# PROGRAM:-

#Packet sniffer in python

#For Linux import socket

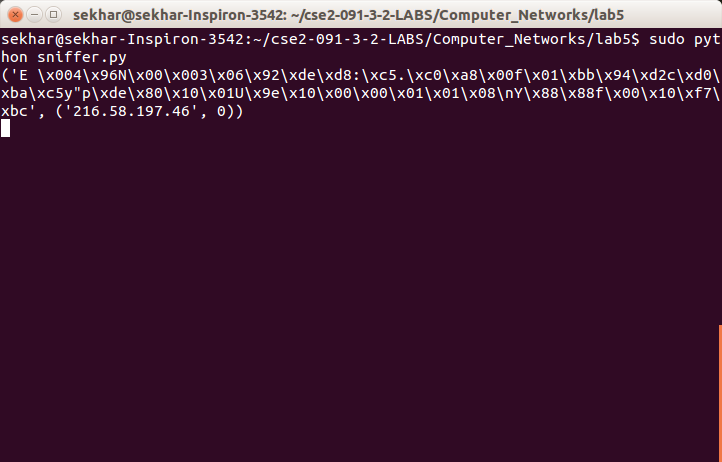
#create an INET, raw socket

s = socket.socket(socket.AF\_INET, socket.SOCK\_RAW, socket.IPPROTO\_TCP)

# receive a packet while True:

print s.recvfrom(65565)

# OUTPUT:-



**PROGRAM:-**

#Packet sniffer in python for Linux

#Sniffs only incoming TCP packet import socket, sys

from struct import \*

#create an INET, STREAMing socket try:

s = socket.socket(socket.AF\_INET, socket.SOCK\_RAW, socket.IPPROTO\_TCP)

except socket.error , msg:

print NSocket could not be created. Error Code : N + str(msg[0]) + N Message N

+msg[1]

sys.exit()

# receive a packet while True:

packet = s.recvfrom(65565)

#packet string from tuple

# print "hi"

packet = packet[0]

#take first 20 characters for the ip header ip\_header = packet[0:20]

#now unpack them :)

iph = unpack(N!BBHHHBBH4s4sN , ip\_header) version\_ihl = iph[0]

version = version\_ihl >> 4 ihl = version\_ihl & 0xF iph\_length = ihl \* 4

ttl = iph[5] protocol = iph[6]

s\_addr = socket.inet\_ntoa(iph[8]); d\_addr = socket.inet\_ntoa(iph[9]);

print NVersion : N + str(version) + N IP Header Length : N + str(ihl) + N TTL : N

+str(ttl) + N Protocol : N + str(protocol) + N Source Address : N + str(s\_addr) + NDestination Address : N + str(d\_addr)

tcp\_header = packet[iph\_length:iph\_length+20]

#now unpack them :)

tcph = unpack(N!HHLLBBHHHN , tcp\_header)

source\_port = tcph[0] dest\_port = tcph[1] sequence = tcph[2] acknowledgement = tcph[3]

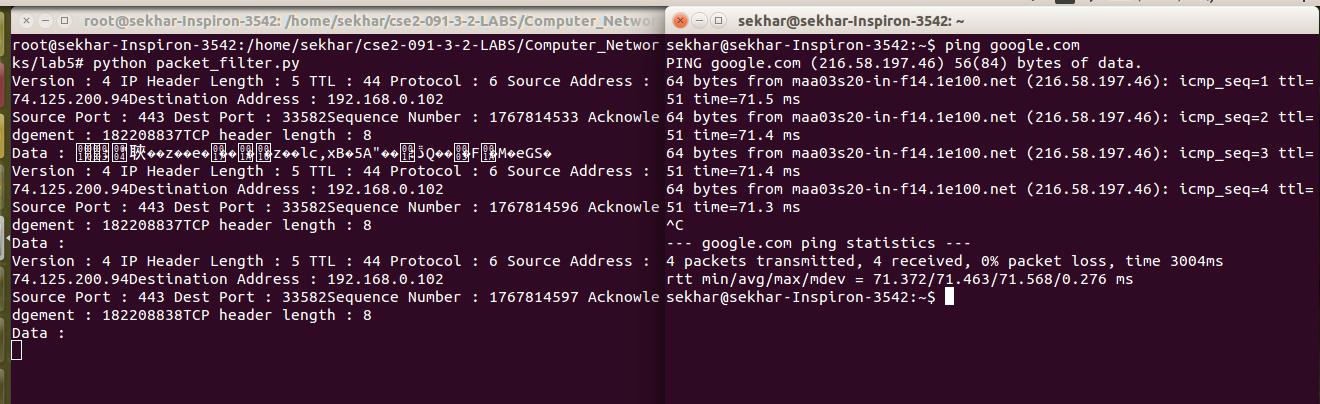
doff\_reserved = tcph[4] tcph\_length = doff\_reserved >> 4

