DOCUMENTATION

PACKET SNIFFER

Detailed Project Description :

While diagnosing the underlying problems of a network, we may want an essential tool that aids in monitoring network traffic and troubleshooting a network. Packet sniffer is one such tool.
Our project focuses on capturing and analyzing packets of data that flow through a particular network.
Packet sniffer can be hardware device or software application. Our development tool is a software application. This tool runs on standard general-purpose computers performing packet sniffing tasks by using the hardware capabilities of the network.
This tool allows the user to be able to view Source of the packets and display the target host and the type of protocol used like UDP OR TCP.

Description of Working Procedure with algorithm :
(in between this para we need to add technical terms and may be some algo which we used in our code)

We know that data travels through a network in the form of packets. In packet-switched networks, the data to be transmitted is broken down into several packets. These packets are reassembled once all the data packets reach their intended destination.

When a packet sniffer is installed in the network, the sniffer intercepts the network traffic and captures the raw data packets. Subsequently, the captured data packet is analyzed by the packet sniffing software and presented to the network manager/technician in a user-friendly format. By user-friendly, we mean the Network Administrator should be able to make sense of it.

(**THIS IS WHAT ACTUALLY HAPPENS IN PACKET SNIFFING CODE**)

1. **SNIFFING** by using a)SOCKET() system call.(practically All the sniffing is done here).
   b)recvfrom loop is used to receive data (input).
   The buffer will hold the data sniffed
2. **read the captured packet**

`icmp_packet(data)`:**ICMP** packets can be analysed by capturing the packet and validating it and filtering is done based on the protocol used.
So we developed different protocol functions like ICMP, TCP (HTTP), UDP, others(IPV4).

3. **analyse it**
   Our packet sniffer tool can sniff TCP,UDP,ICMP and IPv4 packets by unpacking the structure and matching the raw data with the protocol segments.
   When code breaks the input into different sections and unpacks them, it's analysis looks like this.
4. **present it to the user in a readable format.**
   - **Unpack Ethernet Frame : `struct.unpack()`**
   - **Format MAC Address :**
   - **Unpack IPv4 Packets Received : `struct.unpack()`**
   - **In the same way unpacks and formats for TCP, UDP, ICMP packets using `struct.unpack()`.**

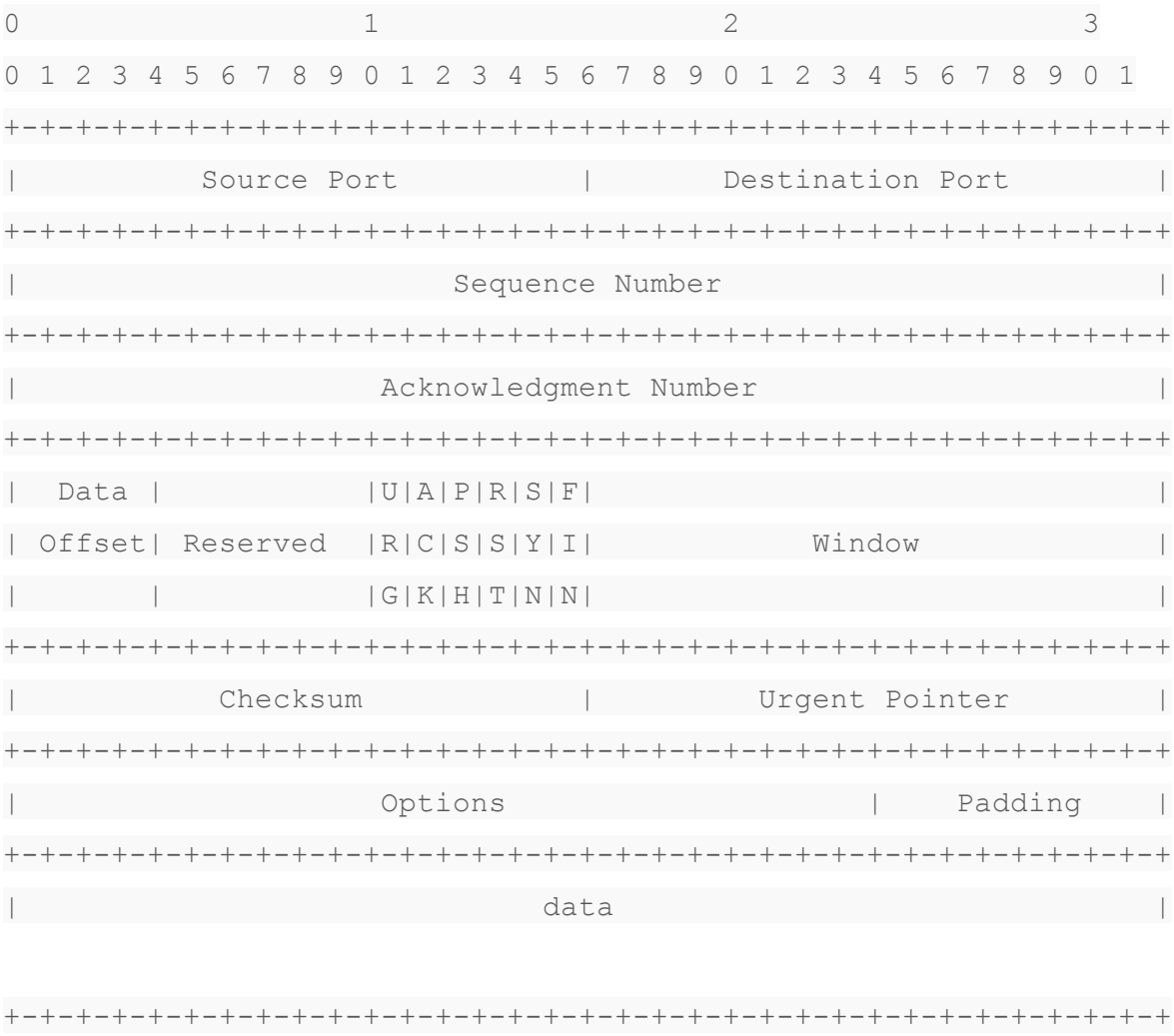
Flowchart (or block diagram): ---------

Implementation:
Requirements :

