a range and collecting responses.

- **How It Applies**:
  - The script creates ARP requests with scapy.ARP() and broadcasts them via scapy.Ether() to identify active devices.
- **Curriculum Source**:
  - "Network Scanner (Solution)" Lesson 1 ("What is ARP.mp4").
- **Practical Prep**:
  - Use arp -a (Windows) or ip neigh (Linux) to view your local ARP cache.
- **Python Code Block**:

```
# Simple ARP request with scapy (requires root/admin privileges)
import scapy.all as scapy

ip = "192.168.1.1"  # Replace with a local IP
arp = scapy.ARP(pdst=ip)
broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
packet = broadcast / arp
result = scapy.srp(packet, timeout=1, verbose=False)[0]
if result:
    print(f"IP: {result[0][1].psrc}, MAC: {result[0][1].hwsrc}")
else:
    print(f"No response from {ip}")
```

-
  - **Run It**: Install scapy (pip install scapy), save as arp_test.py, run with sudo python arp_test.py (Linux) or as admin (Windows). Demonstrates ARP basics, preparing for scan().

2. **Networking Basics – IP Addresses and CIDR Notation**
   - **What to Know**:
     - IP addresses identify devices; CIDR notation (e.g., 192.168.1.0/24) defines network ranges.
     - Host addresses exclude network and broadcast IPs in a subnet.
   - **How It Applies**:

        - The script uses ipaddress.ip_network() to parse CIDR and network.hosts() to iterate over scannable IPs.
    - **Curriculum Source**:
        - "NETWORKING, SOCKETS, AND CYBERSECURITY" PDFs (e.g., "Introduction to basic networking concepts_ TCP, UDP, IP.pdf").
    - **Practical Prep**:
        - Explore IP ranges manually.
    - **Python Code Block**:

```
# Parse CIDR and list host IPs
import ipaddress

cidr = "192.168.1.0/24"
network = ipaddress.ip_network(cidr, strict=False)
for ip in list(network.hosts())[:5]:  # Limit to first 5 for demo
    print(f"Host IP: {ip}")
```

    -
        - **Run It**: Save as ip_test.py, run with python ip_test.py. Shows how to generate IPs from CIDR, like in main().

3. **Functions and Arguments**
    - **What to Know**:
        - Define functions with def and pass arguments (e.g., scan(ip, result_queue)).
        - Return or store results for later use.
    - **How It Applies**:
        - scan() processes each IP, storing results in a queue, while print_result() formats output.
    - **Curriculum Source**:
        - "PYTHON BASICS" Lessons 17-19 (Introduction to Functions, def Keyword, Basics of Functions).
    - **Practical Prep**:
        - Practice function usage.
    - **Python Code Block**:

```
# Function with arguments and return
def add_numbers(a, b):
    return a + b

result = add_numbers(3, 5)
print(f"Result: {result}")
```

  - **Run It**: Save as function_test.py, run with python function_test.py. Prepares for defining scan().

4. **Exception Handling**
   - **What to Know**:
     - Use try/except to handle errors (e.g., socket.herror for hostname resolution failures).
     - Gracefully manage failures without stopping execution.
   - **How It Applies**:
     - The script catches socket.herror when gethostbyaddr() fails, defaulting to "Unknown" hostname.
   - **Curriculum Source**:
     - "PYTHON BASICS" Lesson 21 (Errors and Exception Handling).
   - **Practical Prep**:
     - Test error handling.
   - **Python Code Block**:

```
# Handle socket errors
import socket

ip = "192.168.1.1"  # Replace with a local IP
try:
    hostname = socket.gethostbyaddr(ip)[0]
    print(f"Hostname: {hostname}")
except socket.herror:
    print(f"Hostname for {ip}: Unknown")
```

        - **Run It**: Save as socket_test.py, run with python socket_test.py. Mimics hostname resolution error handling in scan().

