

EXP NO:12.B
DATE:28.10.24

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

[illegible]

```
python3 rawsocket.py
(b'\x08\x00'\xad%\x87RU\n\x00\x02\x02\x08\x00E\x00\x00Z\x01\x8a\x00\x00\x11'\xf8\n\x00\x02\x03\n\x00\x02\x0f\x005\x85\x03\x00F\x88 0\x81\x80\x00\x01\x00\x01\x00\x00\x00\x00\x00\x07contile\x08services\x07mozilla\x03com\x00\x00\x01\x00\x01\xc0\x0c\x00\x01\x00\x01\x00\x00\x00\xe1\x00\x04"u\xbc\xa6', ('eth0', 2048, 0, 1, b'RU\n\x00\x02\x02'))
(b"\x08\x00'\xad%\x87RU\n\x00\x02\x02\x08\x00E\x00\x00J\x01\x8b\x00\x00\x11a\x07\n\x00\x02\x03\n\x00\x02\x0f\x005\x85\x03\x006\xfe7JM\x81\x80\x00\x01\x00\x00\x00\x00\x00\x00\x07contile\x08services\x07mozilla\x03com\x00\x00\x1c\x00\x01", ('eth0', 2048, 0, 1, b'RU\n\x00\x02\x02'))
(b'\x08\x00'\xad%\x87RU\n\x00\x02\x02\x08\x00E\x00\x00,\x01\x8c\x00\x00\x06\x8e\x16"u\xbc\xa6\n\x00\x02\x0f\x01\xbb\xddP\x02\xe0T\x01v\xab0\xee`\x12\xff\xff\xcf\x00\x00\x02\x04\x05\xb4\x00\x00', ('eth0', 2048, 0, 1, b'RU\n\x00\x02\x02'))
(b'\x08\x00'\xad%\x87RU\n\x00\x02\x02\x08\x00E\x00\x00(\x01\x8d\x00\x00\x06\x8e\x19"u\xbc\xa6\n\x00\x02\x0f\x01\xbb\xddP\x02\xe0T\x02v\xab2\xfb3P\x10\xff\xff\xe5\x1d\x00\x00\x00\x00\x00\x00\x00\x00', ('eth0', 2048, 0, 1, b'RU\n\x00\x02\x02'))
(b'\x08\x00'\xad%\x87RU\n\x00\x02\x02\x08\x00E\x00\x0bh\x01\x8e\x00\x00\x06\x82\xd8"u\xbc\xa6\n\x00\x02\x0f\x01\xbb\xddP\x02\xe0T\x02v\xab2\xfb3P\x10\xff\xff\xf6\x84\x00\x00\x16\x03\x03\x00z\x02\x00\x00v\x03\x03uC\xaa\x04 ex\xde\x1b\xefQM\xeb\x8e\xcd\xba\xdb\xb7:\xbff7\x0f\r\xe9\x07H\b5\\M\xd4w, \x17\xb3CC\xc1[\xd0\x93\xc3\x04\\\xd6\xc0\x89\x95\x0f\r"\x0e\xff_n]\xb0\xf9\x1aN\xc59\x12Y\x13\x01\x00\x00.\x003\x00$\x00\x1d\x00 \xa3,\x95\x97\xbe\x97\x18\xa8$}\xa2` \xa1\xe7\xda\xa5\xd4\x04\x9ab\x82\xd8\xedy\xce\'4"\xff\xd9\xaeM\x00+\x00\x02\x03\x04\x14\x03\x03\x00\x01\x01\x17\x03\x03\x0bs$\xbb#\xa3\xe6\xb5"{0\x15\t\x9b\xa7\xfd2Aw3\x9e\x96F\xa5L\xba3\xcfY\xd5\x0e7<\xbbt\x9c\x80u\x1c\xcf\xcf}\xf4!\x95hN\xefa\xaf\xdd\xdes;7\xd8\x19\xd6\xc4\x85\xf9^a\x80\x8b{~\x06_\x8ee|<\x89\xee1Y\x08\xc6\xe3\xdf\xfe\xd5\xb4\xd58\xf0}\x90\xe6\xeb]\xbe\x1d\xf2\x96\xdf\xder\x87\xf3"\xfb\x88OyY\xc2q\x8b39\x99\xe2r2\x1c\x0e}q\xa5m\x10\xf
```

```
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\x9
5\x08fs\x00\x00\x00\x00\x80\x02\xff\xff\x8b\xad\x00\x00\x02\x04\xff\xd7\x01\x03\x03\x08\x0
1\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

```

(root@kali)-[/home/kali]
# python3 rawsocket.py
(b"\x08\x00'\xad%\x87RU\n\x00\x02\x02\x08\x00E\x00\x00\x99\x00\x03\x00\x00\x00\x11b@\n\x00\x02\x03\n\x00\x02\x0f\x005\xe7H\x00\x85\x11\xdd\x98\xfd\x81\x80\x00\x01\x00\x02\x00\x00\x00\x00\x08location\x08services\x07mozilla\x03com\x00\x00\x01\x00\x01\xc0\x0c\x00\x05\x00\x01\x00\x00\x00\x0b\x002\x04prod\x0fclassify-client\x04prod\x0bweb services\x06mozgcp\x03net\x00\xc0;\x00\x01\x00\x01\x00\x00\x00|\x00\x04#\xbeH\xd8", ('eth0', 2048, 0, 1, b'RU\n\x00\x02\x02'))
ETHERNET HEADER
*****
Destination Address
b'080027ad2587'
Source Address
b'52550a000202'
Type
b'0800'
IP HEADER
*****
Destination Address
10.0.2.3
Source Address
10.0.2.15

```

Result:

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