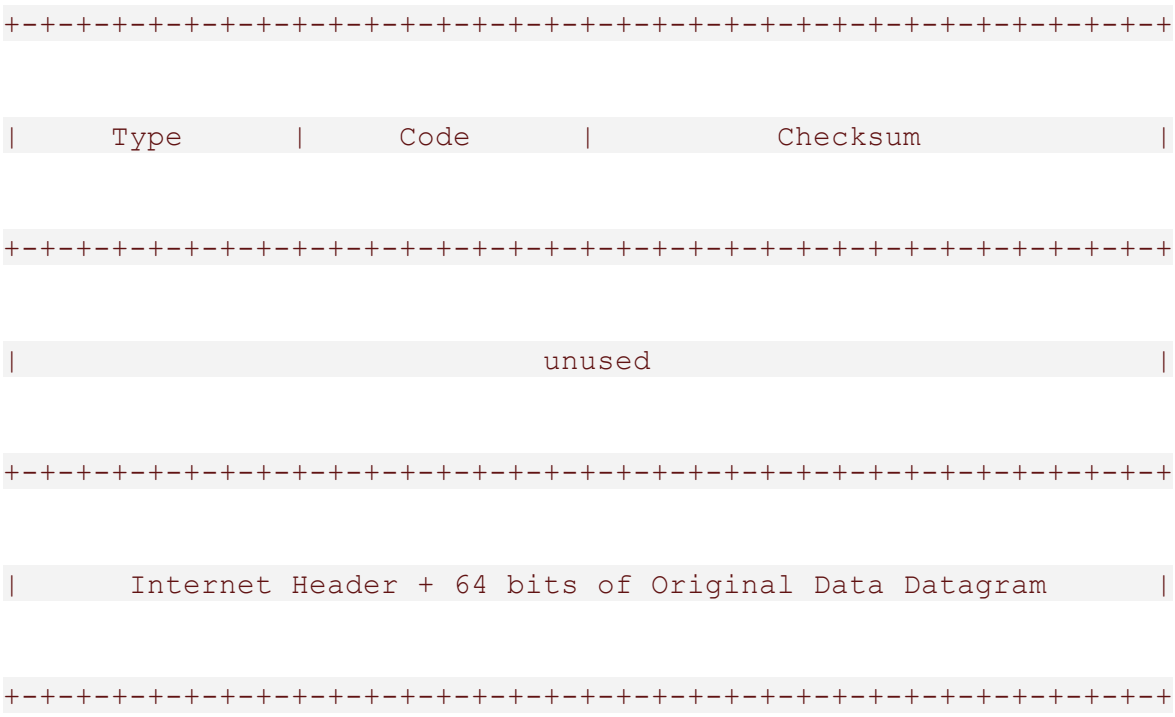
- Python 3.x
- Privileged/Administrative Rights
- Linux or Windows Operating System.

Import functions:
example socket() : the **socket()** function returns a socket objec*t* whose methods implement the various socket system calls. Packet sniffer use sockets api provided by the kernel.

TCP HEADER


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Data Offset
Reserved
U|A|P|R|S|F|
R|C|S|S|Y|I|
G|K|H|T|N|N|
Window
Checksum
Urgent Pointer
Options
Padding
data

```
UDP HEADER

0       7 8     15 16    23 24    31

 +--------+--------+--------+--------+

 |     Source      |   Destination   |

 |      Port       |      Port       |

 +--------+--------+--------+--------+

 |                 |                 |

 |     Length      |    Checksum     |

 +--------+--------+--------+--------+

 |

 |          data octets ...

 +---------------- …

ICMP HEADER

0                   1                   2                   3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```



Various transport layer protocols like TCP, UDP, are implemented for analysis.

packet sniffers are used to troubleshoot and rectify network-related problems.

Monitoring network usage – Packet sniffers are great at monitoring the network usage at any given time, helping Network Managers identify whether a particular network is normal or congested. Also, making it possible to identify bottlenecks within the network and identify and improve the performance with infrastructure upgrades.

Identifying problems – As mentioned earlier, packet sniffers can identify network-related issues. This is possible because a packet sniffer can analyze the conversation between two or more nodes in a network. So, in the event of a network error, the information captured by the packet sniffer can be used to identify the erroneous packets and pinpoint the node that failed to answer the request(s). Making it easy to identify faulty devices within the network in an efficient manner and providing the ability to take swift corrective actions.

Detecting security loopholes – A disturbing fact about packet sniffers is their ability to work as spying tools. They also help the good guys, such as your Network Manager, by testing the vulnerabilities of a network. Once these vulnerabilities are detected, it is easier to remove the loopholes thus preventing the possibilities of hacking attempts.

Results & Discussion:

Result : we have implemented a packet sniffer which can sniff TCP UDP ICMP and IPV4 packets being transmitted on the same network, by sniffing every packet by its domain structure and unpacking it accordingly and presenting to the user in a readable format by formatting.

- **Raw socket is capable of receiving all incoming traffic in the network so we get a dump of network packets, they should be parsed and then unpack function is used.**
- **Packet sniffers can be coded by either using sockets api provided by the kernel, or by using some packet capture library like libpcap.**