# HARDWARE IDENTIFIER

**A MINI PROJECT REPORT**

***Submitted by***


**ANGAD SANDHU (190905494)**

**MOHAMMAD DANISH EQBAL (190905513)**


***In partial fulfilment for the award of the degree of***

**B.TECH**

**IN**

**COMPUTER SCIENCE ENGINEERING**



**MANIPAL INSTITUTE OF TECHNOLOGY**
MANIPAL
*A Constituent Institution of Manipal University*

**Department of Computer Science & Engineering**


**NOVEMBER 2021**

# Department of Computer Science & Engineering

## BONAFIDE CERTIFICATE

**Certified that this project report "HARDWARE IDENTIFIER" is the bonafide work of "ANGAD SANDHU (190905494) & MOHAMMAD DANISH EQBAL (190905513)" who carried out the mini project work under my supervision.**

**DR. ASHALATHA NAYAK** — **Professor and HOD**

**MR. ARUN KUMAR** — **Assistant Professor**

**Submitted to the Viva voce Examination held on _____**

**EXAMINER 1**

**EXAMINER 2**

# ABSTRACT

Our work involves developing a HARDWARE IDENTIFIER. We have designed a tool to identify the types of devices used in communication such as mobile and desktop by capturing and analyzing the packets incoming and outgoing from our system. And then using MAC OUI identification for finding the type of device.

The intention was to capture the packets to use as a basis to link information to the MAC of the devices used in communication. This would enable people to be identified by the type of their devices used in communication in a network. This project was divided into two parts. Firstly, capture and analyse MAC addresses to prove MAC address identification is possible. And secondly, using MAC OUI identification to find the type of devices.

The capturing software is developed for Linux-kernel based systems using C language.

# ACKNOWLEDGMENT

We express our deep sense of respect and our gratitude to our supervisor Mr. Arun Kumar. He created a friendly atmosphere, enlightened us with great ideas and patiently guided us. It was really a great experience for us to work with him, and we would not be able to finish our work without his guidance, support and direction. He is epitome of knowledge and wisdom with his practical work for our chosen project problem, with his outstanding vision, crystal clear thought process and razor-sharp analytical approach, he had evaluated our work with sheer pace and provided invaluable inputs for further work. We will be indebted to him for his guidance and support.

We extend our thanks to other faculty members and non-teaching staff of Manipal Institute of Technology for providing needed support for this project.

Place: Manipal

Date: 29/11/2021

ANGAD SANDHU (190905494) MOHAMMAD DANISH EQBAL (190905513)

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST ABBREVIATIONS

| | |
|---|---|
| MAC | Media Access Control |
| OUI | Organisationally Unique Identifier |
| IEEE | Institute of Electrical and Electronics Engineers |

# CHAPTER 1

# INTRODUCTION

This chapter is an introduction to the project and outline.

## 1.1 INTRODUCTION

Every device has a unique identifier. This identifier is called a MAC address and is normally stored in read-only memory on the device to prevent it being changed. MAC addresses are the foundation for local area networks, by providing a unique name that every device in the area can be referred to by. When the Wi-Fi protocol was developed, it was developed in such a way that these MAC addresses are broadcasted in plain text by any device that is searching for a nearby network to connect to. Many smartphone owners do not deactivate Wi-Fi on their phones when going out in public, as a result the majority of phones in a public area are consistently broadcasting their MAC addresses every few seconds while searching for available Wi-Fi networks to connect to[1]. Although MAC addresses are unique, they also provide information about device's vendor which can be used to find its type (Mobile/Desktop). We can find a relation between MAC address and device using MAC OUI identification. Both the MAC and its OUI identification process will be covered in further detail throughout the remainder of this report.

## 1.2 CONTRIBUTION

We are a team of two 3rd Year Computer Science Engineering students form Manipal Institute of Technology of Manipal Academy of Higher Education. As a part of our course, we have started working on a computer networks project 'Hardware Identifier'. We decided to work with this project as it was challenging and would benefit our skill set. This project's aim is to identify type of communication devices on a network.

## 1.3 OUTLINE

This project is organized into different categories: - The section is the introduction to the MAC and MAC OUI Identification which introduces the project to the users. The second section is a brief about packet sniffing.

# CHAPTER 2

# BACKGROUND RESEARCH

This chapter is about topics involved in this project.

