**A PBL Report**

On

# Face Recognition Based Attendance System

*Submitted to JNTU HYDERABAD*

*In Partial Fulfillment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

Submitted By


**HEENA BEGUM - 208R1A0523**

**D.GREESHMA - 208R1A0516**

**G.PRATHYUSHA - 218R5A0506**

**A.RAJITHA - 218R5A0501**


Under the guidance of

**T.Neha**

Assistant Professor, Department of CSE



## Department of Computer Science & Engineering

## CMR ENGINEERING COLLEGE

(Accredited by NBA, Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad)

Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)


**2022-2023**

# CMR ENGINEERING COLLEGE

*(Accredited by NBA,Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)*

*Kandlakoya, Medchal Road, Hyderabad-501 401*

## Department of Computer Science & Engineering


## *CERTIFICATE*

This is to certify that the project entitled **"PACKET SNIFFER"** is a bonafide work carried out by

**YUKTA MAHESEKAR - 198R1A0560**

**PRANJAL UPADHYAY- 198R1A0544**

**VADLA KSHITHIJA-198R1A0557**

**LAKKARUSU MANASA-198R1A0528**

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this project have been verified and are found to be satisfactory. The results embodied in this project have not been submitted to any other university for the award of any other degree or diploma.

| | |
|---|---|
| _____ | _____ |
| Internal Guide | Head of the Department |
| **Neha Bajaj** | **Dr. Sheo Kumar** |
| Assistant Professor | Professor & HOD |
| Department of CSE, | Department of CSE, |
| CMREC, Hyderabad | CMREC, Hyderabad |

# DECLARATION

This is to certify that the work reported in the present project entitled "**PACKET SNIFFER**" is a record of bonafide work done by me in the Department of Computer Science and Engineering, CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by me and not copied from any other source. I submit my project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

**YUKTA MAHESEKAR-198R1A0560**

**PRANJAL UPADHYAY-198R1A0544**

**VADLA KSHITHIJA-198R1A0557**

**LAKKARUSU MANASA-198R1A0528**

**CONTENTS TITLE**

**PAGE NO**

# 1.0 INTRODUCTION

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 analyzer, 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 analyze the data collected by the device.

A network is a collection of nodes, such as personal computers, servers, and networking hardware that are connected. The network connection allows data to be transferred between these devices. The connections can be physical with cables, or wireless with radio signals. Networks can also be a combination of both types.

As nodes send data across the network, each transmission is broken down into smaller pieces called packets. The defined length and shape allows the data packets to be checked for completeness and usability. Because a network's infrastructure is common to many nodes, packets destined for different nodes will pass through numerous other nodes on the way to their destination. To ensure data is not mixed up, each packet is assigned an address that represents the intended destination of that packet.

A packet's address is examined by each network adapter and connected device to determine what node the packet is destined for. Under normal operating conditions, if a node sees a packet that is not addressed to it, the node ignores that packet and its data. Packet sniffing ignores this standard practice and collects all, or some of the packets, regardless of how they are addressed.

There are two main types of packet sniffers:

1. Hardware Packet Sniffers

A hardware packet sniffer is designed to be plugged into a network and to examine it. A hardware packet sniffer is particularly useful when attempting to see traffic of a specific network segment. By plugging directly into the physical network at the appropriate location, a hardware packet sniffer can ensure that no packets are lost due to filtering, routing, or other deliberate or inadvertent causes.A hardware packet sniffer either stores the collected packets or forwards them on to a collector that logs the data collected by the hardware packet sniffer for further analysis.

2. Software Packet Sniffers

Most packet sniffers these days are of the software variety. While any network interface attached to a network can receive every bit of network traffic that flows by, most are configured not to do so. A software packet sniffer changes this configuration so that the network interface passes all network traffic up the stack. This configuration is known as promiscuous mode for most network adapters.

Once in promiscuous mode, the functionality of a packet sniffer becomes a matter of separating, reassembling, and logging all software packets that pass the interface, regardless of their destination addresses. Software packet sniffers collect all the traffic that flows through the physical network interface. That traffic is then logged and used according to the packet sniffing requirements of the software.

