TASK -1

Basic Network Sniffer

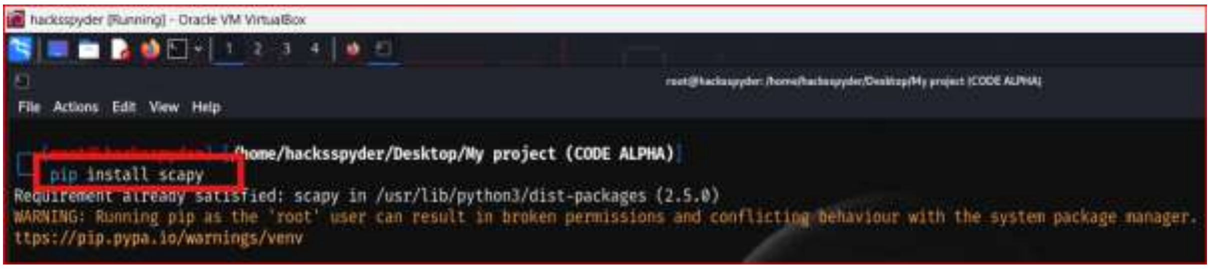
Build a network sniffer in Python that captures and analyzes network traffic. This project will help you understand how data flows on a network and how network packets are structured.

To build a basic network sniffer in Python on Linux, you can use the scapy library, which allows you to interact with network packets, send and receive them, and manipulate network layers easily.

Here's a step-by-step guide to creating a simple network sniffer:

Step 1: Install Scapy

First, you need to install the scapy library. If you haven't already installed it, run this command:

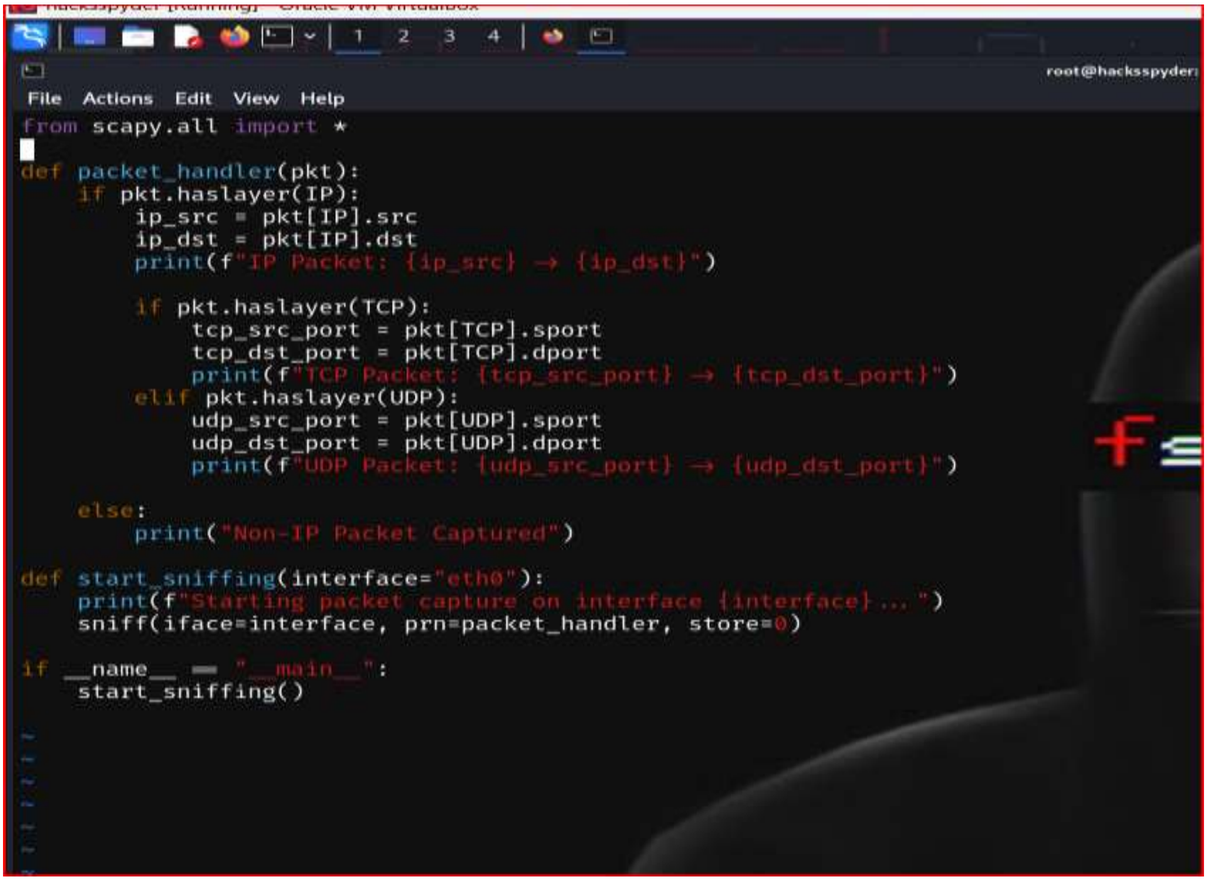


hacksspyder [Running] - Oracle VM VirtualBox
1 2 3 4
root@hacksspyder: /home/hacksspyder/Desktop/My project (CODE ALPHA)
File Actions Edit View Help

```
┌──(root㉿hacksspyder)-[/home/hacksspyder/Desktop/My project (CODE ALPHA)]
└─# pip install scapy
Requirement already satisfied: scapy in /usr/lib/python3/dist-packages (2.5.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager.
ttps://pip.pypa.io/warnings/venv
```

Step 2: Create the Sniffer Script

Now, let's write a Python script to capture and analyze network packets.



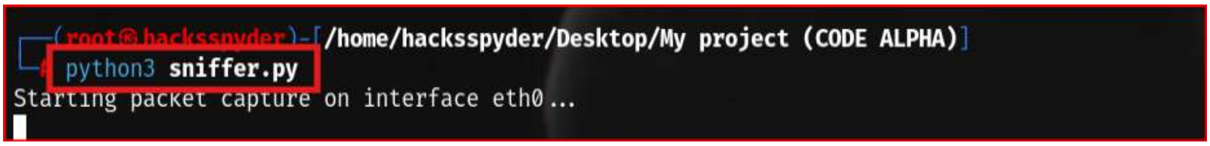
```
from scapy.all import *

def packet_handler(pkt):
    if pkt.haslayer(IP):
        ip_src = pkt[IP].src
        ip_dst = pkt[IP].dst
        print(f"IP Packet: {ip_src} → {ip_dst}")

        if pkt.haslayer(TCP):
            tcp_src_port = pkt[TCP].sport
            tcp_dst_port = pkt[TCP].dport
            print(f"TCP Packet: {tcp_src_port} → {tcp_dst_port}")
        elif pkt.haslayer(UDP):
            udp_src_port = pkt[UDP].sport
            udp_dst_port = pkt[UDP].dport
            print(f"UDP Packet: {udp_src_port} → {udp_dst_port}")

    else:
        print("Non-IP Packet Captured")

def start_sniffing(interface="eth0"):
    print(f"Starting packet capture on interface {interface}...")
    sniff(iface=interface, prn=packet_handler, store=0)

if __name__ == "__main__":
    start_sniffing()

~
~
~
~
~
~
~
```

Step 3: Explanation of Code

- **Imports**: We import everything from scapy.all. scapy is powerful, allowing you to create, manipulate, and analyze network packets.
- **Packet Handler**: The packet_handler() function will be called for every packet captured. It checks if the packet has an IP layer, then extracts the source and destination IP addresses. If it is a TCP or UDP packet, it extracts the source and destination ports as well.
- **sniff()**: The sniff() function is the core of the packet capture process. It starts capturing packets on the given interface (default is eth0). The prn argument specifies the callback function (packet_handler()) that will be called for every captured packet.
- **Interface**: The interface="eth0" specifies that we are sniffing on the Ethernet interface. You might need to change this depending on your network setup (e.g., wlan0 for Wi-Fi).
- **Store**: We set store=0 to avoid storing captured packets in memory. This is useful if you’re analyzing large volumes of traffic in real-time.

