Roll No:241901097 Name: SANGATHAMIZHAN.S.P

EXPT:6 IMPLEMENTATION OF PACKET SNIFFING USING RAW SOCKETS IN PYTHON

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:

SERVER / PROGRAM ALGORITHM:

1. Start
2. Import required modules:
   - socket for network access
   - struct for unpacking binary packet data
   - binascii or textwrap for hex formatting (optional)
3. Get the host machine's IP address using:
   socket.gethostbyname(socket.gethostname())
4. Create a raw socket using:
   socket.socket(AF_INET, SOCK_RAW, IPPROTO_IP)
   which allows capturing raw IP packets.
5. Bind the socket to the host IP and port **0** (capture all incoming packets).
6. Enable IP header inclusion using:
   setsockopt(IP_HDRINCL, 1)
7. Enable promiscuous mode on the network interface using:
   ioctl(SIO_RCVALL, RCVALL_ON)
   so the socket captures **all** packets, not just those addressed to this host.
8. Enter an infinite loop to continuously sniff packets.
9. Receive a raw packet using:
   recvfrom(65536)
   which reads full packet data.
10. Process the raw Ethernet frame:
    - Extract destination MAC
    - Extract source MAC
    - Extract Ethernet protocol

    - Extract payload (IP packet data)

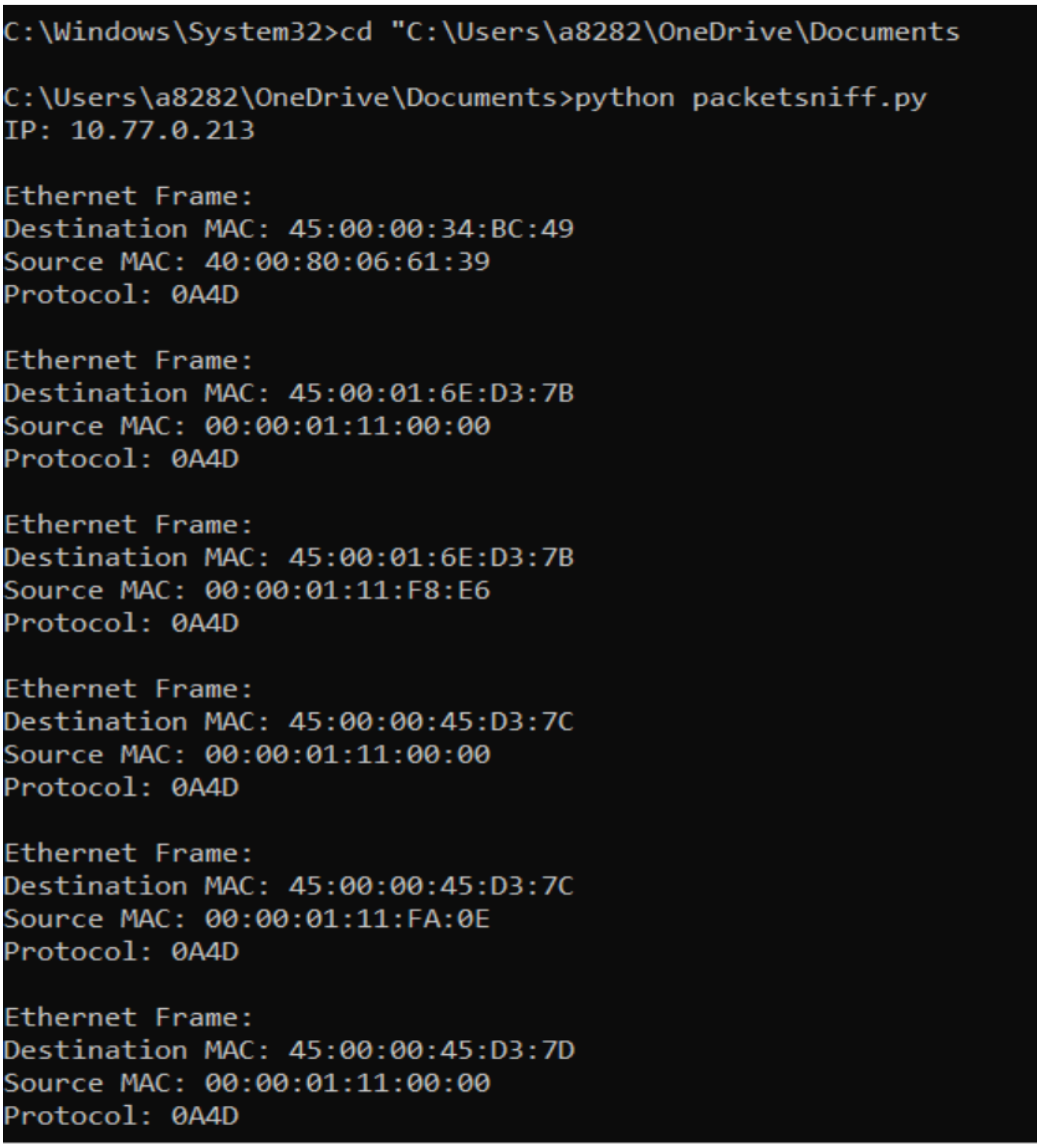
11. Print the extracted Ethernet information (MAC addresses & protocol).
12. If the Ethernet protocol indicates IPv4:
    - Parse the IP header
    - Extract source IP, destination IP, protocol
    - If protocol is TCP/UDP:
        - Extract source and destination ports
    - Print parsed details
13. Repeat the loop until the user stops execution (Ctrl + C).
14. Stop the program.

