AMRITA NETWORK PARSER

A Project Report Submitted to Amrita School of Engineering in partial fulfilment of theRequirements for the Degree of Bachelor of Technology in Computer Science and Engineering (CYBER SECURITY):

CYBER FORENSICS (20CYS311)

Submitted by

ANNADANAM PADMASINI (CH.EN.U4CYS20006)
GUMMANUR ABHISHEK (CH.EN.U4CYS20025)

Under the Guidance of

Dr. Udhaya Kumar S

Assistant Professor

DEPARTMENT OF CYBER SECURITY

Amrita School of Computing

Amrita Vishwa Vidyapeetham

Chennai – 601 103, Tamil Nadu, India.

May, 2023.

ABSTRACT

Network forensics is the use of investigative techniques to analyse data that passes through a network in order to identify, monitor, and investigate security incidents. Network forensics is used to detect malicious activity such as malware, hacking, and data theft, as well as to analyse network performance issues. Network forensics tools are typically used to capture, store, and analyse network packet data, which can help in identifying the source of a security incident, the types of activities that occurred, and the impact of the attack.

Network forensics tools are used to capture, monitor and analyse network traffic or packets for the purpose of network security and troubleshooting. They can be used to investigate suspicious activities like malware, DDoS attacks, unauthorized access and more. They can also be used to detect anomalies in the network, such as unusual traffic patterns or suspicious user activity. Additionally, they can be used to reconstruct past activities, such as discovering the source of a breach or the malicious activity of an attacker.

The Amrita Network Parser is a network forensics project designed to analyze network traffic and extract important information that can aid in the investigation of cyber-attacks. This project includes a suite of tools that can parse and analyze network packets, reconstruct network sessions by using the data and can a replay attack can be performed, and extract various network-related artifacts such as IP addresses, number of bytes of data transferred.

Amrita NetParser analyses network logs using a CSV file as the input. The following elements are present in the csv file's format: time connection made with source, destination, and source and destination ports using the usual epoch value Total bytes transmitted, bytes received (from destination to source), and bytes sent (from source to destination).

INTRODUCTION:

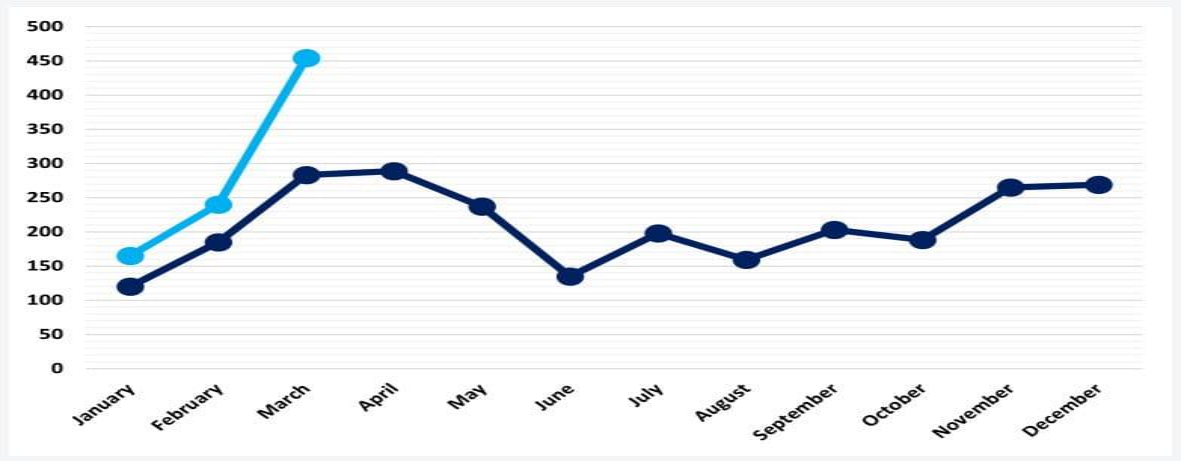
Resource sharing is the main aim of a computer network. It means to make all programs peripherals and data available to any one computer on the network to all other computers in the network without regard to the physical locations of them. Thus user at large distances can share the resources or can see data of a computer in the same way that a local user uses them. Network security is a set of technologies that protects the usability and integrity of a company's infrastructure by preventing the entry or proliferation within a network of a wide variety of potential threats. Network forensics is a subset of digital forensics that deals with the collection and analysis of network traffic with the goal of better understanding and avoiding cybercrime. The importance of network forensics has grown in recent years, according to a report from the European Union Agency for Cybersecurity (ENISA), with the emergence and popularity of network-based services such as e-mails, Directory services, World Wide Web, and others.

