Part 4: Python Network Scanner with Scapy

Contents

Time required: 30 minutes

Python Tabs and Spaces Issue

Visual Studio Code automatically changes a tab into four spaces. Other editors, like geany and nano in Linux, do not. You can end up with a combination of spaces and tabs. Python doesn't like a combination, it wants either one or the other. The preferred method is spaces.

Recommendation:

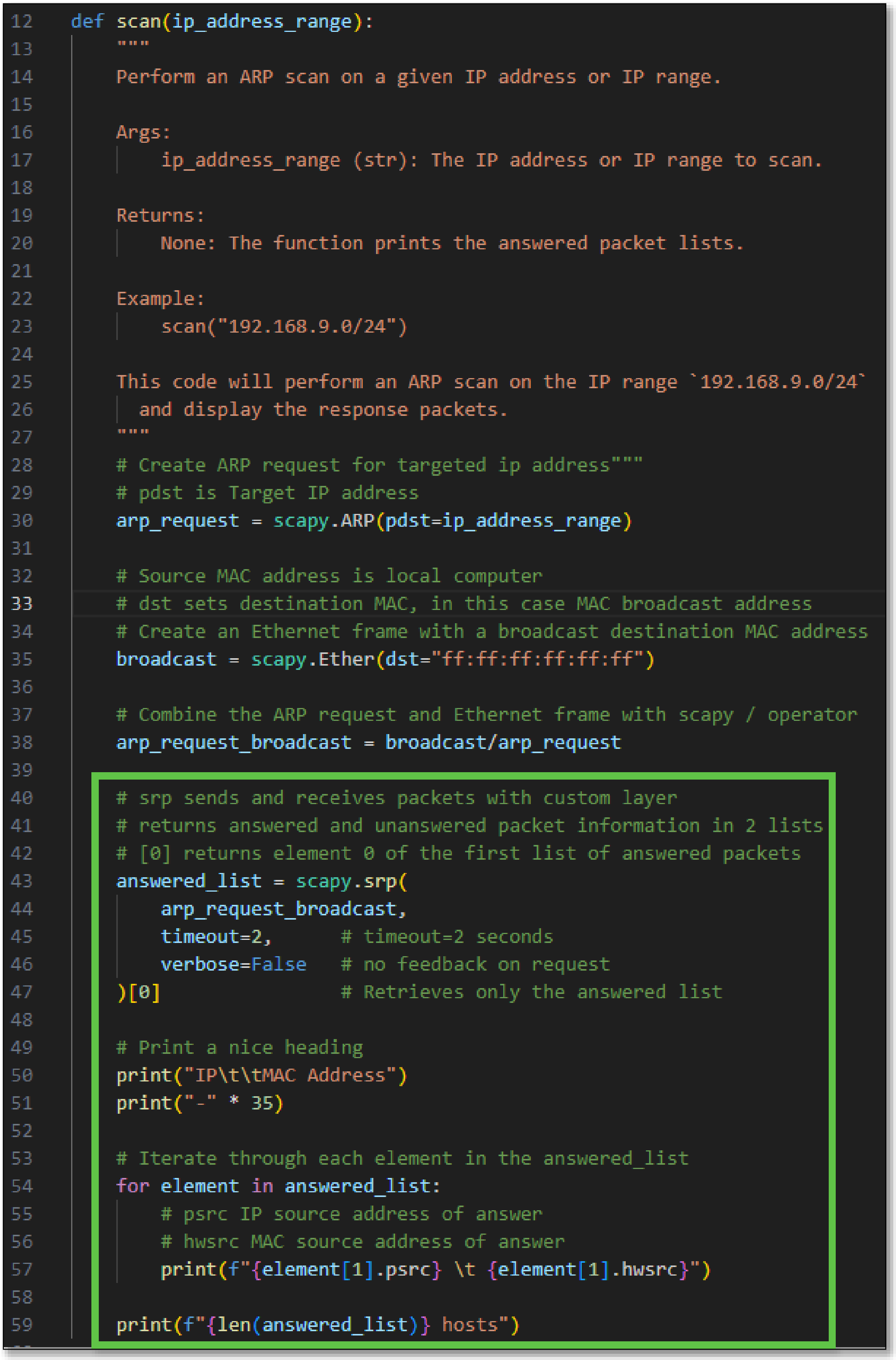
1. Create your Python files in Visual Studio Code in Windows.
2. Copy and paste the code into either nano or geany in Linux.

