Network Packet Analyzer

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Mayank Kumar (191307)

Under the supervision of

Dr. Pankaj Dhiman
(Assistant Professor (SG))

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **" Network Packet Analyzer."** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science and Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat, is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **Dr Pankaj Dhiman (Assistant Professor (SG))**.
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

…………………………
Mayank Kumar (191307).

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

………………………………………
Dr Pankaj Dhiman
Assistant Professor
Computer Science and Engineering
Dated:

Plagiarism certificate
(should be signed by supervisor, student, and LRC officials)

ACKNOWLEDGEMENT

I owe my profound gratitude and indebtedness to our project supervisor **Dr. Pankaj Dhiman**, who took a keen interest and guided us all along in our project work titled **— Network Packet Analyser**, till the completion of our project by providing all the necessary information for developing the project. The project development helped us in research, and we got to know a lot of new things in our domain. We are really thankful to him.

(Student Signature)

Student Name:
Roll no.:

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

S. No	Title	Page No.
1	TCP : Transmission Control Protocol	1
2	UDP : User Datagram Protocol	1
3	DNS : Domain Name Server	11
4	IP : Internet Protocol	14
5	OSI : Open System Interconnection	21
6	TTL : Time To Live	21
7	PCAP : Packet Capture	24
8	DARPA : Defence Advanced Research Projects Agency	27
9	HTTP : HyperText Transfer Protocol	48

LIST OF FIGURES

LIST OF GRAPHS

LIST OF TABLES

ABSTRACT

"Network Traffic Analyzer" is a project with the main objective of designing a helping software capable of monitoring network traffic more precisely while understanding the patterns of real time. Network packets provide a wealth of information about network activities that may be used to describe overall network behaviour. System and network administrators can use network packet analyzers to collect this type of network data. This study will explain the implementation and utilisation of a network packet analyzer based on tcpdump, a common network packet sniffer tool which is widely used in the market. This fully customizable and moldable tool focuses on its flexible input and output choices so that it may be ready as well as optimally integrated into other network tools to accomplish more complex tasks like real-time or offline network intrusion detection. Database support is included as an output option in this program due to its well-known efficiency and simplicity in managing large volumes of data.

This system captures packets on the move while it is transporting to the router and helps in providing essential information on the location of their sources as well as their destinations and recognizes any anomalous/sudden spike in the network activities occurring within transmissions ensuring secure transmissions.

Multiple protocols like TCP/UDP/HTTP have been used in building our platform to ensure efficient and optimal capturing without unnecessary disturbances, lags or delays resulting in better performance achieved and the outcome of faster-saving administrator's time.

However, this software is used explicitly for user convenience and with user-friendly interfaces providing visual outputs and reports effortlessly generated at your fingertips. An effective troubleshooting and error recognition tool perfect for identifying leakages within your infrastructure while minimizing risks through enhancing security measures no wonder it makes our project a valuable and more useful asset for any of the organizations which work on these network infrastructures.

Python Programming Language along with multiple analysis techniques and methods such as packet capture, analysis flow, analysis & protocol detection further marks a solid impact on our opinion behind "Network Traffic Analyzer."

CHAPTER 1

1.1 INTRODUCTION

In the digital era, networks have become a very important and an irreplaceable component of today's organisations. Networks are used to transport or communicate data, interact with other devices and platforms, and distribute or communicate data amongst systems. With network complexity rising, monitoring network traffic is more essentially used than ever to maintain network performance and security. A network traffic analyzer is a software tool that analyses network content to give useful results and help in the monitoring of the network performance and find and help in resolving the abnormalities or security risks. A network traffic analyzer tool catches and monitors real-time network traffic data to discover the origin and destination of packets. The passage of data packets along the network devices is widely referred to as network traffic. Network traffic analysis tools can be used to monitor network utilisation, identify network issues, and identify security leaks. Network traffic analyzers may reveal network traffic trends such as the devices which communicate the most and applications which utilise the most bandwidth. The basic utilisation of the proposed application network traffic analyzer is to gain the insights and use this information and present it in a way that is clear to understand by the human in a much easier form. Network traffic analyzers intend to collect and extract the data from a variety of sources, few of which are listed as networking switches, routers, and firewalls. The analyzer can evaluate the traffic from the network traffic data after it has been recorded and stored to offer insights on network performance. One of the key methods which has been employed by network traffic analyzers to assess network traffic is packet examination.

A packet is a data unit that is used and carried over a network. Packets contain information about the targeted data's source and destination, as well as many extra details such as the interface used in this working and the time that the data was sent by the user. By observing network traffic, network traffic analyzers may give insights into the network's efficiency as well as discover irregularities or dangers to security. Packet analysis can be done at several layers, including the application, transport, network, and data link layers. The transport layer of a network is responsible for providing accurate data transfer services such as TCP and UDP. TCP along with additional reliable data transmission services are provided by the transport layer.. Network traffic analyzers, for example, can identify which apps require the most internet through studying packets at the software layer. Network traffic analyzers are capable of identifying which devices interact the most by studying packets at the network layer. Network traffic analyzers are fully capable of using different methods, such as flow estimation and behaviour evaluation, in addition to packet analysis, to provide information on network performance and discover irregularities or security concerns. Flow analysis is a method that divides packets into flows, which are groupings of packets transferred between the two devices. Network traffic analyzers are capable of providing various useful results and look into network performance and discovering abnormalities such as congestion in the network through analysing flows. Behaviour analysis is a technique to recognise abnormalities in network traffic patterns over time, such as odd spikes in network traffic.

Finally, a network traffic analyzer is a software that collects and evaluates data flowing on the network in order to give insights into network performance and discover flaws and the security concerns which arise from these flows. To give insights into network performance, network traffic analyzers are capable of evaluating packets at multiple layers, which include the application layer, transport layer, network layer, as well as data link layer. Network traffic analyzers are fully capable to utilise other methods, such as flow analysis and behaviour analysis, in parallel with the packet analysis, to give different workings about the network performance and discover irregularities or security concerns. Network traffic analyzers are a very useful tool for network managers to use and company in order to maintain network performance and security.

1.2 Problem Statement

The use of computer networks has become widespread in modern society as organisations rely on them for communication, data sharing, and resource accessibility. Nonetheless, complex networks pose significant challenges to maintaining their performance levels while ensuring security remains optimum. One significant challenge afflicting administrators concerns identifying the source or destination of network packets - which are fundamental units used to transmit data across these networks. This problem remains critical since administrators must guarantee that only authorised parties receive transmitted information free from interference or manipulation by unauthorised entities. Secondly, tracking packet sources/destinations is essential in troubleshooting network issues whilst locating performance limitations within complex systems. Finally, detecting anomalous behaviour through such tracking offers an efficient technique for recognizing probable cybersecurity threats like hacking attacks/breaches into a system's defences. Currently popular methods employ traffic analyzer tools to help locate these distinct endpoints. Unlocking the power of real time network traffic data capture and analysis, our advanced tools offer unmatched understanding into the security and performance of your organisation's network. However, there are several challenges associated with using network traffic analyzer tools, including the following:

- **Complexity**: Network traffic analyzer tools can be complex to set up and configure, requiring a significant amount of technical expertise and specialised knowledge.
- **Scalability**: As networks grow in size and complexity, the amount of data that needs to be analysed by network traffic analyzer tools can become overwhelming, leading to performance issues and data overload.
- **Security**: Network traffic analyzer tools can themselves be a security risk, as they provide a potential attack surface for malicious actors to exploit.
- **Cost**: High-end network traffic analyzer tools can be expensive, making them inaccessible to smaller organisations with limited budgets.

Therefore, there is a need for a network traffic analyzer tool that is easy to use, scalable, secure, and affordable. Such a tool would enable network administrators to identify the source and destination of network packets more effectively and efficiently, leading to improved network performance, increased security, and faster troubleshooting of network issues.

1.3 Objectives

The objective of this project is to develop a network traffic analyzer tool that can capture network traffic data in real-time and analyse it to provide insights into network performance and detect anomalies or security threats. The tool should be able to analyse packets at different levels, such as the application layer, transport layer, network layer, and data link layer, to provide insights into network performance. The primary objective of the tool is to display the source and destination of the packets to identify which devices are communicating the most and which applications are consuming the most bandwidth. To achieve this objective, the network traffic analyzer tool should have the following specific goals:

Packet Capture: Network traffic data from various sources, such as network switches, routers, or firewalls, should be able to be captured by the tool in the form of packets in real-time. A packet capture is a file that contains all packets seen by a specific network box, typically a firewall or router, during a given time period. Packet captures are a powerful and widely used tool for debugging network issues or improving visibility into attack traffic in order to tighten security (for example, by adding firewall rules to block a specific attack pattern). A network engineer may use a pcap file in conjunction with other tools, such as mtr, to troubleshoot network reachability issues.

For example, if an end user reports intermittent connectivity to a specific application, an engineer can set up a packet capture filtered to the user's source IP address to record all packets received from their device. They can then analyse that packet capture and compare it to other sources of information (for example, pcaps from the end user's side of the network path, traffic logs, and analytics) to understand the magnitude of the problem and isolate its source. Packet captures can also be used by security engineers to gain a better understanding of potentially malicious traffic. Assume an engineer notices an unexpected spike in traffic that they suspect is an attempted attack. They can use a packet capture to record traffic as it enters their network and analyse it to see if the packets are valid. If they're not, for example, if the packet payload is randomly generated gibberish, the security engineer can create a firewall rule to block traffic that looks like this from entering their network.

Packet analysis: The tool should be able to analyse packets at multiple layers, including the application layer, transport layer, network layer, and data link layer, in order to provide insights into network performance and detect anomalies or security threats. The growing popularity of online services compels security experts and law enforcement agencies to devise new methods of investigating cybercrime and obtaining evidence admissible in court. Online services send large amounts of data across communication networks in a variety of formats, the most common of which are network packets. These are groups of bits that include data as well as control information (Stallings and Case, 2012), and are generally referred to as a network layer (OSI Layer 3) protocol data unit. They represent the smallest unit of data intercepted and logged about network traffic flow traversing over packet-switched networks at a specific point in time,1 consisting of control information (source and destination IP address, error detection codes, sequencing information) and payload (intended message). A frame is a group of bits that includes data with one or more addresses and other protocol control information (Stallings and Case, 2012). In OSI Layer 4 (transport layer), the equivalent is known as a segment (or datagram). Network packets, when captured, stored, and processed efficiently, can be used in forensic investigations and may even provide admissible evidence against a suspect in a court case. Unless otherwise specified, we will use the term packet analysis throughout this paper, regardless of whether the actual content is a frame, packet, datagram, or session.

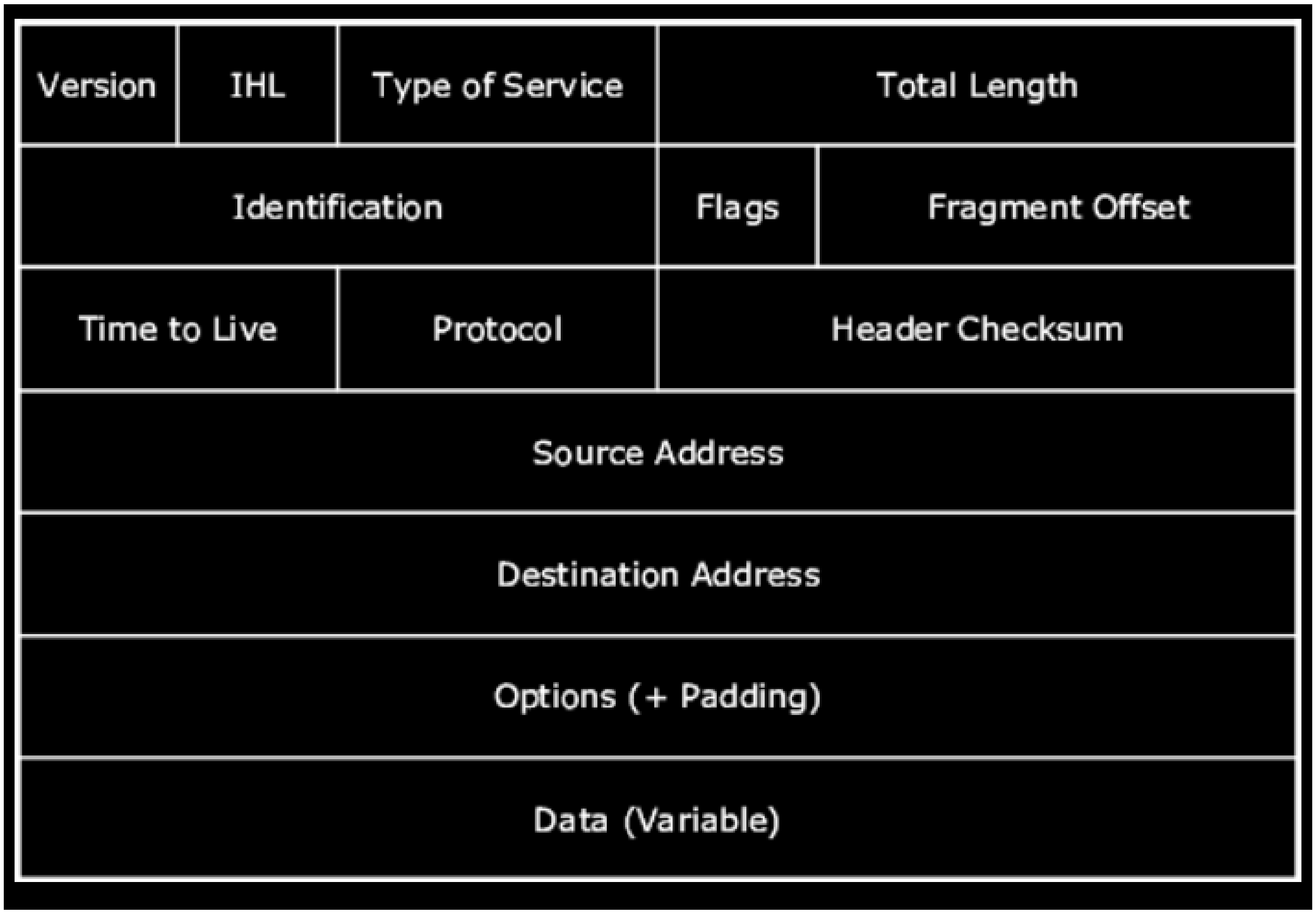


Fig. 1 IPv4 packet structure

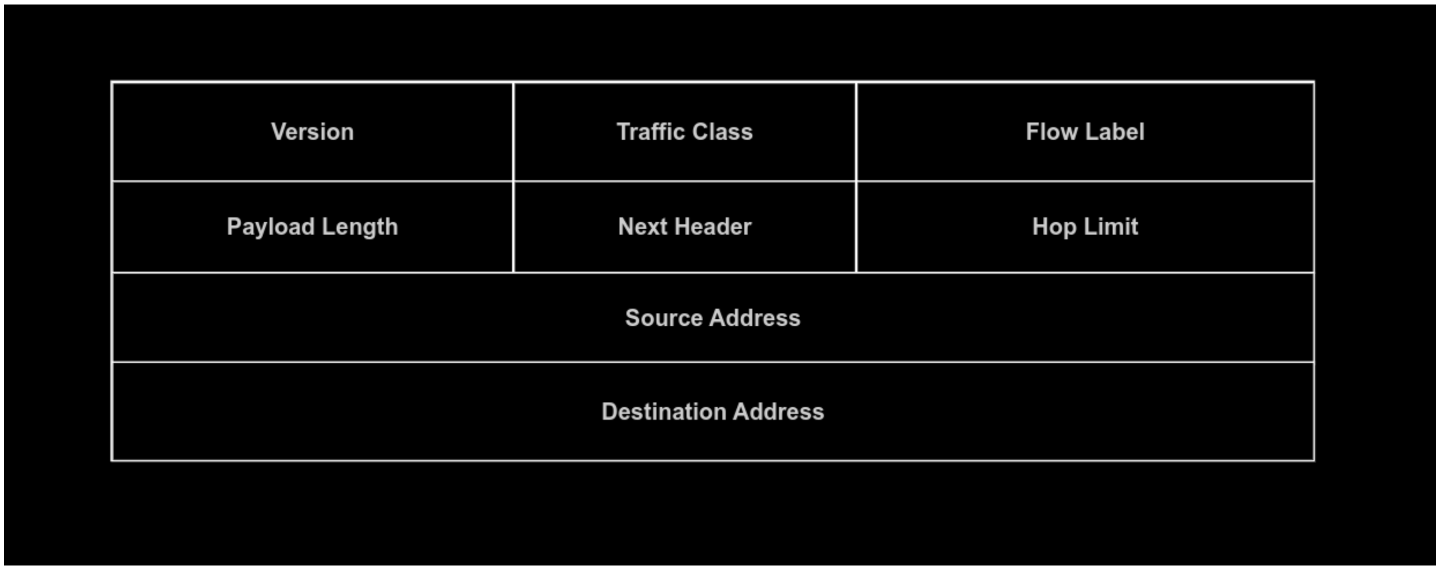


Fig. 2 IPV6 Fixed Header

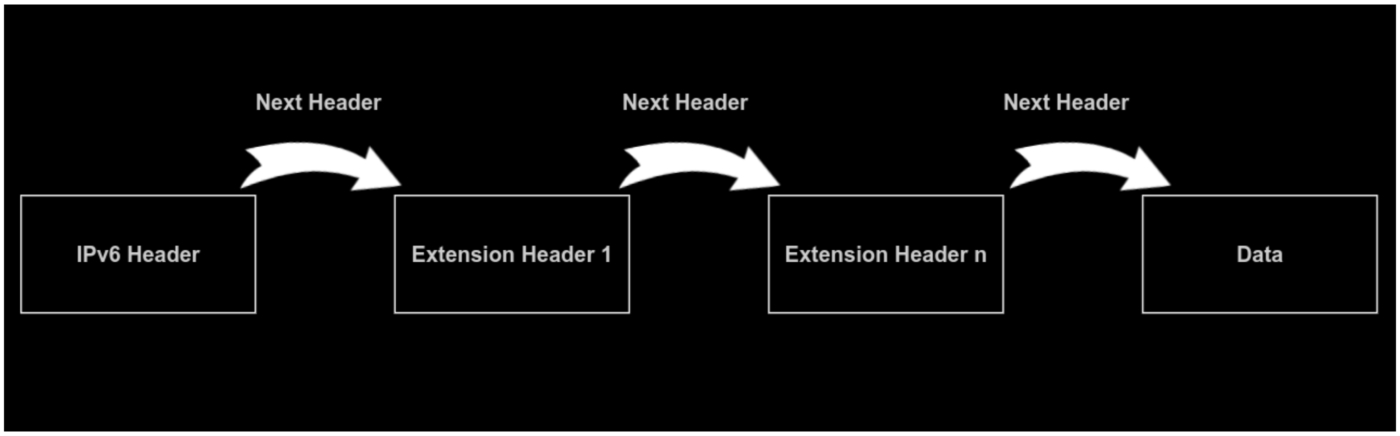


Fig. 3 IPv6 Extension Headers Connected Format

Display of packet source and destination: The tool should be able to display the source and destination of packets in order to identify which devices are communicating the most and which applications are consuming the most bandwidth**.**

Flow analysis: The tool should be able to group packets into flows, which are sequences of packets between two devices, and analyse flows to provide insights into network performance and detect anomalies, such as network congestion. According to traffic flow analysis, the following should be done:

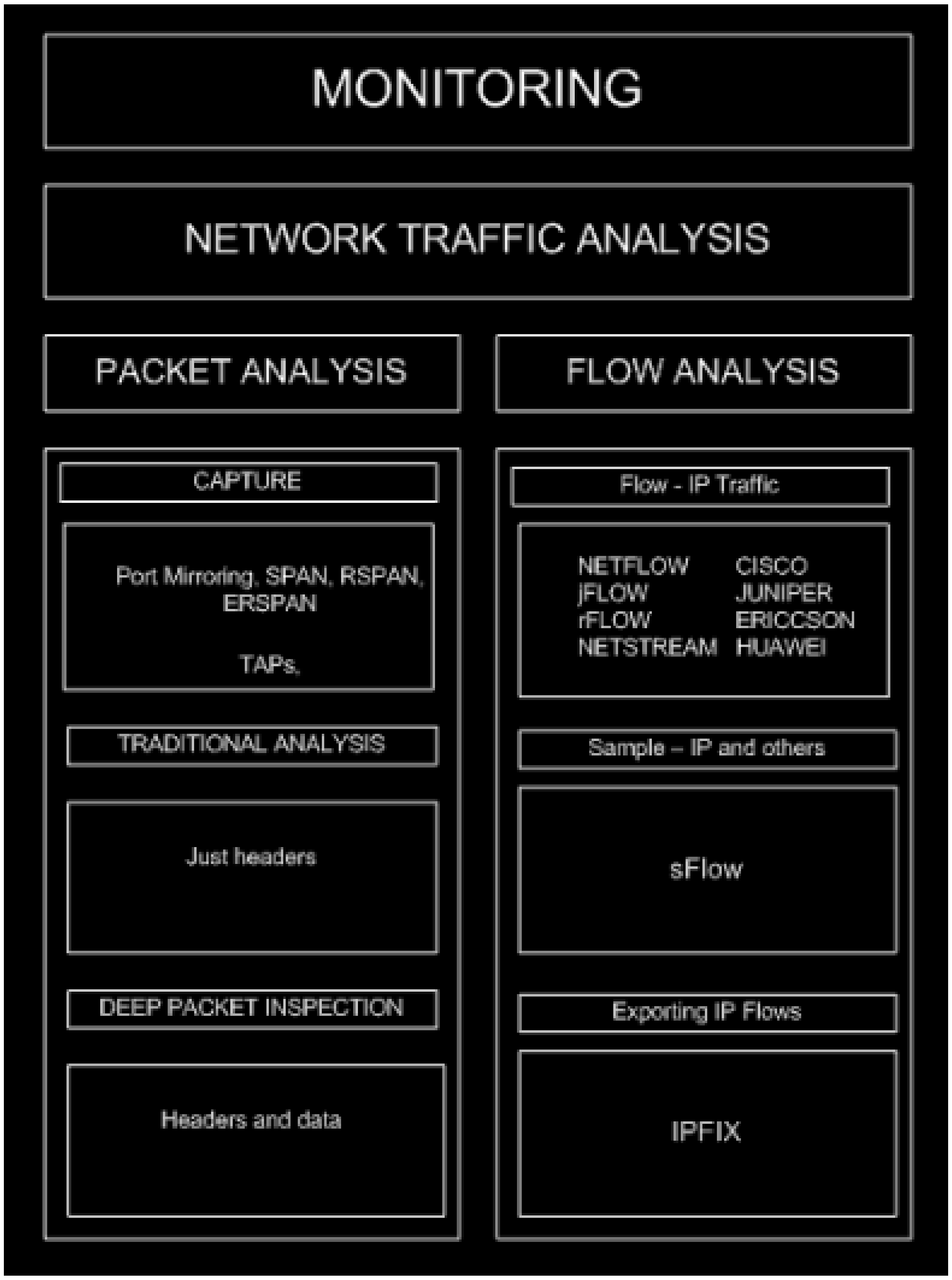


Fig. 4 Network Traffic Analysis

To assess network traffic on the basis of shared characteristics. In other words, the starting point is an abstraction known as "traffic flow," which corresponds to all traffic that shares certain common characteristics and moves from one network host to another. For example, if we consider all of the traffic that a station and a server can share, traffic that is part of the same conversation or has the same goal will be considered flow. Only the metadata is saved, not the flow itself. The idea is to use the devices involved in network traffic passing to generate information about the traffic flow or its metadata without storing the packets that comprise the traffic flow. This metadata must then be saved and reprocessed before it can be displayed with the intention of allowing analysis, such as monitoring, security, forensics, billing, and so on. The traffic flow analysis is based on a set of protocols that enable the generation, transport, storage, and preprocessing of metadata. It is important to note that these protocols do not specify how the analysis should be performed; they leave it to the tools that use metadata to accomplish their goals. NetFlow and sFlow are two protocols that represent two distinct approaches to implementing traffic flow analysis.