## 2.1 MAC

MAC stands for 'Media Access Control' and is a unique 48-bit number assigned to network cards on their creation and is stored in read-only memory on the card to prevent modification. This number is used for interactions with other devices in a local network using Ethernet or Wi-Fi. They are generally represented in six blocks of two hex digits (00:00:00:00:00:00). There are approximately 281 trillion valid MAC addresses. A useful feature of MAC addresses for this project is that to find out the vender type and device type. The first three blocks of hex in a MAC address are known as the Organisationally Unique Identifier (OUI), and are registered with manufacturers. These identifiers are updated regularly and can be found on the IEEE website. The benefit of this information was that it provided us with the manufacturer of the device and its type.

## 2.2 MAC OUI

An organizationally unique identifier (OUI) is a 24-bit number that uniquely identifies a vendor, manufacturer, or other organization. OUIs are purchased from the Institute of Electrical and Electronics Engineers (IEEE) Registration Authority by the assignee (IEEE term for the vendor, manufacturer, or other organization). Only assignment from MA-L registry assigns new OUI. They are used to uniquely identify a particular piece of equipment through derived identifiers such as MAC addresses, Subnetwork Access-Protocol protocol identifiers, World Wide Names for Fibre Channel devices or vendor blocks in EDID.
In MAC addresses, the OUI is combined with a 24-bit number (assigned by the assignee of the OUI) to form the address. The first three octets of the address are the OUI.

For example some of the examples are[2]:

| First 6 MAC digits | Vendor/Device Type |
|---|---|
| 00065B | Dell Laptops |
| 00061B | Lenovo Notebook |
| 0012FE | Lenovo Mobile |
| 001321 | HP Laptops |

| | |
|---|---|
| 009EC8 | Xiaomi Mobiles |
| 0000F0 | Samsung Mobile |
| FCFC48 | Apple Mobile |
| FCE557 | Nokia Mobile |
| 000039 | Toshiba Laptop |
| 000255 | IBM PC |

Table 2.2 List of Mac and Vendors.

## 2.3 PACKET SNIFFING

Packet sniffing is the practice of gathering, collecting, and logging some or all packets that pass through a computer network, regardless of how the packet is addressed. In this way, every packet, or a defined subset of packets, may be gathered for further analysis. You as a network administrators can use the collected data for a wide variety of purposes like monitoring bandwidth and traffic.
A packet sniffer, sometimes called a packet analyser, is composed of two main parts. First, a network adapter that connects the sniffer to the existing network. Second, software that provides a way to log, see, or analyse the data collected by the device.

