Assignment No.10

Problem Statement:

Write a program to implement a packet sniffing tool in C.

Theory

Packet Sniffer

A packet sniffer is a program that can see all of the information passing over the network it is connected to. As data streams back and forth on the network, the program looks at, or "sniffs," each packet. A packet is a part of a message that has been broken up.

Normally, a computer only looks at packets addressed to it and ignores the rest of the traffic on the network. But when a packet sniffer is set up on a computer, the sniffer's network interface is set to promiscuous mode. This means that it is looking at everything that comes through. The amount of traffic largely depends on the location of the computer in the network. A client system out on an isolated branch of the network sees only a small segment of the network traffic, while the main server sees almost all of it.

A packet sniffer can usually be set up in one of two ways:

- Unfiltered - captures all of the packets
- Filtered - captures only those packets containing specific data elements

Packets that contain targeted data are copied onto the disks they pass through. These copies can then be analyzed carefully for specific information or patterns.

When you connect to the Internet, you are joining a network maintained by your Internet service provider (ISP). The ISP's network communicates with networks maintained by other ISPs to form the foundation of the Internet. A packet sniffer located at one of the servers of your ISP would potentially be able to monitor all of your online activities, such as:

- Which Web sites you visit
- What you look at on the site
- Whom you send e-mail to
- What's in the e-mail you send
- What you download from a site
- What streaming events you use, such as audio, video and Internet telephony

From this information, employers can determine how much time a worker is spending online and if that worker is viewing inappropriate material.

Code

We will be implementing a simple packet sniffer. The code will be performing the following steps:

- Sniff the packets from the network.
- Store the sniffed packets and save them in a file as logs.