Capturing data on an entire network may take multiple packet sniffers. Because each collector can only collect the network traffic that is received by the network adapter, it may not be able to see traffic that exists on the other side of routers or switches. On wireless networks, most adapters are capable of connecting to only one channel at a time. In order to capture data on multiple network segments, or multiple wireless channels, a packet sniffer is needed on each segment of the network. Most network monitoring solutions provide packet sniffing as one of the functions of their monitoring agents.

## 1.1 REQUIREMENTS

### 1.1.1 Software Requirements

- Operating System : Windows 11/10/8/7 or Linux
- Tool used : Jupyter notebook
- Python : python3.6

### 1.1.2 Minimum hardware requirements

- RAM : 4GB
- System Architecture : 64-bit
- Hard Disk Space : 3GB
- Processor : Intel Atom or Intel i3 core

# 2.0 BACKGROUND

In today's life networks is playing a very important role in telecommunication. without the network, almost all types of communication and service are useless. hence, this makes network concept more important for all programmers and network administrators.

To maintain and manage the security of network communication, many times network administrators Or network maintainers need to find and control the traffic flowing into the network wire and also find exactly what and which types of data packets are actually flowing into the networks.

For this situation, there are many types of Network analyzing tools are available On the internet. basically, these types of tool come on the ground to help network administrator like Wireshark and other. These tools are fast, easy and reliable to handle many types of network problems but as we know, networking concept is not that easy. so, many time these types of tools do not support our exact situation requirement and we have to find any other solution for our problem and at that time python and its socket module comes on the ground like a big boy to help network administrators. Well, as we know python is really the very awesome language and also very powerful language. With Python, a programmer can do almost any types of programming in fastest and easiest way. So, python and socket module use Packet Sniffing.

Packet analyzing is a form of a Network Intrusion Detection (NID) and has only recently begun to become revolutionized into a useful tool for companies and businesses within the information security world (Elson, 2008). The goal of intrusion detection is to discover anomalous and malicious behavior and misuse of network assets which gained popularity around thirty years ago (Elson, 2008). Over the years, network administrators have used packet sniffing tools to observe networks and conduct analyses as well as troubleshoot problems. Since then, it has evolved into a useful defense system as well as a cause of malicious interception of sensitive data and information traveling along communication lines. As a backup measure, packet sniffing was originally meant to be used as a diagnostic tool to save data and other information being sent across the network (Elson, 2008).

The first network monitors and packet sniffer devices were called Novell LANalyser and Microsoft Network Monitor (Elson, 2008). Once the packets were captured, they could be counted to see how populated the network segment was, or analyzed in detail to see what problems are wrong with the network server (Elson, 2008). New programs developed over time such as Ethereal and improved Microsoft Network Monitor that were able to decipher communication exchanges to other interfaces (Elson, 2008). However, as more advances techniques and technologies advanced, network monitors and packet sniffers began to use their skills to attack networks and deploy schemes to obtain information that should have been kept secure. In order to combat this malicious method of packet sniffing, the use of multiple switches rather than hubs within networks has been proved to reduce the threat of successful attacks such as these because they limit packets from traveling across multiple interfaces thus stopping evil packet sniffers (Elson, 2008).

**What is Packet Sniffer?**

Sniffers are the special programs and tools that can capture network traffic packets from the network and then parse/ analyze them for various purposes. actually, sniffing tools have the ability to capture flowing data packets from networks. Data packets like TCP, UDP, ICMP etc. and after capturing these packets, sniffer also provides the facilities to extract these data packets and represent these packets in easy to understand interface.

