Task 1: Basic Network Sniffer

1. Ensure Python and Pip are Installed:

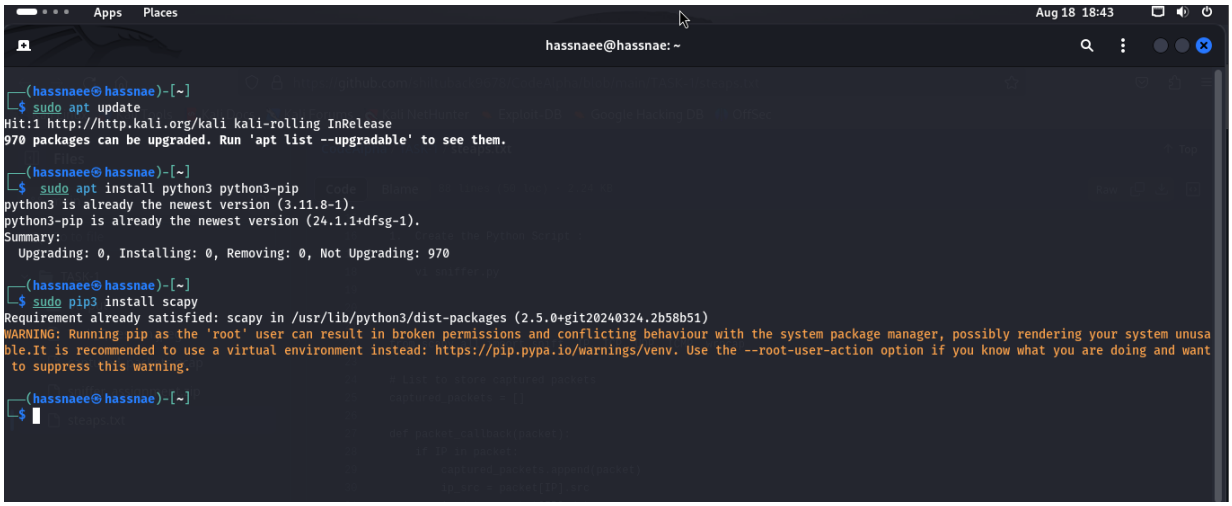
- sudo apt update
- sudo apt install python3 python3-pip

The python3 is already installed in my environment

2. Install Scapy:

Scapy is a Python library used for network packet manipulation, including sniffing, generating, and analysing network packets.
To install the scapy library we need to use this commande line:

- sudo pip3 install scapy



3. Create the Python Script:

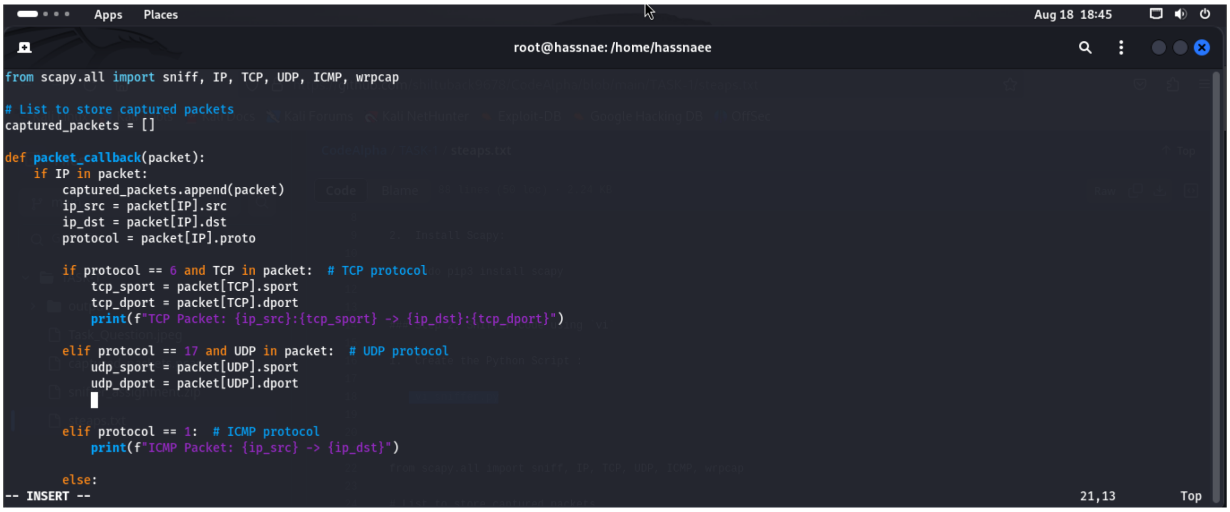
Using the vim text editor to create a new Python script named sniffer.py.
This script will contain the code to sniff and analyze network packets.
To open the vi text editor, we need to use this commande line:

- vi sniffer.py

```
┌──(root㉿hassnae)-[/home/hassnaee]
└─# vi sniffer.py

┌──(root㉿hassnae)-[/home/hassnaee]
└─# chmod +x sniffer.py
```

we need to paste in vim text editor this code:

```
from scapy.all import sniff, IP, TCP, UDP, ICMP, wrpcap

# List to store captured packets
captured_packets = []

def packet_callback(packet):
    if IP in packet:
        captured_packets.append(packet)
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        protocol = packet[IP].proto

        if protocol == 6 and TCP in packet:  # TCP protocol
            tcp_sport = packet[TCP].sport
            tcp_dport = packet[TCP].dport
            print(f"TCP Packet: {ip_src}:{tcp_sport} -> {ip_dst}:{tcp_dport}")

        elif protocol == 17 and UDP in packet:  # UDP protocol
            udp_sport = packet[UDP].sport
            udp_dport = packet[UDP].dport
            print(f"UDP Packet: {ip_src}:{udp_sport} -> {ip_dst}:{udp_dport}")

        elif protocol == 1:  # ICMP protocol
            print(f"ICMP Packet: {ip_src} -> {ip_dst}")

        else:
            print(f"Other IP Packet: {ip_src} -> {ip_dst} (Protocol: {protocol})")
    else:
        print("Non-IP packet detected")

# Start sniffing
sniff(prn=packet_callback, count=50)  # Increase count for more packets

# Save the captured packets to a file
wrpcap('captured_packets.pcap', captured_packets)
```



```
from scapy.all import sniff, IP, TCP, UDP, ICMP, wrpcap

# List to store captured packets
captured_packets = []

def packet_callback(packet):
    if IP in packet:
        captured_packets.append(packet)
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        protocol = packet[IP].proto

        if protocol == 6 and TCP in packet:  # TCP protocol
            tcp_sport = packet[TCP].sport
            tcp_dport = packet[TCP].dport
            print(f"TCP Packet: {ip_src}:{tcp_sport} -> {ip_dst}:{tcp_dport}")

        elif protocol == 17 and UDP in packet:  # UDP protocol
            udp_sport = packet[UDP].sport
            udp_dport = packet[UDP].dport
            

        elif protocol == 1:  # ICMP protocol
            print(f"ICMP Packet: {ip_src} -> {ip_dst}")

        else:
-- INSERT --
```

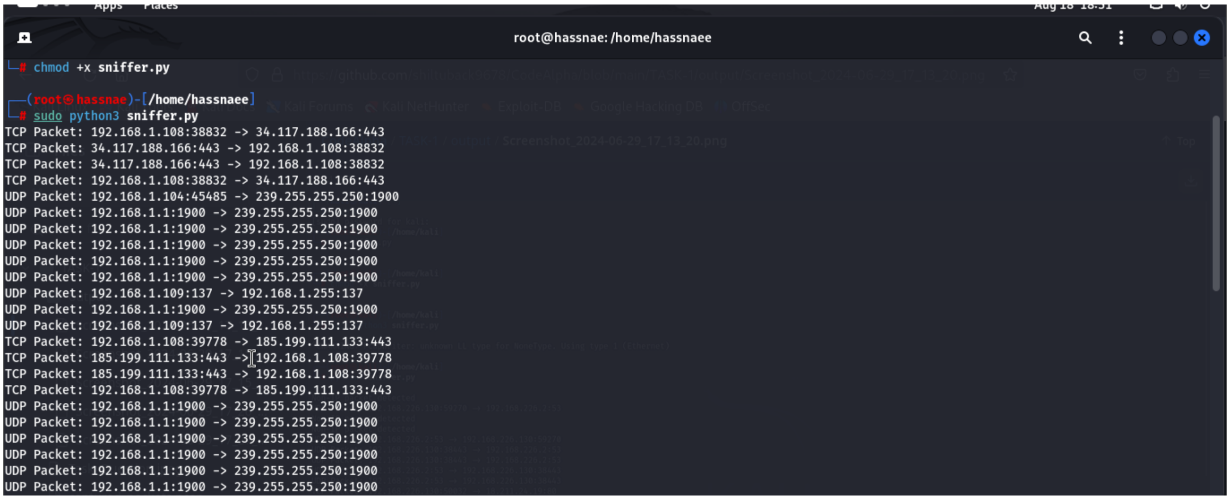
4. Run the Sniffer

To run the sniffer, we need this commande line to make this script executable:

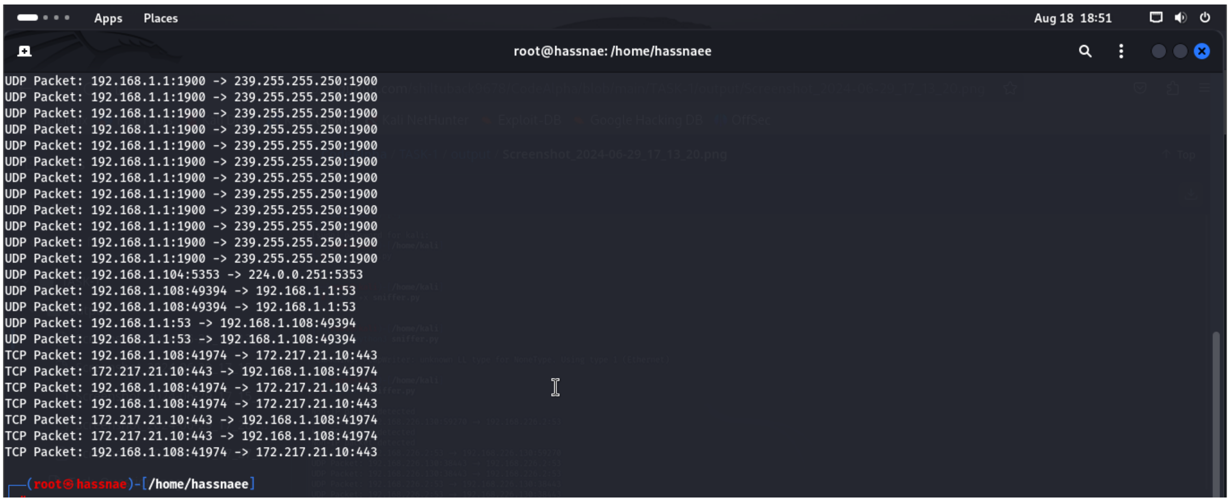
- chmod +x sniffer.py

then we need to Run the Script with Root Privileges

- sudo python3 sniffer.py

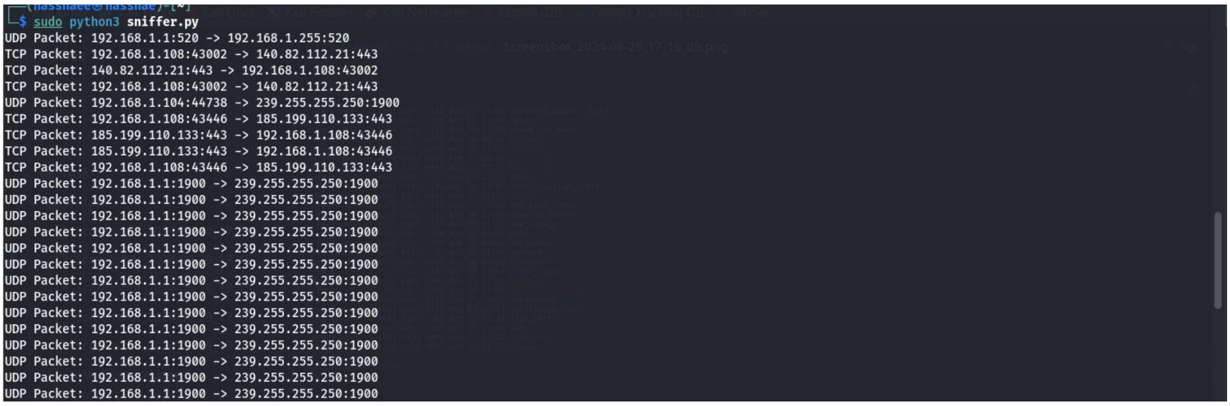


```
└─# chmod +x sniffer.py

┌──(root㉿hassnae)-[/home/hassnaee]
└─# sudo python3 sniffer.py
TCP Packet: 192.168.1.108:38832 -> 34.117.188.166:443
TCP Packet: 34.117.188.166:443 -> 192.168.1.108:38832
TCP Packet: 34.117.188.166:443 -> 192.168.1.108:38832
TCP Packet: 192.168.1.108:38832 -> 34.117.188.166:443
UDP Packet: 192.168.1.104:45485 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.109:137 -> 192.168.1.255:137
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.109:137 -> 192.168.1.255:137
TCP Packet: 192.168.1.108:39778 -> 185.199.111.133:443
TCP Packet: 185.199.111.133:443 -> 192.168.1.108:39778
TCP Packet: 185.199.111.133:443 -> 192.168.1.108:39778
TCP Packet: 192.168.1.108:39778 -> 185.199.111.133:443
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
```