Behaviour analysis: Network behaviour analysis is the process of gathering and analysing enterprise network data to identify unusual entity behaviour that might indicate malicious activity. The tool should be capable of analysing network traffic patterns over time in order to detect anomalies, such as unusual spikes in network traffic. Modern network behaviour analysis solutions collect NetFlow to create a baseline for normal network operations. When a network entity deviates from the set limits for expected behaviour, network behaviour analysis records and highlights the anomalous incident for security operators. Enterprises rely on network behaviour analysis to detect automated threat warnings that security teams cannot easily detect. Using cutting-edge machine learning (ML) techniques, network behaviour analysis solutions can effectively differentiate between different types of network applications, with accuracy levels frequently exceeding 90%.

The study of network traffic characteristics is known as network behaviour analysis. Among these are Flow duration, Packet size, Packet signature, Response time, and Quantities of ACK and SYN packets in each flow. These characteristics are also analysed by the network behaviour analysis solution to classify the type of network traffic and measure network performance, among other applications. Network behaviour analysis enhances network security by tracking traffic patterns and highlighting out-of-place activity. This is a departure from 'traditional' network security operations where conventional solutions such as signature recognition, packet checking, and blocking malicious websites are used to defend networks from harm.

Instead, network behaviour analysis collates information about the way a network operates from numerous sources and leverages machine learning to identify patterns in this data. Any unexpected change in these patterns could suggest the presence of malicious activity. Consider an endpoint used by the accounting department that has never consumed more than one gigabyte of network resources in a single day. One day, this endpoint collects several gigabytes of data from a critical technology database. While it would be possible to spot this anomaly without network behaviour analysis, it could take days or even weeks to identify the problem assuming someone is going through these logs manually.

Network behaviour analysis solutions would highlight this case of data hoarding in near real-time. Advanced cybersecurity setups might even limit the endpoint's permissions and take other automated measures to prevent the exfiltration of the acquired data. Prompt detection and response to such behaviour by network behaviour analysis solutions enable security teams to deal with the threat before widespread damage can be done. The efficacy of network behaviour analysis solutions increases over time as it monitors the network and creates benchmarks for typical network behaviour. The bigger the dataset that network behaviour analysis can work with, the more easily it can identify anomalies. Any deviations identified by the solution are escalated for further action, either to another automated component or a human team.

In the post-pandemic business world, almost all internal processes are carried out online. Therefore, corporations and SMEs rely on network behaviour analysis for valuable insights to help them defend against deadly cyber attacks, particularly zero-day vulnerabilities and malware.

- **User-friendly interface**: The tool should have a user-friendly interface that allows network administrators to easily monitor network traffic and identify performance or security issues.
- **Customizable alerts**: The tool should allow users to set up customizable alerts to notify them of performance or security issues.

By achieving these specific goals, the network traffic analyzer tool can provide network administrators with the information they need to ensure network performance and security. The tool can help network administrators to identify which devices are consuming the most bandwidth, which applications are causing network congestion, and detect security threats such as malware or unauthorised access attempts. Overall, the objective of this project is to develop a powerful and user-friendly network traffic analyzer tool that can help network administrators to monitor and manage their networks effectively.

1.4 Methodology

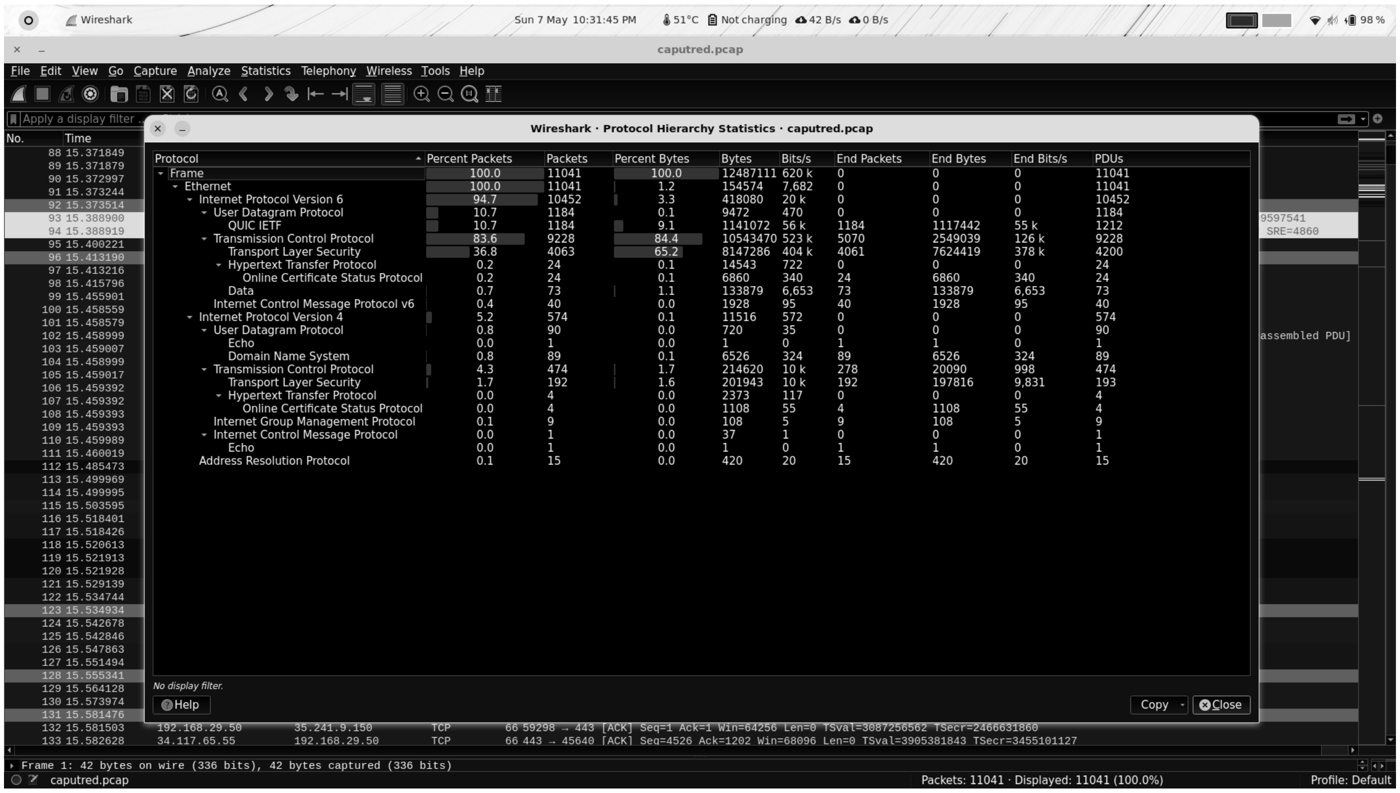
The methodology used in the project that analyses network packets to display their source and destination involves several steps, including packet capture, packet parsing, and packet analysis.

- **Packet Capture**: The first step is to capture the network traffic data in real-time. This is usually done using a network traffic monitoring tool, such as Wireshark, which captures packets as they travel across the network. Wireshark is a popular open-source network protocol analyzer tool that allows users to capture and analyse network traffic data in real time. It provides a graphical user interface (GUI) and supports a wide range of operating systems including Windows, macOS, and Linux. There are several methods for capturing packets using Wireshark:
    - **Local capture**: The first method is to capture packets on the local machine. This involves starting Wireshark and selecting the network interface to capture packets. Once the capture has started, Wireshark will display a real-time stream of captured packets in its GUI.
    - **Remote capture**: The second method is to capture packets remotely. This involves running a Wireshark remote capture service on a remote machine and connecting to it from the local machine. Once connected, Wireshark will display a real-time stream of captured packets in its GUI.
    - **Command-line capture**: The third method is to capture packets using the Wireshark command-line interface (CLI). This involves using the "tshark" command to start a packet capture and specify capture filters and other options. Once the capture has started, Wireshark will display a real-time stream of captured packets in its GUI.

Wireshark captures packets by putting the network interface into "promiscuous mode", which allows it to capture all packets that pass through the interface, not just those intended for the machine on which Wireshark is running. Once the packets have been captured, Wireshark parses them and displays them in its GUI, allowing users to analyse the network traffic in real time. Wireshark also provides a wide range of analysis features, including filtering and searching for specific packets, dissecting protocols to understand their structure and function, and displaying statistical information on network traffic such as packet count and data volume. Additionally, Wireshark allows users to export captured packets in various formats, such as **PCAP**, which can be used for further analysis or to share with other users. Overall, Wireshark is a powerful tool for capturing and analysing network traffic, providing users with a deep understanding of the data flowing across their network. With its user-friendly GUI and advanced features, Wireshark has become a widely used tool in the field of network analysis and troubleshooting.

- **Packet Parsing**: The next step is to parse the captured packets to extract the relevant information, such as the source and destination IP addresses, port numbers, and protocol types. This is typically done using a packet parsing library, such as Scapy or PyShark, which allows developers to write scripts to extract specific fields from the captured packets. Wireshark is a popular network traffic analysis tool that allows users to capture, filter, and analyse network packets. Wireshark is open-source software that is available for free, and it supports multiple operating systems, including Windows, Linux, and macOS. To parse packets in Wireshark, users can utilise various methods, including:
    - **Display Filters**: Display filters are used to filter out packets based on specific criteria, such as IP addresses, ports, and protocols. Users can create display filters using the Wireshark expression language, which allows for complex filtering based on a wide range of attributes.
    - **Protocol Hierarchy**: Wireshark provides a protocol hierarchy view that displays the protocols used in the captured packets in a hierarchical format. This view allows users to drill down into each protocol layer and analyse the details of each packet.
    - **Packet Details**: Wireshark provides a packet details view that displays the contents of each packet in a user-friendly format. This view allows users to see the raw data and decode the contents of each packet, including the source and destination IP addresses, port numbers, and protocol types.
    - **Packet Dissection:** Wireshark uses a packet dissection engine to decode the packets and extract the relevant information. The packet dissection engine can decode a wide range of protocols, including TCP, UDP, HTTP, and DNS.

Working of Wireshark on the network: When Wireshark is launched, it listens on the selected network interface for incoming packets. Once a packet is captured, Wireshark begins parsing the packet using its packet dissection engine. Wireshark then displays the parsed information in various views, including the packet details, protocol hierarchy, and packet list views. Users can apply display filters to the captured packets to filter out unwanted packets and focus on specific network traffic. Users can also create customised columns to display specific packet attributes, such as the source and destination IP addresses, in the packet list view. In addition to packet analysis, Wireshark also provides advanced features, such as protocol decoders, packet reconstruction, and packet exporting. These features allow users to perform in-depth analyses of network traffic and share their findings with other team members. In conclusion, Wireshark provides various methods to parse network packets, including display filters, protocol hierarchy, packet details, and packet dissection. It works by capturing packets on a network interface, parsing the packets using its packet dissection engine, and displaying the parsed information in various views.



Wireshark · Protocol Hierarchy Statistics · caputred.pcap

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	11041	100.0	12487111	620 k	0	0	0	11041
Ethernet	100.0	11041	1.2	154574	7,682	0	0	0	11041
Internet Protocol Version 6	94.7	10452	3.3	418080	20 k	0	0	0	10452
User Datagram Protocol	10.7	1184	0.1	9472	470	0	0	0	1184
QUIC IETF	10.7	1184	9.1	1141072	56 k	1184	1117442	55 k	1212
Transmission Control Protocol	83.6	9228	84.4	10543470	523 k	5070	2549039	126 k	9228
Transport Layer Security	36.8	4063	65.2	8147286	404 k	4061	7624419	378 k	4200
Hypertext Transfer Protocol	0.2	24	0.1	14543	722	0	0	0	24
Online Certificate Status Protocol	0.2	24	0.1	6860	340	24	6860	340	24
Data	0.7	73	1.1	133879	6,653	73	133879	6,653	73
Internet Control Message Protocol v6	0.4	40	0.0	1928	95	40	1928	95	40
Internet Protocol Version 4	5.2	574	0.1	11516	572	0	0	0	574
User Datagram Protocol	0.8	90	0.0	720	35	0	0	0	90
Echo	0.0	1	0.0	1	0	1	1	0	1
Domain Name System	0.8	89	0.1	6526	324	89	6526	324	89
Transmission Control Protocol	4.3	474	1.7	214620	10 k	278	20090	998	474
Transport Layer Security	1.7	192	1.6	201943	10 k	192	197816	9,831	193
Hypertext Transfer Protocol	0.0	4	0.0	2373	117	0	0	0	4
Online Certificate Status Protocol	0.0	4	0.0	1108	55	4	1108	55	4
Internet Group Management Protocol	0.1	9	0.0	108	5	9	108	5	9
Internet Control Message Protocol	0.0	1	0.0	37	1	0	0	0	1
Echo	0.0	1	0.0	1	0	1	1	0	1
Address Resolution Protocol	0.1	15	0.0	420	20	15	420	20	15

No display filter.

```
132 15.581503  192.168.29.50   35.241.9.150    TCP   66 59298 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3087256562 TSecr=2466631860
133 15.582628  34.117.65.55    192.168.29.50   TCP   66 443 → 45640 [ACK] Seq=4526 Ack=1202 Win=68096 Len=0 TSval=3905381843 TSecr=3455101127
Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
caputred.pcap                                  Packets: 11041 · Displayed: 11041 (100.0%)      Profile: Default
```

Fig. 5 Protocol Hierarchy

Wireshark is a powerful network analysis tool that is widely used by network engineers, security analysts, and researchers to analyse network traffic and troubleshoot network issues.

- **Packet Analysis**: Once the packets have been parsed, the next step is to analyse the extracted information to provide insights into network performance and detect anomalies. This is typically done using a network traffic analysis tool, such as tcpdump or NetFlow, which can provide statistics on packet volume, packet size, and packet frequency. Packet analysis is a crucial aspect of network traffic analysis, and Wireshark is a popular tool that offers various methods to perform packet analysis. The following are the packet analysis methods of Wireshark:

    - **Packet Filtering**: Packet filtering is the most common packet analysis method in Wireshark, and it allows users to filter out packets based on specific criteria. Users can create display filters using the Wireshark expression language, which allows for complex filtering based on a wide range of attributes, such as source and destination IP addresses, port numbers, and protocol types.

- **Protocol Decoding**: Wireshark uses a packet dissection engine to decode the packets and extract the relevant information. The packet dissection engine can decode a wide range of protocols, including TCP, UDP, HTTP, and DNS. Wireshark provides a protocol hierarchy view that displays the protocols used in the captured packets in a hierarchical format. This view allows users to drill down into each protocol layer and analyse the details of each packet.

- **Packet Reconstruction**: Wireshark offers a packet reconstruction feature that allows users to reassemble fragmented packets and view their contents. This feature is particularly useful for analysing file transfers and web page requests that are split across multiple packets.

- **Stream Analysis**:

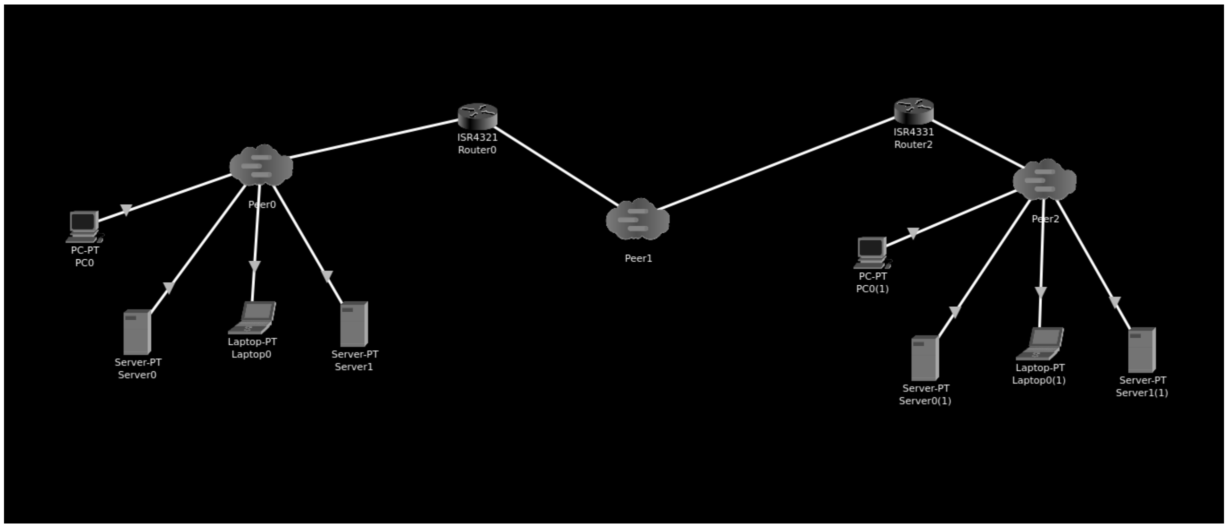


Fig. 6 Packet Travelling Path

Wireshark offers stream analysis features that allow users to analyse the flow of data between network endpoints. Users can view the stream of packets between two endpoints and analyse the various protocols used in the communication.

- **Statistical Analysis**: Wireshark provides various statistical analysis features that allow users to analyse network traffic patterns and identify anomalies. Users can view various statistics, such as the number of packets, bytes, and average packet size for each protocol used in the captured packets.

    ○ **Graphical Analysis**: Wireshark offers graphical analysis features that allow users to visualise network traffic patterns. Users can create charts and graphs to analyse network traffic trends, identify spikes in traffic, and pinpoint network performance issues.

In conclusion, Wireshark provides a comprehensive set of packet analysis methods that allow users to analyse network traffic in depth. These methods include packet filtering, protocol decoding, packet reconstruction, stream analysis, statistical analysis, and graphical analysis. With these methods, network engineers, security analysts, and researchers can identify network issues, troubleshoot problems, and improve network performance.

- **Displaying the Source and Destination**: The final step is to display the source and destination of the packets to identify which devices are communicating the most and which applications are consuming the most bandwidth. This can be achieved using a variety of techniques, such as plotting the source and destination IP addresses on a graph or using colour coding to differentiate between different types of traffic. To implement this methodology, developers can use a variety of programming languages, such as Python or Java, along with relevant libraries and tools to capture, parse, and analyse the packets. For example, they can use Scapy to extract the relevant fields from the captured packets and use Pandas to perform statistical analysis on the extracted data. Additionally, developers can use visualisation libraries such as Matplotlib or Seaborn to create graphs and charts that display the source and destination of the packets in an easily understandable format. Overall, the methodology used in the project that analyses network packets to display their source and destination involves capturing network traffic data in real-time, parsing the packets to extract relevant information, analysing the data to provide insights into network performance, and finally, displaying the source and destination of the packets in a visually appealing manner.

Wireshark is a powerful tool for network traffic analysis that can capture and analyse network packets in real-time or from a saved capture file. One of the most critical pieces of information that Wireshark provides is the source and destination IP addresses of the packets. Wireshark captures network packets by putting the network interface card (NIC) into promiscuous mode. In this mode, the NIC captures all the packets that it sees on the network, including those not intended for the host machine. This allows Wireshark to capture all the packets that pass through the network interface, including those sent between other devices on the network. When Wireshark captures a packet, it analyses the packet header to determine the source and destination IP addresses. The IP header contains the source and destination IP addresses, along with other important information about the packet, such as the protocol used, the time to live (TTL), and the length of the packet. Wireshark uses the IP protocol field in the packet header to determine the protocol used in the packet. If the protocol is TCP or UDP, Wireshark can determine the source and destination port numbers, which are also important for identifying the applications or services that are generating or receiving the packets. Once Wireshark has identified the source and destination IP addresses, it can use this information to filter and analyse the packets. Users can apply filters based on the source and destination IP addresses, allowing them to focus on specific network traffic flows. Users can also sort and group packets based on the source and destination IP addresses, making it easier to identify patterns in the network traffic. In summary, Wireshark captures network packets in promiscuous mode, analyses the packet header to determine the source and destination IP addresses, and uses this information to filter and analyse the packets. The source and destination IP addresses are critical pieces of information that allow users to identify the network devices generating or receiving the traffic and focus their analysis on specific traffic flows.

CHAPTER 2

LITERATURE SURVEY

Network packet analysis is the capture and analysis of network packets in order to gain insight into network traffic behaviour. Network packet analyzers are indispensable tools for network engineers and security analysts because they allow them to diagnose and troubleshoot network issues as well as identify security threats. In this literature review, we will examine the existing literature on network packet analysis and the tools used to perform it.

Capturing and analysing packets at various layers of the network stack is what network packet analysis eThe OSI model defines seven network stack layers, and packet analysis can take place at any of these layers. layers. The data link, network, and transport layers are the most commonly examined layers.

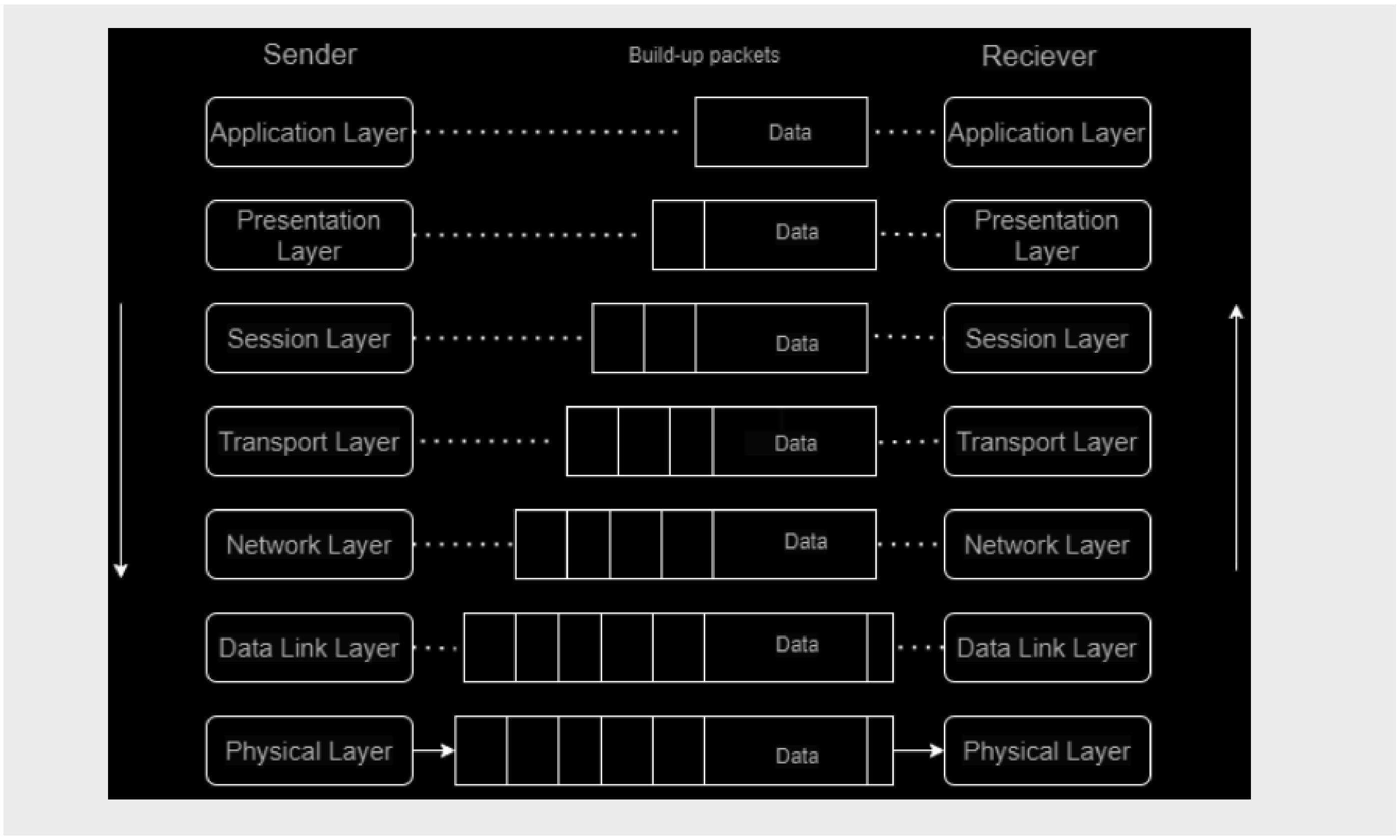


Fig. 7 OSI Model

The Physical Layer:

In terms of network security and hardware support, the physical layer in the OSI model is the fundamental level for the entire network. It defines the hardware needed to connect computers, including wires, devices, frequencies, and pulses. The data is stored in bits and transferred between devices via the nodes of the physical layer. Consider how critical the physical layer is to network security as you learn about it in the OSI model. The physical layer is necessary for network hardware visibility. Layer 1 of the OSI model is frequently ignored by today's software solutions. Because layer 1 devices cannot be identified, rogue devices may be implanted in the hardware, posing a security risk to the entire network. Such bad actors are detected and eliminated by the physical layer. To ensure OSI Model Functions in the Physical Layer, the layer also includes a separate security procedure.