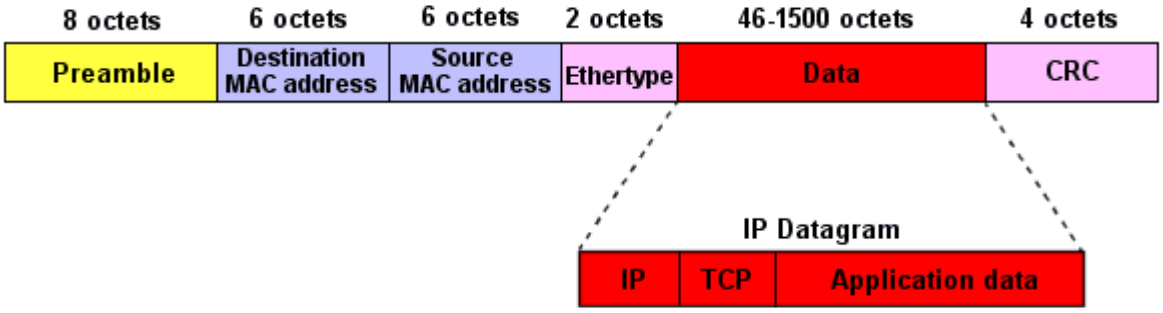
**How This Programs Going to Works?**

Every Client Interact With Server Through Sending and receiving various types of data Packets like TCP, UDP etc. so, our program is going to capture all those data packets from our local computer network and then analyze and represent those packets in easy to understandable ways.

**How To Capture Packets?**

For capturing packets, we are going to use socket.socket module. For sniffing with socket module in python we have to create a socket.socket class object with special configuration.

