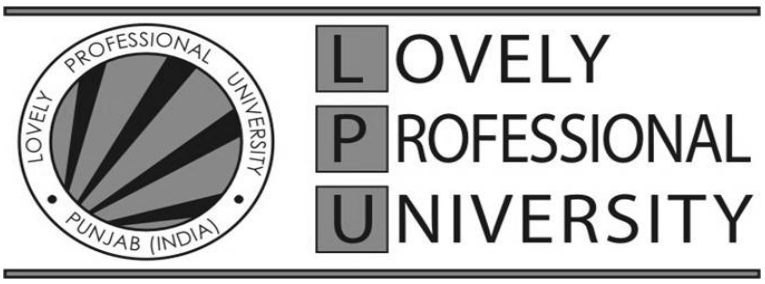
ACADEMIC TASK-3 PROJECT REPORT



NETWORK PACKET NALYSIS USING OPENS OURCE SOFTWARE: CAPTURING, TRACING, AND IDENTIFYING TRAFFIC BURSTS

SUBMITTED BY: Prateek Tripathi

REGISTRATION No.: 11908181

ROLL No.: RKE015B48

SECTION: KE015

SUBMITTED TO: Dr. Manjot Kaur

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

LOVELY PROFESSIONAL UNIVERSITY,

INDIA

TABLE OF CONTENTS

CHAPTER 1:INTRODUCTION

In today's world, networking plays a vital role in facilitating communication between individuals, organizations, and machines. With the increasing dependence on the internet, network security has become a significant concern. Network security breaches can cause severe consequences, such as the loss of sensitive information, financial loss, and reputational damage. Therefore, network administrators must be able to monitor network traffic and detect any suspicious activities.

The purpose of this project report is to demonstrate how open-source software can be used to perform network packet analysis. Specifically, this report focuses on capturing, tracing, and identifying traffic bursts using open-source software. The open-source software used in this project is Wireshark, a widely used network protocol analyser.

The report starts with an overview of network packet analysis and its importance in network security. It then describes the steps involved in capturing network packets and traces connections using Wireshark. The report also discusses how to view the contents of suspect network transactions and how to identify bursts of network traffic. Finally, the report provides some practical use cases for network packet analysis using open-source software.

Overall, this project report serves as a guide for network administrators and security professionals who want to use open-source software to perform network packet analysis. The report highlights the benefits of using open-source software, such as cost-effectiveness and community support. By following the steps outlined in this report, network administrators can improve their network security posture and detect potential threats more efficiently.

CHAPTER 2:PROJECT DETAILS

2.1. OBJECTIVE

The primary objective of this project report is to demonstrate how open-source software can be utilized to perform network packet analysis. The report aims to provide a comprehensive guide for network administrators and security professionals to capture packets from a network connection, trace connections, view the contents of suspect network transactions, and identify bursts of network traffic.

The report aims to highlight the benefits of using open-source software for network packet analysis, such as cost-effectiveness and community support. By utilizing open-source software, network administrators can

perform packet analysis without incurring high costs associated with proprietary software. Furthermore, open-source software provides access to a large community of developers who can aid in case of issues.

The project report aims to provide a step-by-step guide for network administrators and security professionals on how to perform network packet analysis using the Wireshark open-source software. The report provides detailed instructions on how to capture network packets, trace connections, view the contents of suspect network transactions, and identify bursts of network traffic.

Finally, the project report aims to provide practical use cases for network packet analysis using open-source software. The report highlights the importance of network packet analysis in network security and provides examples of how network packet analysis can be utilized to detect potential threats, such as malware infections and suspicious network activity. The report also emphasizes the need for regular network packet analysis to maintain network security and prevent data breach.

2.2. DESCRIPTION

1. Using an Open-Source Software to generate report to capture packets from a network connection.
2. Using the same software trace the network connections and looking into the suspect network by fetching the UDP/TCP data packets.
3. Using the Open-Source software to identify bursts of network traffic.

The project aimed to demonstrate how open-source software can be used for network packet analysis. The report focused on using Wireshark, an open-source network protocol analyser, to capture packets from a network connection, trace connections, view the contents of suspect network transactions, and identify bursts of network traffic.

The project report provided a step-by-step guide for network administrators and security professionals on how to perform network packet analysis using Wireshark. The report started with an overview of network packet analysis and its importance in network security, followed by detailed instructions on how to capture network packets and trace connections using Wireshark.

The project report also explained how to view the contents of suspect network transactions and identify bursts of network traffic using Wireshark. The report described how to filter network traffic and use various features of Wireshark to analyse network traffic data.

To provide practical examples of network packet analysis using open-source software, the project report highlighted some use cases. These included detecting malware infections, identifying unauthorized network traffic, and analysing network traffic patterns to identify potential performance issues. Overall, the project aimed to provide a comprehensive guide for network administrators and security professionals to perform network packet analysis using open-source software. The report highlighted the benefits of using open-source software, such as cost-effectiveness and community support, and emphasized the need for regular network

packet analysis to maintain network security and prevent data breaches. The project aimed to empower network administrators and security professionals to improve their network security posture and detect potential threats more efficiently.