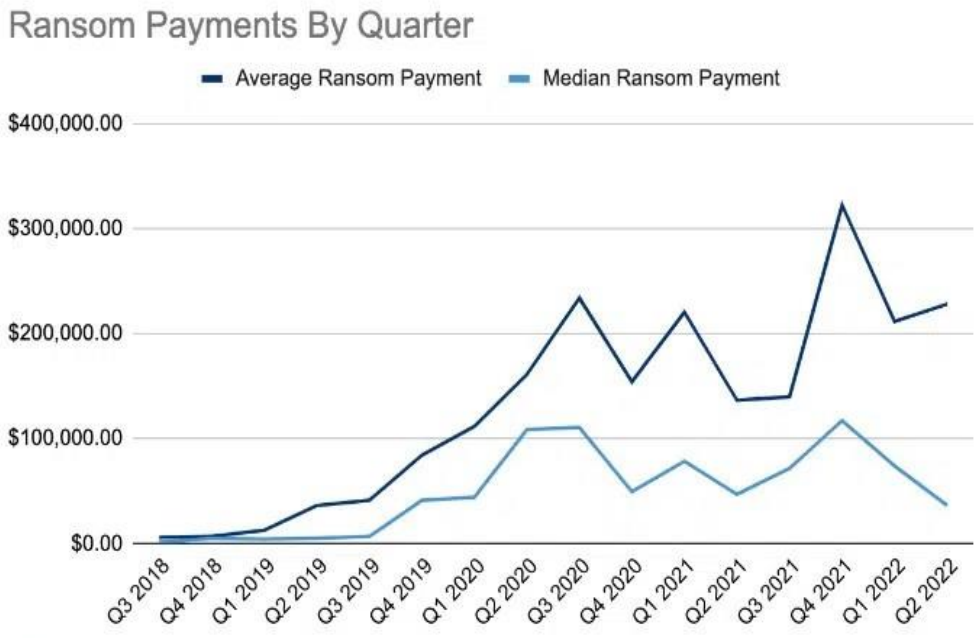
Network forensics is used to detect malicious activity such as malware, hacking, and data theft, as well as to analyse network performance issues. Network forensics tools are typically used to capture, store, and analyse network packet data, which can help in identifying the source of a security incident, the types of activities that occurred, and the impact of the attack.

It relates to the monitoring and analysis of computer network traffic for the purposes of information gathering, legal evidence, or intrusion detection. Unlike other areas of digital forensics, network investigations deal with volatile and dynamic information. Network traffic is transmitted and then lost, so network forensics is often a pro-active investigation.

NEED OF NETWORK PARSER:

In recent years, news stories about malware attacks have virtually become a weekly occurrence around the world. In 2020, the U.S. had a 139% year-over-year increase to 145.2 million documented attacks, while malware attacks decreased in certain other countries (Help Net Security, 2020). According to Sobers (2021) the average malware payment increased by 33% to $111,605 USD in 2020, while malware attacks on corporations typically cost $133,000 USD. The United States is not exempt from this wave of attacks, and Canada has not been an exception. According to Canada's cyber security agency, "attacks against critical Canadian business and infrastructure are 'almost certain' to continue, as Canada has already seen its systems targeted by such attacks in recent years" (Gilmore, 2020).





The Emotet malware, which actively introduces additional software, such as malware, onto the system, has been one of the most effective tools for malware infection. It can itself collect and communicate data about the computers it infects. Malicious actors can now use the Emotet virus in its current form as a "Malware-as-a-Service" that can be customised with a variety of malicious payloads because it has historically been challenging to detect in computer systems (Petcu, 2021).

