[1]

Computer Networks CL3001 Project Report - *A Network Sniffer Tool*

Hassan Abbas 21K-3286

Zaid Syed 21K-3348

Abdullah Siddiqui 21K-3447

I. Introduction

In today's interconnected world, understanding the flow of data across networks is vital for maintaining security, optimizing performance, and diagnosing issues. Packet analyzers serve as indispensable tools for network administrators, security professionals, and researchers, allowing them to inspect, analyze, and interpret the data packets traversing their network interfaces.

This report documents the development of a packet analyzer—a software tool designed to intercept, capture, and dissect network packets in real-time. The primary objective of this project is to empower users to examine and analyze all traffic passing through their computer's network interface, decode packet data, and present various packet fields based on their types.

The packet analyzer is envisioned to provide insights into network behavior, facilitate troubleshooting of network-related problems, and enhance the overall security posture by detecting anomalous or malicious activities. By parsing packet headers, payload data, and applying analytical techniques, users can gain a deeper understanding of their network's dynamics and identify potential threats or performance bottlenecks.

This report outlines the methodology, design considerations, implementation details, and features of the packet analyzer. It also discusses the challenges encountered during development, strategies employed to overcome them, and potential future enhancements to extend the tool's capabilities.

Through the creation of this packet analyzer, we aim to contribute to the arsenal of network monitoring and security tools available to practitioners, researchers, and enthusiasts alike. By empowering users with the ability to inspect and analyze network traffic comprehensively, we endeavor to foster a more secure, efficient, and resilient network infrastructure.

II. PROBLEM STATEMENT

Network administrators, security professionals, and researchers face the challenge of effectively monitoring and analyzing network traffic to ensure security, optimize performance, and troubleshoot issues. However, the lack of accessible and comprehensive tools for real-time packet analysis hinders their ability to gain deep insights into network behavior and identify potential threats or performance bottlenecks.

Existing packet analysis solutions often come with limitations such as high cost, complexity, or lack of customization, making them inaccessible to smaller organizations or individuals with limited resources. Additionally, many available tools may not provide real-time analysis capabilities or fail to decode packet data accurately, limiting their effectiveness in capturing and interpreting diverse network traffic.

Furthermore, as networks grow increasingly complex and dynamic, the need for robust packet analysis tools becomes more pronounced. The ability to examine all traffic passing through a network interface, decode packet data, and present relevant information in a user-friendly manner is essential for maintaining network security, optimizing performance, and diagnosing issues effectively.

Thus, there is a pressing need for the development of a packet analyzer that addresses these challenges by providing a comprehensive, real-time, and customizable solution for network traffic analysis. Such a tool would empower users to inspect and analyze network packets comprehensively, enabling them to detect anomalies, identify patterns, and make informed decisions to enhance the security and performance of their network infrastructure.

III. IDEA

When it is set up on a computer, the network interface of the computer is listening to all the traffic on the network rather than just those packets destined for it. Packet Analyzer will analyze the incoming and outgoing packets without modifying the network's packet in any way. It only makes a copy of each packet flowing through the network interface and finds the source and destination of the packets.

[1]

IV. PROPOSED MODEL

A. *Raw Socket*

A raw socket is used to receive raw packets. This means packets received at the Ethernet layer will directly pass to the raw socket. Stating it precisely as a raw socket bypasses the normal TCP/IP processing and sends the packets to the specific user application. It allows an application to directly access lower-level protocols, which means a raw socket receives unextracted packets. There is no need to provide the port and IP address to a raw socket, unlike in the case of stream and datagram sockets

B. *Opening a raw socket*

The following procedures have been used for sniffing the traffic from Ethernet card using raw socket in monitor mode. To open a socket, three things: the socket family, socket type and protocol are required. For a raw socket, the socket family is AF_PACKET, the socket type is SOCK_RAW and for the protocol, if _ether.h header file is used. To receive all packets, the macro ETH_P_ALL is used.