Network Scanner – The Final Chapter

Save **network_scanner_3c.py** as **network_scanner.py**

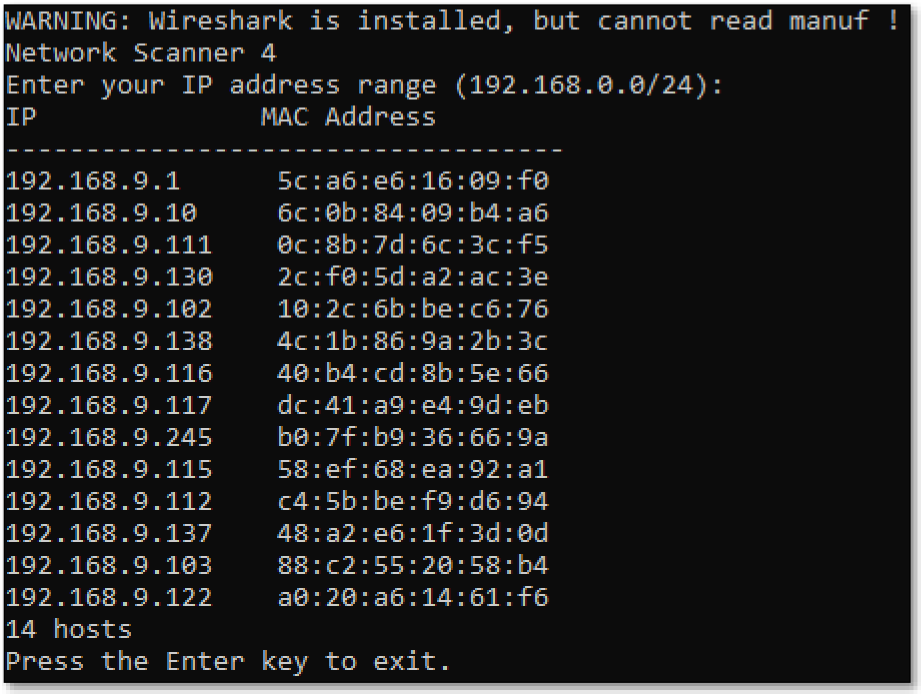
We have everything working. We can make it look better. Let's format our response packets and print a nice title.

- For a cleaner look, let's get rid of the feedback from the scapy.srp packet sending. **verbose=False** turns off all srp feedback.
- Add a nice heading and put the IP and MAC information on the same line.



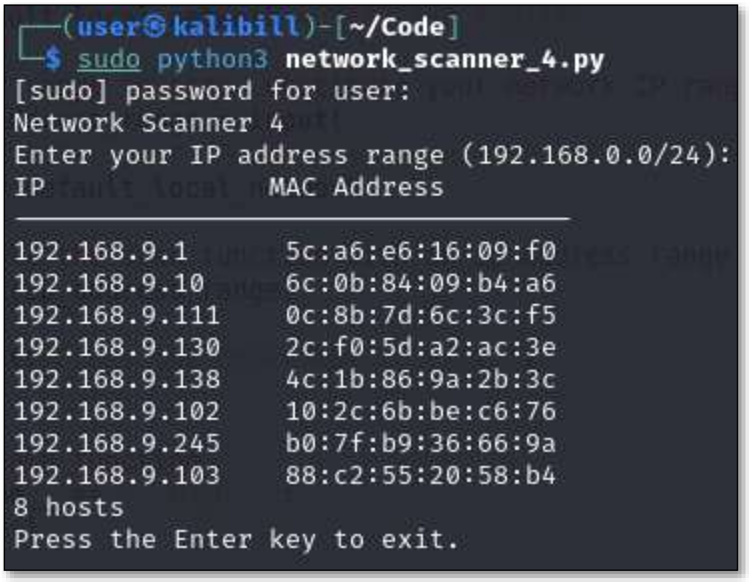
```python
def scan(ip_address_range):
    """
    Perform an ARP scan on a given IP address or IP range.

    Args:
        ip_address_range (str): The IP address or IP range to scan.

    Returns:
        None: The function prints the answered packet lists.

    Example:
        scan("192.168.9.0/24")

    This code will perform an ARP scan on the IP range `192.168.9.0/24`
      and display the response packets.
    """
    # Create ARP request for targeted ip address"""
    # pdst is Target IP address
    arp_request = scapy.ARP(pdst=ip_address_range)

    # Source MAC address is local computer
    # dst sets destination MAC, in this case MAC broadcast address
    # Create an Ethernet frame with a broadcast destination MAC address
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")

    # Combine the ARP request and Ethernet frame with scapy / operator
    arp_request_broadcast = broadcast/arp_request

    # srp sends and receives packets with custom layer
    # returns answered and unanswered packet information in 2 lists
    # [0] returns element 0 of the first list of answered packets
    answered_list = scapy.srp(
        arp_request_broadcast,
        timeout=2,      # timeout=2 seconds
        verbose=False   # no feedback on request
    )[0]                # Retrieves only the answered list

    # Print a nice heading
    print("IP\t\tMAC Address")
    print("-" * 35)

    # Iterate through each element in the answered_list
    for element in answered_list:
        # psrc IP source address of answer
        # hwsrc MAC source address of answer
        print(f"{element[1].psrc} \t {element[1].hwsrc}")

    print(f"{len(answered_list)} hosts")
```