2.3. SCOPE OF THE PROJECT

1. The Wireshark software will be used to capture packets from a network connection, allowing network administrators to monitor network traffic and detect potential threats. The captured packets can be analysed to identify the source and destination of the traffic and the types of protocols used.
2. Wireshark will also be used to trace network connections, allowing network administrators to identify the path of network traffic and pinpoint any bottlenecks or issues. The trace results can be analysed to identify potential performance issues and improve the overall network performance.
3. Suspect network transactions will be analysed by fetching UDP/TCP data packets using Wireshark. This allows network administrators to view the contents of suspect network transactions and identify any potential security threats, such as malware infections or unauthorized access attempts.
4. Wireshark will be used to identify bursts of network traffic, which can indicate potential performance issues or security threats. By analysing the network traffic patterns, network administrators can identify the source of the traffic and take appropriate measures to optimize the network performance or mitigate any security threats.

CHAPTER 3:SYSTEM DESCRIPTION

3.1. TARGET SYSTEM DESCRIPTION

The system targeted by Wireshark for this project is a computer network. This computer network may be a local area network (LAN), a wide area network (WAN), or any other type of network that utilizes the TCP/IP protocol. The network may consist of various devices such as servers, routers, switches, and end-user devices such as desktops, laptops, and mobile devices.

To carry out the project objectives, the Wireshark software will be installed on a device connected to the network, such as a desktop or laptop computer. This device will be used to capture and analyse network traffic data using Wireshark's features. The network traffic data will be collected from various points on the network, such as the network interface of a router, a switch, or a network tap. The project objectives will be carried out by using Wireshark to analyse the network traffic data collected from the targeted system. The Wireshark software will be used to capture packets from the network connection, trace network connections, view the contents of suspect network transactions, and identify bursts of network traffic. By analysing the network traffic data, network administrators can identify potential security threats, such as malware infections or unauthorized access attempts, and take appropriate measures to mitigate them.

Furthermore, analysing the network traffic data can help network administrators to optimize network performance and improve the overall network security posture.

CHAPTER 4: ANALYSIS REPORT

4.1. Capturing Packets

1. To capture packets, over a network, with Wireshark we need to have proper permissions which is administrator and root access in Windows and Linux systems respectively.
2. After opening Wireshark, select Capture >> Options from the main window. This will bring up the Capture Interfaces window, a shown in Figure 1.

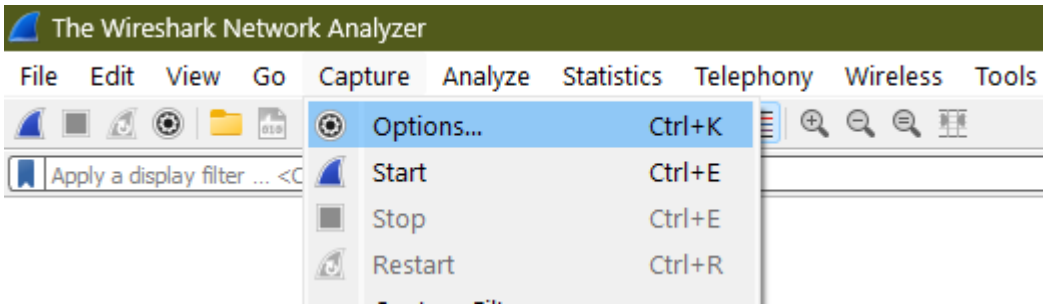


Figure 1: Checking Available Interfaces.

3. The said window will list all available interfaces. Select any of the available Interface, as shown in Figure 2. (Here I am going to go for the Wi-Fi interface)