This report will explore Amrita NETparser, a powerful network analyser, as a forensic tool to examine and discuss the network traffic that is generated by an Emotet infection and suggest methods for early detection.

WHAT IS AMRITA NETWORK PARSER?

Parsers break the input they get into parts such as the nouns (objects), verbs (methods), and their attributes or options. These are then managed by other programming, such as other components in a compiler. A parser may also check to ensure that all the necessary input has been provided.

In parsing, code is taken from the preprocessor, broken into smaller pieces and analyzed so other software can understand it. The parser does this by building a data structure out of the pieces of input.

A network forensics project called the Amrita Network Parser aims to analyse network traffic and extract crucial data that can help with the analysis of cyberattacks. This project offers a set of tools that can parse and analyse network packets, reconstruct network sessions using the data and perform replay attacks, and extract different network-related artefacts including IP addresses and the amount of data transported.

Network logs are analysed by Amrita NetParser using a CSV file as the input. The csv file format includes the following elements: a time connection created with a source, destination,

and source and destination ports using the standard epoch value Total bytes communicated, bytes received (from destination to source), and bytes sent (from source to destination).

LITERATURE SURVEY:

A literature survey on network parser involves researching and analyzing existing research studies, academic papers, conference proceedings, and other relevant publications on the topic. The following are some of the sources that can be used for such a survey:

- "Network Traffic Parsing: A Survey" by B. Cheng, L. Lu, and Y. Luo, published in the IEEE Communications Surveys & Tutorials in 2019.
- "Network Packet Parsing and Protocol Identification" by A. C. Snoeren and K. Conley, published in the ACM SIGCOMM Computer Communication Review in 2005.
- "Network Packet Parsing: From Theory to Practice" by X. Zhang, M. Luo, and W. Zhou, published in the IEEE Communications Magazine in 2018.
- "A Comprehensive Study on Network Packet Parsing Techniques" by S. Shah and S. Kumar, published in the International Journal of Advanced Research in Computer Science in 2016.
- "Deep Packet: A Novel Network Traffic Parser Based on Deep Learning" by Y. Chen, Z. Zhang, and X. Li, published in the IEEE Transactions on Network and Service Management in 2021.
- "A Heuristics-Based Network Protocol Parsing Method" by Y. Li and X. Zhang, published in the Journal of Network and Computer Applications in 2021.
- "A Comparative Study of Network Protocol Parsing Techniques" by M. M. Rahman and M. S. Hossain, published in the Journal of Network and Computer Applications in 2018.
- "Network Forensics: Review of Current Status and Future Directions" by K. R. Kanchana and S. S. Iyengar, published in the International Journal of Computer Applications in 2014.
- "A Review of Network Forensic Tools" by H. Almohri and A. Alnuaimi, published in the International Journal of Computer Science and Mobile Computing in 2015.
- "Network Forensics: A Survey" by P. R. Pacharne and K. N. Kumar, published in the International Journal of Engineering and Technology in 2016.
- "A Survey of Network Forensic Tools and Techniques" by V. K. Jha and D. D. Shetty, published in the International Journal of Engineering Research and Applications in 2017.
- "A Review of Open-Source Network Forensic Tools" by T. A. Gharawi and S. S. Al-Salman, published in the International Journal of Computer Science and Information Security in 2018.
- "A Comprehensive Review of Network Forensic Tools and Techniques" by M. A. Mahmood and M. A. Hossain, published in the International Journal of Computer Networks and Communications in 2019.
- "A Survey of Machine Learning-Based Network Forensic Tools" by S. Ahmed and M. S. Hossain, published in the Journal of Network and Computer Applications in 2020.

These studies provide a comprehensive overview of the existing network parsing techniques, including both traditional and advanced methods. They discuss the latest trends and developments in this field, identify the challenges involved in network parsing, and propose solutions to improve the accuracy and efficiency of network parsing. Additionally, some studies focus on specific techniques such as deep learning-based methods and heuristics-based approaches and compare their performance against traditional methods.

