**CodeAlpha**

**Task 1:**

**Name: Neha shinde**

**Student ID:**

**BASIC NETWORK SNIFFER**:

***Task***: 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.

**About Project:**

This project is a network sniffer tool implemented in Python. It captures and analyzes network traffic, providing detailed information about each packet. The tool utilizes the Npcap library for packet capture on Windows systems**.**

**Features:**

1. Captures network packets in real-time.
2. Provides detailed information about each packet, including source and destination IP addresses, protocol, packet length, time of capture, TTL, flags, and more.
3. Supports analysis of various protocols such as TCP, UDP, and ICMP.
4. Displays source and destination MAC addresses for Ethernet packets.
5. Supports capturing and analyzing fragmented IP packets.
6. Provides TCP-specific information such as sequence numbers, acknowledgment numbers, and TCP flags.

**Installation:**

1. Install the required dependencies:

* *pip install scapy*

1. Install Npcap:

* Visit the Npcap website and download the appropriate installer for your system.
* Run the installer and follow the installation instructions.

1. Navigate to the project directory
2. Run the network sniffer script:

* python sniffer.py

1. The script will start capturing network packets and display detailed information about each packet in real-time.

**Code :**

from scapy.all import \*

def sniff\_packets(packet):

if IP in packet:

ip\_src = packet[IP].src

ip\_dst = packet[IP].dst

proto = packet[IP].proto

ttl = packet[IP].ttl

flags = packet[IP].flags

frag\_offset = packet[IP].frag

length = len(packet)

time = packet.time

print(f"Time: {time}, Source IP: {ip\_src}, Destination IP: {ip\_dst}, Protocol: {proto}, Length: {length}")

print(f" TTL: {ttl}, Flags: {flags}, Fragment Offset: {frag\_offset}")

# Additional information based on protocol

if Ether in packet:

src\_mac = packet[Ether].src

dst\_mac = packet[Ether].dst

print(f" Source MAC: {src\_mac}, Destination MAC: {dst\_mac}")

if proto == 6 and TCP in packet:

sport = packet[TCP].sport

dport = packet[TCP].dport

seq = packet[TCP].seq

ack = packet[TCP].ack

flags = packet[TCP].flags

payload = packet[TCP].payload

print(f" TCP Source Port: {sport}, Destination Port: {dport}")

print(f" Sequence Number: {seq}, Acknowledgment Number: {ack}, Flags: {flags}")

print(f" Payload: {payload}")

elif proto == 17 and UDP in packet:

sport = packet[UDP].sport

dport = packet[UDP].dport

payload = packet[UDP].payload

print(f" UDP Source Port: {sport}, Destination Port: {dport}")

print(f" Payload: {payload}")

elif proto == 1 and ICMP in packet:

icmp\_type = packet[ICMP].type

icmp\_code = packet[ICMP].code

payload = packet[ICMP].payload

print(f" ICMP Type: {icmp\_type}, Code: {icmp\_code}")

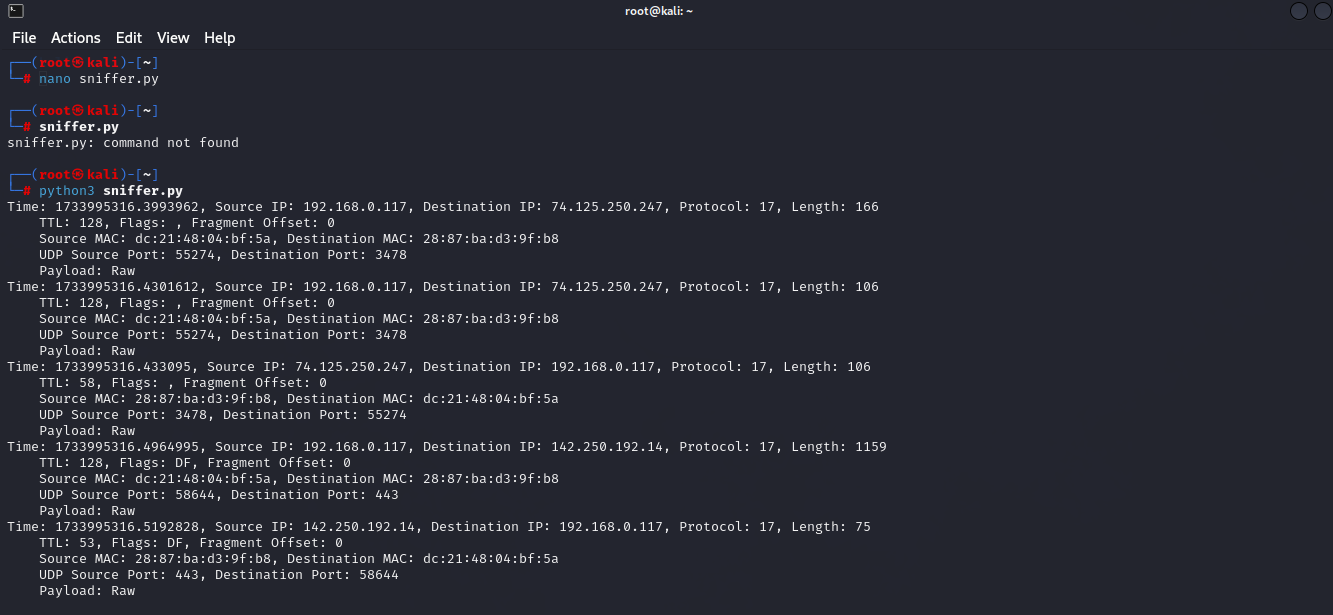
print(f" Payload: {payload}")

else:

print(" Other protocol or unrecognized packet")

sniff(prn=sniff\_packets, store=0)

**Output:**

****