Ex No:12B **DATE:05/11/24**

Roll no:231901004

Packet Sniffing Using raw Socket

AIM:

To study packet sniffing concept and implement it using raw sockets.

DESCRIPTION:

INTRODUCTION TO PACKET SNIFFING

- Packet sniffing is the act of capturing packets of data flowing across a computer network. The software or device used to do this is called a packet sniffer.
- In network management, packet sniffing plays a very crucial role. Network managers and technicians use packet sniffers to diagnose underlying problems in their networks. ● Packet sniffer is essentially a tool that aids in monitoring network traffic and troubleshooting a network.
- It works by capturing and analyzing packets of data that flow through a particular network.

How Do Packet Sniffers Work?

- To understand how a packet sniffer works, first understand that data travels through a network in the form of packets.
- In packet-switched networks, the data to be transmitted is broken down into several packets. These packets are reassembled once all the data packets reach their intended destination.
- When a packet sniffer is installed in the network, the sniffer intercepts the network traffic and captures the raw data packets.
- Subsequently, the captured data packet is analyzed by the packet sniffing software and presented to the network manager/technician in a user-friendly format.
- By user-friendly, it means the Network Administrator should be able to make sense of it.

Uses And Implementations Of Packet Sniffers

- Monitoring network usage

Packet sniffers are great at monitoring the network usage at any given time, helping Network Managers identify whether a particular network is normal or congested. Also, making it possible to identify bottlenecks within the network and identify and improve the performance with infrastructure upgrades.

- Identifying problems

Packet sniffers can identify network-related issues. This is possible because a packet sniffer can analyze the conversation between two or more nodes in a network. So, in the event of a network error, the information captured by the

packet sniffer can be used to identify the erroneous packets and pinpoint the node that failed to answer the request(s). Making it easy to identify faulty devices within the network in an efficient manner and providing the ability to take swift

corrective actions.

● Detecting security loopholes

A disturbing fact about packet sniffers is their ability to work as spying tools. They also help the good guys, such as your Network Manager, by testing the vulnerabilities of a network. Once these vulnerabilities are detected, it is easier to remove the loopholes thus preventing the possibilities of hacking attempts.

PACKET SNIFFING USING PYTHON:

● Python is really the very awesome language and also very powerful language. With Python and socket module, a packet sniffing code can be written.

How To Capture Packets?

For sniffing with socket module in python we have to create a socket.socket class object with special configuration.

In simple words, we have to configure socket.socket class object to capture low-level packets from the network so that it can capture packet from low-level networks and provides us output without doing any type of changes in capture packets. **Code to capture packets (in windows)**

```
import socket s= socket.socket(socket.AF_PACKET,socket.SOCK_RAW,socket.
htons(0x0800))
#s.bind(("127.0.0.1",0))
while True:
print(s.recvfrom(65565))
```

● Raw sockets (or Raw IP sockets), typically available in routers and other network equipment. Here the transport layer is bypassed, and the packet headers are made accessible to the application, and there is no port number in the address, just the IP address.

The above code will capture an IP raw packet and will display the content as such(binary format)

OUTPUT of the CODE:

(b'E\x00\x01H\x00\x00\x00\x00\x80\x119\xa6\x00\x00\x00\x00\xff\xff\xff\xff\x00D\x00C\x014
J\xb5\x01\x01\x06\x00\xfe5*\x85\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x0f\xea\x92\xd2\xc1\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00c\x82Sc5\x01\x0
1t\x01\x01=\x07\x01\x00\x0f\xea\x92\xd2\xc12\x04\xac\x10\x08\xd2\x0c\x03REC<\x08MSFT
5.07\x0b\x01\x0f\x03\x06,./\x1f!\xf9++\x02\xdc\x00\xff\x00\x00\x00\x00\x00\x00', ('0.0.0.0',
0))

How To Parse/Extract Captured Packets?

```
import socket
import struct import binascii s=
socket.socket(socket.AF_PACKET,socket.SOCK_RAW,socket. htons(0x0800))
#s.bind(("127.0.0.1",0)) packet=s.recvfrom(65565)
print(packet) ethernet_header = packet[0][0:14]
eth_header = struct.unpack("!6s6s2s", ethernet_header)
print("ETHERNET HEADER")
print("****************") print("Destination
Address") print( binascii.hexlify(eth_header[0]))
print("Source Address") print(
binascii.hexlify(eth_header[1])) print("Type") print(
binascii.hexlify(eth_header[2])) print("IP HEADER")
print("*********") ipheader = packet[0][14:34]

ip_header = struct.unpack("!12s4s4s", ipheader)
print("Destination Address") print
(socket.inet_ntoa(ip_header[1])) print("Source
Address") print(socket.inet_ntoa(ip_header[2]))
```

OUTPUT:

(b'E\x00\x004\x00\x87@\x00\x80\x06\x00\x00\x7f\x00\x00\x01\x7f\x00\x00\x01\xd8_\x17a\x95\x08fs\x00\x00\x00\x00\x80\x02\xff\xff\x8b\xad\x00\x00\x02\x04\xff\xd7\x01\x03\x03\x08\x01\x01\x04\x02', ('127.0.0.1', 0))

ETHERNET HEADER

****

Destination Address

b'450000340087'

Source Address

b'400080060000'

Type

b'7f00'

IP HEADER

****

Destination Address

102.115.0.0

Source Address

0.0.128.2

Result:

Thus, a study on packet sniffing concept and implement it using raw sockets was done.