Example run Windows:



```
WARNING: Wireshark is installed, but cannot read manuf !
Network Scanner 4
Enter your IP address range (192.168.0.0/24):
IP              MAC Address
-----------------------------------
192.168.9.1      5c:a6:e6:16:09:f0
192.168.9.10     6c:0b:84:09:b4:a6
192.168.9.111    0c:8b:7d:6c:3c:f5
192.168.9.130    2c:f0:5d:a2:ac:3e
192.168.9.102    10:2c:6b:be:c6:76
192.168.9.138    4c:1b:86:9a:2b:3c
192.168.9.116    40:b4:cd:8b:5e:66
192.168.9.117    dc:41:a9:e4:9d:eb
192.168.9.245    b0:7f:b9:36:66:9a
192.168.9.115    58:ef:68:ea:92:a1
192.168.9.112    c4:5b:be:f9:d6:94
192.168.9.137    48:a2:e6:1f:3d:0d
192.168.9.103    88:c2:55:20:58:b4
192.168.9.122    a0:20:a6:14:61:f6
14 hosts
Press the Enter key to exit.
```

Example run Linux:



```
┌──(user㉿kalibill)-[~/Code]
└─$ sudo python3 network_scanner_4.py
[sudo] password for user:
Network Scanner 4
Enter your IP address range (192.168.0.0/24):
IP              MAC Address
───────────────────────────────────

192.168.9.1      5c:a6:e6:16:09:f0
192.168.9.10     6c:0b:84:09:b4:a6
192.168.9.111    0c:8b:7d:6c:3c:f5
192.168.9.130    2c:f0:5d:a2:ac:3e
192.168.9.138    4c:1b:86:9a:2b:3c
192.168.9.102    10:2c:6b:be:c6:76
192.168.9.245    b0:7f:b9:36:66:9a
192.168.9.103    88:c2:55:20:58:b4
8 hosts
Press the Enter key to exit.
```

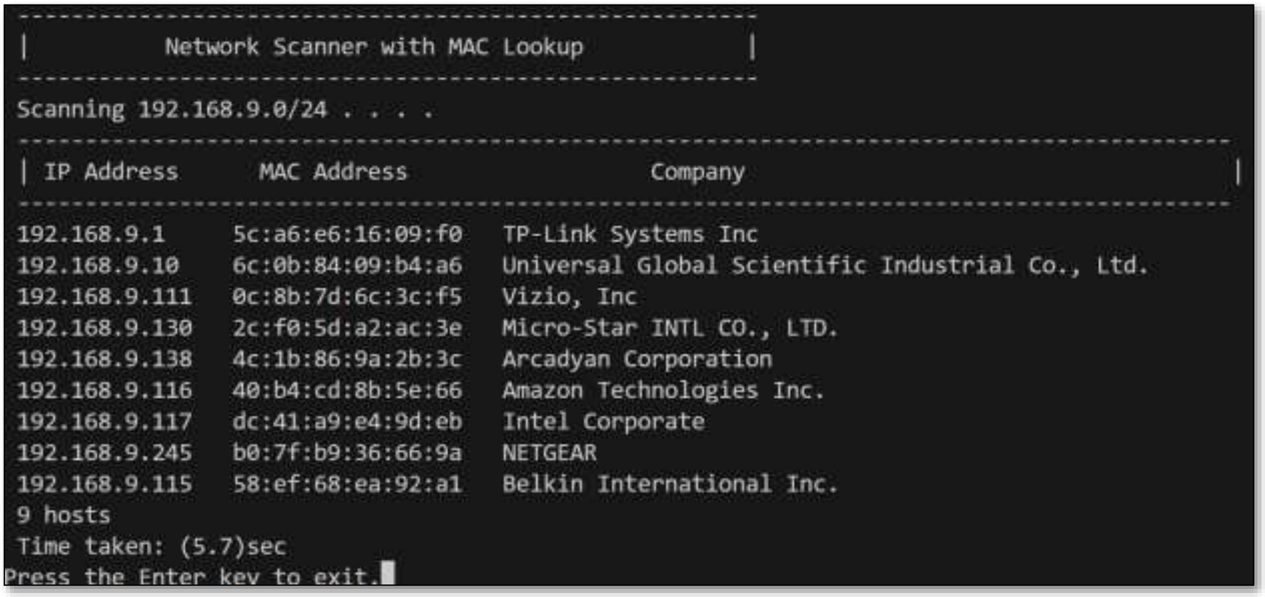
Our finished product looks pretty good!