1. Bit Representation:

The physical layer (Layer 1) in the OSI model is in charge of transmitting individual bits from one node to another over a physical medium. In the case of electrical signals, it specifies how to encode bits, such as how many volts should represent a 0 bit and a 1 bit.

2. Data Transfer Rate:

In the OSI model, the function of Physical Layer maintains the data rate. The number of bits sent per second is defined as the data rate. It is influenced by a variety of factors, including:
Bandwidth: The physical limitation of the underlying medium.
The number of encoding signalling levels.
The frequency with which information is received incorrectly as a result of noise.

3. Synchronisation

Bit synchronisation is one of the functions of the physical layer in the OSI model. Bit synchronisation exists between the sender and the receiver. This is accomplished by including a clock. The sender and receiver are both controlled by this timepiece. This synchronisation method produces bit-level synchronisation.

4. Interface

The transmission interface between devices and the transmission medium is defined by the physical layer of the OSI model. PPP, ATM, and Ethernet are the three most commonly used frames on the physical interface. When considering the standards, it is common, but not required, to divide the physical layer into two parts:

- The Physical Medium (PM) layer is the lowest sublayer of the physical layer..
- Transmission Convergence (TC) layer: The physical layer's highest sublayer.

5. Configuration of Lines

The physical layer in OSI models is in charge of connecting devices to the medium or line configuration. Line configuration, also known as a connection, is the method of connecting two or more devices to a link. A dedicated link connects two devices directly. A device can be a computer, a printer, or any other type of data-transmitting and data-receiving device.

6. Topologies

The OSI model's physical layer specifies how various computing devices in a network should be connected to one another. A network topology is the configuration of computer systems or network devices that are linked together. A network's topology can define both its physical and logical aspects. Device connectivity necessitates the use of Mesh, Star, Ring, and Bus topologies.
7. Modes of Transmission

In the OSI model, the physical layer specifies the transmission direction between two devices. The method of transferring data from one device to another is referred to as transmission mode. The physical layer determines the direction of data travel required to reach the receiver system or node in the OSI model. Transmission modes are classified into three categories:

- The simplex mode
- Half-duplex mode
- Full-duplex mode

Physical topologies:
Mesh Topology is the first step.

It is a highly secure device connection in which every device is linked to every other device in the network via links. A dedicated point-to-point connection is present, which is complex to form.

Star Topology
All devices in this type of device connection are linked to a central hub via a dedicated point-to-point connection. It is easy to install but has no fault tolerance.

Bus Topology

All of the devices in this configuration are linked by a single backbone cable, which is less expensive and easier to replace.

There are several other ways in which the physical layer in the OSI model functions, such as end-to-end configuration.

Data link layer analysis:

Data link layer analysis is a critical aspect of network packet analysis, which involves analysing packets at the Ethernet layer of the OSI model. The Ethernet protocol is the most widely used data link layer protocol, and packet analysis at this layer involves analysing Ethernet frames. The most commonly analysed attributes at the data link layer are the source and destination MAC addresses and the Ethernet frame type. The source MAC address identifies the device that sent the frame, while the destination MAC address identifies the device that is intended to receive the frame. The Ethernet frame type indicates the type of data carried in the frame, such as an IP packet or an ARP request. The data link layer analysis is useful for various network troubleshooting and security tasks, such as identifying network devices and tracking network activity. For example, analysing the source and destination MAC addresses can help identify devices on the network while analysing the Ethernet frame type can help identify the type of traffic being carried on the network. Data link layer analysis can be performed using various tools, such as Wireshark, tcpdump, and Tshark. These tools allow users to capture and analyse Ethernet frames and provide various features, such as filtering, protocol decoding, and statistical analysis. Wireshark is an open-source packet analyzer that is widely used for data link layer analysis. Wireshark offers various features, such as packet filtering, protocol decoding, packet reconstruction, stream analysis, statistical analysis, and graphical analysis. Wireshark provides a graphical user interface that allows users to analyse Ethernet frames in a user-friendly manner. Tcpdump is a command-line packet analyzer that is commonly used for data link layer analysis. Tcpdump offers features such as packet filtering and protocol decoding and is widely used in network troubleshooting and analysis. Tcpdump provides a command-line interface that allows for automation and scripting.

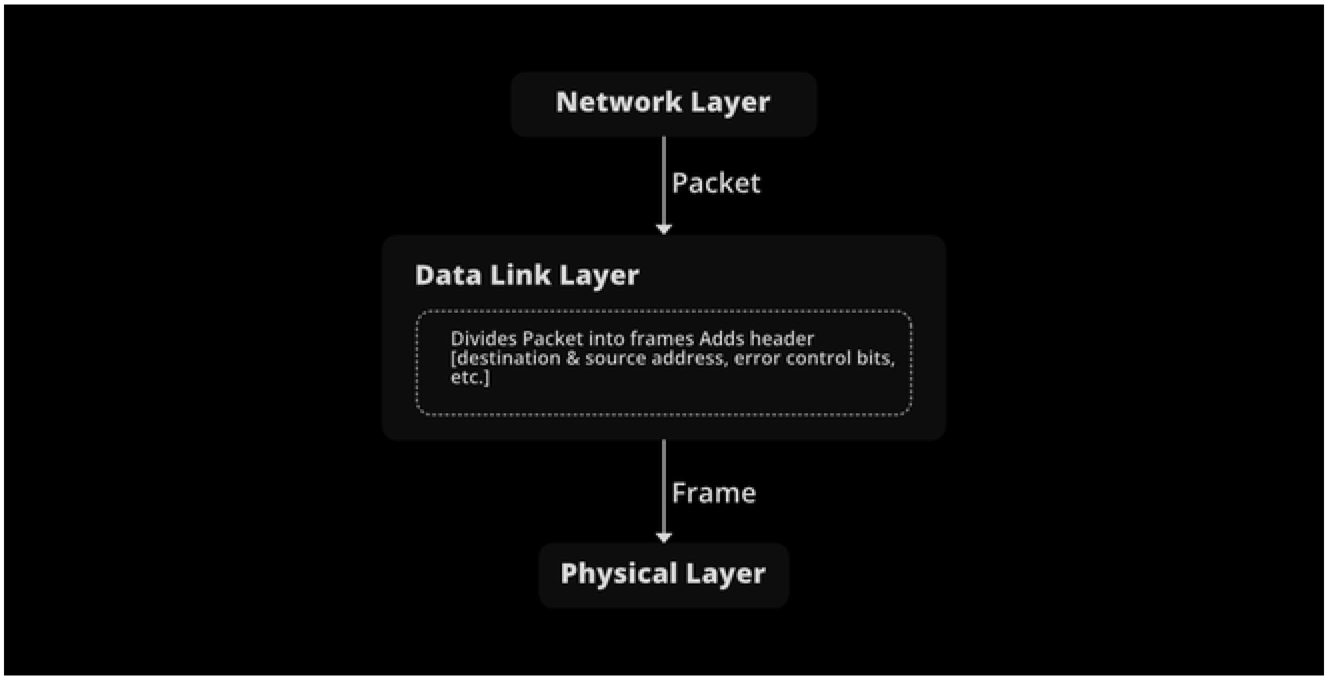

Network Layer
Packet
Data Link Layer
Divides Packet into frames Adds header [destination & source address, error control bits, etc.]
Frame
Physical Layer

Fig. 8 Data Link Layer

Tshark is a command-line version of Wireshark that allows users to capture and analyse Ethernet frames from a terminal window. Tshark offers most of the features of Wireshark, and its command-line interface allows for automation and scripting. In conclusion, data link layer analysis is a critical aspect of network packet analysis that involves analysing Ethernet frames at the data link layer. The most commonly analysed attributes at this layer are the source and destination MAC addresses and the Ethernet frame type. Various tools, such as Wireshark, tcpdump, and Tshark, are available for data link layer analysis and provide various features, such as filtering, protocol decoding, and statistical analysis. Data link layer analysis involves analysing packets at the Ethernet layer. Ethernet is the most widely used data link layer protocol, and packet analysis at this layer involves analysing Ethernet frames. The most commonly analysed attributes at the data link layer are the source and destination MAC addresses and the Ethernet frame type.

Network layer analysis:

Network layer analysis is a key aspect of network packet analysis that involves capturing and analysing network packets at the network layer of the OSI model. The network layer is the third layer of the OSI model, and it is responsible for routing packets between different networks. The most commonly used protocol at the network layer is the Internet Protocol (IP), and packet analysis at this layer involves analysing IP packets. IP packets contain information such as the source and destination IP addresses, the protocol type, and the time to live (TTL). The source and destination IP addresses are the most important attributes of an IP packet. The source IP address identifies the sender of the packet, while the destination IP address identifies the intended recipient of the packet. These addresses are critical for routing the packet through the network, and network layer analysis often involves examining the source and destination IP addresses to identify the origin and destination of network traffic. The protocol type is another important attribute of an IP packet. The protocol type identifies the higher-level protocol that is encapsulated within the IP packet.

For example, if the protocol type is TCP, it indicates that the packet contains a TCP segment. If the protocol type is UDP, it indicates that the packet contains a UDP datagram. Network layer analysis often involves examining the protocol type to determine the type of traffic being carried by the packet. The time to live (TTL) is an attribute of an IP packet that limits the lifetime of the packet. The TTL value is decremented by one each time the packet is forwarded by a router, and if the TTL value reaches zero, the packet is discarded. The TTL attribute is used to prevent packets from circulating indefinitely in the network. Network layer analysis often involves examining the TTL value to determine the number of hops taken by the packet and to identify potential routing issues. There are various tools available for network layer analysis, ranging from open source tools such as Wireshark to commercial tools such as SolarWinds Network Performance Monitor. These tools offer various features and capabilities, such as packet filtering, protocol decoding, and statistical analysis. Commercial packet analyzers, such as SolarWinds Network Performance Monitor and Riverbed SteelCentral Packet Analyzer, offer advanced features such as application performance monitoring, network traffic analysis, and security threat detection. These tools are designed for enterprise-level network analysis and are often used by network engineers and security analysts. Network layer analysis is an essential aspect of network packet analysis that involves capturing and analysing network packets at the network layer of the OSI model. The network layer is responsible for routing packets between different networks, and network layer analysis involves examining attributes such as the source and destination IP addresses, the protocol type, and the TTL attribute. There are various tools available for network layer analysis, ranging from open source tools such as Wireshark to commercial tools such as SolarWinds Network Performance Monitor. The choice of tool depends on the specific requirements of the network analysis and the level of expertise of the user. Network layer analysis involves analysing packets at the IP layer. The IP protocol is the most commonly used network layer protocol, and packet analysis at this layer involves analysing IP packets. The most commonly analysed attributes at the network layer are the source and destination IP addresses, the IP protocol type, and the time to live (TTL).

Transport layer analysis:

Transport layer analysis involves analysing packets at the TCP or UDP layer. TCP and UDP are the most commonly used transport layer protocols, and packet analysis at this layer involves analysing TCP or UDP packets. The transport layer is responsible for providing reliable data transmission and flow control between applications running on different hosts. TCP (Transmission Control Protocol) is a connection-oriented transport layer protocol that provides reliable data transmission. TCP analysis involves analysing TCP segments, which are the units of data transmitted over a TCP connection. The most commonly analysed attributes at the TCP layer are the source and destination port numbers, the sequence and acknowledgement numbers, the TCP flags, the window size, and the options. The source and destination port numbers are used by the TCP layer to identify the sending and receiving applications. The port numbers are 16-bit values that range from 0 to 65535. Well-known port numbers (0-1023) are reserved for standard services such as HTTP, FTP, and Telnet, while the remaining port numbers (1024-65535) are available for dynamic allocation by applications.

TCP uses sequence and acknowledgement numbers to provide reliable data transmission. The sequence number is a 32-bit value that represents the byte offset of the first byte of the segment. The acknowledgement number is a 32-bit value that represents the next expected byte from the other end of the connection. TCP uses flags to control the behaviour of the connection. The most commonly used TCP flags are SYN, ACK, FIN, RST, and URG. The SYN flag is used to initiate a connection, the ACK flag is used to acknowledge data received, the FIN flag is used to terminate a connection, the RST flag is used to reset a connection, and the URG flag is used to indicate urgent data. TCP uses a sliding window mechanism to control the flow of data. The window size is a 16-bit value that represents the maximum amount of data that can be sent without receiving an acknowledgement. UDP (User Datagram Protocol) is a connectionless transport layer protocol that provides unreliable data transmission. UDP analysis involves analysing UDP datagrams, which are the units of data transmitted over a UDP connection. The most commonly analysed attributes at the UDP layer are the source and destination port numbers and the length. The source and destination port numbers are used by the UDP layer to identify the sending and receiving applications. The port numbers are 16-bit values that range from 0 to 65535. The length field in the UDP header represents the length of the datagram, including the header and the data.

Transport layer analysis is a critical aspect of network packet analysis, as it provides insight into the behaviour of the applications running on the network. TCP analysis involves analysing TCP segments, which provide reliable data transmission and flow control, while UDP analysis involves analysing UDP datagrams, which provide fast and lightweight data transmission. The most commonly analysed attributes at the transport layer are the source and destination port numbers, the sequence and acknowledgement numbers, the TCP flags, and the window size for TCP analysis, and the source and destination port numbers and the length for UDP analysis. The choice of tool for transport layer analysis depends on the specific requirements of the network analysis and the level of expertise of the user. Transport layer analysis involves analysing packets at the TCP or UDP layer. TCP and UDP are the most commonly used transport layer protocols, and packet analysis at this layer involves analysing TCP or UDP packets. The most commonly analysed attributes at the transport layer are the source and destination port numbers and the TCP or UDP flags.

Packet analysis tools:

Packet analyzers, also known as packet sniffers, monitor every packet (the smallest unit of information) that flows through a network. They are beneficial to network administrators in terms of traffic logging, monitoring, and analysis. They are also useful in determining the root cause of any network problem. These tools are available as both software and hardware. The software option has become more popular due to its convenience, but both are effective in dealing with troubleshooting and are connected to a network where they store the information they receive.

Packet analyzer tools assist IT professionals in simplifying and speeding up the troubleshooting process for both new and existing information travelling from sender to receiver by decoding the packets. Simply put, they provide visibility into a previously invisible travelling process. These tools help to clarify current issues and anticipate future problems. This improves the end-user experience, whether they are chatting online, gaming, streaming video, or shopping.

There are numerous tools available for network packet analysis, ranging from open-source tools like Wireshark to commercial tools like SolarWinds Network Performance Monitor. These tools include features such as packet filtering, protocol decoding, and statistical analysis.

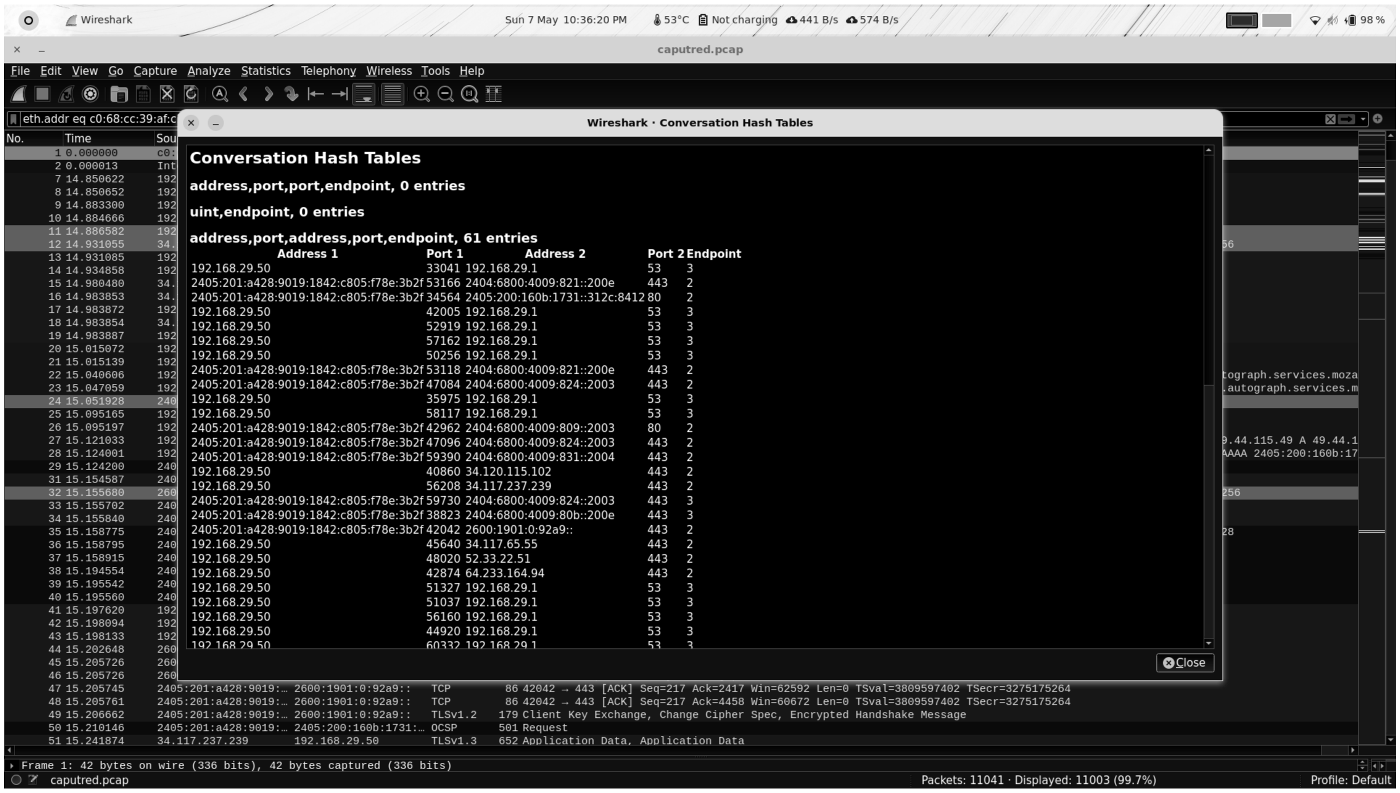
Wireshark:

Wireshark is an open-source packet analyzer that is free to use. It is used for network troubleshooting, analysis, the creation of software and communications protocols, and teaching. Due to trademark difficulties, the project's name was changed to Wireshark in May 2006. Wireshark is cross-platform, implementing its user interface using the Qt widget toolkit in current versions and capturing packets with pcap. It operates on Linux, macOS, BSD, Solaris, certain other Unix-like operating systems, and Microsoft Windows. TShark, a terminal-based (non-GUI) version, is also available. Wireshark and the additional programmes included with it, such as TShark, are free software licenced under the GNU General Public Licence version 2 or later.

PCAP:

Many open-source and commercial network tools, such as protocol analyzers (packet sniffers), network monitors, network intrusion detection systems, traffic generators, and network testers, use libpcap, WinPcap, and Npcap as packet capture and filtering engines.

libpcap, WinPcap, and Npcap also support saving captured packets to files and reading files containing saved packets; applications can be written to capture network traffic and analyse it, or to read a saved capture and analyse it using the same analysis code. A capture file saved in the format supported by libpcap, WinPcap, and Npcap can be read by applications that support the format, such as tcpdump, Wireshark, CA NetMaster, or Microsoft Network Monitor 3.x.



```
Conversation Hash Tables

address,port,port,endpoint, 0 entries

uint,endpoint, 0 entries

address,port,address,port,endpoint, 61 entries
               Address 1                Port 1        Address 2              Port 2 Endpoint
192.168.29.50                           33041 192.168.29.1                   53     3
2405:201:a428:9019:1842:c805:f78e:3b2f 53166 2404:6800:4009:821::200e       443    2
2405:201:a428:9019:1842:c805:f78e:3b2f 34564 2405:200:160b:1731::312c:8412 80     2
192.168.29.50                           42005 192.168.29.1                   53     3
192.168.29.50                           52919 192.168.29.1                   53     3
192.168.29.50                           57162 192.168.29.1                   53     3
192.168.29.50                           50256 192.168.29.1                   53     3
2405:201:a428:9019:1842:c805:f78e:3b2f 53118 2404:6800:4009:821::200e       443    2
2405:201:a428:9019:1842:c805:f78e:3b2f 47084 2404:6800:4009:824::2003       443    2
192.168.29.50                           35975 192.168.29.1                   53     3
192.168.29.50                           58117 192.168.29.1                   53     3
2405:201:a428:9019:1842:c805:f78e:3b2f 42962 2404:6800:4009:809::2003       80     2
2405:201:a428:9019:1842:c805:f78e:3b2f 47096 2404:6800:4009:824::2003       443    2
2405:201:a428:9019:1842:c805:f78e:3b2f 59390 2404:6800:4009:831::2004       443    2
192.168.29.50                           40860 34.120.115.102                 443    2
192.168.29.50                           56208 34.117.237.239                 443    2
2405:201:a428:9019:1842:c805:f78e:3b2f 59730 2404:6800:4009:824::2003       443    3
2405:201:a428:9019:1842:c805:f78e:3b2f 38823 2404:6800:4009:80b::200e       443    3
2405:201:a428:9019:1842:c805:f78e:3b2f 42042 2600:1901:0:92a9::             443    2
192.168.29.50                           45640 34.117.65.55                   443    2
192.168.29.50                           48020 52.33.22.51                    443    2
192.168.29.50                           42874 64.233.164.94                  443    2
192.168.29.50                           51327 192.168.29.1                   53     3
192.168.29.50                           51037 192.168.29.1                   53     3
192.168.29.50                           56160 192.168.29.1                   53     3
192.168.29.50                           44920 192.168.29.1                   53     3
192.168.29.50                           60332 192.168.29.1                   53     3
```

Fig. 9 Conversation Hash Table

The file format created and read by libpcap, WinPcap, and Npcap has the MIME type application/vnd.tcpdump.pcap. The most common file extension is.pcap, though.cap and.dmp are also used. The Network Research Group at Lawrence Berkeley Laboratory created libpcap from the tcpdump developers. The low-level packet capture, capture file reading, and capture file writing code from tcpdump was extracted and turned into a library with which tcpdump was linked. It is now created by the same tcpdump.org team that creates tcpdump.