CODE:

```
import socket
import struct
import binascii
import textwrap


def main():
    # Get host
    host = socket.gethostbyname(socket.gethostname())
    print('IP: {}'.format(host))
    # Create a raw socket and bind it
    conn = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_IP)
    conn.bind((host, 0))
    # Include IP headers
    conn.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
    # Enable promiscuous mode
    conn.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)
    while True:
        # Recive data
        raw_data, addr = conn.recvfrom(65536)
        # Unpack data
        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))
# Unpack ethernet frame
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:]
# Return formatted MAC address AA:BB:CC:DD:EE:FF
def get_mac_addr(bytes_addr):
    bytes_str = map('{:02x}'.format, bytes_addr)
    mac_address = ':'.join(bytes_str).upper()
    return mac_address
# Return formatted protocol ABCD
def get_protocol(bytes_proto):
    bytes_str = map('{:02x}'.format, bytes_proto)
    protocol = ''.join(bytes_str).upper()
    return protocol
main()
```

OUTPUT:



```
C:\Windows\System32>cd "C:\Users\a8282\OneDrive\Documents

C:\Users\a8282\OneDrive\Documents>python packetsniff.py
IP: 10.77.0.213

Ethernet Frame:
Destination MAC: 45:00:00:34:BC:49
Source MAC: 40:00:80:06:61:39
Protocol: 0A4D

Ethernet Frame:
Destination MAC: 45:00:01:6E:D3:7B
Source MAC: 00:00:01:11:00:00
Protocol: 0A4D

Ethernet Frame:
Destination MAC: 45:00:01:6E:D3:7B
Source MAC: 00:00:01:11:F8:E6
Protocol: 0A4D

Ethernet Frame:
Destination MAC: 45:00:00:45:D3:7C
Source MAC: 00:00:01:11:00:00
Protocol: 0A4D

Ethernet Frame:
Destination MAC: 45:00:00:45:D3:7C
Source MAC: 00:00:01:11:FA:0E
Protocol: 0A4D

Ethernet Frame:
Destination MAC: 45:00:00:45:D3:7D
Source MAC: 00:00:01:11:00:00
Protocol: 0A4D
```

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.