print NSource Port : N + str(source\_port) + N Dest Port : N + str(dest\_port) + NSequence Number : N + str(sequence) + N Acknowledgement : N + str(acknowledgement) + NTCP header length : N + str(tcph\_length)

h\_size = iph\_length + tcph\_length \* 4 data\_size = len(packet) - h\_size

#get data from the packet data = packet[h\_size:] print NData : N + data

# OUTPUT:-



**PROGRAM:-**

## SERVER

#include "stdio.h"

#include "stdlib.h"

#include "string.h"

#include "sys/types.h"

#include "sys/socket.h"

#include "arpa/inet.h"

#include "netinet/in.h"

#define SA struct sockaddr struct IPmac {

char ip[100]; char mac[100];

};

int main() { int sockfd,len,i;

struct sockaddr\_in servaddr;

char buff[30],temp[30],ip[30],mac[30]; int flag=0;

struct IPmac in[3]={

{"10.1.1.8","44:dd:22:11:33"},

{"127.0.0.1","33:aa:fe:4e:2d"},

{"10.1.8.5","23:a3:5d:33:9d"}

};

//printing table printf("ip\t\tmac\n"); for(i=0;i<3;i++)

{

printf("%s\t%s\n",in[i].ip,in[i].mac);

}

//create socket

sockfd = socket(AF\_INET,SOCK\_DGRAM,0);

//fill structure servaddr.sin\_family = AF\_INET; servaddr.sin\_port = htons(9999);

servaddr.sin\_addr.s\_addr = INADDR\_ANY;

//bind bind(sockfd,(SA\*)&servaddr,sizeof(servaddr));

//get ip from client len=sizeof(servaddr);

recvfrom(sockfd,ip,sizeof(ip),0,(SA\*)&servaddr,&len); for(i=0;i<strlen(ip)-1;i++) {

temp[i]=ip[i];

}

temp[i]=N\0N;

printf("received IP :%s\n",temp);

//searching in table for equivalent mac for(i=0;i<3;i++) { if(strcmp(temp,in[i].ip)==0) {

strcpy(mac,in[i].mac); break;

}

}

printf("mac address is %s\n",mac); sendto(sockfd,mac,sizeof(mac),0,(SA\*)&servaddr,len);

//rarp simulation

//recv mac address bzero(mac,sizeof(mac));

recvfrom(sockfd,mac,sizeof(mac),0,(SA\*)&servaddr,&len); printf("received mac address :%s",mac);

//store in temp bzero(temp,sizeof(temp)); for(i=0;i<strlen(mac)-1;i++) { temp[i]=mac[i];

}

temp[i]=N\0N; bzero(ip,sizeof(ip));

//check in table for(i=0;i<3;i++) { if(strcmp(temp,in[i].mac)==0) { strcpy(ip,in[i].ip);

break;

}

}

printf("ip address :%s\n",ip); sendto(sockfd,ip,sizeof(ip),0,(SA\*)&servaddr,len); return 0;

}

## CLIENT

#include "stdio.h"

#include "stdlib.h"

#include "string.h"

#include "sys/types.h"

#include "sys/socket.h"

#include "arpa/inet.h"

#include "netinet/in.h"

#define SA struct sockaddr int main()

{

int sockfd,len;

char ip[30],mac[30];

struct sockaddr\_in servaddr;

//creating socket

sockfd = socket(AF\_INET,SOCK\_DGRAM,0);

//fill structure servaddr.sin\_family = AF\_INET; servaddr.sin\_port = htons(9999);

servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

//send ip address

printf("ARP SIMULATION\n"); printf("enter ip address :"); fgets(ip,sizeof(ip),stdin);

sendto(sockfd,ip,sizeof(ip),0,(SA\*)&servaddr,sizeof(servaddr)); len=sizeof(servaddr); recvfrom(sockfd,mac,sizeof(mac),0,(SA\*)&servaddr,&len); printf("MAC address is: %s\n",mac);

printf("RARP simulation\n"); printf("enter mac address :"); bzero(mac,sizeof(mac)); fgets(mac,sizeof(mac),stdin);

sendto(sockfd,mac,sizeof(mac),0,(SA\*)&servaddr,len); recvfrom(sockfd,ip,sizeof(ip),0,(SA\*)&servaddr,&len); printf("IP address is: %s\n",ip);

return 0;

}

# OUTPUT:-