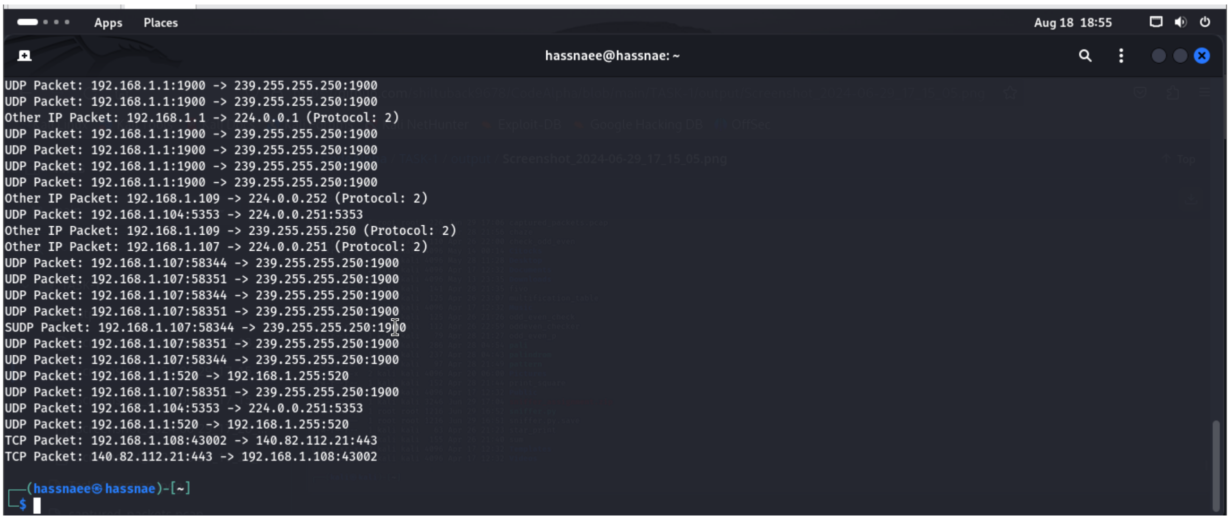


```
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.104:5353 -> 224.0.0.251:5353
UDP Packet: 192.168.1.108:49394 -> 192.168.1.1:53
UDP Packet: 192.168.1.108:49394 -> 192.168.1.1:53
UDP Packet: 192.168.1.1:53 -> 192.168.1.108:49394
UDP Packet: 192.168.1.1:53 -> 192.168.1.108:49394
TCP Packet: 192.168.1.108:41974 -> 172.217.21.10:443
TCP Packet: 172.217.21.10:443 -> 192.168.1.108:41974
TCP Packet: 192.168.1.108:41974 -> 172.217.21.10:443
TCP Packet: 192.168.1.108:41974 -> 172.217.21.10:443
TCP Packet: 172.217.21.10:443 -> 192.168.1.108:41974
TCP Packet: 172.217.21.10:443 -> 192.168.1.108:41974
TCP Packet: 192.168.1.108:41974 -> 172.217.21.10:443

┌──(root㉿hassnae)-[/home/hassnaee]
```