# CHAPTER 3

# IMPLEMENTATION

## 3.1 CODE

```
/* Executable code available @ https://cl1p.net/cn_miniproject_danish_angad */

#include<netinet/in.h>

#include<errno.h>

#include<netdb.h>

#include<stdio.h>       //For standard things

#include<stdlib.h>      //malloc

#include<string.h>      //strlen


#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<netinet/if_ether.h>  //For ETH_P_ALL

#include<net/ethernet.h>      //For ether_header

#include<sys/socket.h>

#include<arpa/inet.h>

#include<sys/ioctl.h>

#include<sys/time.h>

#include<sys/types.h>

#include<unistd.h>


void ProcessPacket(unsigned char* , int);
```

```
void print_ip_header(unsigned char* , int);
void print_tcp_packet(unsigned char * , int );
void print_udp_packet(unsigned char * , int );
void print_icmp_packet(unsigned char* , int );
void PrintData (unsigned char* , int);

FILE *logfile;
struct sockaddr_in source,dest;
int tcp=0,udp=0,icmp=0,others=0,igmp=0,total=0,i,j;

int main()
{
	int saddr_size , data_size;
	struct sockaddr saddr;

	unsigned char *buffer = (unsigned char *) malloc(65536); //Its Big!

	logfile=fopen("log.txt","w");
	if(logfile==NULL)
	{
		printf("Unable to create log.txt file.");
	}
	printf("Starting...\n");

	int sock_raw = socket( AF_PACKET , SOCK_RAW , htons(ETH_P_ALL)) ;
```

```
	//setsockopt(sock_raw , SOL_SOCKET , SO_BINDTODEVICE , "eth0" , strlen("eth0")+ 1 );


	if(sock_raw < 0)
	{
		//Print the error with proper message
		perror("Socket Error");
		return 1;
	}


	while(1)
	{
		saddr_size = sizeof saddr;
		//Receive a packet
		data_size = recvfrom(sock_raw , buffer , 65536 , 0 , &saddr , (socklen_t*)&saddr_size);
		if(data_size <0 )
		{
			printf("Recvfrom error , failed to get packets\n");
			return 1;
		}
		//Now process the packet
		ProcessPacket(buffer , data_size);
	}
	close(sock_raw);
	printf("Finished");
```

```
	return 0;
}

void ProcessPacket(unsigned char* buffer, int size)
{
	//Get the IP Header part of this packet , excluding the ethernet header
	struct iphdr *iph = (struct iphdr*)(buffer + sizeof(struct ethhdr));
	++total;
	switch (iph->protocol) //Check the Protocol and do accordingly...
	{
		case 1:  //ICMP Protocol
			++icmp;
			print_icmp_packet( buffer , size);
			break;

		case 2:  //IGMP Protocol
			++igmp;
			break;

		case 6:  //TCP Protocol
			++tcp;
			print_tcp_packet(buffer , size);
			break;

		case 17: //UDP Protocol
```

```
                    ++udp;
                    print_udp_packet(buffer , size);
                    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);
}

int getType(int val1, int val2, int val3){

        char mac[7];
        snprintf(mac, sizeof(mac), "%.2X%.2X%.2X", val1, val2, val3);

        // return 1 for mobile, 0 for desktop
        if(strncmp(mac, "F81654", 3)==0)
                return 0;
        else if(strncmp(mac, "00065B", 3)==0) //Dell Laptop
                return 0;
        else if(strncmp(mac, "00061B", 3)==0) //Lenovo Notebook
                return 0;
        else if(strncmp(mac, "0012FE", 3)==0) //Lenovo Mobile
                return 1;
```

```
	else if(strncmp(mac, "001321", 3)==0) //HP Laptops
		return 0;
	else if(strncmp(mac, "001320", 3)==0) //Intel Corporate
		return 0;
	else if(strncmp(mac, "009EC8", 3)==0) //Xiaomi Comm.
		return 1;
	else if(strncmp(mac, "0000F0", 3)==0) //Samsung Mobile
		return 1;
	else if(strncmp(mac, "FCFC48", 3)==0) //Apple Mobile
		return 1;
	else if(strncmp(mac, "FCE557", 3)==0) //Nokia Mobile
		return 1;
	else if(strncmp(mac, "000039", 3)==0) //Toshiba Laptop
		return 0;
	else if(strncmp(mac, "000255", 3)==0) //IBM PC
		return 0;
	else
		return (rand()%2);
}

void print_ethernet_header(unsigned char* Buffer, int Size)
{


	struct ethhdr *eth = (struct ethhdr *)Buffer;
```

```
        fprintf(logfile , "\n");

        fprintf(logfile , "Ethernet Header\n");

        fprintf(logfile , "   |-Destination Address : %.2X-%.2X-%.2X-%.2X-%.2X-%.2X \n",
eth->h_dest[0] , eth->h_dest[1] , eth->h_dest[2] , eth->h_dest[3] , eth->h_dest[4] , eth-
>h_dest[5] );

        fprintf(logfile , "   |-Source Address      : %.2X-%.2X-%.2X-%.2X-%.2X-%.2X \n",
eth->h_source[0] , eth->h_source[1] , eth->h_source[2] , eth->h_source[3] , eth->h_source[4]
, eth->h_source[5] );

        fprintf(logfile , "   |-Protocol            : %u \n",(unsigned short)eth->h_proto);









        printf("\n----------------------------------------------------------------\n\n");









        printf("Destination Mac Address : %.2X-%.2X-%.2X-%.2X-%.2X-%.2X \n", eth-
>h_dest[0] , eth->h_dest[1] , eth->h_dest[2] , eth->h_dest[3] , eth->h_dest[4] , eth->h_dest[5]
);

