

BASIC NETWORK SNIFFER

Title: Building a Python Packet Sniffer to Capture and Analyze Network Traffic

Task Objective

To build a Python program that captures network traffic packets, analyzes them to understand their structure and content, and displays useful information such as:

- Source IP
 - Destination IP
 - Protocol (TCP/UDP/ICMP)
 - Payload data
-

Tools and Technologies Used

- **Operating System:** Kali Linux
 - **Programming Language:** Python 3
 - **Libraries Used:** `scapy`
 - **Terminal:** Kali Terminal
 - **Environment:** Python Virtual Environment (`venv`)
-

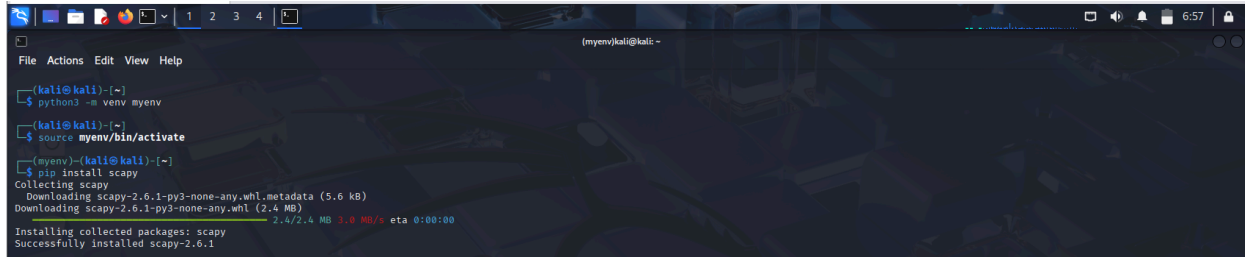
Step-by-Step Implementation

Step 1: Create a Virtual Environment

```
python3 -m venv myenv
```

Activate it:

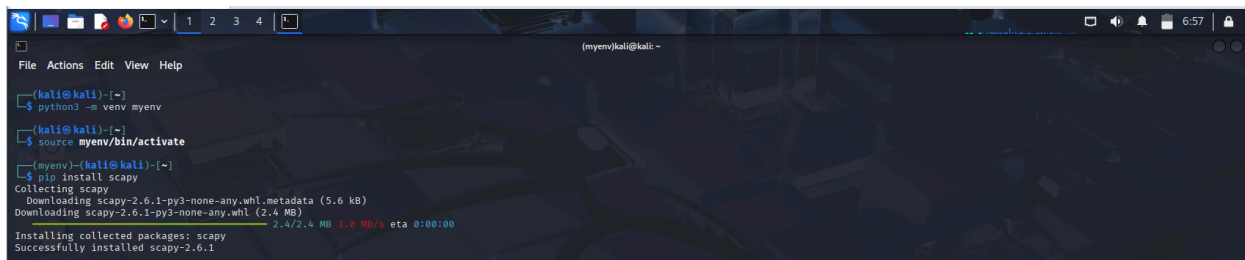
```
source myenv/bin/activate
```



```
(kali@kali)-[~]
$ python3 -m venv myenv
(kali@kali)-[~]
$ source myenv/bin/activate
(myenv)-(kali@kali)-[~]
$ pip install scapy
Collecting scapy
  Downloading scapy-2.6.1-py3-none-any.whl.metadata (5.6 kB)
  Downloading scapy-2.6.1-py3-none-any.whl (2.4 MB)
    2.4/2.4 MB 3.8 MB/s eta 0:00:00
Installing collected packages: scapy
Successfully installed scapy-2.6.1
```

Step 2: Install Scapy Library Inside the Virtual Environment

```
pip install scapy
```



```
(kali@kali)-[~]
$ python3 -m venv myenv
(kali@kali)-[~]
$ source myenv/bin/activate
(myenv)-(kali@kali)-[~]
$ pip install scapy
Collecting scapy
  Downloading scapy-2.6.1-py3-none-any.whl.metadata (5.6 kB)
  Downloading scapy-2.6.1-py3-none-any.whl (2.4 MB)
    2.4/2.4 MB 3.8 MB/s eta 0:00:00
Installing collected packages: scapy
Successfully installed scapy-2.6.1
```

Step 3: Create the Packet Sniffer Script

Create a Python file:

```
nano packet_sniffer.py
```

Paste the following code:

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

```
def analyze_packet(packet):
    if IP in packet:
        ip_layer = packet[IP]
        src_ip = ip_layer.src
```

```
dst_ip = ip_layer.dst
protocol = ip_layer.proto

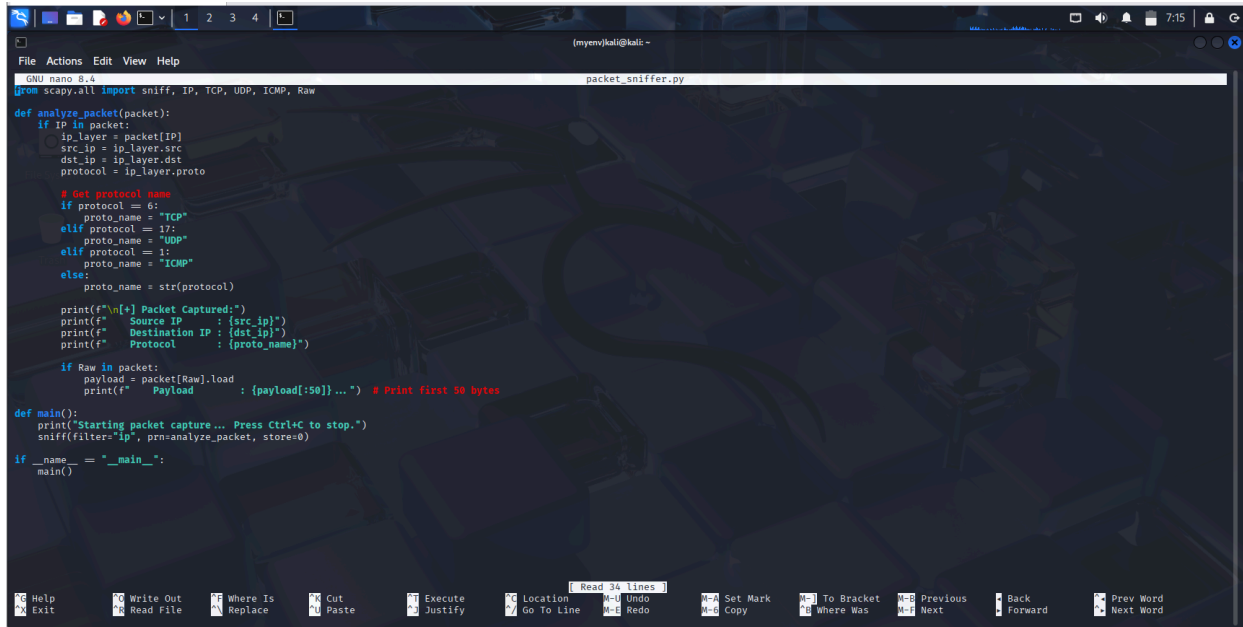
if protocol == 6:
    proto_name = "TCP"
elif protocol == 17:
    proto_name = "UDP"
elif protocol == 1:
    proto_name = "ICMP"
else:
    proto_name = str(protocol)

print(f"\n[+] Packet Captured:")
print(f"    Source IP      : {src_ip}")
print(f"    Destination IP : {dst_ip}")
print(f"    Protocol       : {proto_name}")

if Raw in packet:
    payload = packet[Raw].load
    print(f"    Payload          : {payload[:50]}...")

def main():
    print("Starting packet capture... Press Ctrl+C to stop.")
    sniff(filter="ip", prn=analyze_packet, store=0)

if __name__ == "__main__":
    main()
```



```
GNU nano 2.8.4 packet_sniffer.py
from scapy.all import sniff, IP, TCP, UDP, ICMP, Raw