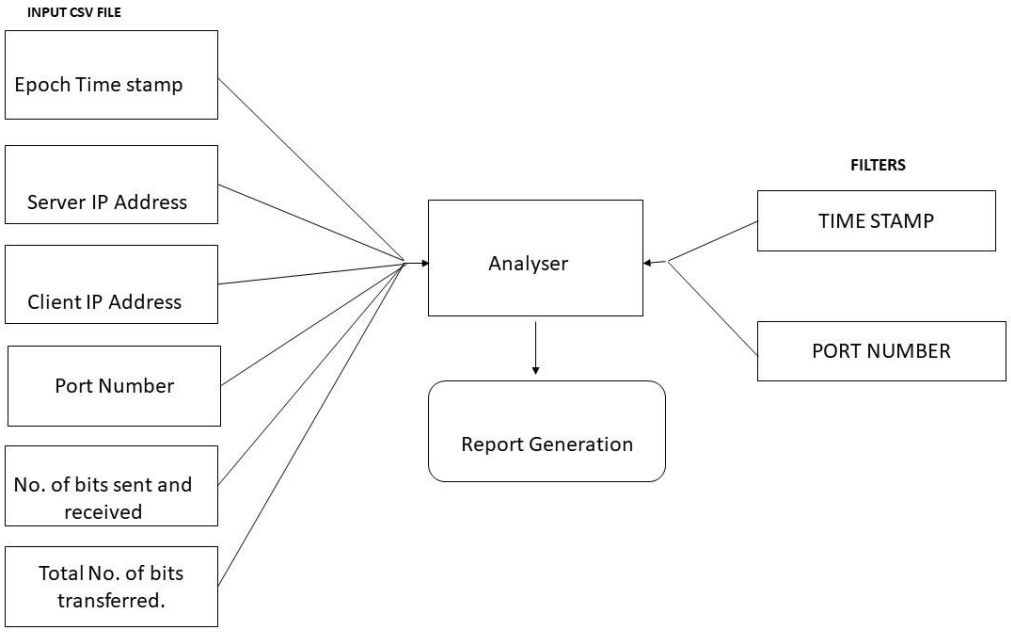
MAPPING WITH AVAILABLE TOOLS:

Network parsers are tools used to extract meaningful information from network traffic data. Several previous works have been done on network parsers, and some of them are discussed below:

- ✓ Tcpdump: Tcpdump is a widely used network packet analyzer that can capture, filter, and display network traffic. It is a command-line tool that operates by capturing packets directly from the network interface card (NIC).
- ✓ Wireshark: Wireshark is a popular network protocol analyzer that allows users to capture, view, and analyze network traffic. It can capture and decode multiple protocols, including TCP, UDP, HTTP, and DNS, among others.
- ✓ Bro: Bro is an open-source network analysis framework that can capture, parse, and analyze network traffic in real-time. It supports a wide range of protocols and can generate detailed network activity logs.
- ✓ Zeek: Zeek, formerly known as Bro, is an open-source network security monitoring tool that can analyze network traffic in real-time. It can detect and analyze various network protocols, including FTP, HTTP, DNS, and SMTP, among others.
- ✓ Suricata: Suricata is an open-source network intrusion detection and prevention system that can analyze network traffic in real-time. It can detect and analyze various network protocols and can generate detailed network activity logs.
- ✓ NetworkMiner: NetworkMiner is a free and open-source network forensic analysis tool that can parse network traffic and extract files and other relevant information from network packets. It can analyze various protocols, including FTP, HTTP, DNS, and SMTP, among others.
- ✓ Snort: Snort is an open-source network intrusion detection and prevention system that can analyze network traffic in real-time. It can detect and analyze various network protocols and can generate alerts for potential security threats.

These tools have been extensively used and researched for network parsing, and each of them has its own strengths and weaknesses. Researchers have also developed several techniques to improve the accuracy and efficiency of network parsing, such as deep learning-based methods and heuristics-based approaches.