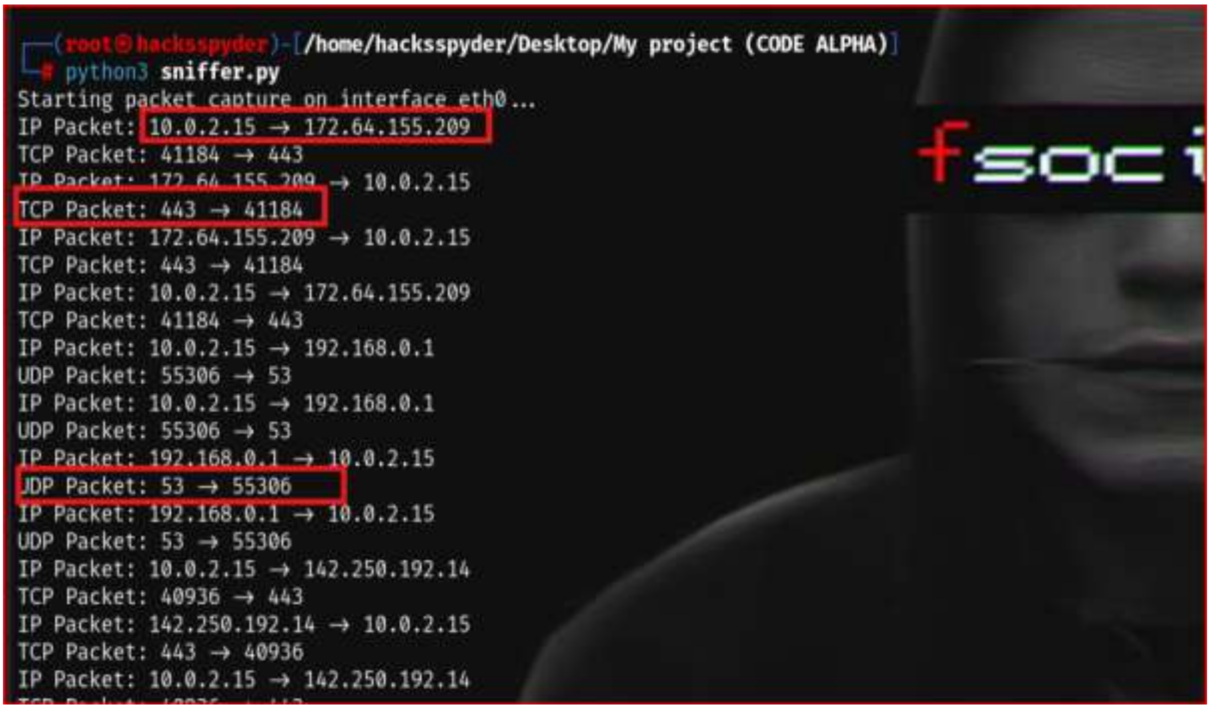
Step 4: Run the Script

1. **Permissions**: Running this script may require root privileges, as sniffing packets typically requires administrative access.



```
┌──(root㉿hacksspyder)-[/home/hacksspyder/Desktop/My project (CODE ALPHA)]
└─# python3 sniffer.py
Starting packet capture on interface eth0 ...
```