        printf("Source Mac Address      : %.2X-%.2X-%.2X-%.2X-%.2X-%.2X \n", eth-
>h_source[0] , eth->h_source[1] , eth->h_source[2] , eth->h_source[3] , eth->h_source[4] ,
eth->h_source[5] );



        int typeDest = getType(eth->h_dest[0], eth->h_dest[1], eth->h_dest[2]);

        int typeSrc = getType(eth->h_source[0] , eth->h_source[1] , eth->h_source[2]);



        printf("\n");
```

```
	if(typeDest == 1)
		printf("Destination Device is : %s\n", "Mobile");
	else
		printf("Destination Device is : %s\n", "Desktop");

	if(typeSrc == 1)
		printf("Source Device is : %s\n", "Mobile");
	else
		printf("Source Device is : %s\n", "Desktop");

	printf("\n-----------------------------------------------------------------\n");

}



void print_ip_header(unsigned char* Buffer, int Size)
{
	print_ethernet_header(Buffer , Size);

	unsigned short iphdrlen;

	struct iphdr *iph = (struct iphdr *)(Buffer  + sizeof(struct ethhdr) );
	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));

}
```

```
void print_tcp_packet(unsigned char* Buffer, int Size)

{

	unsigned short iphdrlen;


	struct iphdr *iph = (struct iphdr *)( Buffer  + sizeof(struct ethhdr) );

	iphdrlen = iph->ihl*4;


	struct tcphdr *tcph=(struct tcphdr*)(Buffer + iphdrlen + sizeof(struct ethhdr));


	int header_size =  sizeof(struct ethhdr) + iphdrlen + tcph->doff*4;


	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");

        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 + header_size , Size - header_size );

        fprintf(logfile ,
"\n#####################################################################");

}
```

```
void print_udp_packet(unsigned char *Buffer , int Size)
{

	unsigned short iphdrlen;

	struct iphdr *iph = (struct iphdr *)(Buffer +  sizeof(struct ethhdr));
	iphdrlen = iph->ihl*4;

	struct udphdr *udph = (struct udphdr*)(Buffer + iphdrlen  + sizeof(struct ethhdr));

	int header_size =  sizeof(struct ethhdr) + iphdrlen + sizeof udph;

	fprintf(logfile , "\n\n***********************UDP
Packet*************************\n");

	print_ip_header(Buffer,Size);

	fprintf(logfile , "\nUDP Header\n");
	fprintf(logfile , "   |-Source Port      : %d\n" , ntohs(udph->source));
	fprintf(logfile , "   |-Destination Port : %d\n" , ntohs(udph->dest));
	fprintf(logfile , "   |-UDP Length       : %d\n" , ntohs(udph->len));
	fprintf(logfile , "   |-UDP Checksum     : %d\n" , ntohs(udph->check));

	fprintf(logfile , "\n");
	fprintf(logfile , "IP Header\n");
```

```
        PrintData(Buffer , iphdrlen);

        fprintf(logfile , "UDP Header\n");
        PrintData(Buffer+iphdrlen , sizeof udph);

        fprintf(logfile , "Data Payload\n");

        //Move the pointer ahead and reduce the size of string
        PrintData(Buffer + header_size , Size - header_size);

        fprintf(logfile ,
"\n#####################################################################");
}

void print_icmp_packet(unsigned char* Buffer , int Size)
{
        unsigned short iphdrlen;

        struct iphdr *iph = (struct iphdr *)(Buffer  + sizeof(struct ethhdr));
        iphdrlen = iph->ihl * 4;

        struct icmphdr *icmph = (struct icmphdr *)(Buffer + iphdrlen  + sizeof(struct
ethhdr));

        int header_size =  sizeof(struct ethhdr) + iphdrlen + sizeof icmph;
```

```
	fprintf(logfile , "\n\n**********************ICMP
Packet*************************\n");


	print_ip_header(Buffer , Size);


	fprintf(logfile , "\n");


	fprintf(logfile , "ICMP Header\n");
	fprintf(logfile , "   |-Type : %d",(unsigned int)(icmph->type));


	if((unsigned int)(icmph->type) == 11)
	{
		fprintf(logfile , "  (TTL Expired)\n");
	}
	else if((unsigned int)(icmph->type) == ICMP_ECHOREPLY)
	{
		fprintf(logfile , "  (ICMP Echo Reply)\n");
	}


	fprintf(logfile , "   |-Code : %d\n",(unsigned int)(icmph->code));
	fprintf(logfile , "   |-Checksum : %d\n",ntohs(icmph->checksum));
	//fprintf(logfile , "   |-ID       : %d\n",ntohs(icmph->id));
	//fprintf(logfile , "   |-Sequence : %d\n",ntohs(icmph->sequence));
	fprintf(logfile , "\n");


	fprintf(logfile , "IP Header\n");
```

```
        PrintData(Buffer,iphdrlen);

        fprintf(logfile , "UDP Header\n");
        PrintData(Buffer + iphdrlen , sizeof icmph);

        fprintf(logfile , "Data Payload\n");

        //Move the pointer ahead and reduce the size of string
        PrintData(Buffer + header_size , (Size - header_size) );

        fprintf(logfile ,
"\n#####################################################################");
}

void PrintData (unsigned char* data , int Size)
{
        int i , j;
        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" );
            }
        }
}
```