C. *Receive the network packet*

After successfully opening a raw socket, network packets are received using recvfrom api. recv api can also be used in place of recvfrom api but recvfrom provides additional information. So, I have used recvfrom api in my code.

D. *Extracting the Ethernet header*

After receiving network packets in buffer, the information about the Ethernet header is extracted. The Ethernet header contains the physical address of the source and destination, or the MAC address and protocol of the receiving packet. The if_ether.h header contains the structure of the Ethernet header.

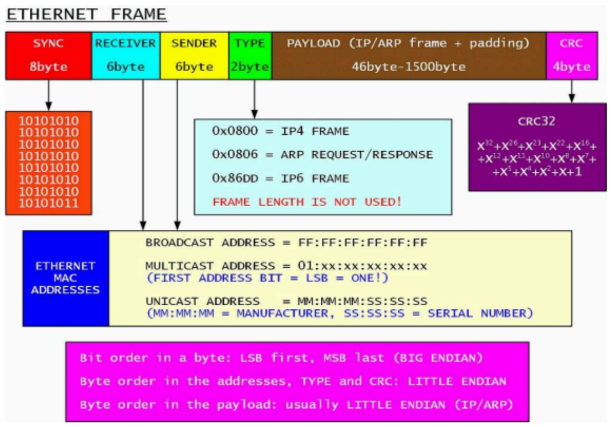
E. *Extracting the IP header*

The IP layer gives various pieces of information like the source and destination IP addresses, the transport layer protocol, etc. The structure of the IP header is defined in the ip.h header file. Now, to get the information about IP addresses, buffer pointer is incremented by the size of the Ethernet header because the IP header comes after the Ethernet header.

If you wish, you may write in the first person singular or plural and use the active voice ("I observed that ..." or "We observed that ..." instead of "It was observed that ..."). Remember to check spelling. If your native language is not English, please get a native English-speaking colleague to carefully proofread your paper.

V. IMPLEMENTATION

Implementation had been done on python. First, we will develop components and interfaces to capture packets and then implement a GUI over it using Tkinter to make it more interactive. This is implemented by putting the Ethernet interface in promiscuous mode, receiving all the packets in the local network and displaying useful information like source, destination and the protocol used


ETHERNET FRAME
SYNC 8byte
RECEIVER 6byte
SENDER 6byte
TYPE 2byte
PAYLOAD (IP/ARP frame + padding) 46byte-1500byte
CRC 4byte
10101010 10101010 10101010 10101010 10101010 10101010 10101010 10101011
0x0800 = IP4 FRAME
0x0806 = ARP REQUEST/RESPONSE
0x86DD = IP6 FRAME
FRAME LENGTH IS NOT USED!
CRC32
$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$
ETHERNET MAC ADDRESSES
BROADCAST ADDRESS = FF:FF:FF:FF:FF:FF
MULTICAST ADDRESS = 01:xx:xx:xx:xx:xx (FIRST ADDRESS BIT = LSB = ONE!)
UNICAST ADDRESS = MM:MM:MM:SS:SS:SS (MM:MM:MM = MANUFACTURER, SS:SS:SS = SERIAL NUMBER)
Bit order in a byte: LSB first, MSB last (BIG ENDIAN)
Byte order in the addresses, TYPE and CRC: LITTLE ENDIAN
Byte order in the payload: usually LITTLE ENDIAN (IP/ARP)