def analyze_packet(packet):
    if IP in packet:
        ip_layer = packet[IP]
        src_ip = ip_layer.src
        dst_ip = ip_layer.dst
        protocol = ip_layer.proto

        # Get protocol name
        if protocol == 6:
            proto_name = "TCP"
        elif protocol == 17:
            proto_name = "UDP"
        elif protocol == 1:
            proto_name = "ICMP"
        else:
            proto_name = str(protocol)

        print(f"\n[*] Packet Captured:")
        print(f"   Source IP   : {src_ip}")
        print(f"   Destination IP : {dst_ip}")
        print(f"   Protocol    : {proto_name}")

        if Raw in packet:
            payload = packet[Raw].load
            print(f"   Payload      : {payload[:50]} ... " # Print first 50 bytes

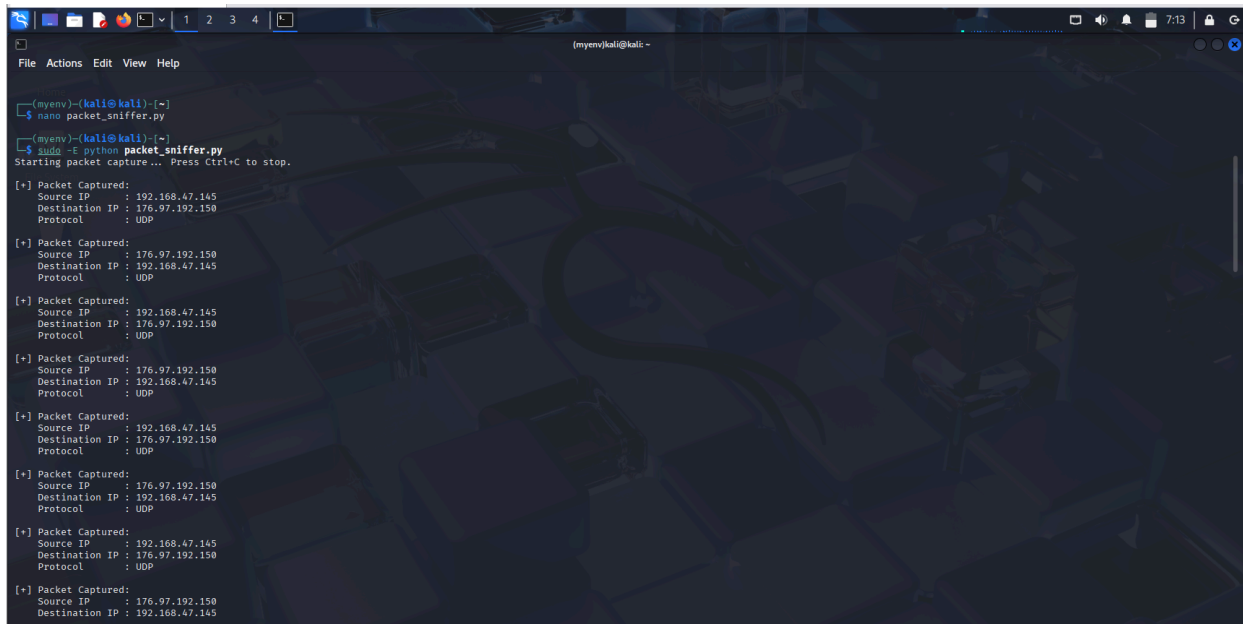
def main():
    print("Starting packet capture... Press Ctrl+C to stop.")
    sniff(filter="ip", prn=analyze_packet, store=0)

if __name__ == "__main__":
    main()
```

