NAME : PRAVEEN V

ROLL NO.: 241901082

# Exercise 6

IMPLEMENTATION OF PACKET SNIFFING USING RAW SOCKETS IN PYTHON

INTRODUCTION:

Packet sniffing reads raw network packets from your NIC so you can see headers (Ethernet/IP/TCP/UDP) and a bit of payload. This simple experiment uses Linux raw sockets to capture and print a brief summary for each IPv4 packet.

AIM:

Write a minimal Python program that captures packets using a raw socket and prints source/destination IP, protocol, ports (if TCP/UDP) and a short hex dump of the payload.

ALGORITHM :

1. Open a raw AF\_PACKET socket (capture all EtherTypes).

2. Loop: receive a packet.

3. Parse Ethernet header; if IPv4, parse IP header.

4. If TCP/UDP, parse ports. Print a one-line summary + short hex of payload.

5. Repeat until Ctrl+C.

CODE:

import socket

import struct

import binascii

import textwrap

def main():

host = socket.gethostbyname(socket.gethostname())

print('IP: {}'.format(host))

conn = socket.socket(socket.AF\_INET, socket.SOCK\_RAW, socket.IPPROTO\_IP)

conn.bind((host, 0))

conn.setsockopt(socket.IPPROTO\_IP, socket.IP\_HDRINCL, 1)

conn.ioctl(socket.SIO\_RCVALL, socket.RCVALL\_ON)

while True:

raw\_data, addr = conn.recvfrom(65536)

dest\_mac, src\_mac, eth\_proto, data = ethernet\_frame(raw\_data)

print('\nEthernet Frame:')

print("Destination MAC: {}".format(dest\_mac))

print("Source MAC: {}".format(src\_mac))

print("Protocol: {}".format(eth\_proto))

def ethernet\_frame(data):

dest\_mac, src\_mac, proto = struct.unpack('!6s6s2s', data[:14])

return get\_mac\_addr(dest\_mac), get\_mac\_addr(src\_mac), get\_protocol(proto), data[14:]

def get\_mac\_addr(bytes\_addr):

bytes\_str = map('{:02x}'.format, bytes\_addr)

mac\_address = ':'.join(bytes\_str).upper()

return mac\_address

def get\_protocol(bytes\_proto):

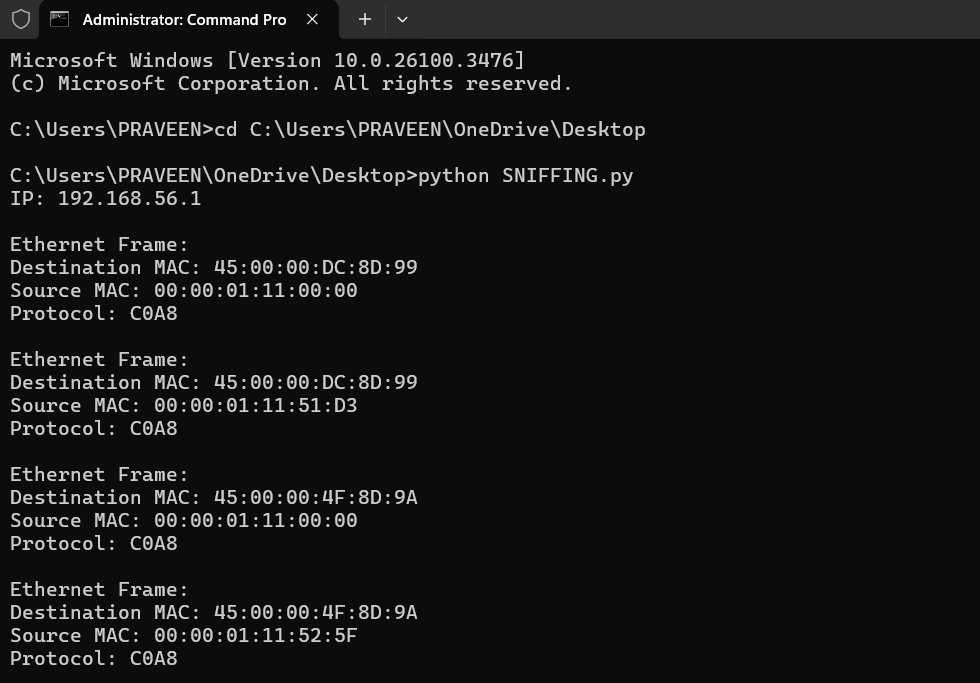
bytes\_str = map('{:02x}'.format, bytes\_proto)

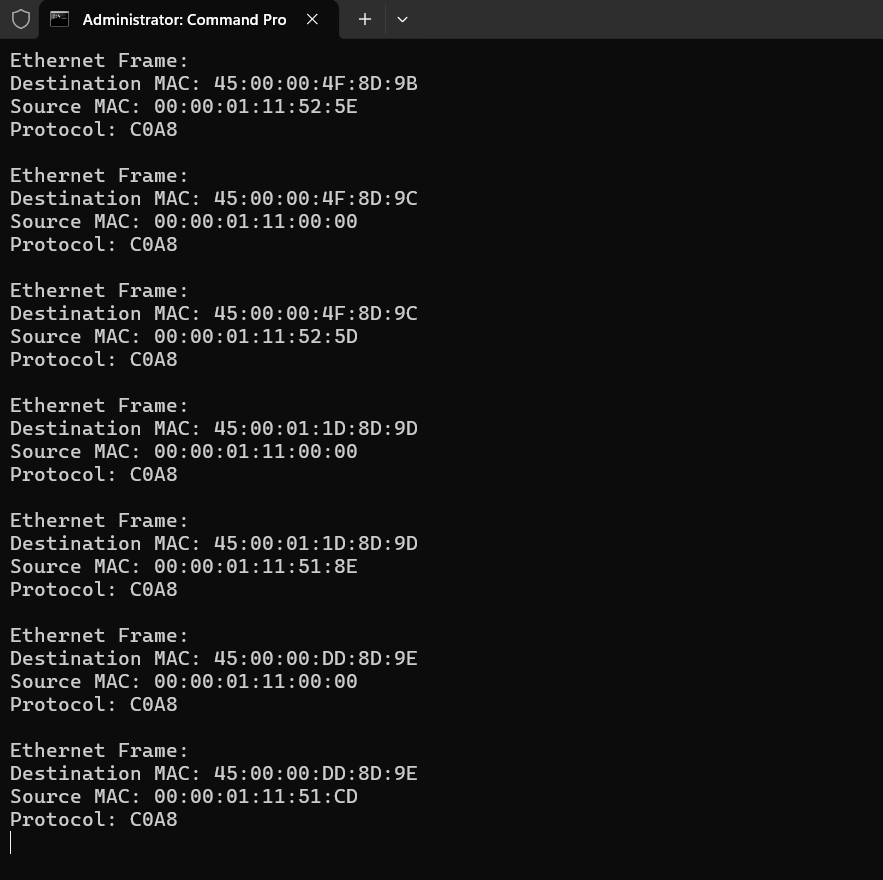
protocol = ''.join(bytes\_str).upper()

return protocol

main()

OUTPUT:





RESULT:

The Python program for packet sniffing using raw sockets was executed successfully.

It captured live network packets and displayed the source IP, destination IP, protocol type, port numbers, and part of the data in hexadecimal form.