VI. RESULTS.

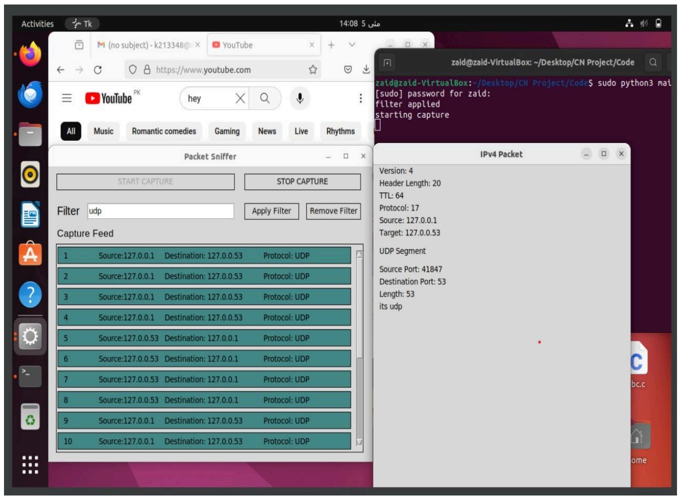
A. *Uses*

1. Detecting network intrusion attempts.
2. Monitor data in transit.
3. Learning about fields and values in different types of protocols.
4. Identify suspect content in network traffic.
5. Gather and Report Statistics.

B. *Core Features*

1. Creates a capture.pcap file to view in Wireshark.
2. Color coded packets for simplicity.
3. Inspect all fields of the packet.
4. Filter packets according to your preference

C. *GUI Outputs*



```
zaid@zaid-VirtualBox:~/Desktop/CN Project/Code$ sudo python3 mai
[sudo] password for zaid:
filter applied
starting capture
```

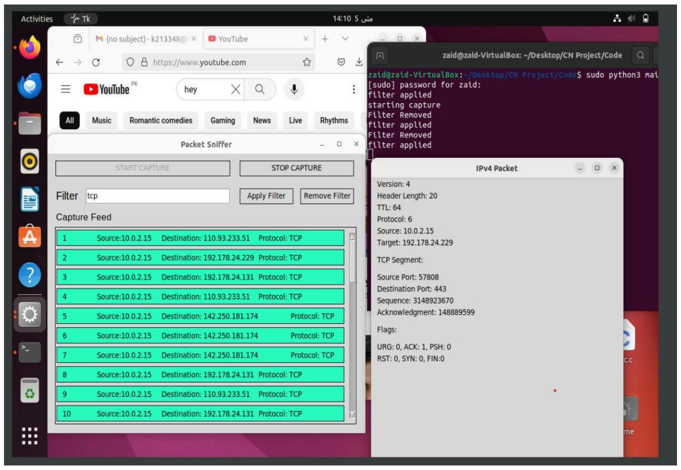
Filter: udp

```
1    Source:127.0.0.1     Destination: 127.0.0.53    Protocol: UDP
2    Source:127.0.0.1     Destination: 127.0.0.53    Protocol: UDP
3    Source:127.0.0.1     Destination: 127.0.0.53    Protocol: UDP
4    Source:127.0.0.1     Destination: 127.0.0.53    Protocol: UDP
5    Source:127.0.0.53    Destination: 127.0.0.1     Protocol: UDP
6    Source:127.0.0.53    Destination: 127.0.0.1     Protocol: UDP
7    Source:127.0.0.53    Destination: 127.0.0.1     Protocol: UDP
8    Source:127.0.0.53    Destination: 127.0.0.1     Protocol: UDP
9    Source:127.0.0.1     Destination: 127.0.0.53    Protocol: UDP
10   Source:127.0.0.1     Destination: 127.0.0.53    Protocol: UDP
```

```
Version: 4
Header Length: 20
TTL: 64
Protocol: 17
Source: 127.0.0.1
Target: 127.0.0.53

UDP Segment

Source Port: 41847
Destination Port: 53
Length: 53
its udp
```