########################## **C Code for Packet Sniffer** ##################################
#sniffer.c
###################### **Code Begins** ##
```
#include<stdio.h> //For standard things
#include<stdlib.h> //malloc
#include<string.h> //memset
#include<netinet/ip_icmp.h> //Provides declarations for icmp header
#include<netinet/udp.h> //Provides declarations for udp header
#include<netinet/tcp.h> //Provides declarations for tcp header
#include<netinet/ip.h> //Provides declarations for ip header
#include<sys/socket.h>
#include<arpa/inet.h>
void ProcessPacket(unsigned char* , int);
void print_ip_header(unsigned char* , int);
void print_tcp_packet(unsigned char* , int);
void PrintData (unsigned char* , int);
int sock_raw;
FILE *logfile;
int tcp=0,udp=0,icmp=0,others=0,igmp=0,total=0,i,j;
struct sockaddr_in source,dest;
//char IP_src[100];
//int flg_syn;
int main()
{
  int saddr_size , data_size;
  struct sockaddr saddr;
  struct in_addr in;
unsigned char *buffer = (unsigned char *)malloc(65536); //Its Big!
logfile=fopen("log.txt","w");
if(logfile==NULL) printf("Unable to create file.");
printf("Starting...\n");
//Create a raw socket that shall sniff
sock_raw = socket(AF_INET , SOCK_RAW , IPPROTO_TCP);
if(sock_raw < 0)
{
   printf("Socket Error\n");
   return 1;
}
while(1)
{
   saddr_size = sizeof saddr;
   //Receive a packet
```

```
   data_size = recvfrom(sock_raw , buffer , 65536 , 0 , &saddr , &saddr_size);
   if(data_size <0 )
   {
      printf("Recvfrom error , failed to get packets\n");
      return 1;
   }
   //Now process the packet
   ProcessPacket(buffer , data_size);
     /*if(flg_syn != 0)
     {
              printf("%s sent Syn Request Packet\n",IP_src);
     }*/
}
close(sock_raw);
printf("Finished");
return 0;
}
void ProcessPacket(unsigned char* buffer, int size)
{
  //Get the IP Header part of this packet
  struct iphdr *iph = (struct iphdr*)buffer;
  ++total;
  switch (iph->protocol) //Check the Protocol and do accordingly...
  {
     case 1: //ICMP Protocol
       ++icmp;
break;
case 2: //IGMP Protocol
  ++igmp;
  break;
case 6: //TCP Protocol
  ++tcp;
  print_tcp_packet(buffer , size);
  break;
case 17: //UDP Protocol
  ++udp;
Break
default: //Some Other Protocol like ARP etc.
  ++others;
  break;
}
  printf("TCP : %d UDP : %d ICMP : %d IGMP : %d Others : %d Total :
%d\r",tcp,udp,icmp,igmp,others,total);
}
void print_ip_header(unsigned char* Buffer, int Size)
{
  unsigned short iphdrlen;
struct iphdr *iph = (struct iphdr *)Buffer;
iphdrlen =iph->ihl*4;
memset(&source, 0, sizeof(source));
source.sin_addr.s_addr = iph->saddr;
```

```
memset(&dest, 0, sizeof(dest));
dest.sin_addr.s_addr = iph->daddr;
  fprintf(logfile,"\n");
  fprintf(logfile,"IP Header\n");
  fprintf(logfile," |-IP Version: %d\n",(unsigned int)iph->version);
  fprintf(logfile," |-IP Header Length : %d DWORDS or %d Bytes\n",(unsigned int)iph->ihl,((unsigned
int)(iph->ihl))*4);
  fprintf(logfile," |-Type Of Service : %d\n",(unsigned int)iph->tos);
  fprintf(logfile," |-IP Total Length : %d Bytes(Size of Packet)\n",ntohs(iph->tot_len));
  fprintf(logfile," |-Identification : %d\n",ntohs(iph->id));
  //fprintf(logfile," |-Reserved ZERO Field : %d\n",(unsigned int)iphdr->ip_reserved_zero);
  //fprintf(logfile," |-Dont Fragment Field : %d\n",(unsigned int)iphdr->ip_dont_fragment);
  //fprintf(logfile," |-More Fragment Field : %d\n",(unsigned int)iphdr->ip_more_fragment);
  fprintf(logfile," |-TTL : %d\n",(unsigned int)iph->ttl);
  fprintf(logfile," |-Protocol : %d\n",(unsigned int)iph->protocol);
  fprintf(logfile," |-Checksum : %d\n",ntohs(iph->check));
  fprintf(logfile," |-Source IP: %s\n",inet_ntoa(source.sin_addr));
fprintf(logfile," |-Destination IP : %s\n",inet_ntoa(dest.sin_addr));
//strcpy(IP_src,inet_ntoa(source.sin_addr));
}
void print_tcp_packet(unsigned char* Buffer, int Size)
{
  unsigned short iphdrlen;
// flg_syn=0;
  struct iphdr *iph = (struct iphdr *)Buffer;
  iphdrlen = iph->ihl*4;
struct tcphdr *tcph=(struct tcphdr*)(Buffer + iphdrlen);
fprintf(logfile,"\n\n**********************TCP Packet*************************\n");
print_ip_header(Buffer,Size);
  fprintf(logfile,"\n");
  fprintf(logfile,"TCP Header\n");
  fprintf(logfile," |-Source Port : %u\n",ntohs(tcph->source));
  fprintf(logfile," |-Destination Port : %u\n",ntohs(tcph->dest));
  fprintf(logfile," |-Sequence Number : %u\n",ntohl(tcph->seq));
  fprintf(logfile," |-Acknowledge Number : %u\n",ntohl(tcph->ack_seq));
  fprintf(logfile," |-Header Length : %d DWORDS or %d BYTES\n" ,(unsigned int)tcph->doff,(unsigned
int)tcph->doff*4);
  //fprintf(logfile," |-CWR Flag : %d\n",(unsigned int)tcph->cwr);
  //fprintf(logfile," |-ECN Flag : %d\n",(unsigned int)tcph->ece);
  fprintf(logfile," |-Urgent Flag: %d\n",(unsigned int)tcph->urg);
  fprintf(logfile," |-Acknowledgement Flag : %d\n",(unsigned int)tcph->ack);
  fprintf(logfile," |-Push Flag: %d\n",(unsigned int)tcph->psh);
  fprintf(logfile," |-Reset Flag: %d\n",(unsigned int)tcph->rst);
  fprintf(logfile," |-Synchronise Flag : %d\n",(unsigned int)tcph->syn);
  fprintf(logfile," |-Finish Flag: %d\n",(unsigned int)tcph->fin);
  fprintf(logfile," |-Window: %d\n",ntohs(tcph->window));
  fprintf(logfile," |-Checksum: %d\n",ntohs(tcph->check));
  fprintf(logfile," |-Urgent Pointer : %d\n",tcph->urg_ptr);
  fprintf(logfile,"\n");
  fprintf(logfile,"DATA Dump");
  fprintf(logfile,"\n");
```

```
// flg_syn = (unsigned int)tcph->syn;
fprintf(logfile,"IP Header\n");
PrintData(Buffer,iphdrlen);
fprintf(logfile,"TCP Header\n");
PrintData(Buffer+iphdrlen,tcph->doff*4);
fprintf(logfile,"Data Payload\n");
PrintData(Buffer + iphdrlen + tcph->doff*4 , (Size - tcph->doff*4-iph->ihl*4) );
fprintf(logfile,"\n###########################################################");
}
void PrintData (unsigned char* data , int Size)
{
for(i=0 ; i < Size ; i++)
{
  if( i!=0 && i%16==0) //if one line of hex printing is complete...
  {
     fprintf(logfile,"");
     for(j=i-16 ; j<i ; j++)
     {
        if(data[j]>=32 && data[j]<=128)
           fprintf(logfile,"%c",(unsigned char)data[j]); //if its a number or alphabet
else fprintf(logfile,"."); //otherwise print a dot
}
fprintf(logfile,"\n");
}
if(i%16==0) fprintf(logfile," ");
   fprintf(logfile," %02X",(unsigned int)data[i]);
if( i==Size-1) //print the last spaces
{
   for(j=0;j<15-i%16;j++)
 fprintf(logfile," "); //extra spaces
fprintf(logfile," ");
for(j=i-i%16 ; j<=i ; j++)
{
  if(data[j]>=32 && data[j]<=128) fprintf(logfile,"%c",(unsigned char)data[j]);
  else fprintf(logfile,".");
}
fprintf(logfile,"\n");
}
}
}
```

###################### **Code Ends** ##

Compiling and executing the code:

$ gcc –o sniffer sniffer.c

$ ./sniffer

Conclusion

Hence, we have studied & implemented packet sniffing program.