

EXPNO:10.B Customized ping command to test the server connectivity
DATE:1.10.2024 without using socket

AIM:

To study packet sniffing concept and implement it without using raw sockets.

Algorithm:

1 . Define a Packet Callback Function:

- Define a function packet_callback(packet) that processes each captured packet.
- Check if the packet contains an IP layer (IP in packet).

2 . Extract Packet Details:

- If the packet contains the IP layer, retrieve the protocol number, source IP, and destination IP from the IP layer (packet[IP]).
- Initialize protocol_name as an empty string.

3 . Determine Protocol Type:

- Use conditional statements to map protocol numbers to protocol names:
 - 1 for ICMP
 - 6 for TCP
 - 17 for UDP
 - Any other protocol number as "Unknown Protocol".

4 . Display Packet Details:

- Print the protocol name, source IP, and destination IP for each captured packet.
- Print a separator line to distinguish between different packets.

5 . Main Function:

- Use a try block to handle exceptions.
- Set the interface name (e.g., "Ethernet" or "Wi-Fi") based on the system's network configuration.
- Call the sniff function to capture packets on the specified network interface with:
 - iface=interface_name for the interface name.
 - prn=packet_callback to call the callback function for each packet.
 - filter="ip" to capture only IP packets.
 - store=0 to avoid storing packets in memory.

6 . Error Handling:

- In the except block, print an error message if an exception occurs, and advise running with elevated privileges or checking the interface name.

7 . Execute the Program:

- In the main function, call main() to start the packet-sniffing process.

CODE:

```

from scapy.all import sniff
from scapy.layers.inet import IP, TCP, UDP, ICMP

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

        # Determine the protocol
        protocol_name = ""
        if protocol == 1:
            protocol_name = "ICMP"
        elif protocol == 6:
            protocol_name = "TCP"
        elif protocol == 17:
            protocol_name = "UDP"
        else:
            protocol_name = "Unknown Protocol"

        # Print packet details
        print(f"Protocol: {protocol_name}")
        print(f"Source IP: {src_ip}")
        print(f"Destination IP: {dst_ip}")
        print("-" * 50)

def main():
    try:
        # Replace 'Ethernet' with your actual network interface name from ipconfig output
        interface_name = "Ethernet" # or "Wi-Fi" if using wireless

        # Capture packets on the specified network interface
        sniff(iface=interface_name, prn=packet_callback, filter="ip", store=0)

    except Exception as e:

```

```
    print(f"Error: {e}")
    print("Make sure you are running the script with elevated privileges (e.g., sudo) and check
the interface name.")

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

OUTPUT:

```
"C:\Users\Kaviya V\PycharmProjects\pythonProject2\.venv\Scripts
Connected to pydev debugger (build 242.23339.19)
Protocol: UDP
Source IP: 172.16.53.110
Destination IP: 224.0.0.251
-----
Protocol: UDP
Source IP: 172.16.53.110
Destination IP: 224.0.0.251
-----
Protocol: UDP
Source IP: 172.16.53.187
Destination IP: 224.0.0.251
-----
Protocol: UDP
Source IP: 172.16.53.198
Destination IP: 224.0.0.251
-----
Protocol: UDP
Source IP: 172.16.53.110
Destination IP: 224.0.0.252
-----
Protocol: UDP
Source IP: 172.16.53.110
Destination IP: 224.0.0.252
-----
Protocol: UDP
Source IP: 172.16.53.42
Destination IP: 172.16.53.255
```

RESULT :

packet sniffing concept and implement it without using raw sockets is studied.