That’s it, we are done. We can use this hand-built network scanner on any network.

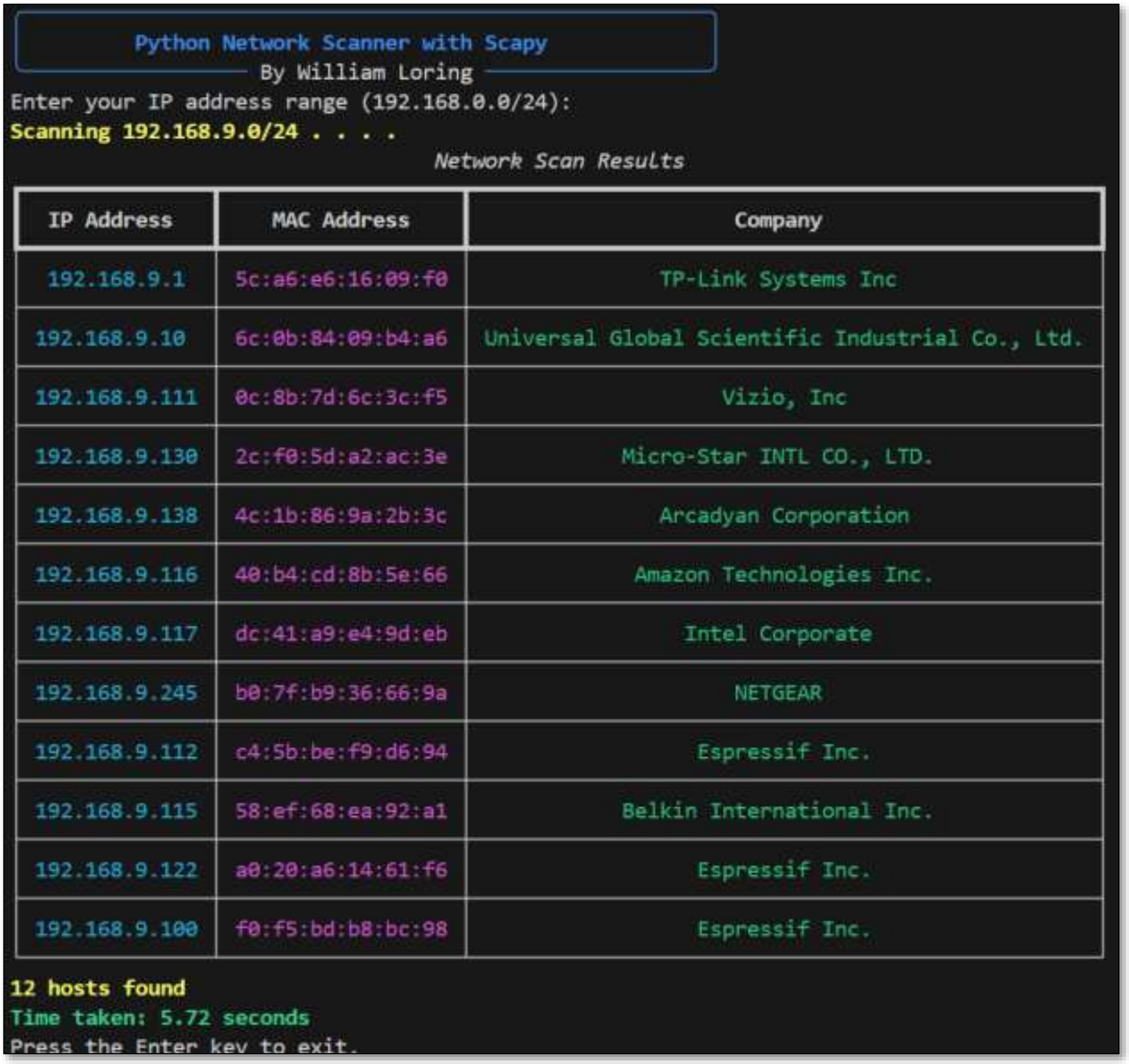
Test your Network Scanner file on Windows and Kali Linux using a bridged adapter.