Tshark:

Tshark is a command-line version of Wireshark that allows users to capture and analyse network traffic from a terminal window. Tshark offers most of the features of Wireshark, and its command-line interface allows for automation and scripting.

Tcpdump:

Tcpdump is a packet analyzer that allows users to capture and analyse network traffic from a command line. Tcpdump offers features such as packet filtering and protocol decoding and is widely used in network troubleshooting and analysis.

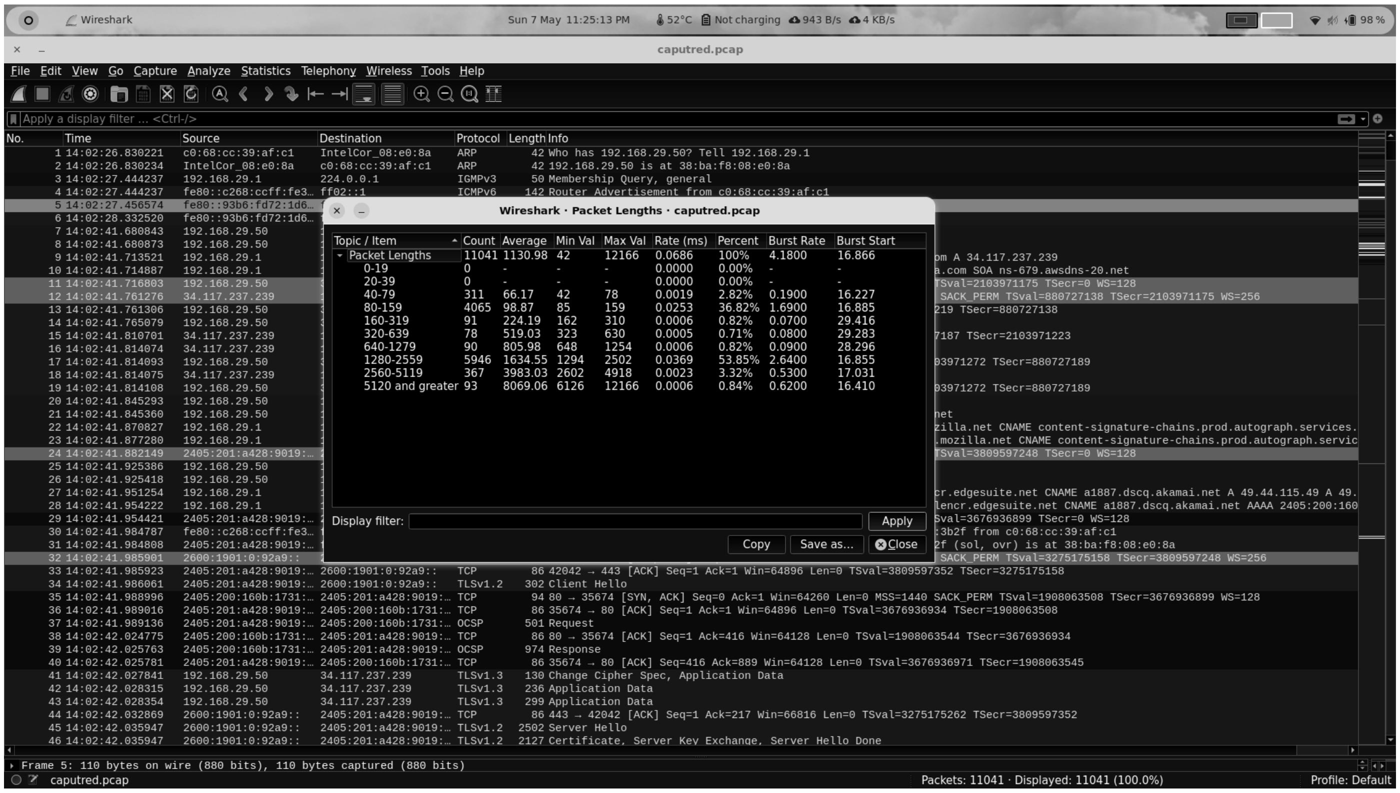
Commercial packet analyzers:

Commercial packet analyzers, such as SolarWinds Network Performance Monitor and Riverbed SteelCentral Packet Analyzer, offer advanced features such as application performance monitoring, network traffic analysis, and security threat detection. These tools are designed for enterprise-level network analysis and are often used by network engineers and security analysts.

Network packet analysis is a critical aspect of network troubleshooting and security analysis. There are various tools available for packet analysis, ranging from open-source tools such as Wireshark to commercial tools such as SolarWinds Network Performance Monitor. The choice of tool depends on the specific requirements of the network analysis and the level of expertise of the user. We investigated the existing literature on network packet analysis and the tools used for this purpose in this literature review.

TCP/IP Protocols :

Because they can be used to communicate across any set of interconnected networks and are equally well suited for LAN and WAN communications, the Internet protocols are the world's most popular open-system (nonproprietary) protocol suite. Internet protocols are a collection of communication protocols, the most well-known of which are the Transmission Control Protocol (TCP) and the Internet Protocol (IP). The Internet protocol suite includes not only lower3 layer protocols (such as TCP and IP), but also common applications such as e-mail, terminal emulation, and file transfer. This document provides an overview of the specifications that comprise the Internet protocols.



```
caputred.pcap
Apply a display filter … <Ctrl-/>
No.  Time             Source                  Destination             Protocol Length Info
   1 14:02:26.830221  c0:68:cc:39:af:c1       IntelCor_08:e0:8a       ARP          42 Who has 192.168.29.50? Tell 192.168.29.1
   2 14:02:26.830234  IntelCor_08:e0:8a       c0:68:cc:39:af:c1       ARP          42 192.168.29.50 is at 38:ba:f8:08:e0:8a
   3 14:02:27.444237  192.168.29.1            224.0.0.1               IGMPv3       50 Membership Query, general
   4 14:02:27.444237  fe80::c268:ccff:fe3…    ff02::1                 ICMPv6      142 Router Advertisement from c0:68:cc:39:af:c1
   5 14:02:27.456574  fe80::93b6:fd72:1d6…
   6 14:02:28.332520  fe80::93b6:fd72:1d6…
   7 14:02:41.680843  192.168.29.50
   8 14:02:41.680873  192.168.29.50
   9 14:02:41.713521  192.168.29.1                                            om A 34.117.237.239
  10 14:02:41.714887  192.168.29.1                                            a.com SOA ns-679.awsdns-20.net
  11 14:02:41.716803  192.168.29.50                                           TSval=2103971175 TSecr=0 WS=128
  12 14:02:41.761276  34.117.237.239                                           SACK_PERM TSval=880727138 TSecr=2103971175 WS=256
  13 14:02:41.761306  192.168.29.50                                           219 TSecr=880727138
  14 14:02:41.765079  192.168.29.50
  15 14:02:41.810701  34.117.237.239                                          7187 TSecr=2103971223
  16 14:02:41.814074  34.117.237.239
  17 14:02:41.814093  192.168.29.50                                           03971272 TSecr=880727189
  18 14:02:41.814075  34.117.237.239
  19 14:02:41.814108  192.168.29.50                                           03971272 TSecr=880727189
  20 14:02:41.845293  192.168.29.50
  21 14:02:41.845360  192.168.29.50                                           net
  22 14:02:41.870827  192.168.29.1                                            zilla.net CNAME content-signature-chains.prod.autograph.services.
  23 14:02:41.877280  192.168.29.1                                            .mozilla.net CNAME content-signature-chains.prod.autograph.servic
  24 14:02:41.882149  2405:201:a428:9019:…                                    TSval=3809597248 TSecr=0 WS=128
  25 14:02:41.925386  192.168.29.50
  26 14:02:41.925418  192.168.29.50
  27 14:02:41.951254  192.168.29.1                                            cr.edgesuite.net CNAME a1887.dscq.akamai.net A 49.44.115.49 A 49.
  28 14:02:41.954222  192.168.29.1                                            lencr.edgesuite.net CNAME a1887.dscq.akamai.net AAAA 2405:200:160
  29 14:02:41.954421  2405:201:a428:9019:…                                    Sval=3676936899 TSecr=0 WS=128
  30 14:02:41.984787  fe80::c268:ccff:fe3…                                    :3b2f from c0:68:cc:39:af:c1
  31 14:02:41.984808  2405:201:a428:9019:…                                    2f (sol, ovr) is at 38:ba:f8:08:e0:8a
  32 14:02:41.985901  2600:1901:0:92a9::                                       SACK_PERM TSval=3275175158 TSecr=3809597248 WS=256
  33 14:02:41.985923  2405:201:a428:9019:…  2600:1901:0:92a9::      TCP          86 42042 → 443 [ACK] Seq=1 Ack=1 Win=64896 Len=0 TSval=3809597352 TSecr=3275175158
  34 14:02:41.986061  2405:201:a428:9019:…  2600:1901:0:92a9::      TLSv1.2     302 Client Hello
  35 14:02:41.988996  2405:200:160b:1731:…  2405:201:a428:9019:…    TCP          94 80 → 35674 [SYN, ACK] Seq=0 Ack=1 Win=64260 Len=0 MSS=1440 SACK_PERM TSval=1908063508 TSecr=3676936899 WS=128
  36 14:02:41.989016  2405:201:a428:9019:…  2405:200:160b:1731:…    TCP          86 35674 → 80 [ACK] Seq=1 Ack=1 Win=64896 Len=0 TSval=3676936934 TSecr=1908063508
  37 14:02:41.989136  2405:201:a428:9019:…  2405:200:160b:1731:…    OCSP        501 Request
  38 14:02:42.024775  2405:200:160b:1731:…  2405:201:a428:9019:…    TCP          86 80 → 35674 [ACK] Seq=1 Ack=416 Win=64128 Len=0 TSval=1908063544 TSecr=3676936934
  39 14:02:42.025763  2405:200:160b:1731:…  2405:201:a428:9019:…    OCSP        974 Response
  40 14:02:42.025781  2405:201:a428:9019:…  2405:200:160b:1731:…    TCP          86 35674 → 80 [ACK] Seq=416 Ack=889 Win=64128 Len=0 TSval=3676936971 TSecr=1908063545
  41 14:02:42.027841  192.168.29.50           34.117.237.239          TLSv1.3     130 Change Cipher Spec, Application Data
  42 14:02:42.028315  192.168.29.50           34.117.237.239          TLSv1.3     236 Application Data
  43 14:02:42.028354  192.168.29.50           34.117.237.239          TLSv1.3     299 Application Data
  44 14:02:42.032869  2600:1901:0:92a9::      2405:201:a428:9019:…    TCP          86 443 → 42042 [ACK] Seq=1 Ack=217 Win=66816 Len=0 TSval=3275175262 TSecr=3809597352
  45 14:02:42.035947  2600:1901:0:92a9::      2405:201:a428:9019:…    TLSv1.2    2502 Server Hello
  46 14:02:42.035947  2600:1901:0:92a9::      2405:201:a428:9019:…    TLSv1.2    2127 Certificate, Server Key Exchange, Server Hello Done
Frame 5: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
caputred.pcap                                      Packets: 11041 · Displayed: 11041 (100.0%)      Profile: Default
```

Wireshark · Packet Lengths · caputred.pcap

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	11041	1130.98	42	12166	0.0686	100%	4.1800	16.866
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	311	66.17	42	78	0.0019	2.82%	0.1900	16.227
80-159	4065	98.87	85	159	0.0253	36.82%	1.6900	16.885
160-319	91	224.19	162	310	0.0006	0.82%	0.0700	29.416
320-639	78	519.03	323	630	0.0005	0.71%	0.0800	29.283
640-1279	90	805.98	648	1254	0.0006	0.82%	0.0900	28.296
1280-2559	5946	1634.55	1294	2502	0.0369	53.85%	2.6400	16.855
2560-5119	367	3983.03	2602	4918	0.0023	3.32%	0.5300	17.031
5120 and greater	93	8069.06	6126	12166	0.0006	0.84%	0.6200	16.410

Fig. 10 Packet Lengths

The Defence Advanced Research Projects Agency (DARPA) became interested in establishing a packet-switched network that would facilitate communication between dissimilar computer systems at research institutions in the mid-1970s, which led to the development of Internet protocols. DARPA-funded research at Stanford University and Bolt, Beranek, and Newman (BBN) with the goal of heterogeneous connectivity in mind. The Internet protocol suite, which was completed in the late 1970s, was the result of this development effort. Internet protocols (including new or revised protocols) and policies are documented in technical reports known as Request for Comments (RFCs), which are published and then reviewed and analysed by the Internet community. Protocol enhancements are documented in the new RFCs. It depicts the scope of the Internet protocols by mapping many of the protocols in the Internet protocol suite and their corresponding OSI layers.

The Internet Protocol (IP) is a network-layer (Layer 3) protocol that contains addressing information and some control information that enables packets to be routed. IP is the primary network-layer protocol in the Internet protocol suite. Along with the Transmission Control Protocol (TCP), IP represents the heart of the Internet protocols. IP has two primary responsibilities: providing connectionless, best-effort delivery of datagrams through an inter-network; and providing fragmentation and reassembly of datagrams to support data links with different maximum-transmission unit (MTU) sizes.

IP Packet Format:

The following discussion describes the IP packet fields illustrated as

- Version— Indicates the version of IP currently used.
- IP Header Length (IHL) — Indicates the datagram header length in 32-bit words.
- Type-of-Service— Specifies how an upper-layer protocol would like a current datagram to be handled, and assigns data grams various levels of importance.
- Total Length — Specifies the length, in bytes, of the entire IP packet, including the data and header.
- Identification — Contains an integer that identifies the current datagram. This field is used to help piece together datagram fragments.
- Flags — Consists of a 3-bit field of which the two low-order (least significant) bits control fragmentation. The low-order bit specifies whether the packet can be fragmented. The middle bit specifies whether the packet is the last fragment in a series of fragmented packets. The third or high-order bit is not used.
- Fragment Offset — Indicates the position of the fragment's data relative to the beginning of the data in the original datagram, which allows the destination IP process to properly reconstruct the original datagram.
- Time-to-Live — Maintains a counter that gradually decrements down to zero, at which point the datagram is discarded. This keeps packets from looping endlessly.
- Protocol — Indicates which upper-layer protocol receives incoming packets after IP processing is complete.
- Header Checksum— Helps ensure IP header integrity.
- Source Address — Specifies the sending node.
- Destination Address — Specifies the receiving node.
- Options — Allows IP to support various options, such as security.
- Data — Contains upper-layer information.

IP Addressing:

As with any other network-layer protocol, the IP addressing scheme is integral to the process of routing IP datagram through an inter-network.

Class	Address Range	Subnet masking	Example IP	Leading Bits	Max No. of networks	Application
A	1 to 126	255.0.0.0	1.1.1.1	8	128	Used for large no. of hosts.
B	128 to 191	255.255.0.0	128.1.1.1	16	16384	Used for medium size network.
C	192 to 223	255.255.255.0	192.1.11.	24	2097157	Used for local area networks.
D	224 to 239	NA	NA	NA	NA	Reserve for multi - tasking.
E	240 to 254	NA	NA	NA	NA	This class is reserved for research and Development Purposes.

Table 1: IP header class

Each IP address has specific components and follows a basic format. These IP addresses can be subdivided and used to create addresses for sub-networks. Each host on a TCP/IP network is assigned a unique 32-bit logical address that is divided into two main parts: the network number and the host number. The network number identifies a network and must be assigned by the Internet Network Information Center (InterNIC) if the network is to be part of the Internet. An Internet Service Provider (ISP) can obtain blocks of network addresses from the InterNIC and can itself assign address space as necessary. The host number identifies a host on a network and is assigned by the local network administrator.

IP Address Format:

The 32-bit IP address is grouped eight bits at a time, separated by dots, and represented in decimal format (known as dotted decimal notation). Each bit in the octet has a binary weight (128, 64, 32, 16, 8, 4, 2, 1). The minimum value for an octet is 0, and the maximum value for an octet is 255. Figure.3 illustrates the basic format of an IP address.

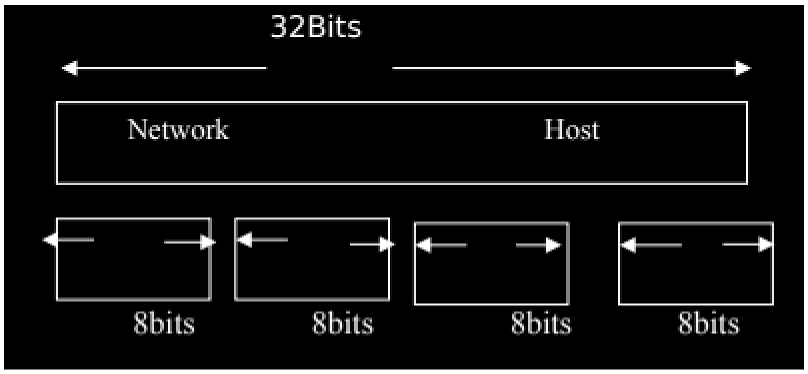


Fig. 11 IP Format

IP Address Classes:

IP addressing supports five different address classes: A, B, C, D, and E. Only classes A, B, and C are available for commercial use. The left-most (high-order) bits indicate the network class.

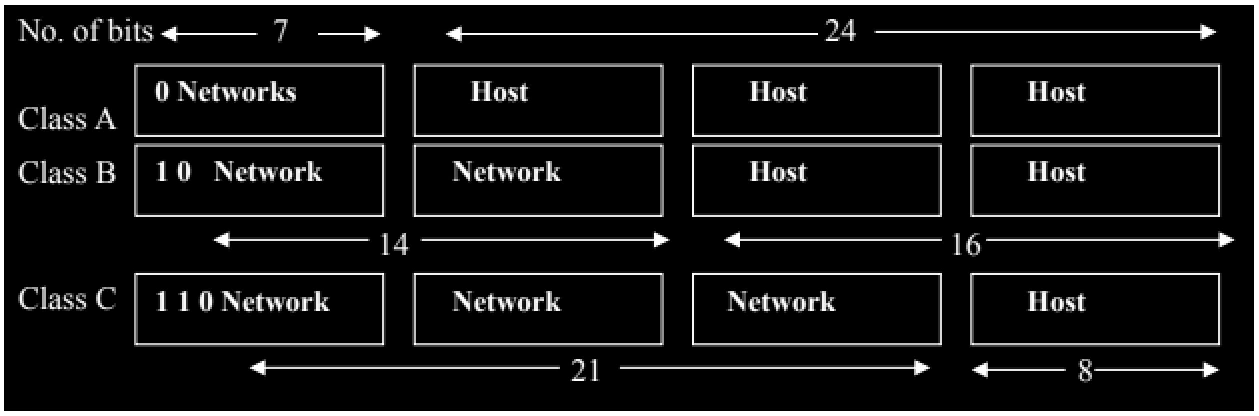


Fig. 12 IP classes

The class of address can be determined easily by examining the first octet of the address and mapping that value to a class range in the following table. In an IP address of 172.31.1.2, for example, the first octet is 172. Because 172 falls between 128 and 191, 172.31.1.2 is a Class B address. Figure 5 summarises the range of possible values for the first octet of each address class.

IP Subnet Addressing:

IP networks can be divided into smaller networks called sub-networks (or subnets). Sub-netting provides the network administrator with several benefits, including extra flexibility, more efficient use of network addresses, and the capability to contain broadcast traffic (a broadcast will not cross a router). Subnets are under local administration. As a result, the outside world perceives an organisation as a single network with no detailed knowledge of its internal structure. A network address can be divided into numerous sub-networks. For example, subnets 172.16.1.0, 172.16.2.0, 172.16.3.0, and 172.16.4.0 are all part of network 171.16.0.0. (An address with all 0s in the host portion specifies the entire network.)

Address Resolution Protocol (ARP) Overview:

To communicate, two machines on the same network must know each other's physical (or MAC) addresses. A host can dynamically discover the MAC-layer address corresponding to a specific IP network-layer address by broadcasting Address Resolution Protocols (ARPs). When an IP device receives a MAC-layer address, it creates an ARP cache to store the recently acquired IP-to-MAC address mapping, avoiding the need to broadcast ARPS when re-contacting a device. If the device does not respond within a specified time frame, the cache entry is flushed. The Reverse Address Resolution Protocol (RARP) is also used to map MAC-layer addresses to IP addresses. The logical inverse of ARP, RARP, may be used by diskless workstations that do not know their IP addresses when they boot. RARP is dependent on the presence of a RARP server with MAC-layer to IP address mapping table entries.

Internet Routing:

Internet routing devices are commonly referred to as gateways. A gateway, on the other hand, is a device that performs application-layer protocol translation between devices in today's jargon. Interior gateways are devices that perform these protocol functions between machines or networks that are under the same administrative control or authority, such as the internal network of a corporation. These are referred to as self-contained systems. Exterior gateways carry out protocol functions between separate networks. The

Internet's routers are arranged in a hierarchical order. Interior routers are routers that are used for information exchange within autonomous systems.