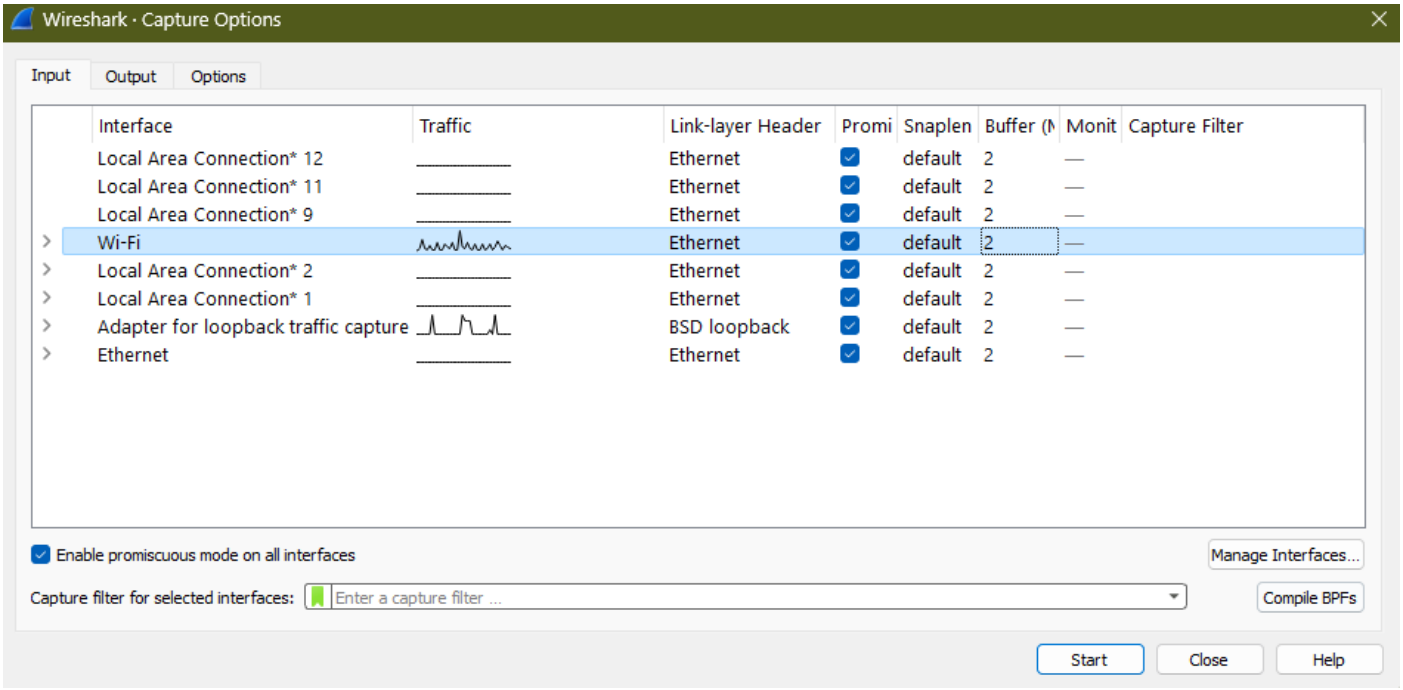
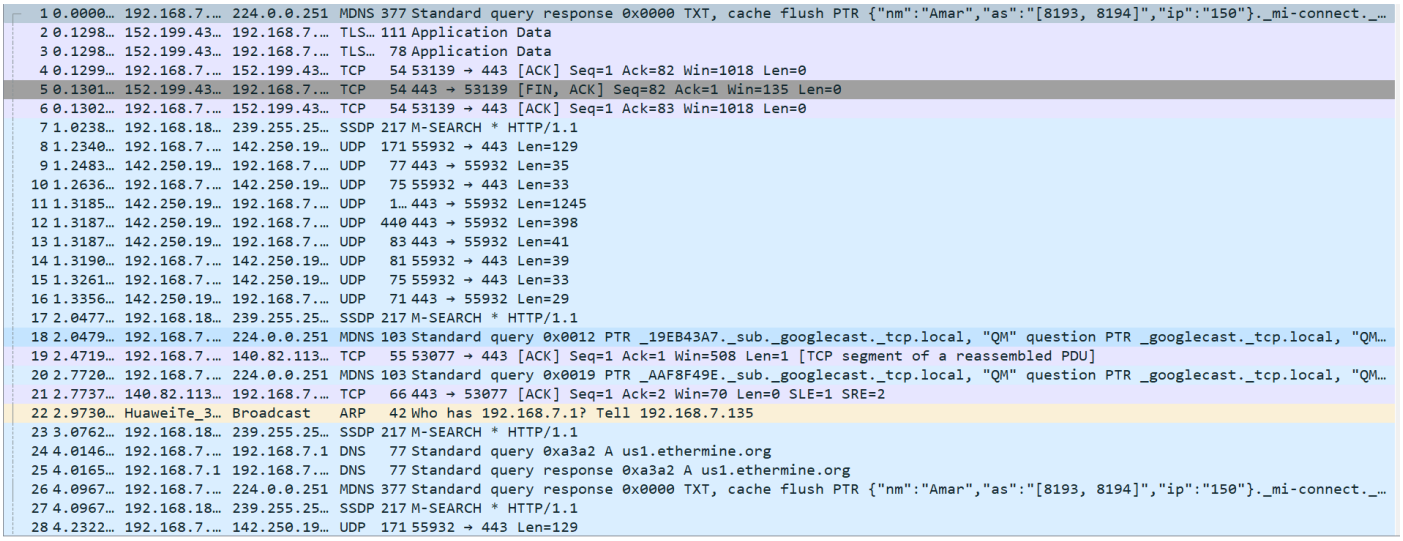


Figure 2: Selecting Desired Interface.

4. Once the network interface is selected, I clicked on the Start button to begin the capture of packets. The captures packets start getting listed on the screen, as shown in Figure 3.