**How To Parse/Extract Captured Packets?**

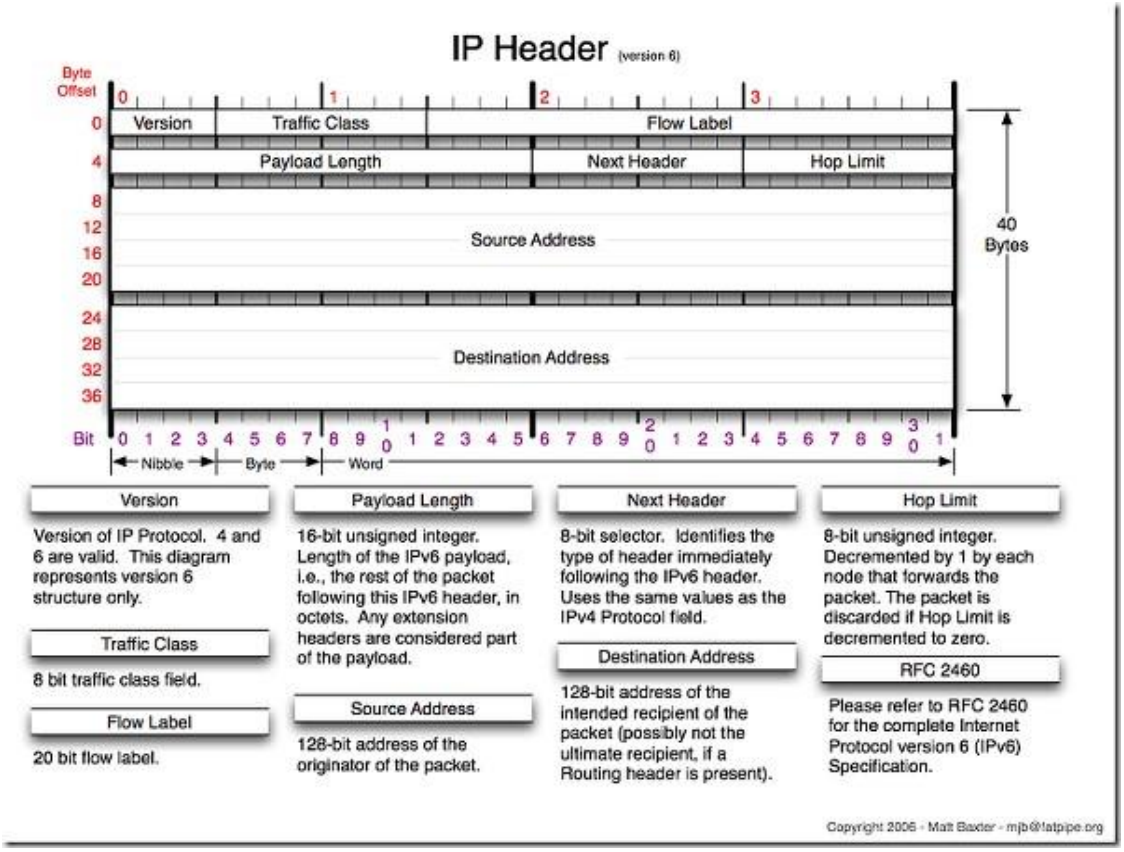



As you can see in Ethernet Frame Format Diagram There are more than 3 fields to extract but here, for this project we are only going to extract only 3 fields, Source Mac Address, Destination Mac Address and Ethernal Protocol Type.To Extract Source Address, Destination Address, and Ethernet Type Address, We have to use struct module which can unpack network packets for us.

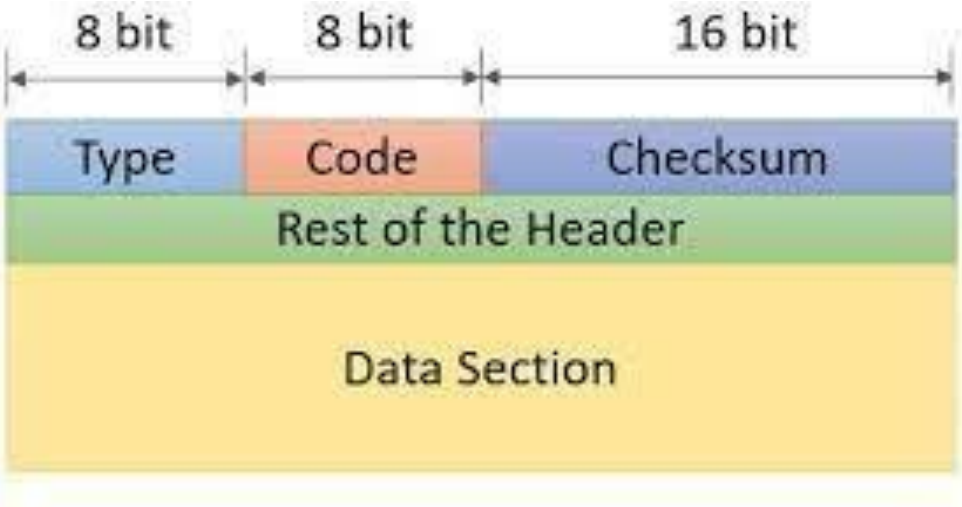
Basically, To Extract Data From Network Packets we have to pass an argument that going to represent field types, we want to extract in struct.unpack function.

The following figures help us to understand header formats of protocols:

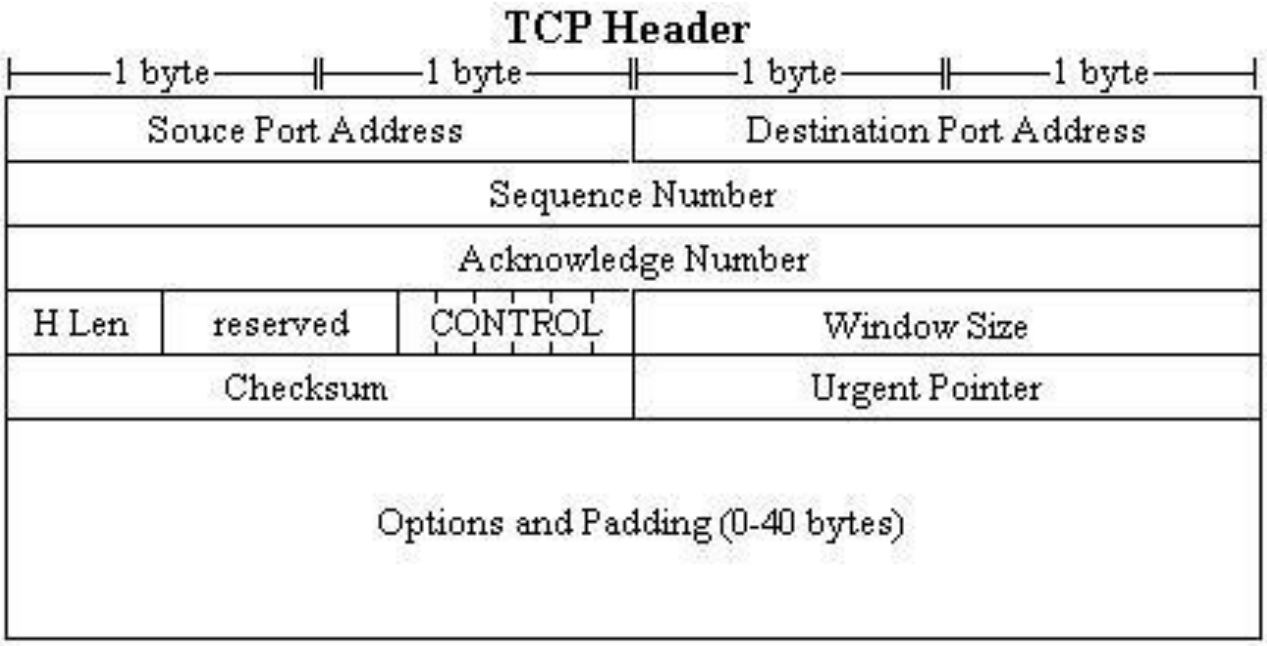
IP Header Format



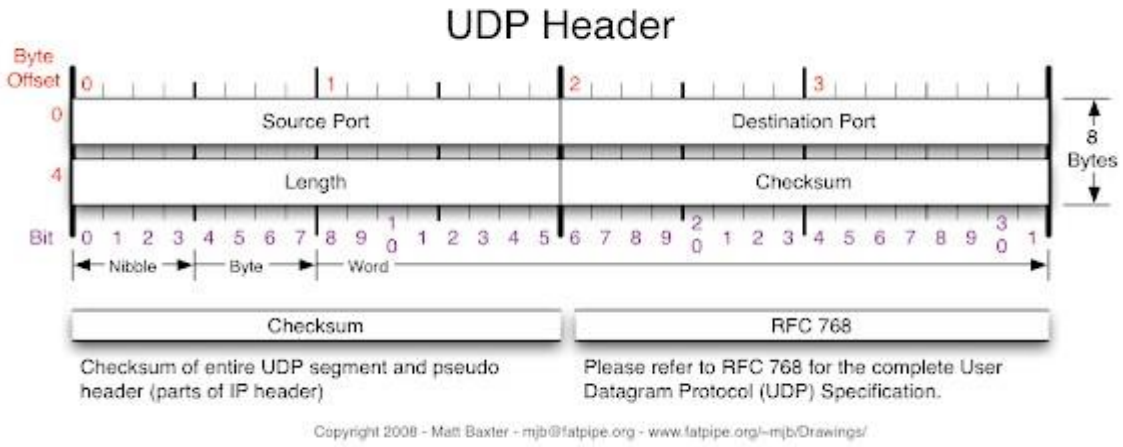

ICMP Header Format




TCP Header Format




UDP Header Format




## 3.0 CODE BASE

We will Create 2 Scripts.

1. For Capturing Packets (pypackets.py)

```
import socket
import struct
import binascii
import os
import pye
# if operating system is windows if
os.name == "nt":
    s = socket.socket(socket.AF_INET,socket.SOCK_RAW,socket.IPPROTO_IP)
    s.bind(("YOUR_INTERFACE_IP",0))
    s.setsockopt(socket.IPPROTO_IP,socket.IP_HDRINCL,1)
    s.ioctl(socket.SIO_RCVALL,socket.RCVALL_ON)

# if operating system is linux
else:
    s=socket.socket(socket.PF_PACKET, socket.SOCK_RAW,
socket.ntohs(0x0800))

# create loop  while
True:

    # Capture packets from network
pkt=s.recvfrom(65565)

    # extract packets with the help of pye.unpack class
unpack=pye.unpack()
     print ("\n\n===&gt;&gt; [+] ------------ Ethernet Header----- [+]")
    # print data on terminal     for i in
unpack.eth_header(pkt[0][0:14]).iteritems():
        a,b=i
        print "{} : {} | ".format(a,b),     print ("\n===&gt;&gt; [+]
------------ IP Header ------------[+]")     for i in
unpack.ip_header(pkt[0][14:34]).iteritems():
        a,b=i         print ("{} : {} | ".format(a,b) ),     print
("\n===&gt;&gt; [+] ------------ Tcp Header ----------- [+]")     for
i in unpack.tcp_header(pkt[0][34:54]).iteritems():
        a,b=i
        print ("{} : {} | ".format(a,b) ),
```