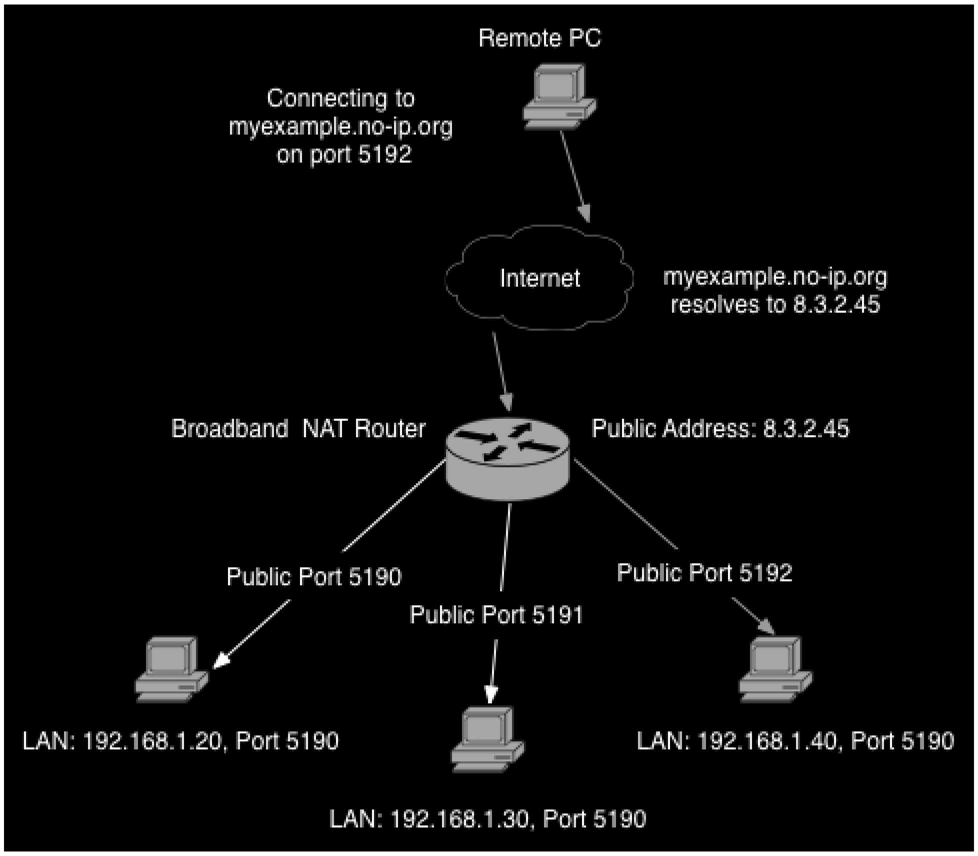

Remote PC
Connecting to myexample.no-ip.org on port 5192
Internet
myexample.no-ip.org resolves to 8.3.2.45
Broadband NAT Router
Public Address: 8.3.2.45
Public Port 5190
Public Port 5191
Public Port 5192
LAN: 192.168.1.20, Port 5190
LAN: 192.168.1.30, Port 5190
LAN: 192.168.1.40, Port 5190

Fig. 13 Internet Routing

They use a variety of Interior Gateway Protocols (IGPs) to accomplish this. The Routing Information Protocol (RIP) is an example of an IGP. Exterior routers are routers that move data between autonomous systems. To exchange data between autonomous systems, these routers employ an exterior gateway protocol. The Border Gateway Protocol (BGP) is an example of an exterior gateway protocol. IP Routing Protocols: IP routing protocols are dynamic in nature. Dynamic routing requires software in routing devices to calculate routes automatically at regular intervals. This is in contrast to static routing, in which routers are set up by the network administrator and do not change until the network administrator changes them. An IP routing table composed of destination address/next hop pairs is used to enable dynamic routing. is tSend the packet out Ethernet interface 0 (E0) to connect to the network 172.31.0.0. Send the packet out Ethernet interface 0 (E0) to network 172.31.0.0. IP routing specifies that IP datagrams travel through internetworks one hop at a time. The entire route, however, is unknown at the start of the journey. Instead, at each stop, the next destination is determined by matching the datagram's destination address with an entry in the current node's routing table. The only role of each node in the routing process is to forward packets based on internal information. The nodes do not monitor whether the packets get to their final destination, nor does IP provide for error reporting back to the source when routing anomalies occur. This task is left to another Internet protocol, the Internet Control-Message Protocol (ICMP), which is discussed in the following section.

ICMP stands for Internet Control Message Protocol.

The Internet Control Message Protocol (ICMP) is a network-layer Internet protocol that sends message packets to the source to report errors and other information about IP packet processing. RFC 792 contains information about ICMP.

ICMP Messages:

ICMPs generate a variety of useful messages, including Destination Unreachable, Echo Request and Reply, Redirect, Time Exceeded, and Router Advertisement and Router Solicitation. If an ICMP message is unable to be delivered, no additional ones are generated. This is done to avoid a never-ending flood of ICMP messages. When a router sends an ICMP destination-unreachable message, it means that the router is unable to deliver the package to its final destination. The original packet is then dropped by the router. There are two possible reasons why a destination may be inaccessible. Most of the time, the source host has specified an invalid address. Occasionally, the router lacks a route to the destination. There are four types of destination-unreachable messages: network unreachable, host unreachable, protocol unreachable, and port unreachable. Unreachable network messages typically indicate a failure in packet routing or addressing. Typically, host-unreachable messages indicate a delivery failure, such as an incorrect subnet mask. Messages marked as protocol-unreachable generally indicate that the destination does not support the upper-layer protocol specified in the packet. Port Unreachable messages indicate that the TCP socket or port is inaccessible. Any host can send an ICMP echo-request message generated by the ping command to test node reachability across an inter network. The ICMP echo reply message indicates that the node was successfully reached. The router sends an ICMP Redirect message to the source host to encourage more efficient routing. The original packet is still forwarded to the destination by the router. Because only one router's address is required, even if that router does not provide the best path, ICMP redirects allow host routing tables to remain small. Even after receiving an ICMP Redirect message, some devices may continue to take the inefficient route. If the Time-toLive field of an IP packet (expressed in hops or seconds) reaches zero, the router sends an ICMP Time-exceeded message. If the internetwork contains a routing loop, the Time-to-Live field prevents packets from circulating indefinitely. The router then discards the original packet.

Transmission Control Protocol (TCP):

TCP ensures dependable data transmission in an IP environment. The transport layer (Layer 4) of the OSI reference model is TCP. TCP offers several services, including stream data transfer, reliability, efficient flow control, full duplex operation, and multiplexing. When using stream data transfer, TCP sends an unstructured stream of bytes identified by sequence numbers. This service saves time for applications by removing the need to divide data into blocks before passing it to TCP. TCP, on the other hand, divides bytes into segments and delivers them via IP. TCP ensures reliability by sending connection-oriented, end-to-end dependable packets across an internetwork. This is accomplished by preceding bytes with a forwarding acknowledgment number, which indicates to the destination the next byte that the source expects to receive. Bytes that are not acknowledged within a certain amount of time are resent. The reliability mechanism of TCP allows devices to deal with lost, delayed, duplicate, or misread packets. A time-out mechanism allows devices to detect lost packets and request retransmission. Because TCP provides efficient flow control, when acknowledgments are returned to the source, the receiving TCP process indicates the maximum sequence number it can receive without overflowing its internal buffers. Full-duplex TCP processes can send and receive data simultaneously. Finally, TCP multiplexing allows for multiple simultaneous upper-layer conversations to be carried out over a single connection.

TCP Connection Establishment:

To use dependable transport services, TCP hosts must establish a connection-oriented session with one another. To make connections, a "three-way handshake" mechanism is used. A three-way handshake allows both ends of a connection to agree on initial sequence numbers, allowing them to be synchronised. This mechanism also ensures that both sides are prepared to transmit data and are aware that the other side is also prepared to transmit. This is required to prevent packets from being sent or resent during session establishment or termination. Each host chooses a random sequence number to track bytes in the stream it is sending and receiving. Following that, the three-way handshake is performed as follows: The first host

(Host A) starts the connection by sending a packet with the initial sequence number (X) and the SYN bit, which indicates a connection request. Host B receives the SYN, records the sequence number X, and responds by acknowledging the SYN (with an ACK = X + 1). Host B has its own unique initial sequence number (SEQ = Y). An ACK = 20 indicates that the host has received bytes 0 through 19 and is waiting for byte 20. This method is known as forward acknowledgment. Host A then acknowledges all bytes sent by Host B with a forward acknowledgment indicating the next byte expected by Host A (ACK = Y + 1). Data transfer then can begin.

PAR (Positive Acknowledgement and Retransmission):

A simple transport protocol could use a reliability-and-flow-control technique in which the source sends one packet, starts a timer, and waits for an acknowledgement before sending another packet. If the acknowledgement is not received before the timer expires, the packet is retransmitted by the source. This method is known as positive acknowledgment and retransmission (PAR). PAR assigns a sequence number to each packet, allowing hosts to track lost or duplicate packets caused by network delays that result in premature retransmission. The sequence numbers are returned in the acknowledgments so that they can be tracked. However, because a host must wait for an acknowledgement before sending a new packet, and only one packet can be sent at a time, PAR is an inefficient use of bandwidth.

TCP Sliding Window:

TCP sliding windows use network bandwidth more efficiently than PAR because they allow hosts to send multiple bytes or packets before waiting for an acknowledgement. TCP requires the receiver to specify the current window size in every packet. Window sizes are expressed in bytes because TCP is a byte-stream connection. This means that a window is the maximum number of data bytes that a sender can send before waiting for an acknowledgment. Initial window sizes are specified during connection setup, but may vary during data transfer to provide flow control. A window size of zero, for example, indicates "Send no data." In a TCP sliding-window operation, for example, the sender may have a sequence of bytes to send (numbered 1 to 10) to a receiver with a window size of five. The sender would then create a window around the first five bytes and send them all at once. It would then wait for an acknowledgement. The receiver would respond with an ACK = 6, indicating that it has received bytes 1–5 and is waiting for byte 6. The receiver would indicate its window size as 5 in the same packet. The sender would then slide the sliding window to the right five bytes and transmit bytes 6 to 10. The receiver would respond with an ACK = 11, indicating that it is looking forward to sequenced byte 11. In this packet, the receiver may indicate that its window size is 0 (for example, because its internal buffers are full). The sender is now unable to send any more bytes until the receiver sends another packet with a window size greater than 0.

TCP Packet Field Descriptions:

The TCP packet fields are summarised in the following descriptions:

- **Source Port and Destination Port** — Identifies points at which upper-layer source and destination processes receive TCP services.
- **Sequence Number** — Usually specifies the number assigned to the first byte of data in the current message. In the connection-establishment phase, this field also can be used to identify an initial sequence number to be used in an upcoming transmission.
- **Acknowledgment Number** — Contains the sequence number of the next byte of data the sender of the packet expects to receive.
- **Data Offset** — Indicates the number of 32-bit words in the TCP header. Reserved—Remains reserved for future use.
- **Flags** — Carries a variety of control information, including the SYN and ACK bits used for connection establishment, and the FIN bit used for connection termination.
- **Window** — Specifies the size of the sender's receive window (that is, the buffer space available for incoming data).
- **Checksum —** Indicates whether the header was damaged in transit.
- **Urgent Pointer** — Points to the first urgent data byte in the packet.
- **Options** — Specifies various TCP options.
- **Data** — Contains upper-layer information.

User Datagram Protocol (UDP):

The Internet protocol family includes the User Datagram Protocol (UDP), which is a connectionless transport-layer protocol (Layer 4). UDP serves as a bridge between IP and upper-layer processes. Multiple applications running on a single device are distinguished by UDP protocol ports. UDP, unlike TCP, does not add any reliability, flow control, or error recovery functions to IP. UDP headers contain fewer bytes and consume less network overhead than TCP because of their simplicity. UDP is useful in situations where TCP's reliability mechanisms are not required, such as when a higher-layer protocol provides error and flow control. UDP is the transport Network File System (NFS), Simple Network Management Protocol (SNMP), Domain Name System (DNS), and Trivial File Transfer Protocol (TFTP) are all application-layer protocols that use UDP as their transport protocol. As shown in Figure 7, the UDP packet format has four fields. These include source and destination ports, length, and checksum fields

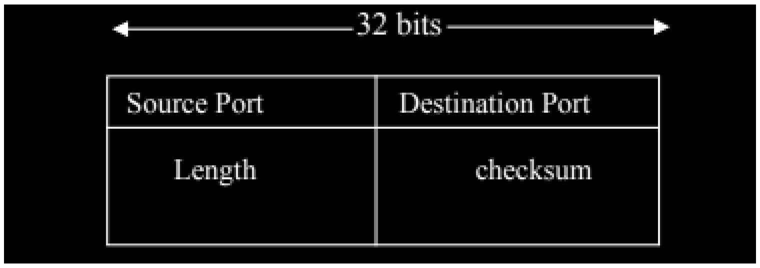


Fig. 14 UDP Header

The 16-bit UDP protocol port numbers used to de-multiplex datagrams for receiving application-layer processes are contained in the source and destination ports. The length of the UDP header and data is specified by the length field. Checksum checks the integrity of the UDP header and data (optionally).

Internet Protocols: Application Layer Protocols

Many application-layer protocols that represent a wide range of applications are included in the Internet protocol suite, including the following:

FTP (File Transfer Protocol)—Transfers files between devices.

Simple Network-Management Protocol (SNMP)—Reports abnormal network conditions and establishes network thresholds.

Telnet—A protocol for terminal emulation.

X Windows—A distributed windowing and graphics system that allows X terminals to communicate with UNIX workstations.

Network File System (NFS), External Data Representation (XDR), and Remote Procedure Call (RPC)—These protocols collaborate to provide transparent access to remote network resources.

Simple Mail Transfer Protocol (SMTP)—This protocol is used to provide electronic mail services.

Domain Name System (DNS)—Converts network node names into network addresses**.** The following are the higher-layer protocols and the applications that they support: Application Protocols File transfer FTP Terminal emulation Telnet Electronic mail SMTP Network management SNMP Distributed file services NFS, XDR, RPC, X Windows.
The list of the higher-layer protocols and the applications that they support is as follows:

Application	Protocol
File transfer	FTP
Terminal emulation	Telnet
Network management	SNMP
Distributed file services	NFS, XDR, RPC, X Windows

Table 2 Supported app and its protocol

CHAPTER 3
SYSTEM DEVELOPMENT

The systematic approach to the development, operation, maintenance, and retirement of software is known as software engineering. All software products can be developed using a Software Process, also known as the Software Life Cycle. This Software Process is nothing more than a series of distinct stages that a software product goes through during its lifetime. This series begins with a Feasibility Study Stage, followed by Requirement Analysis and Specification, Design, Coding, Testing, and Maintenance. Each of these stages is referred to as a Life Cycle Phase. This Software process is accomplished with the assistance of a software life cycle model (or process model). A Process Model is a descriptive and graphical representation of a software process. A process model identifies all of the activities required to develop and maintain a software product, as well as the precedence ordering of those activities.

Each phase's entry and exit criteria are defined by a process model. For example, the Software Requirement Specification phase's corresponding phase-entry criteria could be that the software Requirement Specification document has been completed, internally reviewed, and approved by the customer. With such well-defined entry and exit criteria for various phases, managing and monitoring the project's progress becomes easier. As a result, life cycle models promote systematic and disciplined software development. Because of the foregoing, the developer should follow a well-defined life cycle model. Thus, one major benefit of adhering to a well-defined life cycle model is that it aids in controlling and organising systematically various activities of the product under development. When a life cycle model is followed, the developer can easily determine which stage of development (e.g., design, code, testing) the project is currently in. If no life cycle model is used, it becomes extremely difficult to track the progress of the project, and the developer may encounter a problem known as the 99% complete syndrome. In this syndrome, which appears when there is no definite way to assess the project's progress, the optimistic developer believes that the project is 99% complete, even if it is far from finished. The success or failure of a project is heavily dependent on the life cycle model to which the developer will adhere. As a result, a life cycle model is critical to the successful completion of a project. When developing a software product, five different types of life cycle models are used. Model of a traditional waterfall. The waterfall model is iterative. Model for Evolutionary Prototyping. Spiral Design. I used the Classical Waterfall Model while developing the project Packet Sniffer. This model divides the life cycle of a software development process into the following phases: Development phase = Feasibility Study + Requirement Analysis and Specification + Design + Coding and Unit Testing + Integration and System Testing.

Data Flow Diagrams:

A data flow diagram is made up of several symbols that represent system components. The most common data flow modelling methods employ four types of symbols. These symbols represent four different types of system components: processes, data stores, data flows, and external entities. The data flow diagrams for the current project are shown in the figure below. It is the overall process data flow diagram. It specifies the major transformation points in the software development process. This is the first step in the structured design method. In the project, the inputs are the packets that are flowing through the network interface that is set to promiscuous mode. The output is the human-readable information contained in the packets, which is saved in the output file. The context diagram and data flow diagram of the proposed system are given as follows:

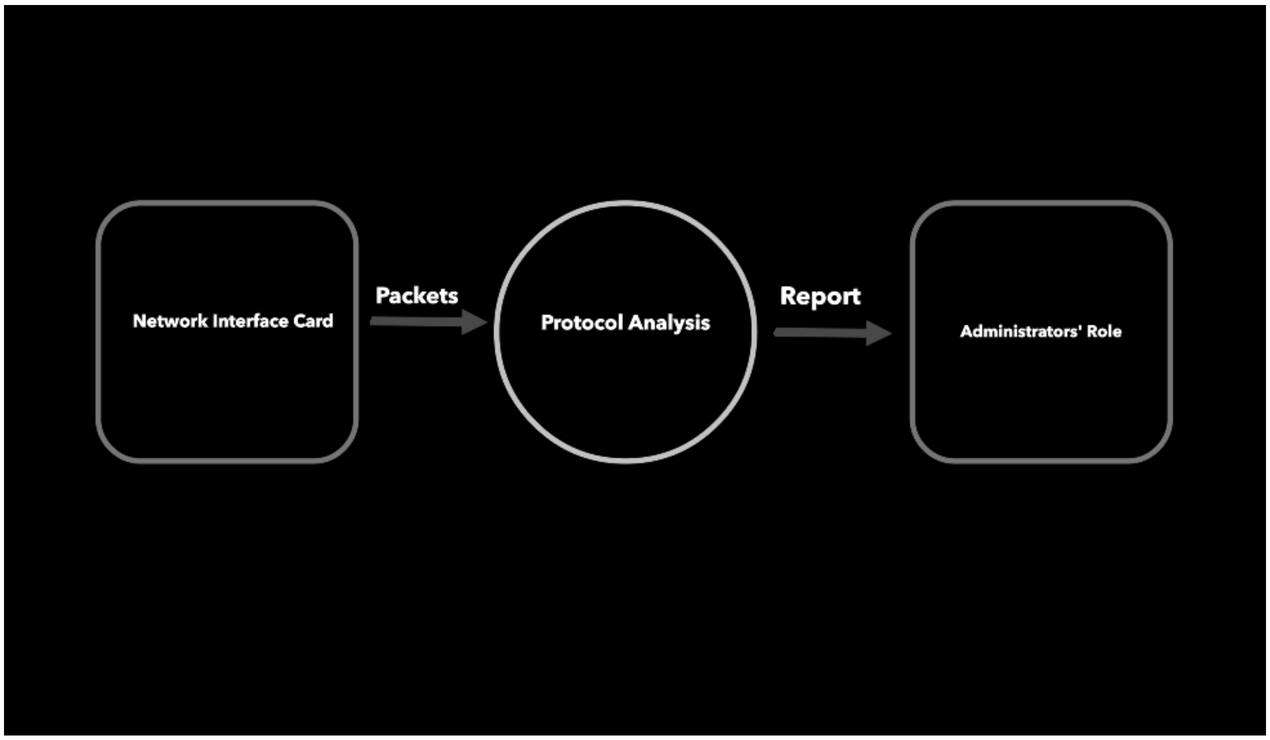


Fig. 15 Data flow diagram

The data flow diagrams for the current project are depicted in the figure below. It is the overall process data flow diagram. It specifies the major transformation points in the software development process. The structured design method begins with this step. In the project, the inputs are the packets that are flowing through the network interface that is set to promiscuous mode. The output is the human-readable information contained in the packets, which is saved in the output file.

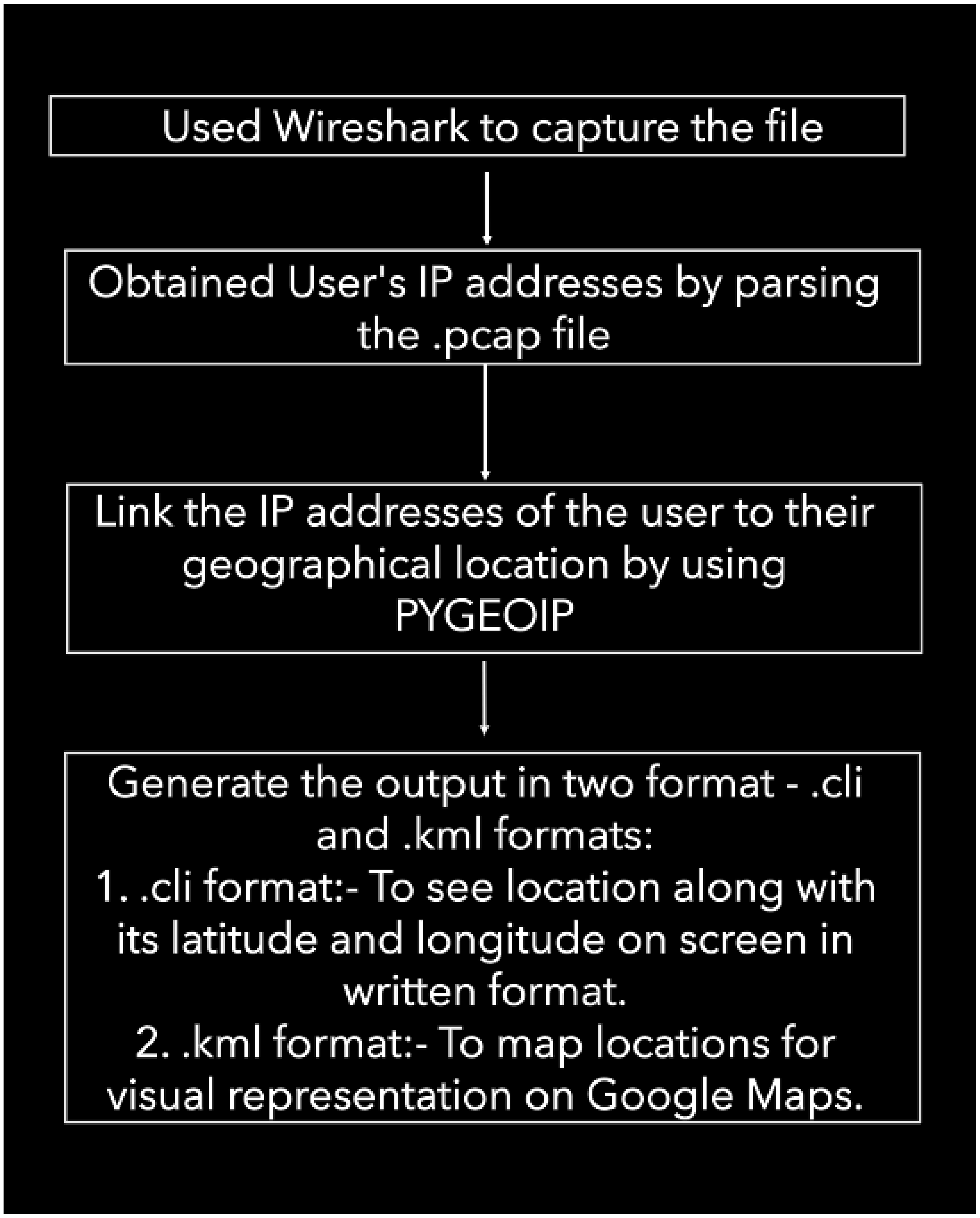

Used Wireshark to capture the file
Obtained User's IP addresses by parsing the .pcap file
Link the IP addresses of the user to their geographical location by using PYGEOIP
Generate the output in two format - .cli and .kml formats:
1. .cli format:- To see location along with its latitude and longitude on screen in written format.
2. .kml format:- To map locations for visual representation on Google Maps.

Fig. 16 Application Flow Diagram

The 'Get packets' process in the diagram obtains input as packets from the network interface. This process creates a packet socket, retrieves raw packets from the network interface, and stores them in a buffer. The buffer containing the packets is passed to the'separate header' process, which strips off various headers of the packet and passes them to the 'analyse headers' process, which analyses them and passes the information to the 'update output file' process. The output file will be updated with the most recent information obtained from the subsequent processes. The most abstract inputs are the stripped off headers and the most abstract output is the information in the headers in human readable form.