1	0.0000…	192.168.7.…	224.0.0.251	MDNS	377	Standard query response 0x0000 TXT, cache flush PTR {"nm":"Amar","as":"[8193, 8194]","ip":"150"}._mi-connect._…
2	0.1298…	152.199.43…	192.168.7.…	TLS…	111	Application Data
3	0.1298…	152.199.43…	192.168.7.…	TLS…	78	Application Data
4	0.1299…	192.168.7.…	152.199.43…	TCP	54	53139 → 443 [ACK] Seq=1 Ack=82 Win=1018 Len=0
5	0.1301…	152.199.43…	192.168.7.…	TCP	54	443 → 53139 [FIN, ACK] Seq=82 Ack=1 Win=135 Len=0
6	0.1302…	192.168.7.…	152.199.43…	TCP	54	53139 → 443 [ACK] Seq=1 Ack=83 Win=1018 Len=0
7	1.0238…	192.168.18…	239.255.25…	SSDP	217	M-SEARCH * HTTP/1.1
8	1.2340…	192.168.7.…	142.250.19…	UDP	171	55932 → 443 Len=129
9	1.2483…	142.250.19…	192.168.7.…	UDP	77	443 → 55932 Len=35
10	1.2636…	192.168.7.…	142.250.19…	UDP	75	55932 → 443 Len=33
11	1.3185…	142.250.19…	192.168.7.…	UDP	1…	443 → 55932 Len=1245
12	1.3187…	142.250.19…	192.168.7.…	UDP	440	443 → 55932 Len=398
13	1.3187…	142.250.19…	192.168.7.…	UDP	83	443 → 55932 Len=41
14	1.3190…	192.168.7.…	142.250.19…	UDP	81	55932 → 443 Len=39
15	1.3261…	192.168.7.…	142.250.19…	UDP	75	55932 → 443 Len=33
16	1.3356…	142.250.19…	192.168.7.…	UDP	71	443 → 55932 Len=29
17	2.0477…	192.168.18…	239.255.25…	SSDP	217	M-SEARCH * HTTP/1.1
18	2.0479…	192.168.7.…	224.0.0.251	MDNS	103	Standard query 0x0012 PTR _19EB43A7._sub._googlecast._tcp.local, "QM" question PTR _googlecast._tcp.local, "QM…
19	2.4719…	192.168.7.…	140.82.113…	TCP	55	53077 → 443 [ACK] Seq=1 Ack=1 Win=508 Len=1 [TCP segment of a reassembled PDU]
20	2.7720…	192.168.7.…	224.0.0.251	MDNS	103	Standard query 0x0019 PTR _AAF8F49E._sub._googlecast._tcp.local, "QM" question PTR _googlecast._tcp.local, "QM…
21	2.7737…	140.82.113…	192.168.7.…	TCP	66	443 → 53077 [ACK] Seq=1 Ack=2 Win=70 Len=0 SLE=1 SRE=2
22	2.9730…	HuaweiTe_3…	Broadcast	ARP	42	Who has 192.168.7.1? Tell 192.168.7.135
23	3.0762…	192.168.18…	239.255.25…	SSDP	217	M-SEARCH * HTTP/1.1
24	4.0146…	192.168.7.…	192.168.7.1	DNS	77	Standard query 0xa3a2 A us1.ethermine.org
25	4.0165…	192.168.7.1	192.168.7.…	DNS	77	Standard query response 0xa3a2 A us1.ethermine.org
26	4.0967…	192.168.7.…	224.0.0.251	MDNS	377	Standard query response 0x0000 TXT, cache flush PTR {"nm":"Amar","as":"[8193, 8194]","ip":"150"}._mi-connect._…
27	4.0967…	192.168.18…	239.255.25…	SSDP	217	M-SEARCH * HTTP/1.1
28	4.2322…	192.168.7.…	142.250.19…	UDP	171	55932 → 443 Len=129

Figure 3: Captured Packets.

5. Once enough packets are captured, I clicked on Stop button (red square button on the top). Now we have a static packet capture to investigate.
6. The Table drawn below explains the colour coding Used by Wireshark.

Colour in Wireshark	Packet Type
Light Purple	TCP
Light Blue	UDP
Black	Packets with Errors
Light Green	HTTP Traffic
Light Yellow	Windows Specific Traffic
Dark Yellow	Routing
Dark Gray	TCP SYN, FIN and ACK Traffic

Table 1: Colour Coding in Wireshark

7. To have better insights about the packets, we can look into the I/O Stats for an entire packet capture by going to Statistics >> I/O Graph, as shown in Figure 4.