Challenges

- Use the socket.gethostbyaddress() function to resolve the host names.
- Use rich formatting to spice up the program.
- Use the https://pypi.org/project/mac-vendor-lookup/ library to lookup the manufacturer's name from the MAC address.



```
-------------------------------------------------------
|          Network Scanner with MAC Lookup            |
-------------------------------------------------------
Scanning 192.168.9.0/24 . . . .
------------------------------------------------------------------------------------------
| IP Address      MAC Address                  Company                                  |
------------------------------------------------------------------------------------------
192.168.9.1     5c:a6:e6:16:09:f0   TP-Link Systems Inc
192.168.9.10    6c:0b:84:09:b4:a6   Universal Global Scientific Industrial Co., Ltd.
192.168.9.111   0c:8b:7d:6c:3c:f5   Vizio, Inc
192.168.9.130   2c:f0:5d:a2:ac:3e   Micro-Star INTL CO., LTD.
192.168.9.138   4c:1b:86:9a:2b:3c   Arcadyan Corporation
192.168.9.116   40:b4:cd:8b:5e:66   Amazon Technologies Inc.
192.168.9.117   dc:41:a9:e4:9d:eb   Intel Corporate
192.168.9.245   b0:7f:b9:36:66:9a   NETGEAR
192.168.9.115   58:ef:68:ea:92:a1   Belkin International Inc.
9 hosts
Time taken: (5.7)sec
Press the Enter key to exit.
```



```
            Python Network Scanner with Scapy
─────────────────── By William Loring ───────────────────
Enter your IP address range (192.168.0.0/24):
Scanning 192.168.9.0/24 . . . .
                                 Network Scan Results
┌───────────────┬───────────────────┬──────────────────────────────────────────────────┐
│  IP Address   │    MAC Address    │                     Company                      │
├───────────────┼───────────────────┼──────────────────────────────────────────────────┤
│  192.168.9.1  │ 5c:a6:e6:16:09:f0 │                 TP-Link Systems Inc              │
│ 192.168.9.10  │ 6c:0b:84:09:b4:a6 │ Universal Global Scientific Industrial Co., Ltd. │
│ 192.168.9.111 │ 0c:8b:7d:6c:3c:f5 │                    Vizio, Inc                    │
│ 192.168.9.130 │ 2c:f0:5d:a2:ac:3e │              Micro-Star INTL CO., LTD.           │
│ 192.168.9.138 │ 4c:1b:86:9a:2b:3c │               Arcadyan Corporation               │
│ 192.168.9.116 │ 40:b4:cd:8b:5e:66 │             Amazon Technologies Inc.             │
│ 192.168.9.117 │ dc:41:a9:e4:9d:eb │                 Intel Corporate                  │
│ 192.168.9.245 │ b0:7f:b9:36:66:9a │                     NETGEAR                      │
│ 192.168.9.112 │ c4:5b:be:f9:d6:94 │                  Espressif Inc.                  │
│ 192.168.9.115 │ 58:ef:68:ea:92:a1 │             Belkin International Inc.            │
│ 192.168.9.122 │ a0:20:a6:14:61:f6 │                  Espressif Inc.                  │
│ 192.168.9.100 │ f0:f5:bd:b8:bc:98 │                  Espressif Inc.                  │
└───────────────┴───────────────────┴──────────────────────────────────────────────────┘
12 hosts found
Time taken: 5.72 seconds
Press the Enter key to exit.
```

Assignment Submission

Attach all program files and screenshots of your results from both operating systems to the assignment in BlackBoard.