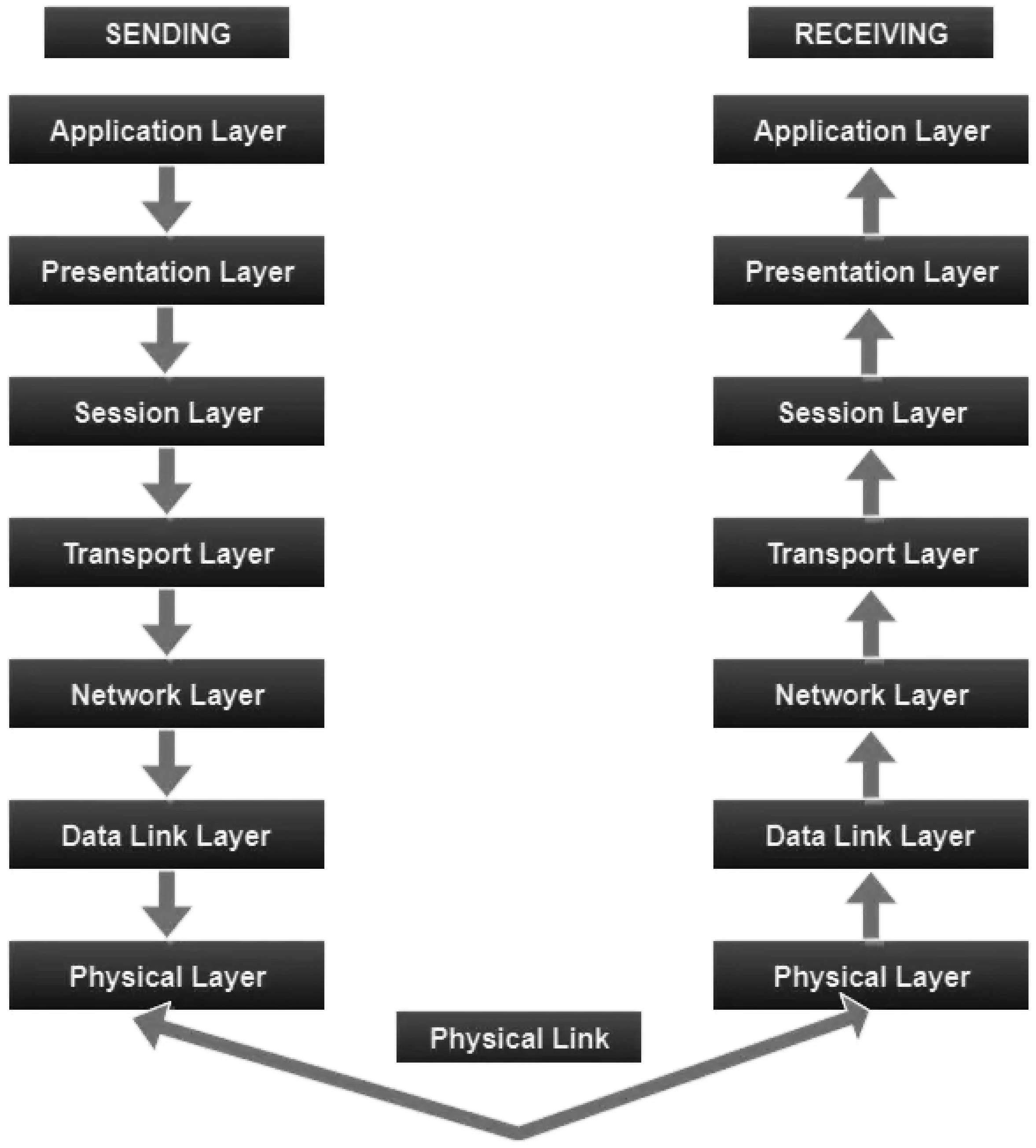


Fig. 17 Network flow

Python:-

Python is one of the most widely used and popular programming languages in the world. It's strong, versatile, and simple to learn. Python is widely used in a variety of applications, including the following:

- Web development
- Data Science
- Data analysis
- Machine learning
- Artificial Intelligence (AI)
- Scripting and tooling

Python is a general-purpose programming language that can be used in almost any domain or application. It can be used to build a website, train machine learning models, perform complex financial calculations, or write quick automation scripts; the possibilities are endless. Various AI programmes, such as ChatGPT, MidJourney, and Dall-E, have recently turned the Internet on its head. Python is a popular programming language for artificial intelligence and data science. You can build and train accurate models using its extensive library and framework library, such as Scikit or TensorFlow. Python provides frameworks and libraries that help developers in the more traditional domain of web development. Django, for example, is used by major corporations such as INSTAGRAM. Python is also used in financial predictive modelling and automated fraud detection; it is even used in the entertainment industry for game development.

If you are young and have only recently discovered Python, you may be surprised to learn that it has been around since the early 1990s! Python was well-known as a scripting language before it became popular for the applications mentioned above, and it is still used for that. A script enables the rapid automation of complex tasks on a server. Scripts can be run without having to compile or restart anything. Consider the following scenario: you want to reduce your website's churn, so you decide to send an email every day to remind users who are nearing their renewal date to renew the service. You can easily accomplish this with a Python script linked to a CRON JOB.

You're probably aware that every application or platform you create must be tested before it can be released. Python includes a variety of testing frameworks and libraries. You can easily create any type of test case your application requires using libraries like UNITTEST, PYTEST, or DJANGO.TEST. You can also link your tests to the powerful COVERAGE tool, which will determine which parts of your code have not been tested and will detect potential flaws in your system. Please don't be concerned about the language's complexity! Python is an excellent choice for a first programming language because it is simple to understand and write, with a straightforward syntax.

Many people believe Python is powered by batteries. It's a clever way of saying that it comes with a substantial base library. Furthermore, because Python is so widely used, there are hundreds of thousands of high-quality libraries and frameworks available to assist you in accomplishing your goals quickly and easily. A little Python code can get you a long way!

Python Timeline:-

Python 2 and 3 have been developed and maintained concurrently for a long time, as evidenced by the Python history timeline. The main reason for this is that Python 3 code is not completely backward compatible with Python 2 code. Because of this incompatibility, the adoption rate was delayed. Many users were content with version 2 and saw no reason to upgrade.

Furthermore, Python 3 was initially slower than Python 2. Python 3 eventually took off as it continued to improve and gain new features. Python 3 is now faster than ever thanks to recent efforts led by Guido. Furthermore, Python 3 adds many useful features to the language, making it easier and more enjoyable. Unless you need to maintain a legacy code base, avoid Python 2.

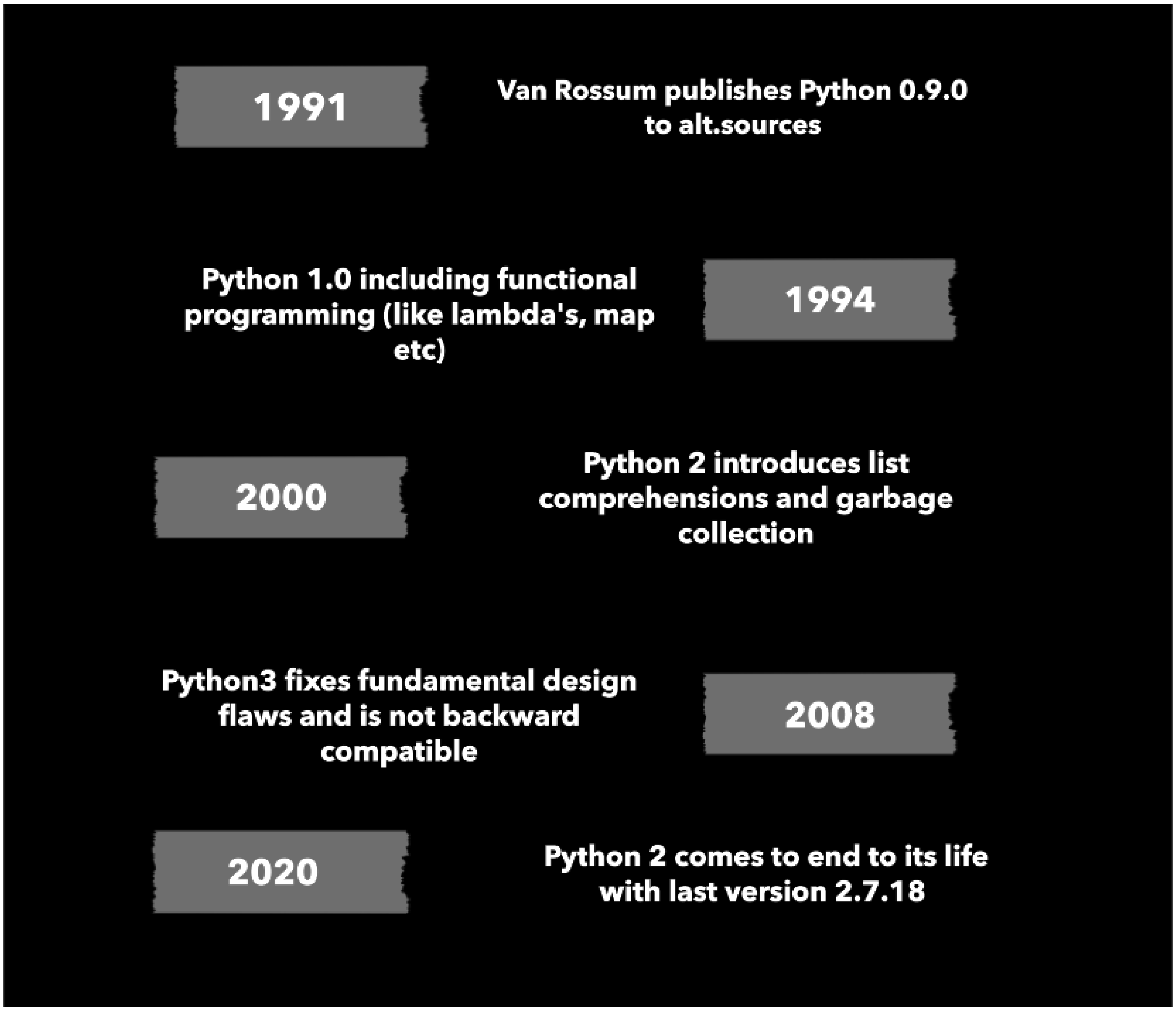

1991
Van Rossum publishes Python 0.9.0 to alt.sources
Python 1.0 including functional programming (like lambda's, map etc)
1994
2000
Python 2 introduces list comprehensions and garbage collection
Python3 fixes fundamental design flaws and is not backward compatible
2008
2020
Python 2 comes to end to its life with last version 2.7.18

Fig. 18 Python Timeline

Django:-

A "web framework" is a set of tools that abstracts away much of the difficulty and repetition that comes with web development. Most websites, for example, require the same basic functionality: the ability to connect to a database, set URL routes, display content on a page, properly handle security, and so on. er than reinventing the wheel, programmers have created web frameworks in all of the major programming languages, including Django in Python, Rails in Ruby, and Laravel in PHP, among many, many others.

Django adopted Python's "batteries-included" approach and includes out-of-the-box support for common web development tasks such as:

- User authentication
- Testing
- Database models, forms, URL routes, and templates
- Admin user interface
- Enhancements to security and performance
- Multiple database backends are supported.

Instead of reinventing the wheel every time, this approach allows web developers to concentrate on what makes a web application unique.

In contrast, some web frameworks, such as Flask, take a microframework approach, providing only the bare necessities for a simple webpage. Flask is far more lightweight than Django and allows for maximum flexibility; however, the developer pays a price for this. Building a simple website necessitates the installation of a dozen or more third-party packages, which may or may not be current, secure, or dependable. Because there are no guardrails, Flask's project structure varies greatly, making it difficult to maintain best practises when switching between projects.

Django continues to be actively developed, with monthly security/bug fixes and a major new release every eight months. Django has already been used by millions of programmers to build websites. It makes no sense to rewrite the same code--and make the same mistakes--when a large community of brilliant developers has already solved these issues for us.

And, best of all, Django is written in the incredibly readable yet powerful Python programming language. In short, if you're starting from scratch, Django is a fantastic choice. The most recent version of Django, 4.2, was released in April 2023. Python versions 3.8, 3.9, 3.10, and 3.11 are officially supported. The official Django website contains information on all prior releases and detailed 4.2 release notes.

One of the significant efforts for Django since 3.0 has been adding asynchronous support. Django 3.0 introduced ASGI (Asynchronous Server Gateway Interface), and Django 3.1 added asynchronous views, middleware, tests, and a test client. Django 4.0 introduced async cache backends, and Django 4.1 added asynchronous handlers for class-based views as well as an asynchronous ORM interface. Psycopg 3, a

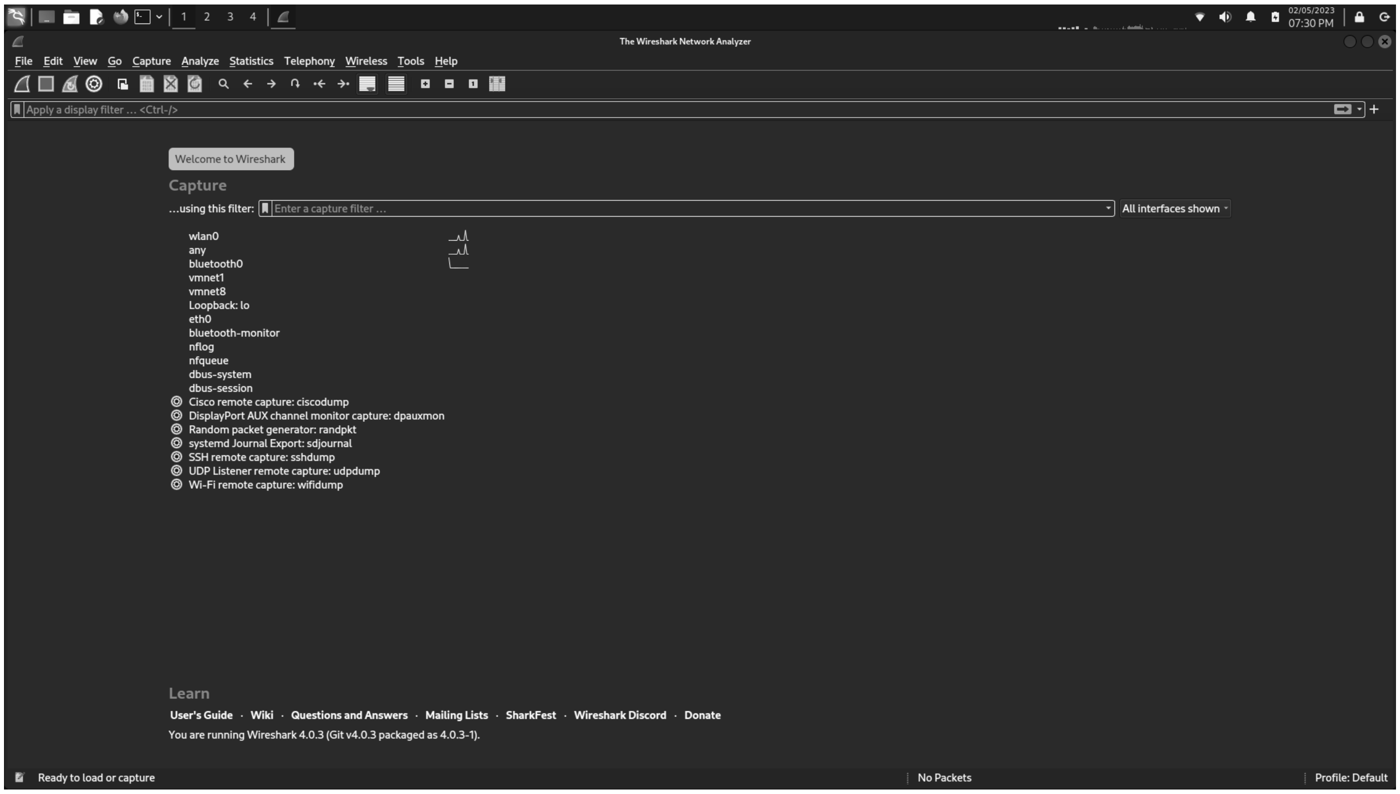
PostgreSQL database adapter for Python with async support, is supported by Django 4.2. These iterative enhancements bring Django ever closer to full asynchronous functionality.

Django 4.2 is an LTS (Long Term Support) release, which means it will be supported for at least three years after its initial release. The previous LTS was Django 3.2, which will be decommissioned in April 2024. As with all major releases, the 4.2 release includes numerous new enhancements, such as a light or dark colour theme for the admin.

CHAPTER-4

EXPERIMENTS & RESULT ANALYSIS

Our network security project's goal was to carry out a passive attack on the LAN by finding and analysing network traffic with a network protocol analyzer tool. Following that, a detailed analysis of the packet captures was arranged and performed to determine whether blacklisted websites were accessed by LAN users and to identify the geographical locations of those users and websites. In addition, we analysed and processed the HTTP traffic in the capture file and were able to identify the HTTP Request and HTTP Response contents in order to monitor Internet user behaviour and their access details. In the field of network security, our approach was an attempt to detect and prevent illegal activities such as visiting and downloading inappropriate content from the Internet. This is referred to as Cyberslacking. Inappropriate content includes, but is not limited to, media elements such as audio and video files that violate copyrights, pornography, prohibited resources, and so on. To achieve the project's goal, we used several network security tools.


1 2 3 4
02/05/2023
07:30 PM
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Welcome to Wireshark
Capture
…using this filter:
Enter a capture filter …
All interfaces shown
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate

wlan0
any
bluetooth0
vmnet1
vmnet8
Loopback: lo
eth0
bluetooth-monitor
nflog
nfqueue
dbus-system
dbus-session
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
Wi-Fi remote capture: wifidump

You are running Wireshark 4.0.3 (Git v4.0.3 packaged as 4.0.3-1).

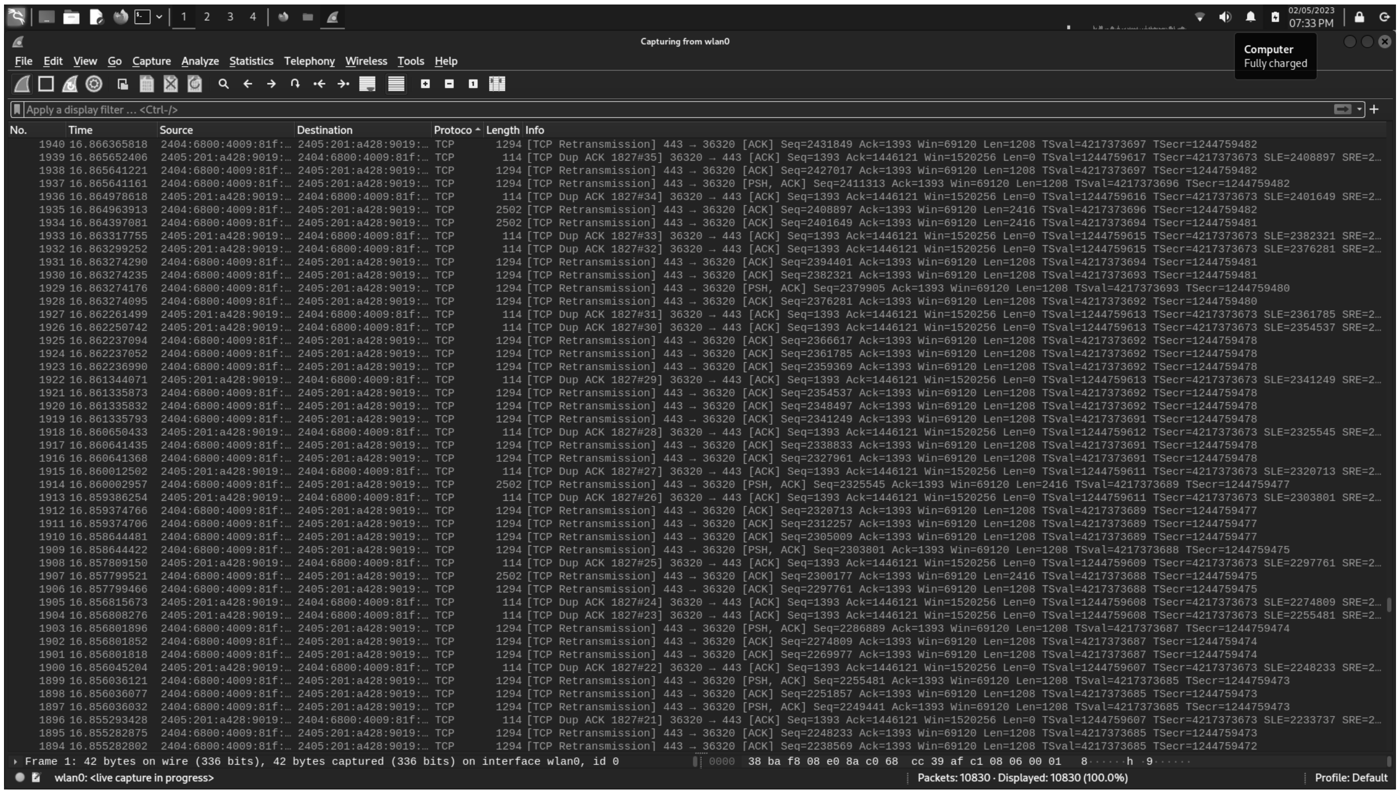
Ready to load or capture | No Packets | Profile: Default

Fig. 19 Wireshark Home

We created a programme in Python to parse the network capture (.pcap) file obtained at the LAN interface using the dpkt python package. The tool gathered and extracted IP addresses from the file and filtered out those that had been flagged as potentially blacklisted websites. Later, we used the pyGeoIP geolocation tool to determine the physical or the geographical locations of the blacklisted websites and their users. The following are the project's implementation steps:

1. A network packet capture file is required to perform network traffic analysis. Wireshark is the most widely and the most popular used network protocol analyzer tool for capturing packets at the LAN interface. The packet capture setup is depicted in the diagram below.