5. **Multithreading with threading and Queue**
    - **What to Know**:
        - threading.Thread runs tasks concurrently; Queue safely shares data between threads.
        - Start threads with start(), wait with join(), and use Queue.put()/get() for results.
    - **How It Applies**:
        - The script spawns threads to scan IPs in parallel, storing results in a Queue for later processing.
    - **Curriculum Source**:
        - "Network Scanner (Solution)" Lesson 4 ("Working with Threads for our program.mp4").
    - **Practical Prep**:
        - Experiment with threads and queues.
    - **Python Code Block**:

```
# Use threads and a queue
import threading
from queue import Queue
import time

def worker(num, q):
    time.sleep(1)  # Simulate work
    q.put(f"Worker {num} done")

q = Queue()
threads = []
for i in range(3):
    t = threading.Thread(target=worker, args=(i, q))
    t.start()
    threads.append(t)

for t in threads:
    t.join()

while not q.empty():
    print(q.get())
```

-
    - **Run It**: Save as thread_queue_test.py, run with python thread_queue_test.py. Shows threading and queue usage, like in main().

6. **Working with scapy Library**
    - **What to Know**:
        - scapy is a packet manipulation tool; scapy.ARP() creates ARP requests, scapy.Ether() sets Ethernet frames, and scapy.srp() sends/receives packets.
        - Requires root/admin privileges to send packets.
    - **How It Applies**:
        - The script uses scapy to craft and send ARP packets, parsing responses for IP and MAC data.
    - **Curriculum Source**:
        - "Network Scanner (Solution)" Lessons 2-3 ("Creating ARP request.mp4", "Extracting info from the answer.mp4").
        - Note: Students need pip install scapy.
    - **Practical Prep**:
        - Test a basic scapy scan (see ARP code block above).
7. **Socket Library for Hostname Resolution**
    - **What to Know**:
        - socket.gethostbyaddr() resolves an IP to a hostname via reverse DNS lookup.
        - May fail if no DNS record exists, requiring error handling.
    - **How It Applies**:
        - The script uses socket.gethostbyaddr() to enrich scan results with hostnames.
    - **Curriculum Source**:
        - "NETWORKING, SOCKETS, AND CYBERSECURITY" PDF ("Introduction to Sockets.pdf").
    - **Practical Prep**:
        - See the socket_test.py code block above.

How Network Scanning Works

- **Concept**:
    - Network scanning identifies active devices on a LAN by sending ARP requests to all IPs in a range (e.g., 192.168.1.0/24). Responses reveal IP and MAC addresses, with optional hostname resolution.
    - This is a reconnaissance technique to map network topology.
- **Script Workflow**:
    - Accept a CIDR range (e.g., 192.168.1.0/24) from user input.
    - Parse the range with ipaddress and generate host IPs.
    - Spawn threads to send ARP requests to each IP, storing results in a Queue.
    - Collect responses, resolve hostnames, and display in a table.
- **Why Multithreading?**:
    - Speeds up scanning by querying multiple IPs simultaneously, critical for large networks.

How to Write the Network Scanner Script

Using these prerequisites, students can build the script step-by-step:

1. **Setup**:
    - Import scapy.all, socket, threading, Queue, and ip address.
    - Define scan() to send ARP requests and collect responses.
2. **Scan Function**:
    - Create ARP packet with scapy.ARP(pdst=ip) and broadcast with scapy.Ether().
    - Send with scapy.srp(), parse answers, and resolve hostnames with socket.gethostbyaddr().
    - Store results in result_queue.
3. **Print Results**:
    - Write print_result() to format and display IP, MAC, and hostname.
4. **Main Function**:

    - Parse CIDR with ipaddress.ip_network(), spawn threads for each host IP, and collect results from the queue.
5. **Entry Point**:
    - Use if __name__ == "__main__": to prompt for CIDR and run main().

Notes for Students

- **Setup**: Install scapy (pip install scapy), run the script with admin/root privileges (sudo python script.py on Linux), and use your local network's CIDR (e.g., 192.168.1.0/24).
- **Engagement**: Run each code block to practice ARP, threading, and IP parsing. Test the final script on a small range (e.g., /29) to avoid overwhelming your network.
- **Tips**: Ensure your network allows ARP traffic; some routers may block excessive requests.

Jump to the Network Scanner Project Description if you are ready now.