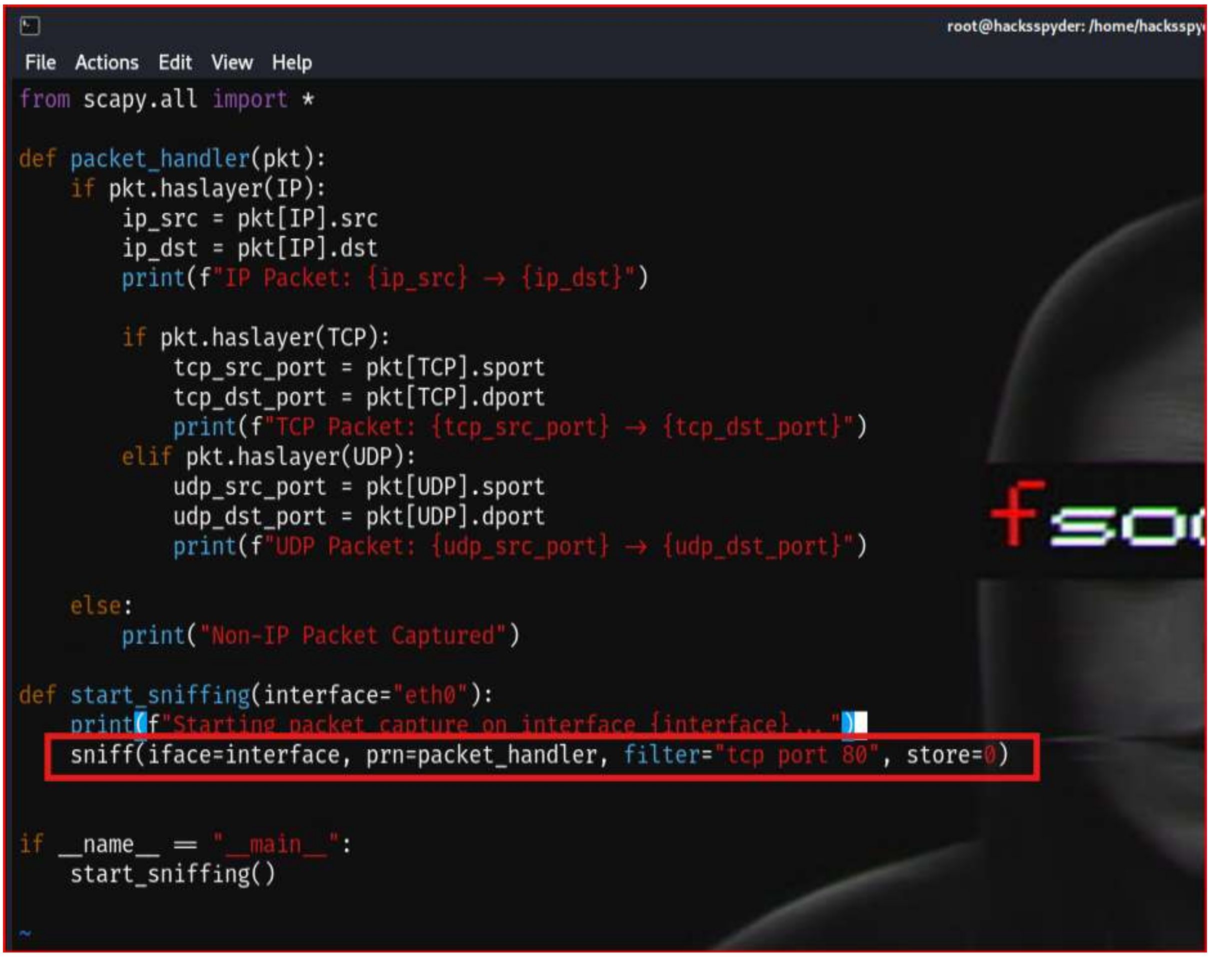
2. **Output**: The output will look something like this:



```
┌──(root㉿hacksspyder)-[/home/hacksspyder/Desktop/My project (CODE ALPHA)]
└─# python3 sniffer.py
Starting packet capture on interface eth0 ...
IP Packet: 10.0.2.15 → 172.64.155.209
TCP Packet: 41184 → 443
IP Packet: 172.64.155.209 → 10.0.2.15
TCP Packet: 443 → 41184
IP Packet: 172.64.155.209 → 10.0.2.15
TCP Packet: 443 → 41184
IP Packet: 10.0.2.15 → 172.64.155.209
TCP Packet: 41184 → 443
IP Packet: 10.0.2.15 → 192.168.0.1
UDP Packet: 55306 → 53
IP Packet: 10.0.2.15 → 192.168.0.1
UDP Packet: 55306 → 53
IP Packet: 192.168.0.1 → 10.0.2.15
UDP Packet: 53 → 55306
IP Packet: 192.168.0.1 → 10.0.2.15
UDP Packet: 53 → 55306
IP Packet: 10.0.2.15 → 142.250.192.14
TCP Packet: 40936 → 443
IP Packet: 142.250.192.14 → 10.0.2.15
TCP Packet: 443 → 40936
IP Packet: 10.0.2.15 → 142.250.192.14
```

Step 5: Customize the Sniffer

- **Filter**: You can apply filters to capture only specific packets, such as only capturing HTTP traffic. For example:


root@hackssspyder: /home/hacksspy
File Actions Edit View Help
fsoc

```python
from scapy.all import *

def packet_handler(pkt):
    if pkt.haslayer(IP):
        ip_src = pkt[IP].src
        ip_dst = pkt[IP].dst
        print(f"IP Packet: {ip_src} → {ip_dst}")

        if pkt.haslayer(TCP):
            tcp_src_port = pkt[TCP].sport
            tcp_dst_port = pkt[TCP].dport
            print(f"TCP Packet: {tcp_src_port} → {tcp_dst_port}")
        elif pkt.haslayer(UDP):
            udp_src_port = pkt[UDP].sport
            udp_dst_port = pkt[UDP].dport
            print(f"UDP Packet: {udp_src_port} → {udp_dst_port}")

    else:
        print("Non-IP Packet Captured")

def start_sniffing(interface="eth0"):
    print(f"Starting packet capture on interface {interface}...")
    sniff(iface=interface, prn=packet_handler, filter="tcp port 80", store=0)


if __name__ == "__main__":
    start_sniffing()

~
```

- **Packet Analysis**: You can extend the packet_handler() function to analyze more layers of the packet (e.g., DNS, HTTP headers). Scapy can automatically decode a large number of protocols.

- **GUI**: For a more advanced project, you can create a graphical user interface (GUI) using libraries like Tkinter or PyQt5 to visualize captured traffic.

Step 6: Important Notes

- **Root Privileges**: Packet sniffing usually requires root privileges, so ensure you run the script with sudo.

- **Performance**: Capturing network traffic on busy networks can overwhelm your system. Consider adding filters to narrow down the traffic you're analyzing.

- **Legal Considerations**: Only capture traffic on networks you own or have explicit permission to monitor. Unauthorized sniffing can be illegal in many places.