Filter: tcp

```
1    Source:10.0.2.15    Destination: 110.93.233.51    Protocol: TCP
2    Source:10.0.2.15    Destination: 192.178.24.229   Protocol: TCP
3    Source:10.0.2.15    Destination: 192.178.24.131   Protocol: TCP
4    Source:10.0.2.15    Destination: 110.93.233.51    Protocol: TCP
5    Source:10.0.2.15    Destination: 142.250.181.174          Protocol: TCP
6    Source:10.0.2.15    Destination: 142.250.181.174          Protocol: TCP
7    Source:10.0.2.15    Destination: 142.250.181.174          Protocol: TCP
8    Source:10.0.2.15    Destination: 192.178.24.131   Protocol: TCP
9    Source:10.0.2.15    Destination: 110.93.233.51    Protocol: TCP
10   Source:10.0.2.15    Destination: 192.178.24.131   Protocol: TCP
```

```
zaid@zaid-VirtualBox:~/Desktop/CN Project/Code$ sudo python3 mai
[sudo] password for zaid:
filter applied
starting capture
Filter Removed
filter applied
Filter Removed
filter applied
```

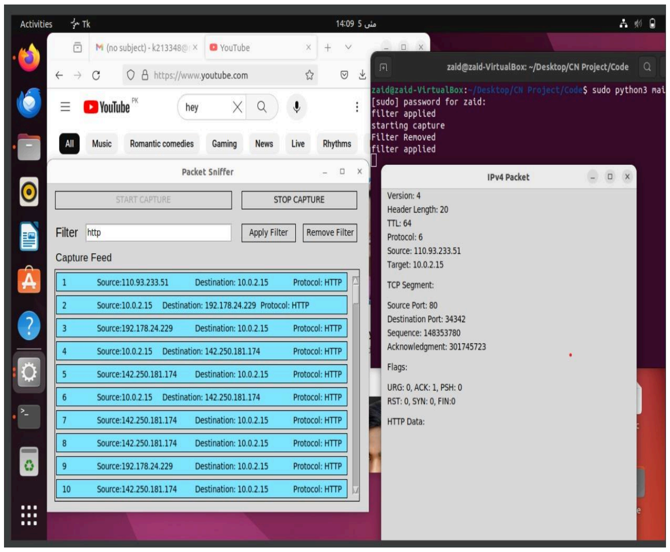
```
Version: 4
Header Length: 20
TTL: 64
Protocol: 6
Source: 10.0.2.15
Target: 192.178.24.229

TCP Segment:

Source Port: 57808
Destination Port: 443
Sequence: 3148923670
Acknowledgment: 148889599

Flags:

URG: 0, ACK: 1, PSH: 0
RST: 0, SYN: 0, FIN:0
```



Filter: http

```
1    Source:110.93.233.51      Destination: 10.0.2.15         Protocol: HTTP
2    Source:10.0.2.15     Destination: 192.178.24.229  Protocol: HTTP
3    Source:192.178.24.229     Destination: 10.0.2.15         Protocol: HTTP
4    Source:10.0.2.15     Destination: 142.250.181.174        Protocol: HTTP
5    Source:142.250.181.174    Destination: 10.0.2.15         Protocol: HTTP
6    Source:10.0.2.15     Destination: 142.250.181.174        Protocol: HTTP
7    Source:142.250.181.174    Destination: 10.0.2.15         Protocol: HTTP
8    Source:142.250.181.174    Destination: 10.0.2.15         Protocol: HTTP
9    Source:192.178.24.229     Destination: 10.0.2.15         Protocol: HTTP
10   Source:142.250.181.174    Destination: 10.0.2.15         Protocol: HTTP
```

```
zaid@zaid-VirtualBox:~/Desktop/CN Project/Code$ sudo python3 mai
[sudo] password for zaid:
filter applied
starting capture
Filter Removed
filter applied
```

```
Version: 4
Header Length: 20
TTL: 64
Protocol: 6
Source: 110.93.233.51
Target: 10.0.2.15

TCP Segment:

Source Port: 80
Destination Port: 34342
Sequence: 148353780
Acknowledgment: 301745723

Flags:

URG: 0, ACK: 1, PSH: 0
RST: 0, SYN: 0, FIN:0

HTTP Data:
```