This script will capture 50 packets and save them to a file named captured_packets. pcap.



```
└─$ sudo python3 sniffer.py
UDP Packet: 192.168.1.1:520 -> 192.168.1.255:520
TCP Packet: 192.168.1.108:43002 -> 140.82.112.21:443
TCP Packet: 140.82.112.21:443 -> 192.168.1.108:43002
TCP Packet: 192.168.1.108:43002 -> 140.82.112.21:443
UDP Packet: 192.168.1.104:44738 -> 239.255.255.250:1900
TCP Packet: 192.168.1.108:43446 -> 185.199.110.133:443
TCP Packet: 185.199.110.133:443 -> 192.168.1.108:43446
TCP Packet: 185.199.110.133:443 -> 192.168.1.108:43446
TCP Packet: 192.168.1.108:43446 -> 185.199.110.133:443
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
```



```
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
Other IP Packet: 192.168.1.1 -> 224.0.0.1 (Protocol: 2)
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:1900 -> 239.255.255.250:1900
Other IP Packet: 192.168.1.109 -> 224.0.0.252 (Protocol: 2)
UDP Packet: 192.168.1.104:5353 -> 224.0.0.251:5353
Other IP Packet: 192.168.1.109 -> 239.255.255.250 (Protocol: 2)
Other IP Packet: 192.168.1.107 -> 224.0.0.251 (Protocol: 2)
UDP Packet: 192.168.1.107:58344 -> 239.255.255.250:1900
UDP Packet: 192.168.1.107:58351 -> 239.255.255.250:1900
UDP Packet: 192.168.1.107:58344 -> 239.255.255.250:1900
UDP Packet: 192.168.1.107:58351 -> 239.255.255.250:1900
SUDP Packet: 192.168.1.107:58344 -> 239.255.255.250:1900
UDP Packet: 192.168.1.107:58351 -> 239.255.255.250:1900
UDP Packet: 192.168.1.107:58344 -> 239.255.255.250:1900
UDP Packet: 192.168.1.1:520 -> 192.168.1.255:520
UDP Packet: 192.168.1.107:58351 -> 239.255.255.250:1900
UDP Packet: 192.168.1.104:5353 -> 224.0.0.251:5353
UDP Packet: 192.168.1.1:520 -> 192.168.1.255:520
TCP Packet: 192.168.1.108:43002 -> 140.82.112.21:443
TCP Packet: 140.82.112.21:443 -> 192.168.1.108:43002

┌──(hassnaee㉿hassnae)-[~]
└─$ █
```

To verify the installation of this package, we have to use this commande:

- Ls -l

Ls : is used to list the directories and files in the current directories.
l : is used to list all the files and directories files in the directories.

```
┌──(hassnaee㉿hassnae)-[~]
└─$ ls -l
total 52
drwxr-xr-x 2 hassnaee hassnaee  4096 Aug 18 17:21 Desktop
drwxr-xr-x 2 hassnaee hassnaee  4096 Aug 18 17:21 Documents
drwxr-xr-x 2 hassnaee hassnaee  4096 Aug 18 17:21 Downloads
drwxr-xr-x 2 hassnaee hassnaee  4096 Aug 18 17:21 Music
drwxr-xr-x 2 hassnaee hassnaee  4096 Aug 18 17:21 Pictures
drwxr-xr-x 2 hassnaee hassnaee  4096 Aug 18 17:21 Public
drwxr-xr-x 2 hassnaee hassnaee  4096 Aug 18 17:21 Templates
drwxr-xr-x 2 hassnaee hassnaee  4096 Aug 18 17:21 Videos
-rw-r--r-- 1 root     root     12430 Aug 18 18:54 captured_packets.pcap
-rwxrwxr-x 1 hassnaee hassnaee  1215 Aug 18 18:38 sniffer.py
```

5. Install Wireshark (if not already installed)

Using this next commande we can install the Wireshark if is not exist

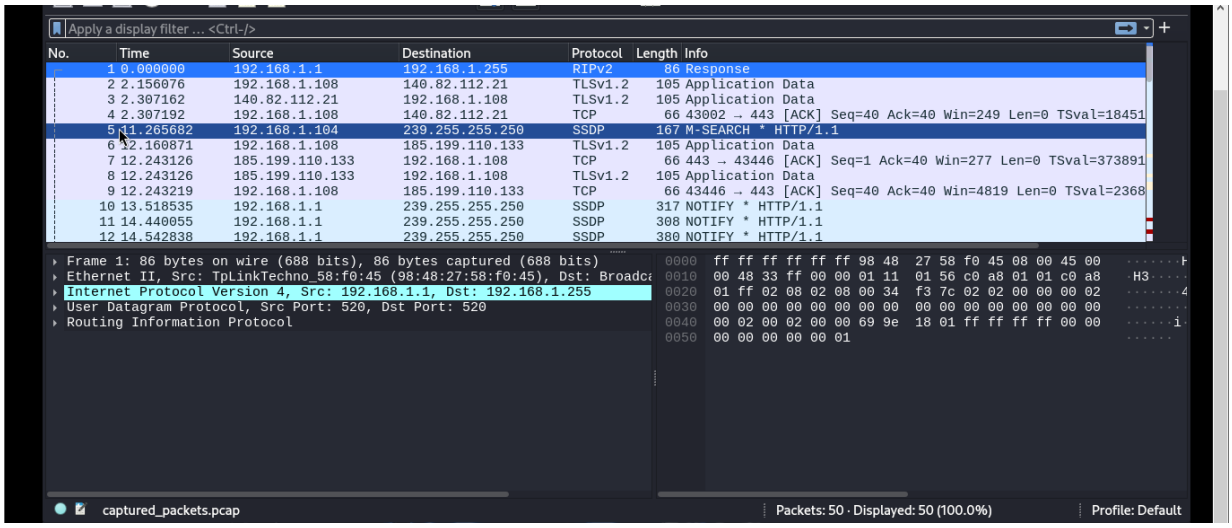
- sudo apt install Wireshark

```
┌──(hassnaee㉿hassnae)-[~]
└─$ sudo apt install wireshark
wireshark is already the newest version (4.2.5-1).
wireshark set to manually installed.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 970
```