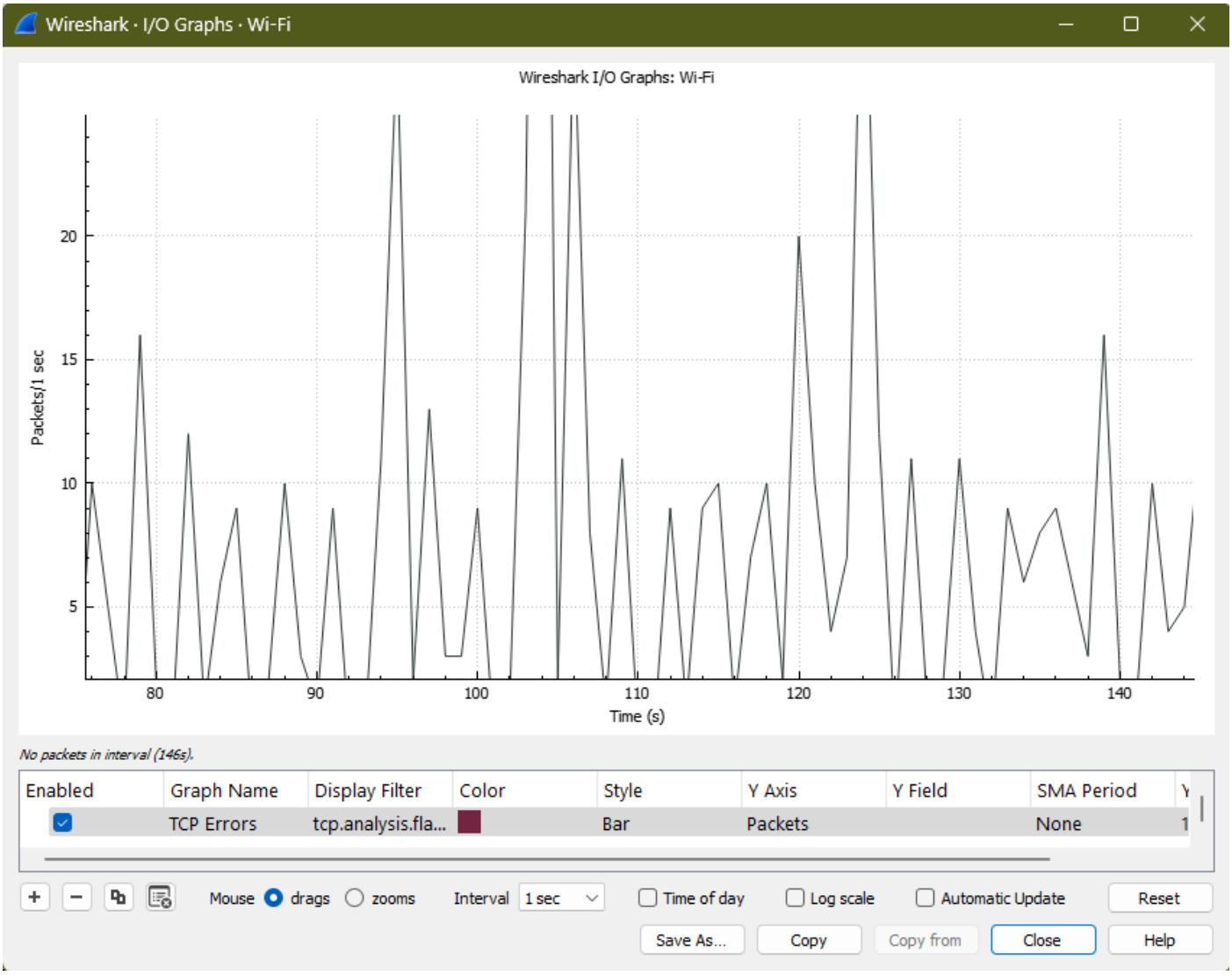


Figure 4: I/O Graphs depicting Bursts in Network Packets.

8. This I/O graph is widely used to find bursts of traffic in the network.

4.2. Tracing Network Connections

After capturing the packets in the previous step, we will start inspecting and tracing the captures network packets by filtering out the packets and inspecting them.

1. For inspecting something specific, such as the traffic a program sends when phoning home, it helps to close all other applications using the network so we can narrow down the traffic.
2. The previous step can be performed with a little accuracy by using Filters available in Wireshark. We can apply filters by typing it into the filter box at the top of the window and applying it on the captured data. For Example, if you type in TCP or DNS, the captured data will be filtered out as per the filter conditions, as shown in Figure 5.

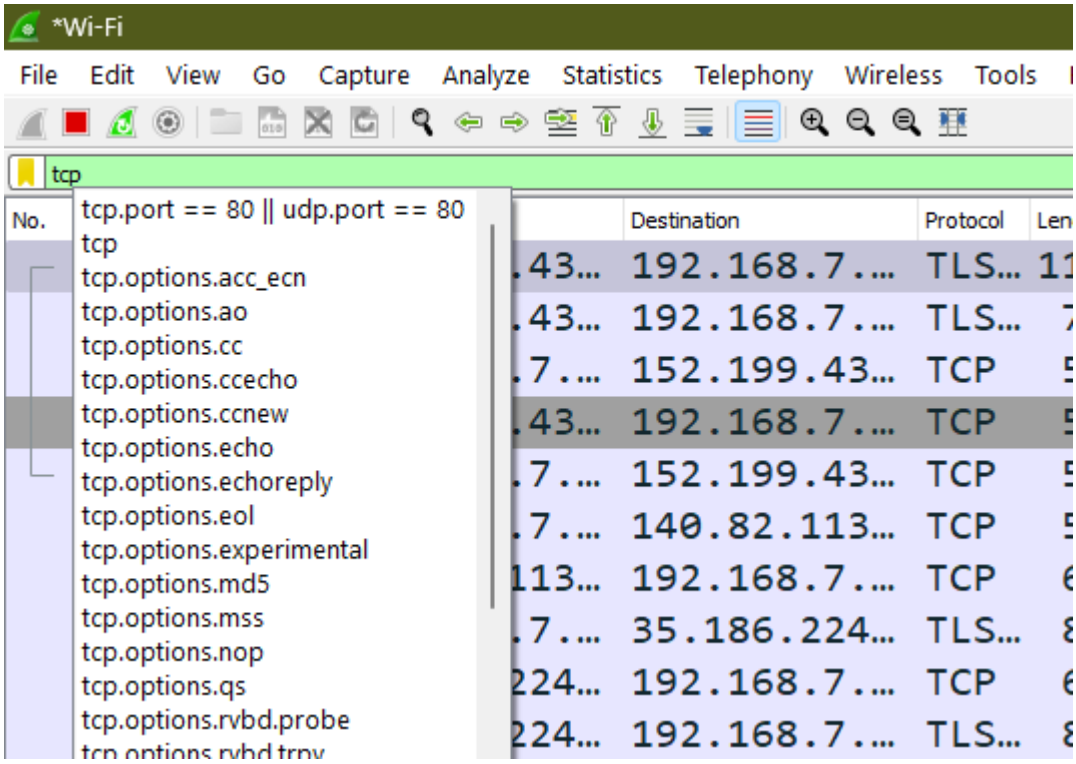


Figure 5: Filtering network packets.