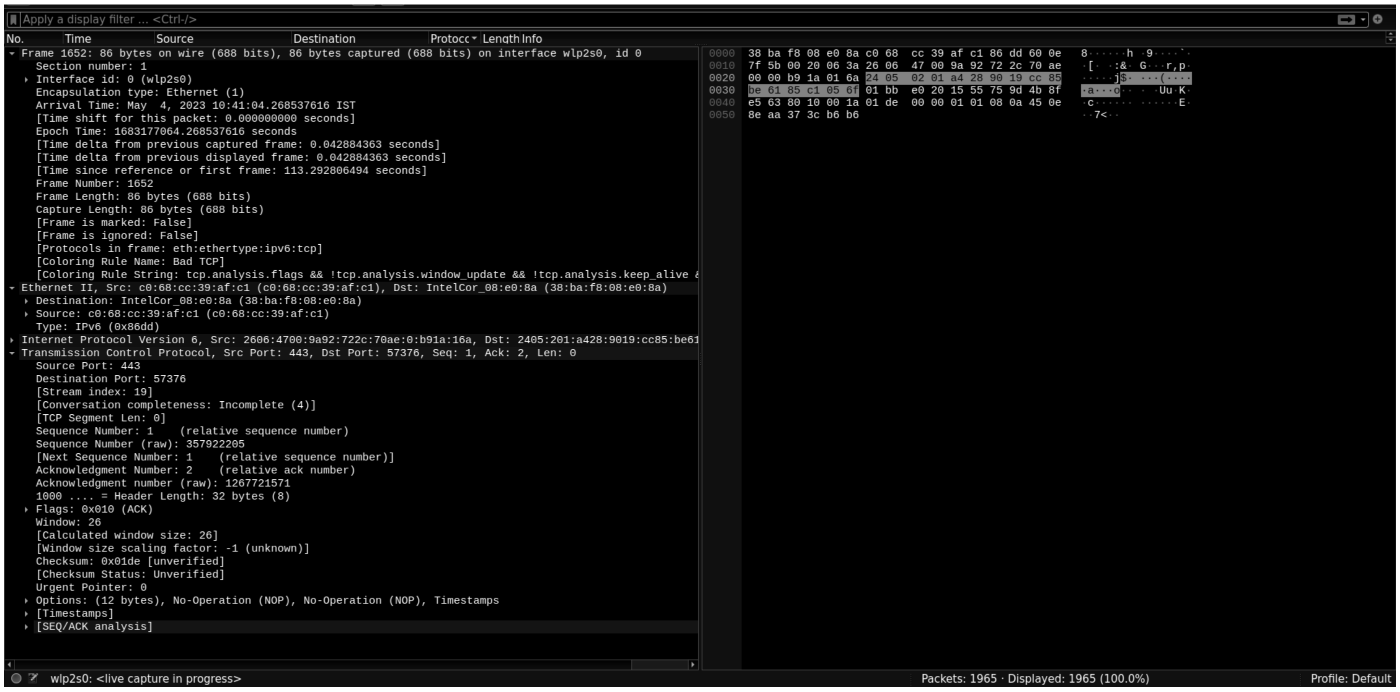
The following figure shows the packet capture by Wireshark packet capture tool:


02/05/2023
07:33 PM
Capturing from wlan0
Computer
Fully charged
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time          Source              Destination         Protoco Length Info
 1940 16.866365818  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2431849 Ack=1393 Win=69120 Len=1208 TSval=4217373697 TSecr=1244759482
 1939 16.865652406  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#35] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759617 TSecr=4217373673 SLE=2408897 SRE=2…
 1938 16.865641221  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2427017 Ack=1393 Win=69120 Len=1208 TSval=4217373697 TSecr=1244759482
 1937 16.865641161  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [PSH, ACK] Seq=2411313 Ack=1393 Win=69120 Len=1208 TSval=4217373696 TSecr=1244759482
 1936 16.864978618  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#34] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759616 TSecr=4217373673 SLE=2401649 SRE=2…
 1935 16.864963913  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  2502 [TCP Retransmission] 443 → 36320 [ACK] Seq=2408897 Ack=1393 Win=69120 Len=2416 TSval=4217373696 TSecr=1244759482
 1934 16.864397081  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  2502 [TCP Retransmission] 443 → 36320 [ACK] Seq=2401649 Ack=1393 Win=69120 Len=2416 TSval=4217373694 TSecr=1244759481
 1933 16.863317755  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#33] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759615 TSecr=4217373673 SLE=2382321 SRE=2…
 1932 16.863299252  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#32] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759615 TSecr=4217373673 SLE=2376281 SRE=2…
 1931 16.863274290  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2394401 Ack=1393 Win=69120 Len=1208 TSval=4217373694 TSecr=1244759481
 1930 16.863274235  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2382321 Ack=1393 Win=69120 Len=1208 TSval=4217373693 TSecr=1244759481
 1929 16.863274176  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [PSH, ACK] Seq=2379905 Ack=1393 Win=69120 Len=1208 TSval=4217373693 TSecr=1244759480
 1928 16.863274095  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2376281 Ack=1393 Win=69120 Len=1208 TSval=4217373692 TSecr=1244759480
 1927 16.862261499  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#31] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759613 TSecr=4217373673 SLE=2361785 SRE=2…
 1926 16.862250742  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#30] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759613 TSecr=4217373673 SLE=2354537 SRE=2…
 1925 16.862237094  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2366617 Ack=1393 Win=69120 Len=1208 TSval=4217373692 TSecr=1244759478
 1924 16.862237052  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2361785 Ack=1393 Win=69120 Len=1208 TSval=4217373692 TSecr=1244759478
 1923 16.862236990  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2359369 Ack=1393 Win=69120 Len=1208 TSval=4217373692 TSecr=1244759478
 1922 16.861344071  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#29] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759613 TSecr=4217373673 SLE=2341249 SRE=2…
 1921 16.861335873  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2354537 Ack=1393 Win=69120 Len=1208 TSval=4217373692 TSecr=1244759478
 1920 16.861335832  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2348497 Ack=1393 Win=69120 Len=1208 TSval=4217373692 TSecr=1244759478
 1919 16.861335793  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2341249 Ack=1393 Win=69120 Len=1208 TSval=4217373691 TSecr=1244759478
 1918 16.860650433  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#28] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759612 TSecr=4217373673 SLE=2325545 SRE=2…
 1917 16.860641435  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2338833 Ack=1393 Win=69120 Len=1208 TSval=4217373691 TSecr=1244759478
 1916 16.860641368  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2327961 Ack=1393 Win=69120 Len=1208 TSval=4217373691 TSecr=1244759478
 1915 16.860012502  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#27] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759611 TSecr=4217373673 SLE=2320713 SRE=2…
 1914 16.860002957  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  2502 [TCP Retransmission] 443 → 36320 [PSH, ACK] Seq=2325545 Ack=1393 Win=69120 Len=2416 TSval=4217373689 TSecr=1244759477
 1913 16.859386254  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#26] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759611 TSecr=4217373673 SLE=2303801 SRE=2…
 1912 16.859374766  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2320713 Ack=1393 Win=69120 Len=1208 TSval=4217373689 TSecr=1244759477
 1911 16.859374706  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2312257 Ack=1393 Win=69120 Len=1208 TSval=4217373689 TSecr=1244759477
 1910 16.858644481  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2305009 Ack=1393 Win=69120 Len=1208 TSval=4217373689 TSecr=1244759477
 1909 16.858644422  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [PSH, ACK] Seq=2303801 Ack=1393 Win=69120 Len=1208 TSval=4217373688 TSecr=1244759475
 1908 16.857809150  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#25] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759609 TSecr=4217373673 SLE=2297761 SRE=2…
 1907 16.857799521  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  2502 [TCP Retransmission] 443 → 36320 [ACK] Seq=2300177 Ack=1393 Win=69120 Len=2416 TSval=4217373688 TSecr=1244759475
 1906 16.857799466  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2297761 Ack=1393 Win=69120 Len=1208 TSval=4217373688 TSecr=1244759475
 1905 16.856815673  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#24] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759608 TSecr=4217373673 SLE=2274809 SRE=2…
 1904 16.856808276  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#23] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759608 TSecr=4217373673 SLE=2255481 SRE=2…
 1903 16.856801896  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [PSH, ACK] Seq=2286889 Ack=1393 Win=69120 Len=1208 TSval=4217373687 TSecr=1244759474
 1902 16.856801852  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2274809 Ack=1393 Win=69120 Len=1208 TSval=4217373687 TSecr=1244759474
 1901 16.856801818  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2269977 Ack=1393 Win=69120 Len=1208 TSval=4217373687 TSecr=1244759474
 1900 16.856045204  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#22] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759607 TSecr=4217373673 SLE=2248233 SRE=2…
 1899 16.856036121  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [PSH, ACK] Seq=2255481 Ack=1393 Win=69120 Len=1208 TSval=4217373685 TSecr=1244759473
 1898 16.856036077  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2251857 Ack=1393 Win=69120 Len=1208 TSval=4217373685 TSecr=1244759473
 1897 16.856036032  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [PSH, ACK] Seq=2249441 Ack=1393 Win=69120 Len=1208 TSval=4217373685 TSecr=1244759473
 1896 16.855293428  2405:201:a428:9019:… 2404:6800:4009:81f:… TCP   114 [TCP Dup ACK 1827#21] 36320 → 443 [ACK] Seq=1393 Ack=1446121 Win=1520256 Len=0 TSval=1244759607 TSecr=4217373673 SLE=2233737 SRE=2…
 1895 16.855282875  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2248233 Ack=1393 Win=69120 Len=1208 TSval=4217373685 TSecr=1244759473
 1894 16.855282802  2404:6800:4009:81f:… 2405:201:a428:9019:… TCP  1294 [TCP Retransmission] 443 → 36320 [ACK] Seq=2238569 Ack=1393 Win=69120 Len=1208 TSval=4217373685 TSecr=1244759472
▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface wlan0, id 0    0000  38 ba f8 08 e0 8a c0 68  cc 39 af c1 08 06 00 01   8······h ·9······
wlan0: <live capture in progress>    Packets: 10830 · Displayed: 10830 (100.0%)    Profile: Default
```

Fig. 20 Packet capture

The Python parser application reads and parse the pcap file and extracts and presents the source and destination IP addresses from all the packets. The extracted destination IP addresses are compared to a blacklist of IP addresses. If an address match is found, the corresponding source and destination IP addresses are used to obtain geographical location information via the pyGeoIP python module, and the geolocation details are printed to the output.



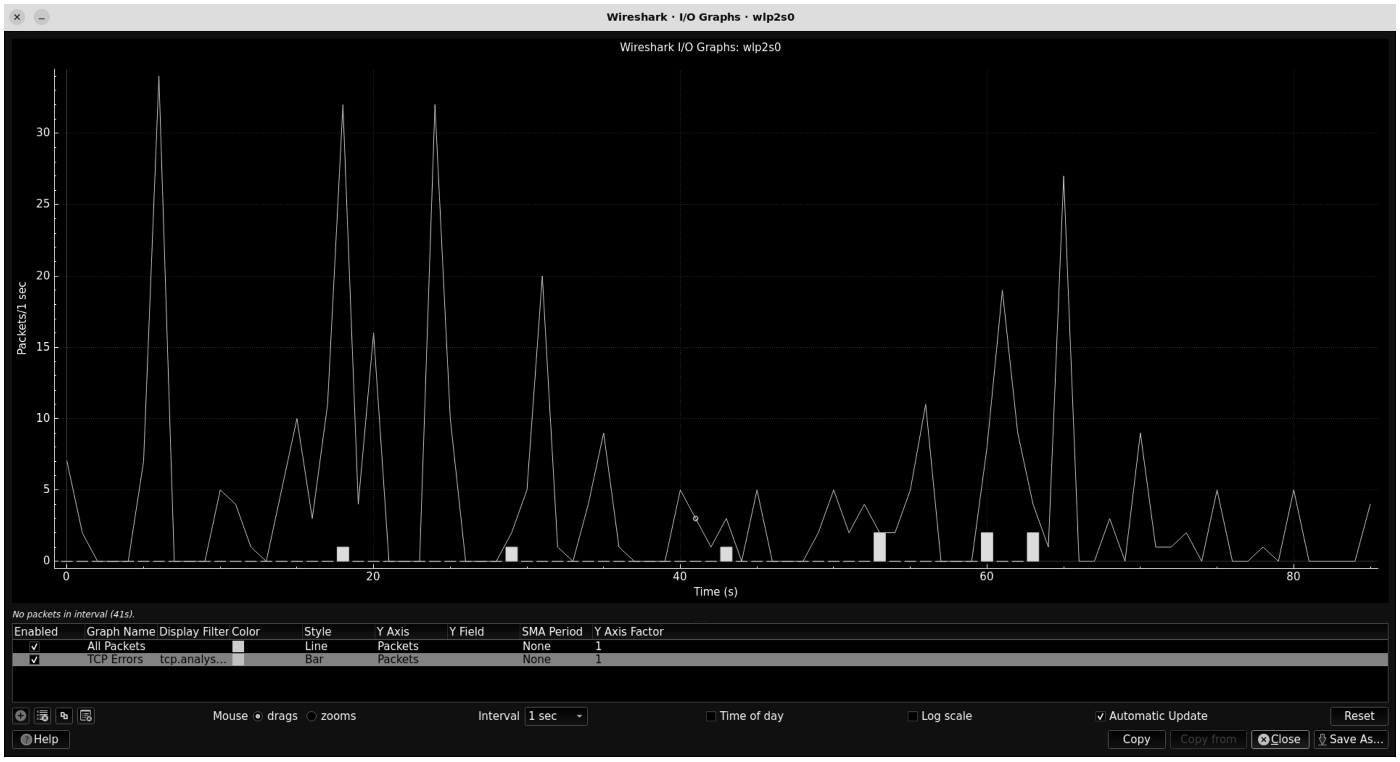
```
Apply a display filter ... <Ctrl-/>
No.   Time   Source   Destination   Protocol   Length Info
▾ Frame 1652: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface wlp2s0, id 0
    Section number: 1
  ▸ Interface id: 0 (wlp2s0)
    Encapsulation type: Ethernet (1)
    Arrival Time: May  4, 2023 10:41:04.268537616 IST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1683177064.268537616 seconds
    [Time delta from previous captured frame: 0.042884363 seconds]
    [Time delta from previous displayed frame: 0.042884363 seconds]
    [Time since reference or first frame: 113.292806494 seconds]
    Frame Number: 1652
    Frame Length: 86 bytes (688 bits)
    Capture Length: 86 bytes (688 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ipv6:tcp]
    [Coloring Rule Name: Bad TCP]
    [Coloring Rule String: tcp.analysis.flags && !tcp.analysis.window_update && !tcp.analysis.keep_alive &
▾ Ethernet II, Src: c0:68:cc:39:af:c1 (c0:68:cc:39:af:c1), Dst: IntelCor_08:e0:8a (38:ba:f8:08:e0:8a)
  ▸ Destination: IntelCor_08:e0:8a (38:ba:f8:08:e0:8a)
  ▸ Source: c0:68:cc:39:af:c1 (c0:68:cc:39:af:c1)
    Type: IPv6 (0x86dd)
▸ Internet Protocol Version 6, Src: 2606:4700:9a92:722c:70ae:0:b91a:16a, Dst: 2405:201:a428:9019:cc85:be61
▾ Transmission Control Protocol, Src Port: 443, Dst Port: 57376, Seq: 1, Ack: 2, Len: 0
    Source Port: 443
    Destination Port: 57376
    [Stream index: 19]
    [Conversation completeness: Incomplete (4)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 357922205
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 2    (relative ack number)
    Acknowledgment number (raw): 1267721571
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window: 26
    [Calculated window size: 26]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x01de [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [Timestamps]
  ▸ [SEQ/ACK analysis]
```

```
0000  38 ba f8 08 e0 8a c0 68  cc 39 af c1 86 dd 60 0e   8······h ·9····`·
0010  7f 5b 00 20 06 3a 26 06  47 00 9a 92 72 2c 70 ae   ·[· ·:&· G···r,p·
0020  00 00 b9 1a 01 6a 24 05  02 01 a4 28 90 19 cc 85   ·····j$· ···(····
0030  be 61 85 c1 05 6f 01 bb  e0 20 15 55 75 9d 4b 8f   ·a···o·· · ·Uu·K·
0040  e5 63 80 10 00 1a 01 de  00 00 01 01 08 0a 45 0e   ·c······ ······E·
0050  8e aa 37 3c b6 b6                                  ··7<··
```

```
wlp2s0: <live capture in progress>        Packets: 1965 · Displayed: 1965 (100.0%)        Profile: Default
```

Fig. 21 Header Details

HTTP Traffic analysis:



```
No packets in interval (41s).
Enabled  Graph Name   Display Filter  Color  Style  Y Axis   Y Field  SMA Period  Y Axis Factor
✓        All Packets                  ⬜     Line   Packets           None        1
✓        TCP Errors   tcp.analys...   ⬜     Bar    Packets           None        1

Mouse ● drags ○ zooms    Interval 1 sec    ☐ Time of day    ☐ Log scale    ✓ Automatic Update
```

Graph 1. Data Flow Graph

In addition to determining and finding the geolocations of blacklisted IP addresses feeded by the user, our project examines the HTTP traffic in the pcap file. This approach is adopted to analyze the user behaviour on the network or the Internet. The following figure shows the sample HTTP Request and Response in a packet in the capture file or the pacp file extracted with the help of wireshark.

The HTTP Request obtained in the packet from the pcap file is depicted in the screenshot above.

Major Project
Last edit was seconds ago
Add layer
Share
Preview
Untitled layer
Import
Add places to this layer by drawing or importing data. Learn more
Base map
Google My Maps

Fig. 22 Google Map

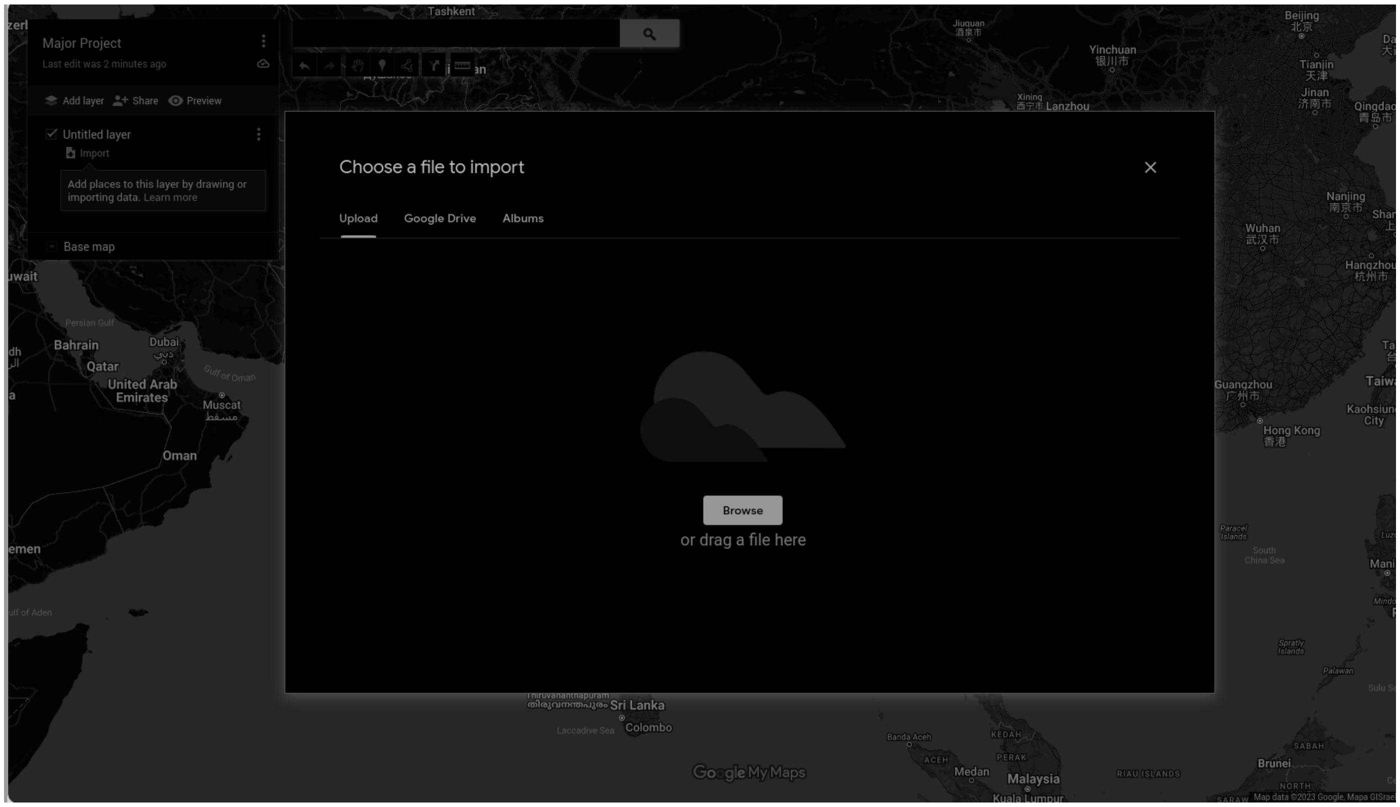
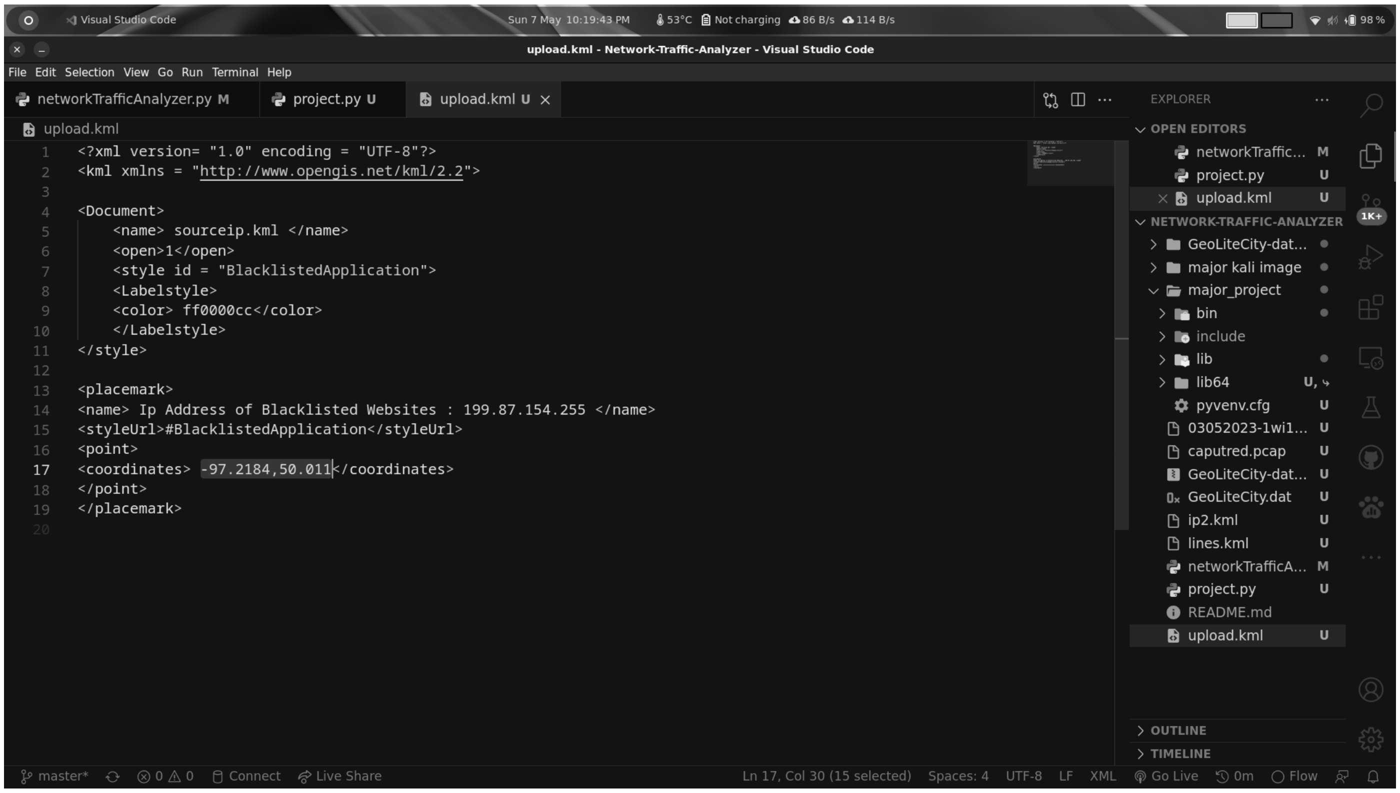


Fig. 23 Upload File

Keyhole Markup Language (KML) format:

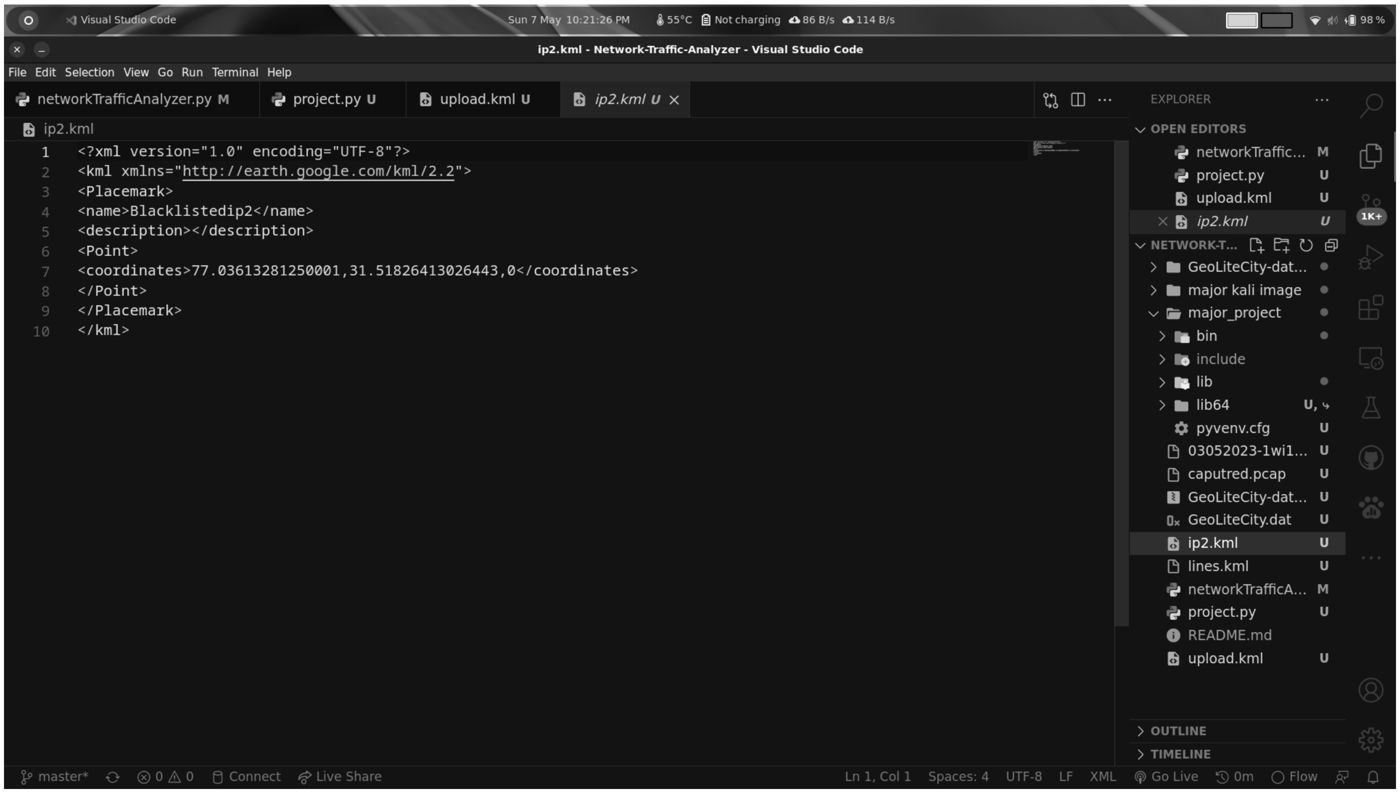
The figure shown below depicts the KML format output from the Python parser application. Keyhole Markup Language (KML) is an XML notation markup language developed by Google which can be assumed as an extension of the xml language that is used to display geolocation on Maps using latitude and longitude values. The Python parser reads the pcap file and extracts the source and destination IP addresses from each packet. The extracted destination IP addresses are compared to a list of IP addresses that have been blacklisted form the users perspective. If the addresses match, the corresponding source and destination are returned.

KML is a file format for displaying geographic data in an Earth browser such as Google Earth. KML is a tag-based structure with nested elements and attributes that is based on the XML standard. All tags are case-sensitive and must be written exactly as they appear in the KML Reference. The Reference section specifies which tags are optional. Within a given element, tags must appear in the systematic order as shown in the Reference.



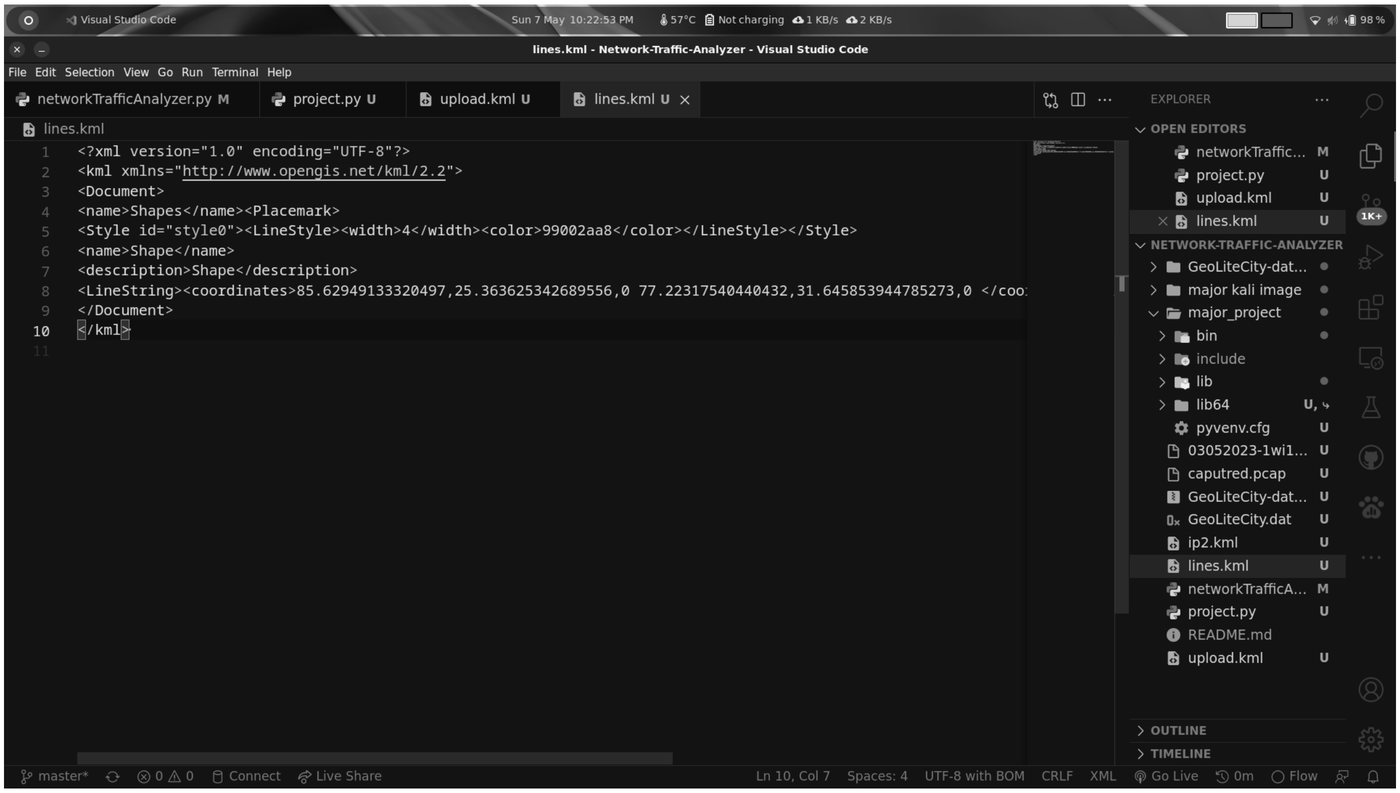
```
<?xml version= "1.0" encoding = "UTF-8"?>
<kml xmlns = "http://www.opengis.net/kml/2.2">