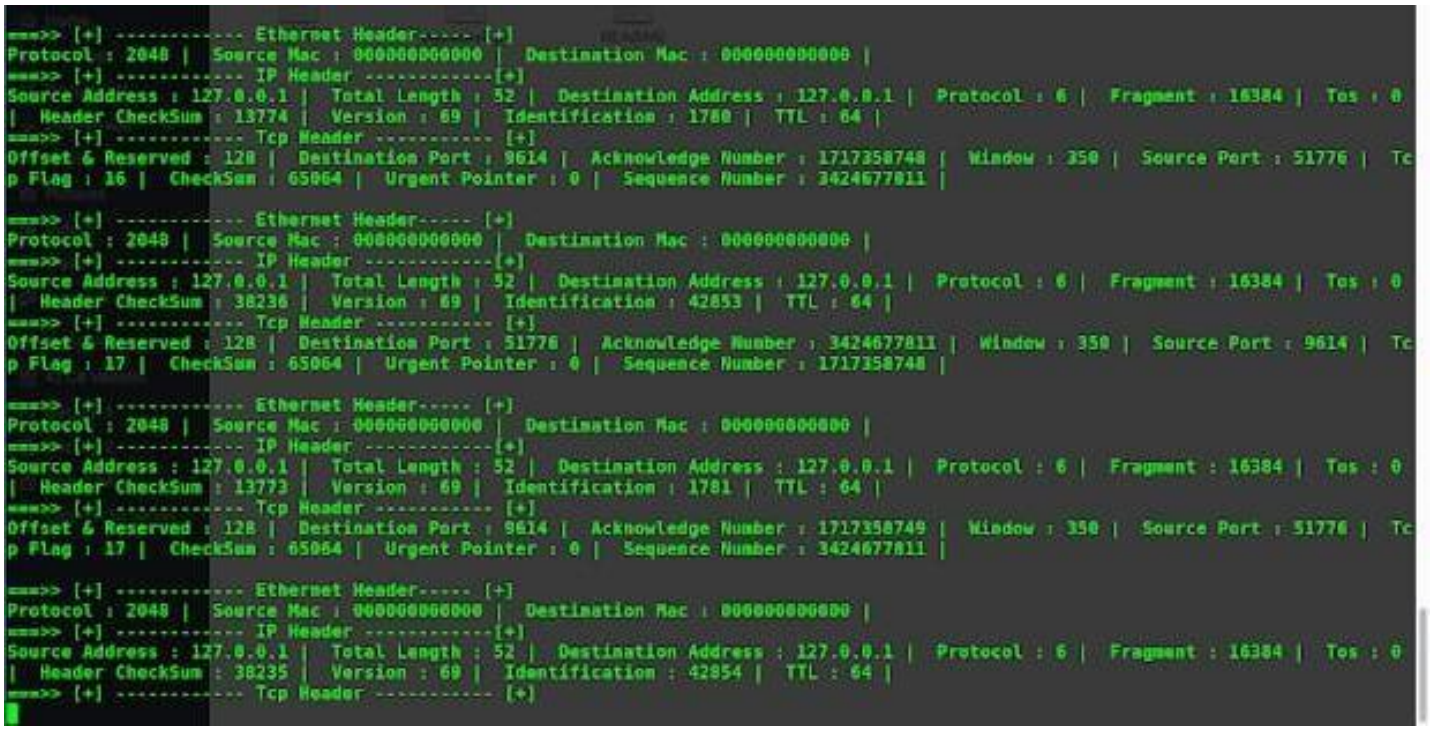
2. For Extracting Captured Data (pye.py)

```
import socket, struct, binascii
 class unpack:  def
__cinit__(self):
  self.data=None

 # Ethernet Header  def
eth_header(self, data):
  storeobj=data
storeobj=struct.unpack("!6s6sH",storeobj)
destination_mac=binascii.hexlify(storeobj[0])
source_mac=binascii.hexlify(storeobj[1])
eth_protocol=storeobj[2]   data={"Destination
Mac":destination_mac,
  "Source Mac":source_mac,
"Protocol":eth_protocol}   return
data

 # ICMP HEADER Extraction
def icmp_header(self, data):
  icmph=struct.unpack('!BBH', data)
icmp_type = icmph[0]   code =
icmph[1]   checksum = icmph[2]
data={'ICMP Type':icmp_type,
"Code":code,   "CheckSum":checksum}
return data

 # UDP Header Extraction  def
udp_header(self, data):
storeobj=struct.unpack('!HHHH', data)
source_port = storeobj[0]   dest_port =
storeobj[1]   length = storeobj[2]
checksum = storeobj[3]   data={"Source
Port":source_port,
  "Destination Port":dest_port,
```

```
  "Length":length,
"CheckSum":checksum}   return
data

 # IP Header Extraction  def ip_header(self,
data):
storeobj=struct.unpack("!BBHHHBBH4s4s", data)
  _version=storeobj[0]
  _tos=storeobj[1]
  _total_length =storeobj[2]
  _identification =storeobj[3]
  _fragment_Offset =storeobj[4]
  _ttl =storeobj[5]
  _protocol =storeobj[6]
  _header_checksum =storeobj[7]
  _source_address =socket.inet_ntoa(storeobj[8])
  _destination_address =socket.inet_ntoa(storeobj[9])

  data={'Version':_version,
  "Tos":_tos,
  "Total Length":_total_length,
  "Identification":_identification,
  "Fragment":_fragment_Offset,
  "TTL":_ttl,
  "Protocol":_protocol,
  "Header CheckSum":_header_checksum,
