**Prologue**

* Packet sniffers are programs that intercept the network traffic flowing in and out of a system through network interfaces. So if you are browsing the internet then traffic is flowing and a packet sniffer would be able to catch it in the form of packets and display them for whatever reasons required.
* Packet sniffers are used for various needs like analysing protocols, monitoring network, and assessing the security of a network. Wireshark for example is the most popular packet sniffer out there and is available for all platforms. It’s GUI based and very easy to use.
* In this program "exp.c" we are going to talk about how to code and make our own packet sniffer in C and on the linux platform. By Linux it means that the code sample shown here would work only on linux and not windows.
* Packet sniffers can be coded by either using sockets api provided by the kernel, or by using some packet capture library like libpcap. In this program “exp.c” we shall be covering the first method that is by using sockets.

**How to code**

* To code a very simply sniffer in C the steps would be

1. Create a raw socket.
2. Put it in a recvfrom loop and receive data on it.

* A raw socket when put in recvfrom loop receives all incoming packets. This is because it is not bound to a particular address or port.

sock\_raw = socket(AF\_INET , SOCK\_RAW , IPPROTO\_TCP);

while(1)

{

data\_size = recvfrom(sock\_raw , buffer , 65536 , 0 , &saddr , &saddr\_size);

}

* That’s all. The buffer will hold the data sniffed or picked up. The sniffing part is actually complete over here. The next task is to actually read the captured packet, analyse it and present it to the user in a readable format. Our code in "exp.c" will do so. Note that it sniffs only incoming packets.

**Compile and Run**

$ gcc sniffer.c && sudo ./a.out

* The program must be run as root user or superuser privileges. e.g. sudo ./a.out in Ubuntu
* The program creates raw sockets which require root access.

**Note**

1. The above sniffer picks up only TCP packets, because of the declaration :

sock\_raw = socket(AF\_INET , SOCK\_RAW , IPPROTO\_TCP);

For UDP and ICMP the declaration has to be :

sock\_raw = socket(AF\_INET , SOCK\_RAW , IPPROTO\_UDP);

sock\_raw = socket(AF\_INET , SOCK\_RAW , IPPROTO\_ICMP);

You might be tempted to think of doing :

sock\_raw = socket(AF\_INET , SOCK\_RAW , IPPROTO\_IP);

but this will not work , since IPPROTO\_IP is a dummy protocol not a real one.

1. This sniffer picks up only incoming packets.
2. It provides the application with IP frames, which means that ethernet headers are not available.
3. It is not very accurate since it misses out some packets even in the incoming ones.