3. After this, select any packet you want to inspect. You will be able to see the full conversation between the client and the server as shown in Figure 6.

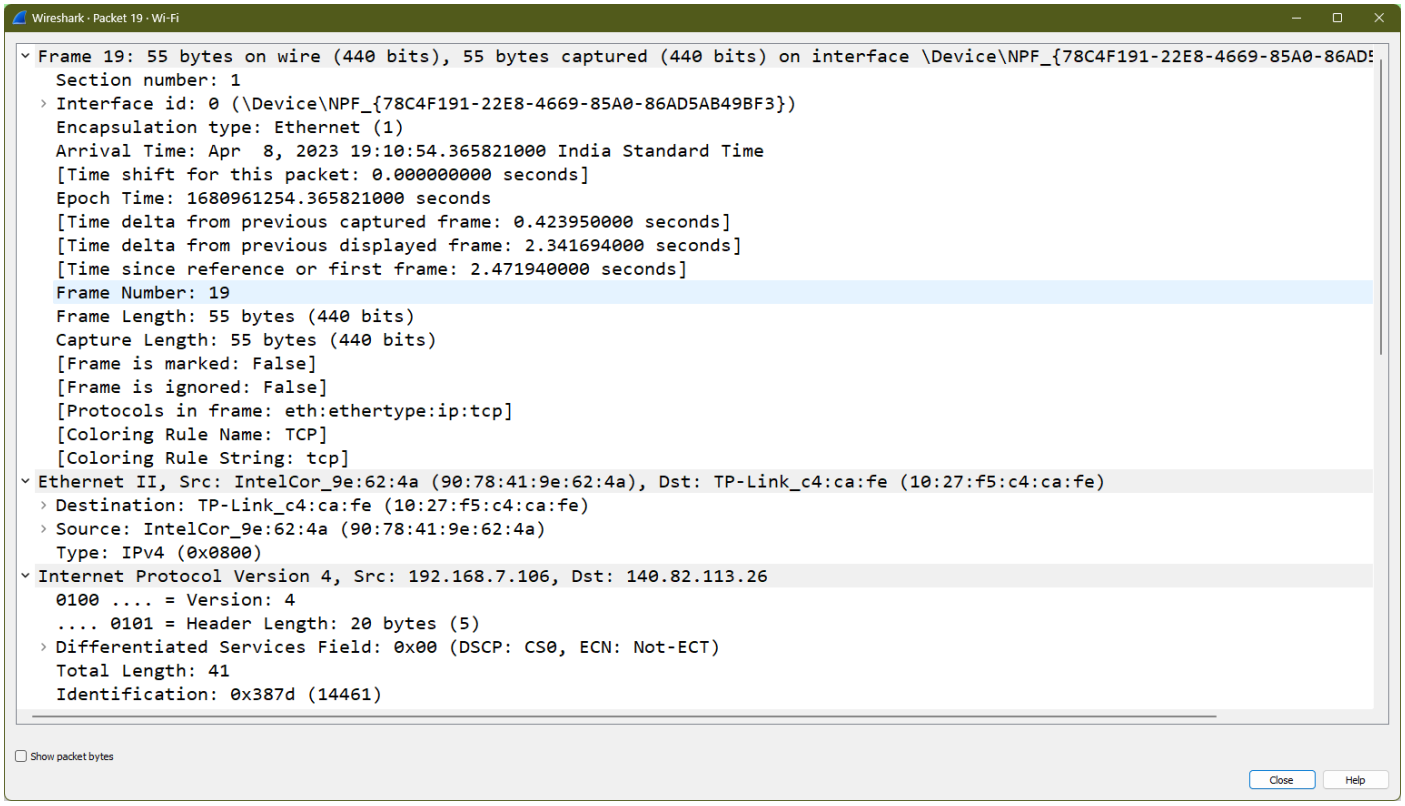


Figure 6: Inspecting and tracing Desired network packet.

4. Table 2 given below holds information about some search criteria which can be used to filter out a very specific packet trace. These filters can help in deep and accurate Tracing of network packets.

Filters	Description
ip.addr	Specifies an IPv4 address
ipv6.addr	Specifies an IPv6 address
src	Source – Where the packet came from
dst	Destination – Where the packet is going.
tcp.port == value	Filters packets to show a port of our choosing.
!(ip.src == value)	Shows all packets except those originating from 'value'
!(ipv6.dst == value)	Shows all packets except those going to the IPv6 address 'value'
ip.addre == value && ip.addr == value1	Shows both value and value1 packets.
http.request	Shows only http requests – useful when troubleshooting or visualizing Web Traffic

Table 2: Some filter conditions which can be used for accurate network tracing and inspection.