## 3.2 OUTPUT SNAPSHOTS

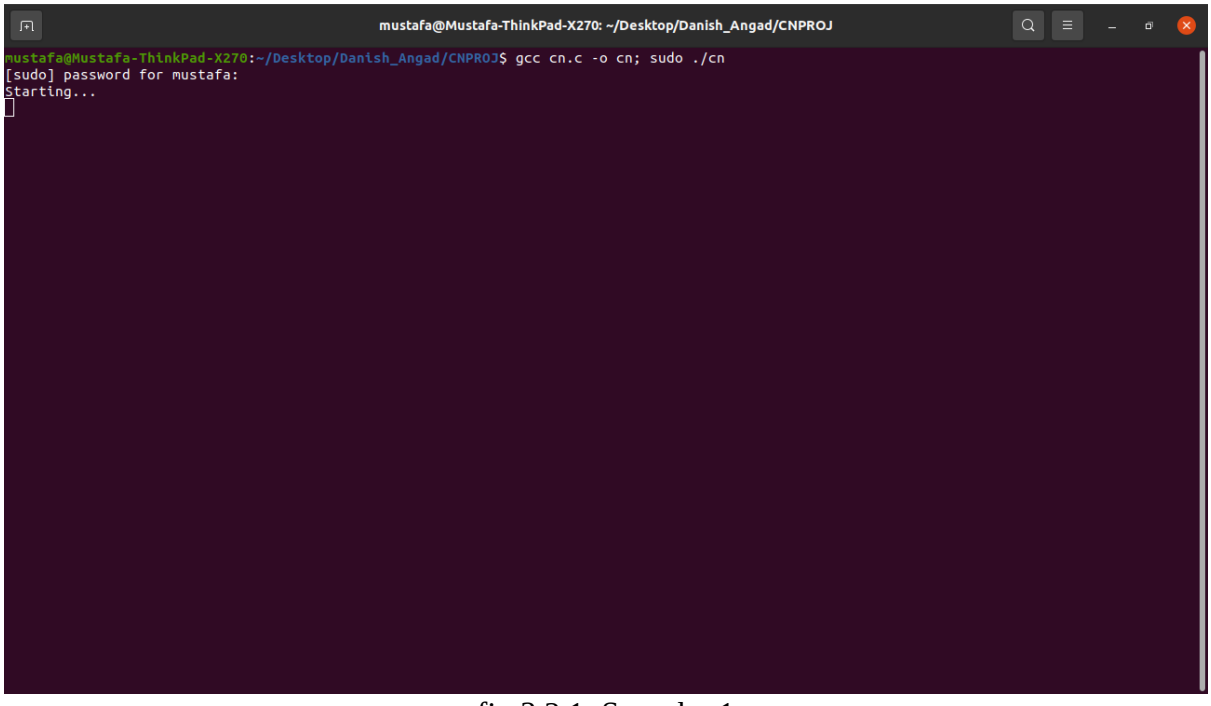


fig 3.2.1: Snapshot1

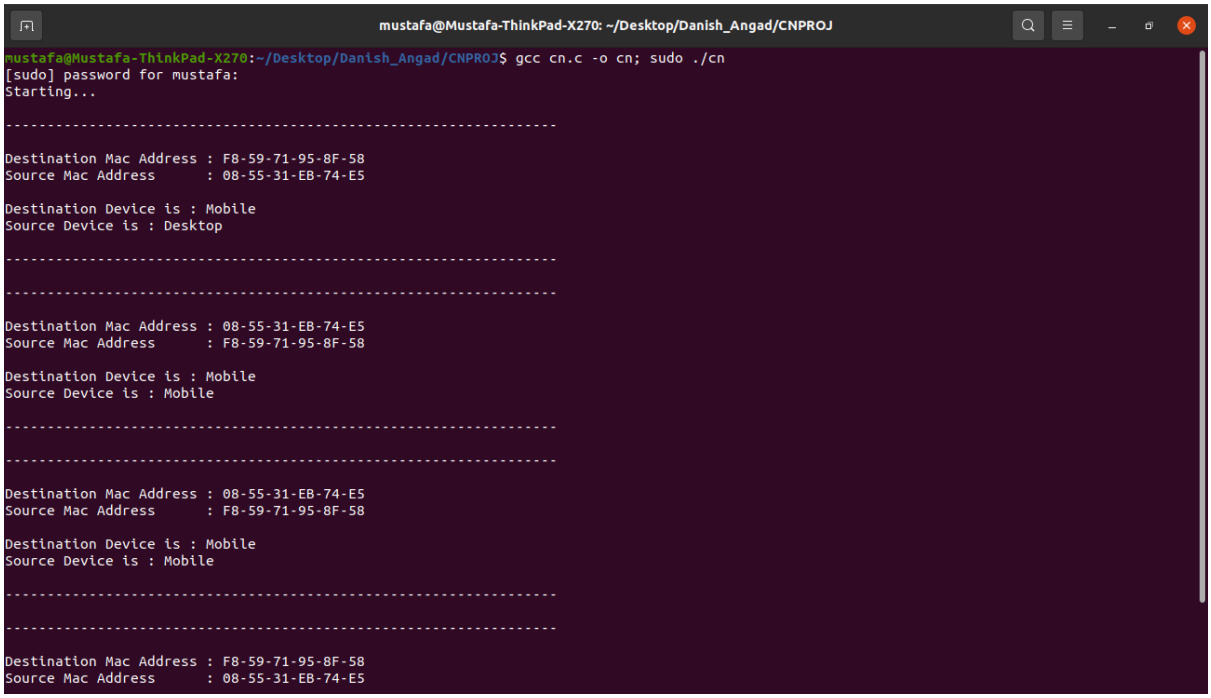


fig 3.2.2: Snapshot2

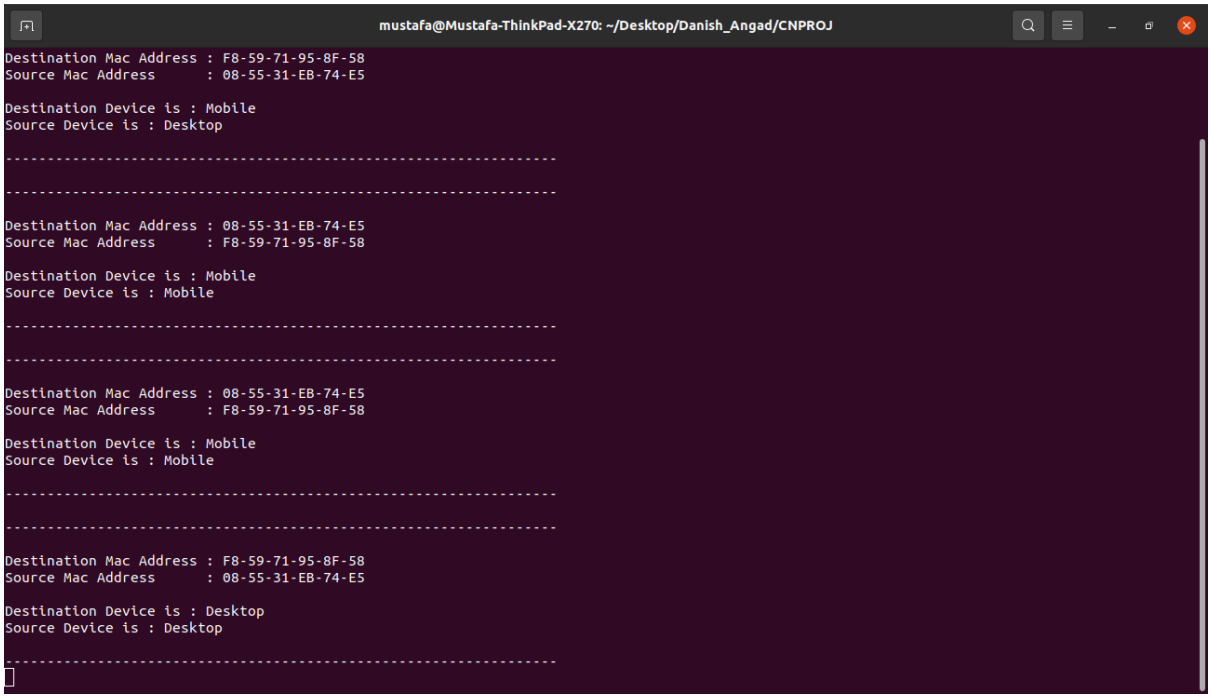