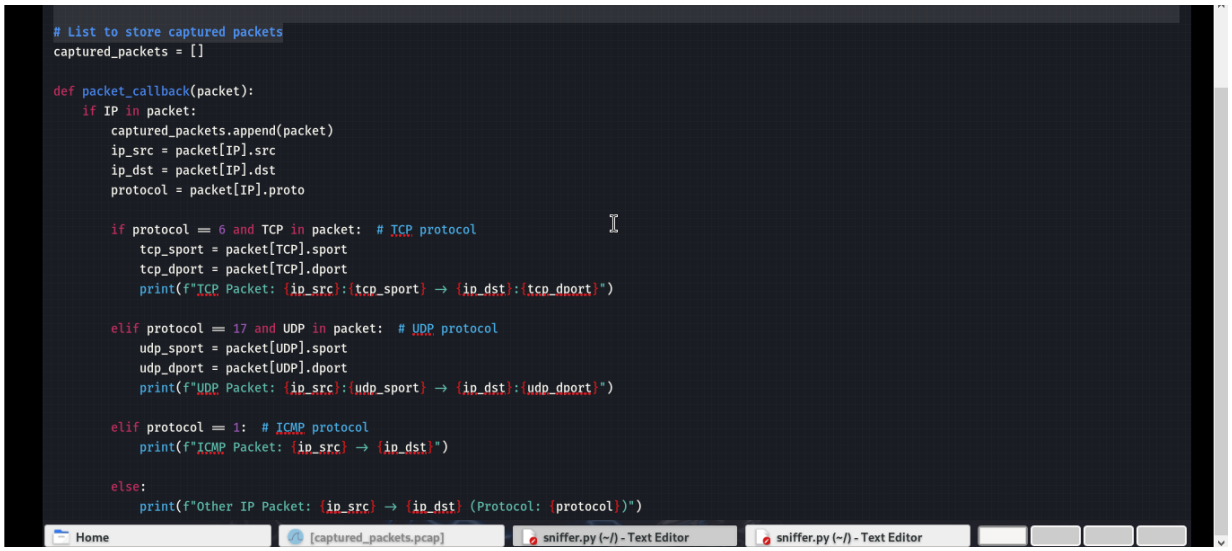
In this case the Wireshark had already installed
"**Wireshark** is a free and open-source packet analyser. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues."

6. Open the Captured Packets in Wireshark:

- **captured_packets. pcap**



- **sniffer.py**



```python
# List to store captured packets
captured_packets = []

def packet_callback(packet):
    if IP in packet:
        captured_packets.append(packet)
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        protocol = packet[IP].proto

        if protocol == 6 and TCP in packet:  # TCP protocol
            tcp_sport = packet[TCP].sport
            tcp_dport = packet[TCP].dport
            print(f"TCP Packet: {ip_src}:{tcp_sport} → {ip_dst}:{tcp_dport}")

        elif protocol == 17 and UDP in packet:  # UDP protocol
            udp_sport = packet[UDP].sport
            udp_dport = packet[UDP].dport
            print(f"UDP Packet: {ip_src}:{udp_sport} → {ip_dst}:{udp_dport}")

        elif protocol == 1:  # ICMP protocol
            print(f"ICMP Packet: {ip_src} → {ip_dst}")

        else:
            print(f"Other IP Packet: {ip_src} → {ip_dst} (Protocol: {protocol})")
```

7. Prepare Submission

Package the Python script and the .pcap file into a ZIP archive for submission.

Compress the Files using this commande:

- zip sniffer_assignment.zip sniffer.py captured_packets. Pcap