5. Valid filters are visible in GREEN colour, the invalid one appears in VIVID PINK colour, as shown in Figure 7.

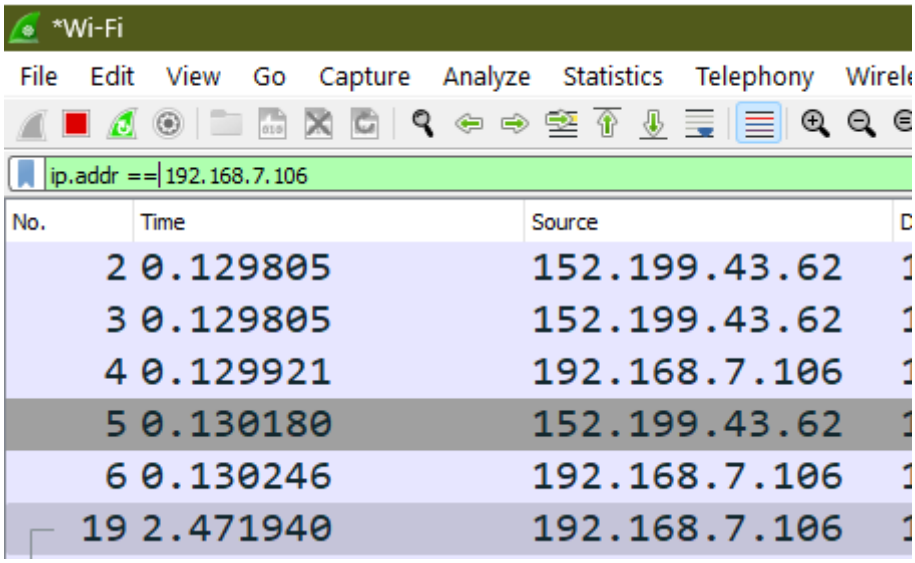


Figure 7: Implementing Filter Condition to trace and inspect a certain Network.

6. There is also an other way around, apparently we can highlight the IP address of a packet and then create a filter for it. Once we select the IP address, right click, and then select the Apply as Filter option as shown in Figure 8.

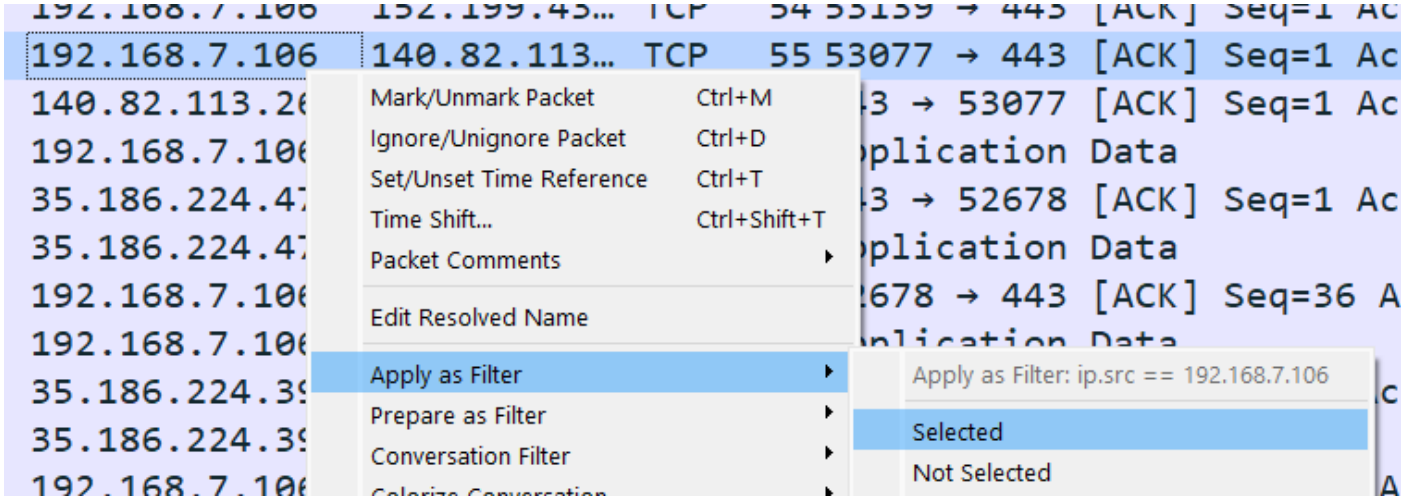


Figure 8: Another way to apply filter.

4.3. Identifying Bursts of Network Traffic

Sometimes an unexpected or sudden network traffic volume peaks and troughs based on seasonal factors. These peaks and troughs signify a burst in network traffic. The most elicit reason can be a DDoS attack.

1. After capturing packets for a required duration, Click on Statistics button from the Status bar on the Top.
2. After that click on I/O Graph. This I/O graph will show the flow of packets and the bursts in network traffic (if any), As shown in Figure 9.