fig 3.2.3 Snapshot3

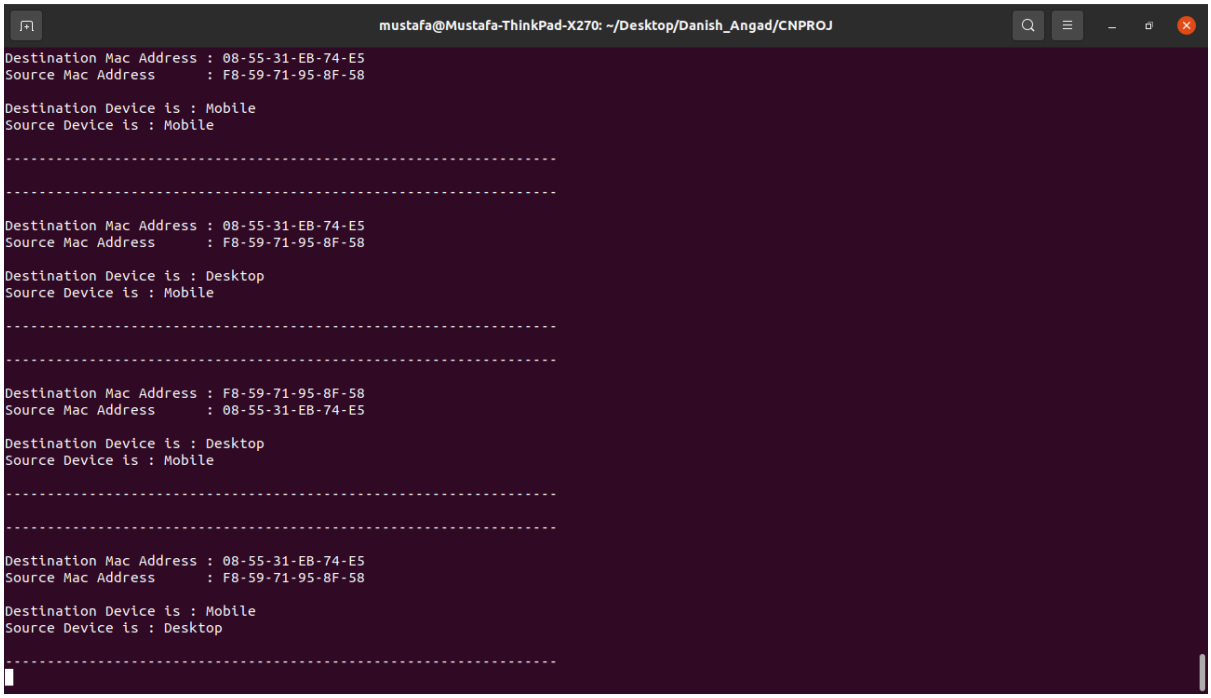


fig 3.2.4 Snapshot4

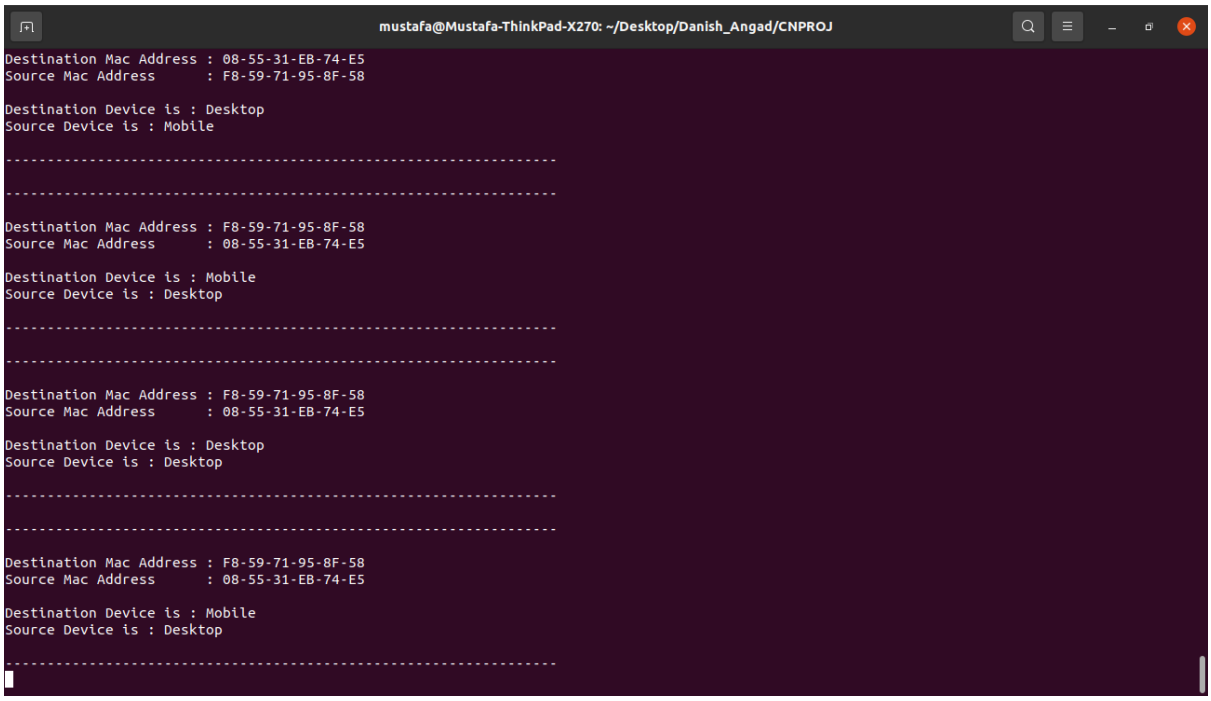


fig 3.2.5 Snapshot5

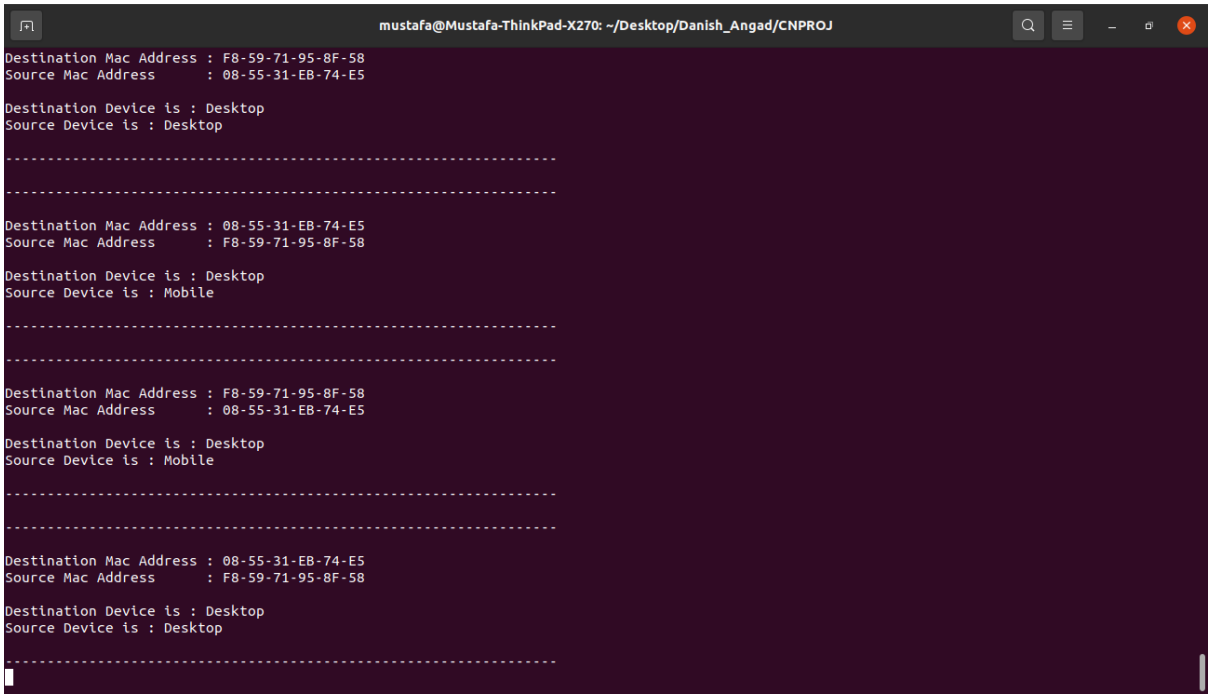


fig 3.2.6 Snapshot6

# REFERENCES

[1] Freudiger, J. (2015) How Talkative is your Mobile Device? An Experimental Study of Wi-Fi Probe Requests. Available at: https://frdgr.ch/wpcontent/uploads/2015/06/Freudiger15.pdf

[2] Allan, A. List of MAC addresses with Vendor identities. Available at: https://gist.github.com/aallan/b4bb86db86079509e6159810ae9bd3e4