<Document>
    <name> sourceip.kml </name>
    <open>1</open>
    <style id = "BlacklistedApplication">
    <Labelstyle>
    <color> ff0000cc</color>
    </Labelstyle>
</style>

<placemark>
<name> Ip Address of Blacklisted Websites : 199.87.154.255 </name>
<styleUrl>#BlacklistedApplication</styleUrl>
<point>
<coordinates> -97.2184,50.011</coordinates>
</point>
</placemark>
```

Fig. 24 Upload.kml



```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Placemark>
<name>Blacklistedip2</name>
<description></description>
<Point>
<coordinates>77.03613281250001,31.51826413026443,0</coordinates>
</Point>
</Placemark>
</kml>
```

Fig. 25 Upload2.kml



lines.kml - Network-Traffic-Analyzer - Visual Studio Code
File Edit Selection View Go Run Terminal Help
networkTrafficAnalyzer.py M
project.py U
upload.kml U
lines.kml U
EXPLORER
OPEN EDITORS
NETWORK-TRAFFIC-ANALYZER
OUTLINE
TIMELINE

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
<name>Shapes</name><Placemark>
<Style id="style0"><LineStyle><width>4</width><color>99002aa8</color></LineStyle></Style>
<name>Shape</name>
<description>Shape</description>
<LineString><coordinates>85.62949133320497,25.363625342689556,0 77.22317540440432,31.645853944785273,0 </coo
</Document>
</kml>
```

Fig. 26 Lines.kml

If you're new to KML, read this document and the sample files that go with it (SamplesInEarth) to get a sense of the basic structure of a KML file and the most commonly used tags. The firThe first section describes the features that can be added to Google Earth using the useThe second section describes features that necessitate the use of a text editor to create KML. gons. The second section describes features that require authoring KML with a text editor. When a text file is saved with a .kml or .kmz extension, Earth browsers know how to display it.

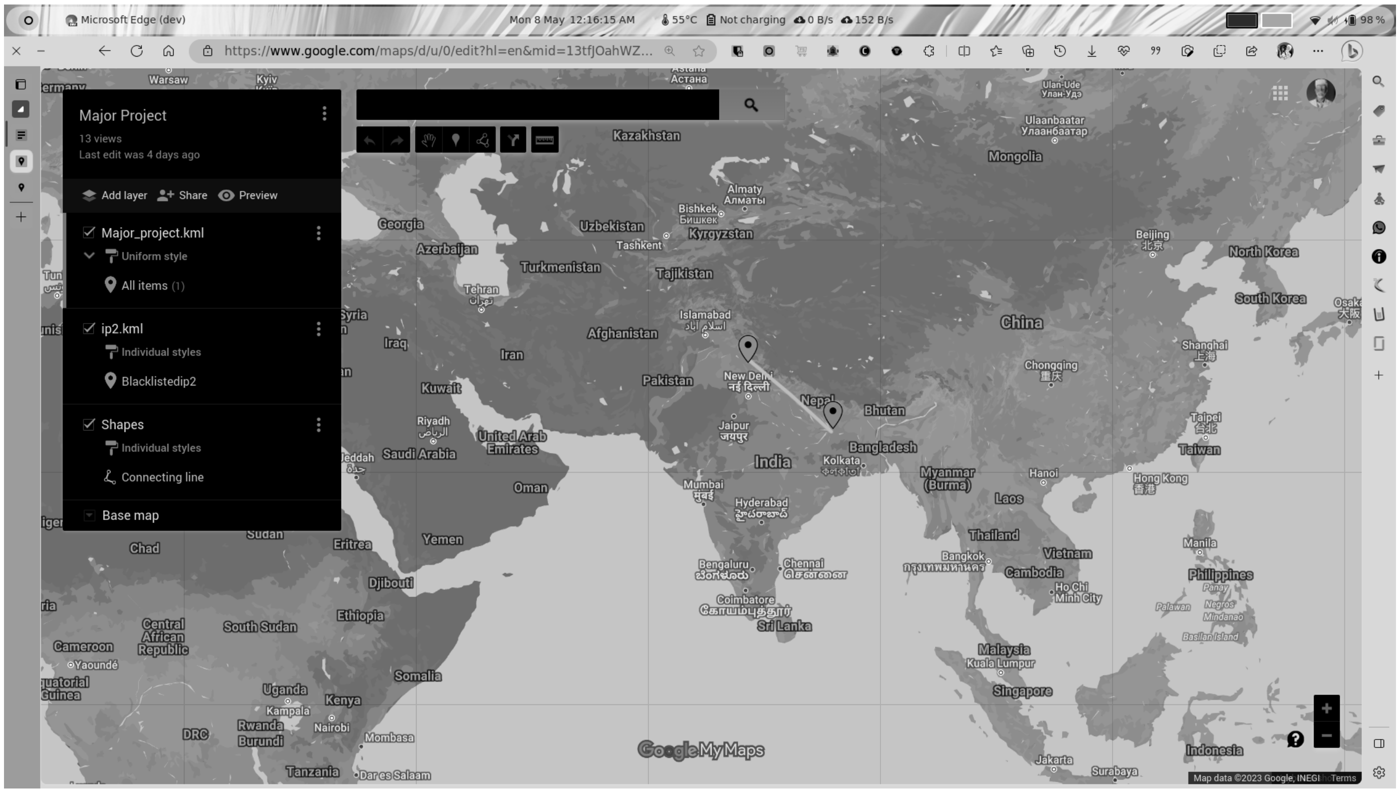


Fig. 27 Pointed Location on map

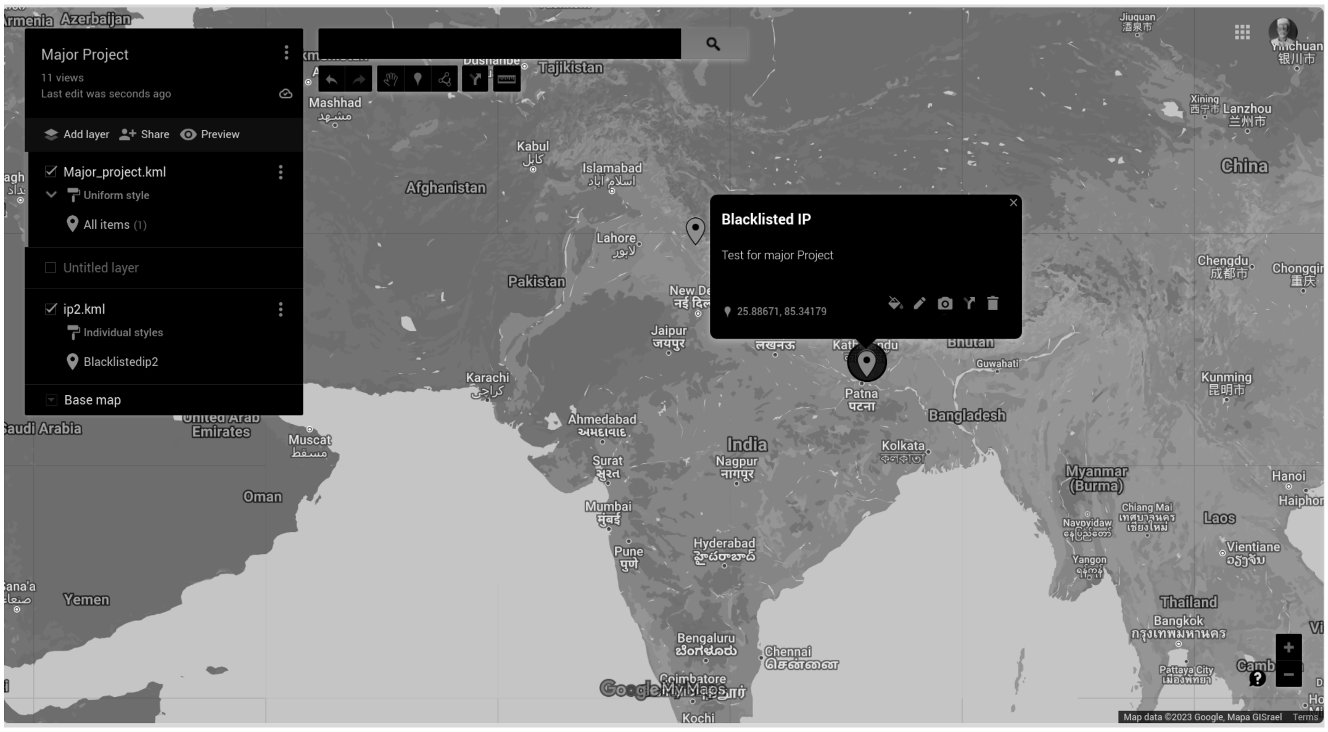


Fig. 28 Pointer indicating blacklisted Ip

The pyGeoIP python module is used to obtain geographical location information and details from destination IP addresses.This geolocation data from the database, which consists of latitude and longitude, is used to create the KML file which is used to present the information on the google maps. This geolocation data, which consists of latitude and longitude, is used to create the KML file.

The KML file is then uploaded into Google Maps to display IP addresses with map-marker icons representing every source or destination IP address that is either accessing or belongs to the blacklisted category of websites. The output of Google Maps after loading a KML file is shown below.

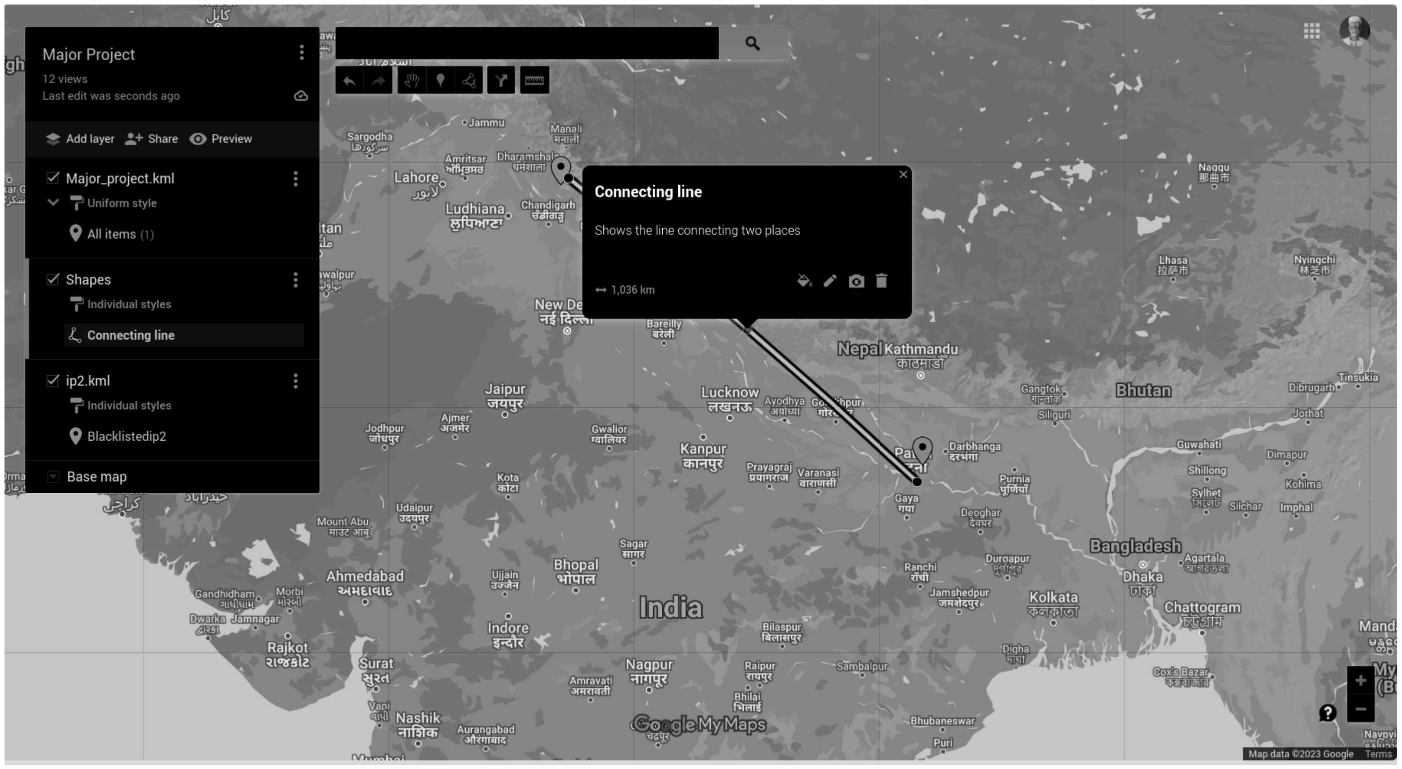


Fig. 29 Line on map

CHAPTER - 5

5.1 CONCLUSIONS

In this project, we created and implemented a Network Traffic Analyzer that help in parsing network traffic using an application code script written in python, extracted IP addresses and other useful information from users who are attempting to access malicious websites, and also identify the IP address of malicious websites using a Command line interface (.cli) format file. The geographical locations were also obtained with the help of a Keyhole Markup Language (.kml) format file. We choose to create this application using a Python script and the database named as GeoLiteCity database, which contains all of the information about locations all over the world. We also used helpful tools like Google Maps and Wireshark. PyGeoIP and DPKT python modules were among the libraries which we used extensively.

Talking about the practical approach, there isn't a single common network problem that can't be discovered and solved with packet tracing technology. It can be used as the first line of protection or defence against a variety of problems ranging from overloaded networks to unresponsive switches to lost or dropped packets. As the number of networks and nodes grows and network speeds increase, it becomes increasingly difficult to monitor a LAN using traditional tools like RMON (Remote Monitoring) probes. Packet sniffers, on the other hand, monitor network traffic down to the Header information on each several series of data. This means that you can follow as well as track data from its beginning to its end. Packet sniffers can also be used to identify and detect the types of packets on a network and determine whether a particular packet contains any errors or is mislead.

5.2 FUTURE SCOPE

The scope of our project can be extended in the following ways:

- Adding more functionalities to the current features
- Adding new features

1. Adding more features to the existing ones.

We are using the GeoLiteCity database by MaxMind.com in its free version inside our project. GeoIP database, a commercially used and complete version of the IP geolocation database, can be used in its place which provides more precise geographical location which is the value of longitude and the latitude information for blacklisted websites which we are intended to locate and their users.

The application has been installed and tested on the local device and servers. This can be deployed on cloud infrastructure like Amazon AWS, google cloud and accessed from various different systems and can also be accessed from different locations or IPs'.

2. Introducing new functions

A pcap file is basically a dump of tcp packets or it can also be called as a repository that contains a large amount of information about computer network communication between different nodes. The packets can contain sensitive and personal information which includes user credentials (username and password). The packets in the pcap file can be used to examine network attack behaviour and methods. When a large number of packets originate from a single source IP address, a Denial Of Service (DOS) attack is detected and hence restrictive measures could be taken to prevent it. This feature can also be added to the current application.

REFERENCES

- AL.Jeeva, Dr.V.Palanisamy, K.Kanagaram “Comparative Analysis Of Performance Efficiency And Security Measures Of Some Algorithms” International Journal Of Engineering Research And Applications (IJERA), Vol. 2, Issue 3, pp.3033-3037, May-June 2012.
- What’s New In Python 3.11 — Python 3.11.3 documentation
- Oliver Michel, Roberto Bifulco, Gabor Retvari, Stefan schmid. “The Programmable Data Plane : Abstraction, Architecture, Algorithms. And Applications”. Institute of Electrical and Electronics Engineers(IEEE), 2020
- W.D. Ivancic, D. Brooks, B. Frantz, D. Hoder, D. shell, D.Beering. “NASA’s broadband satellite networking research” IEEE communications Magazine, 1999
- Fei Liu, Ming Yang. "Verification and validation of all simulation systems", Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826), 2004
- A. Gill Waters. "LANs: protocols above the medium access layer", Institution of Engineering and Technology (IET), 1994
- R CAMEROON. "Networking, Security, and the Firewall", Configuring Juniper Networks NetScreen & SSG Firewalls, 2006
- "Information Security", Springer Science and Business Media LLC, 2009
- Vacca, . "Enterprise High-Speed LAN/WAN Cisco Internetworking Technology", High Speed Cisco Networks Planning Design and Implementation, 2001.