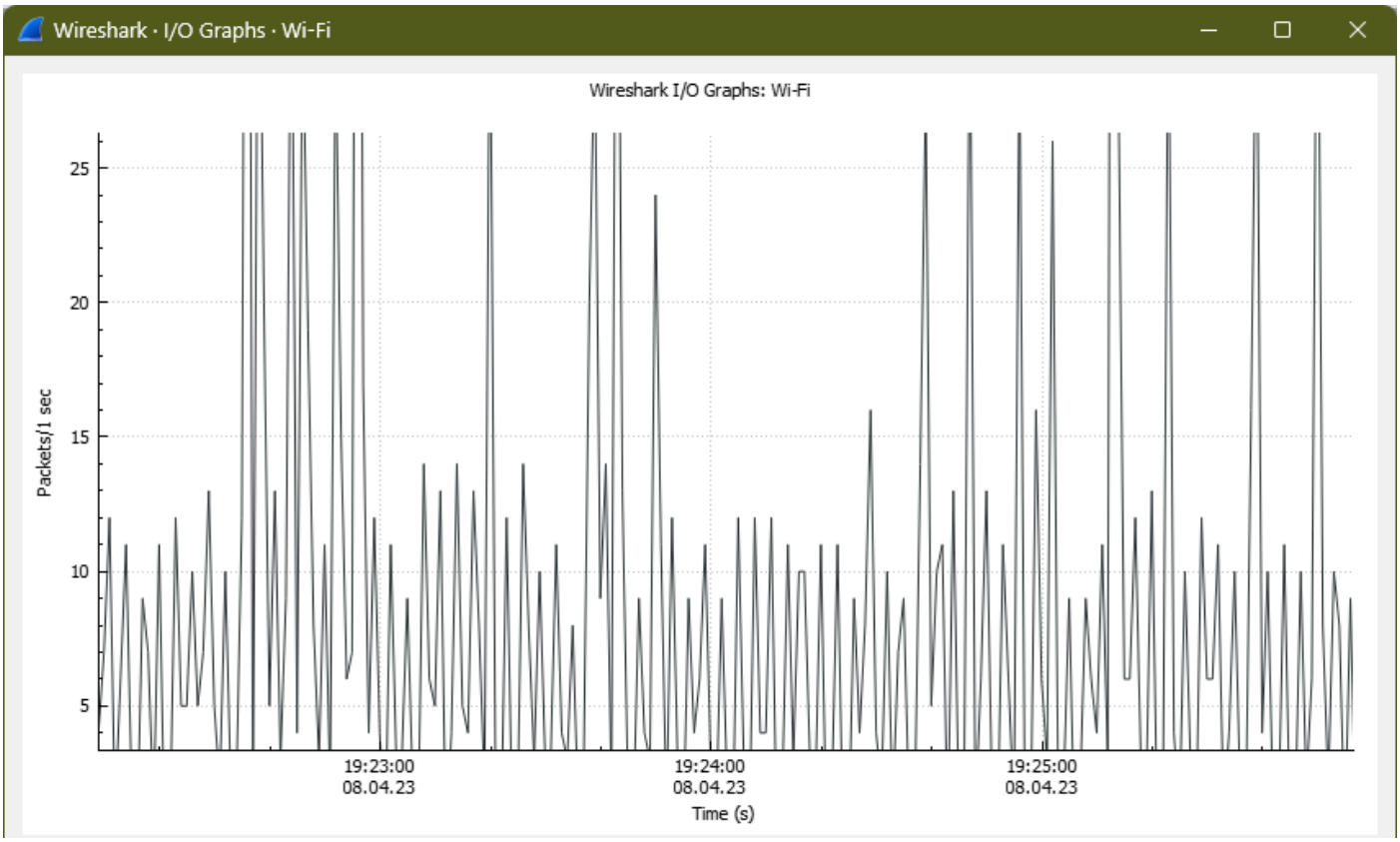


Figure 9: I/O Graph.

CHAPTER 5: REFERENCES

[1] Wireshark Documentation. https://www.wireshark.org/docs/

[2] https://protocoholic.com/2018/05/24/wireshark-how-to-identify-burst-of-traffic-in-network/

[3] CompTIA. https://www.comptia.org/content/articles/what-is-wireshark-and-how-to-use-it#:~:text=Cybersecurity%20professionals%20often%20use%20Wireshark,identify%20bursts%20of%20network%20traffic.

[4] https://support.microfocus.com/kb/doc.php?id=3892415

[5] https://pitstop.manageengine.com/portal/en/kb/articles/how-to-use-wireshark-to-capture-and-inspect-network-trace-18-6-2019

[6] R. Das and G. Tuna, "Packet tracing and analysis of network cameras with Wireshark," 2017 5th International Symposium on Digital Forensic and Security (ISDFS), Tirgu Mures, Romania, 2017, pp. 1-6, doi: 10.1109/ISDFS.2017.7916510.

[7] T. Baba and S. Matsuda, "Tracing network attacks to their sources," in IEEE Internet Computing, vol. 6, no. 2, pp. 20-26, March-April 2002, doi: 10.1109/4236.991439.

[8] Network forensics analysis using Wireshark Vivens Ndatinya, Zhifeng Xiao, Vasudeva Rao Manepalli, Ke Meng, and Yang Xiao International Journal of Security and Networks 2015 10:2, 91-106, https://doi.org/10.1504/IJSN.2015.070421

[9]