EXP NO: 4	Implement packet sniffing using raw sockets in Python

Aim

Capture network packets on an interface and print basic protocol headers (Ethernet, IP, TCP/UDP) for analysis. Useful for learning how packets look and for debugging authorized networks.

Procedure / Algorithm

1. Open a raw socket (requires admin/root).
2. Receive raw frames in a loop.
3. Parse Ethernet header → check Ethertype (e.g., 0x0800 = IPv4).
4. Parse IP header → get protocol, source/destination IPs.
5. If TCP or UDP, parse transport header and print ports + flags.
6. Log or save to file (pcap writing is optional).

Code (Python, raw sockets)

```
# Requires Python 3, run as root/administrator.
import socket
import struct
import textwrap

def mac_addr(bytes_addr):
    return ':'.join(f'{b:02x}' for b in bytes_addr)

def ipv4(addr_bytes):
    return '.'.join(map(str, addr_bytes))

def main():
    # Create raw socket to capture all protocols (Linux)
    conn = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(3))

    while True:
        raw_data, addr = conn.recvfrom(65535)
        eth_header = raw_data[:14]
        dest, src, proto = struct.unpack('!6s6sH', eth_header)
        print(f'\nEthernet: {mac_addr(src)} -> {mac_addr(dest)}, proto=0x{proto:04x}')
        if proto == 0x0800:  # IPv4
```

```
            ip_header = raw_data[14:34]
            iph = struct.unpack('!BBHHHBBH4s4s', ip_header)
            version_ihl = iph[0]
            version = version_ihl >> 4
            ihl = (version_ihl & 0xF) * 4
            proto_num = iph[6]
            src_ip = ipv4(iph[8])
            dst_ip = ipv4(iph[9])
            print(f'IPv{version} {src_ip} -> {dst_ip} proto={proto_num}')
            if proto_num == 6:  # TCP
                tcp_start = 14 + ihl
                tcp_header = raw_data[tcp_start:tcp_start+20]
                src_port, dst_port, seq, ack, offset_reserved_flags =
struct.unpack('!HHLLH', tcp_header[:14])
                flags = offset_reserved_flags & 0x01FF
                print(f'TCP {src_port} -> {dst_port} flags=0x{flags:03x}')
            elif proto_num == 17:  # UDP
                udp_start = 14 + ihl
                udp_header = raw_data[udp_start:udp_start+8]
                src_port, dst_port, length = struct.unpack('!HHH', udp_header[:6])
                print(f'UDP {src_port} -> {dst_port} len={length}')
        else:
            print('Non-IPv4 frame')
if __name__ == '__main__':
    main()
```

Output:

```
Ethernet: 00:00:00:00:00:00 -> 00:00:00:00:00:00, proto=0x0800
IPv4 172.28.0.12 -> 172.28.0.12 proto=6
TCP 9000 -> 44710 flags=0x018

Ethernet: 00:00:00:00:00:00 -> 00:00:00:00:00:00, proto=0x0800
IPv4 172.28.0.12 -> 172.28.0.12 proto=6
TCP 9000 -> 44710 flags=0x018

Ethernet: 00:00:00:00:00:00 -> 00:00:00:00:00:00, proto=0x0800
IPv4 172.28.0.12 -> 172.28.0.12 proto=6
TCP 44710 -> 9000 flags=0x010

Ethernet: 00:00:00:00:00:00 -> 00:00:00:00:00:00, proto=0x0800
IPv4 172.28.0.12 -> 172.28.0.12 proto=6
TCP 44710 -> 9000 flags=0x010
```

Result:

The network packets captured on an interface and displayed basic protocol headers