  "Source Address":_source_address,
"Destination Address":_destination_address}
return data

 # Tcp Header Extraction  def
tcp_header(self, data):
storeobj=struct.unpack('!HHLLBBHHH',data)
  _source_port =storeobj[0]
  _destination_port  =storeobj[1]
  _sequence_number  =storeobj[2]
  _acknowledge_number  =storeobj[3]
  _offset_reserved  =storeobj[4]
  _tcp_flag  =storeobj[5]
_window  =storeobj[6]
```

```
  _checksum                =storeobj[7]
_urgent_pointer            =storeobj[8]
data={"Source       Port":_source_port,
"Destination Port":_destination_port,
  "Sequence Number":_sequence_number,
  "Acknowledge Number":_acknowledge_number,
  "Offset & Reserved":_offset_reserved,
  "Tcp Flag":_tcp_flag,
"Window":_window,
  "CheckSum":_checksum,
  "Urgent Pointer":_urgent_pointer
  }
  return data

# Mac Address Formating def
mac_formater(a):
 b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]), ord(a[1]), ord(a[2]),
ord(a[3]), ord(a[4]) , ord(a[5]))  return b
 def
get_host(q):
try:
  k=socket.gethostbyaddr(q)
except:   k='Unknown'
 return k
```

## 4.0 OUTPUT




## 5.0 CONCLUSIONS

Packet Sniffer developed was able to capture packets like TCP, UDP, ICMP. For each component the information like source ip, destination ip, source port, destination port is printed out. This script is handy to capture packets from the terminal. This tool is extremely handy for infosec and network professionals to capture packets by running a script with small size.

# 6.0 REFERENCES


https://www.bitforestinfo.com/blog/02/15/how-to-write-simple-packet-sniffer.html

https://www.ukessays.com/essays/information-technology/the-history-of-packet-sniffing-informationtechnology-essay.php