Step 4: Run the Script with Root Privileges

Use this command:

```
sudo -E python packet_sniffer.py
```



```
(myenv)-(kali@kali)-[~]
└─$ nano packet_sniffer.py
(myenv)-(kali@kali)-[~]
└─$ sudo -E python packet_sniffer.py
Starting packet capture... Press Ctrl+C to stop.

[*] Packet Captured:
Source IP   : 192.168.47.145
Destination IP : 176.97.192.150
Protocol    : UDP

[*] Packet Captured:
Source IP   : 176.97.192.150
Destination IP : 192.168.47.145
Protocol    : UDP

[*] Packet Captured:
Source IP   : 192.168.47.145
Destination IP : 176.97.192.150
Protocol    : UDP

[*] Packet Captured:
Source IP   : 176.97.192.150
Destination IP : 192.168.47.145
Protocol    : UDP

[*] Packet Captured:
Source IP   : 192.168.47.145
Destination IP : 176.97.192.150
Protocol    : UDP

[*] Packet Captured:
Source IP   : 176.97.192.150
Destination IP : 192.168.47.145
Protocol    : UDP

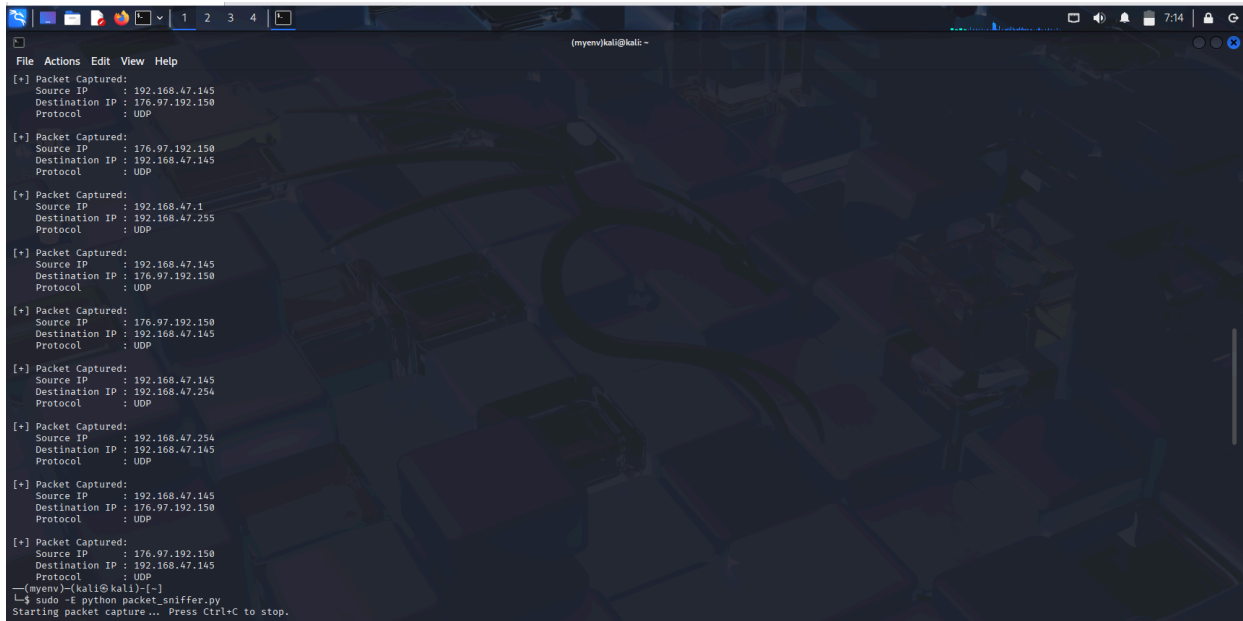
[*] Packet Captured:
Source IP   : 192.168.47.145
Destination IP : 176.97.192.150
Protocol    : UDP

[*] Packet Captured:
Source IP   : 176.97.192.150
Destination IP : 192.168.47.145
Protocol    : UDP
```

Step 5: Stop the Capture

When enough packets are captured, stop the program using:

Ctrl + C



```
(myenv)kali@kali:~$ python packet_sniffer.py
[*] Packet Captured:
Source IP : 192.168.47.145
Destination IP : 176.97.192.150
Protocol : UDP

[*] Packet Captured:
Source IP : 176.97.192.150
Destination IP : 192.168.47.145
Protocol : UDP

[*] Packet Captured:
Source IP : 192.168.47.1
Destination IP : 192.168.47.255
Protocol : UDP

[*] Packet Captured:
Source IP : 192.168.47.145
Destination IP : 176.97.192.150
Protocol : UDP

[*] Packet Captured:
Source IP : 176.97.192.150
Destination IP : 192.168.47.145
Protocol : UDP

[*] Packet Captured:
Source IP : 192.168.47.145
Destination IP : 192.168.47.254
Protocol : UDP

[*] Packet Captured:
Source IP : 192.168.47.254
Destination IP : 192.168.47.145
Protocol : UDP

[*] Packet Captured:
Source IP : 192.168.47.145
Destination IP : 176.97.192.150
Protocol : UDP

[*] Packet Captured:
Source IP : 176.97.192.150
Destination IP : 192.168.47.145
Protocol : UDP

(myenv)kali@kali:~$ sudo -E python packet_sniffer.py
Starting packet capture... Press Ctrl+C to stop.
```

Summary of What I Learned

- **Scapy Usage:** How to use Scapy to sniff and analyze real-time network packets.
- **Protocols:**
 - **TCP** ensures reliable delivery of data.
 - **UDP** sends fast but doesn't guarantee delivery.
 - **ICMP** helps with network diagnostics (e.g., ping).
- **Data Flow:** Data flows through packets that have headers (with source/destination info) and payload (the actual content).
- **Packet Analysis:** I learned how to extract and print data from raw packet headers and payloads.