SYSTEM ARCHITECTURE:



PROPOSED ALGORITHM:

1. CSV file using the "csv.reader" function.
2. Create four empty lists called "ipdict", "iplist", "plist", and "flist", and set the "conn" variable to 0.
3. Print the name of the source file that was specified as a command-line argument.
4. Display a menu to ask the user how they want to filter the data.
5. If the user chooses to filter Import necessary libraries such as csv, sys, operator, datetime, re, time, itertools, and colorama.
6. Define a function called "animate" to animate the program output.
7. Create a dictionary called "MONTH_DICT" to map month names with their corresponding numbers.
8. Check if the command-line arguments are valid or not.
9. Open the specified by port numbers:

a. Ask the user how many ports they want to evaluate and create an empty list called "ports".

b. Get the port numbers from the user and append them to the "ports" list.

c. Loop through each row of the CSV file and check if the port number is in the "ports" list.

d. If the port number matches, then check if the IP address is already in the "iplist" list.

e. If it is not in the list, then add it to the "iplist" list and check if the source IP address is already in the "slist" list.

f. If it is not in the list and the IP address is already in the "iplist" list, then add it to the "slist" list.

g. If the "conn" variable is 0, then set it to the current row's timestamp.

10. If the user chooses to filter by timestamps:

a. Ask the user to input the start and end timestamps in epoch format and store them in variables called "a" and "b".

b. Loop through each row of the CSV file and check if the timestamp is between the start and end timestamps provided by the user.

c. If the timestamp is within the range, then check if the IP address is already in the "iplist" list.

d. If it is not in the list, then add it to the "iplist" list and check if the source IP address is already in the "slist" list.

e. If it is not in the list and the IP address is already in the "iplist" list, then add it to the "slist" list.

f. If the "conn" variable is 0, then set it to the current row's timestamp.

11. If the user selects an invalid option, print "invalid input."
12. Create a new list called "plist" by splitting the IP addresses in "iplist" by ".".
13. Sort the "plist" list based on the last octet of the IP address.
14. Create a new list called "flist" by joining the elements of each IP address in "plist" with ".".
15. Print the final output.

EXPERIMENTAL ANALYSIS:

The csv file 'malwarelogsample.csv' that is been taken as the input for the python code that analysed based on the filter given by the user generates the document about the system.The csv file that is taken as input is as follows

Tabular representation of the csv file is as follows:

Time Stamp (Epoch Value)	Source IP ADDRESS	Destination IP ADDRESS	PORT NUMBER SOURCE	PORT NUMBER DESTINATION	Number of bits (data) sent	Number of bits (data) received	Total number of bits transferred (Sent + Received)
1.59E+09	10.10.10.67	255.81.33.119	49958	53	1733	147750	149483
1.59E+09	10.10.10.19	255.244.13.39	49255	1337	29381980	242	29382222
1.59E+09	10.10.10.32	255.166.53.137	56501	23	3808	300133	303941
1.59E+09	10.10.10.80	255.126.237.185	51235	22	2199	5214230	5216429

The data in the csv file is huge in number, we have 1004636 entries that includes the data about each system, therefore the administrator or the security analyst will take very long time to verify log of each system, hence, we need this tool in order to analyse the data and provide precise information and generate required document for forensic use.

This is a Python script that analyzes a log file and provides information about the IP addresses and ports mentioned in it. The script takes a log file as input, and then performs some analysis on the data in the file.

The script uses the following modules:

- **csv**: for reading the log file as a CSV file.
- **sys:** for accessing command-line arguments and standard input/output.
- **operator**: for sorting lists.
- **datetime**: for working with dates and times.
- **re**: for working with regular expressions.
- **time:** for working with time intervals.
- **itertools:** for creating animation cycles.
- **colorama**: for adding colour to the output text.

The script starts by defining a dictionary that maps month numbers to month names. It then checks the number of command-line arguments and prints an error message if there are too many or too few arguments.

If there is one command-line argument, the script opens the log file and initializes several lists and variables that will be used for storing data and counting occurrences. It then displays an animation that indicates that it is processing the log file.

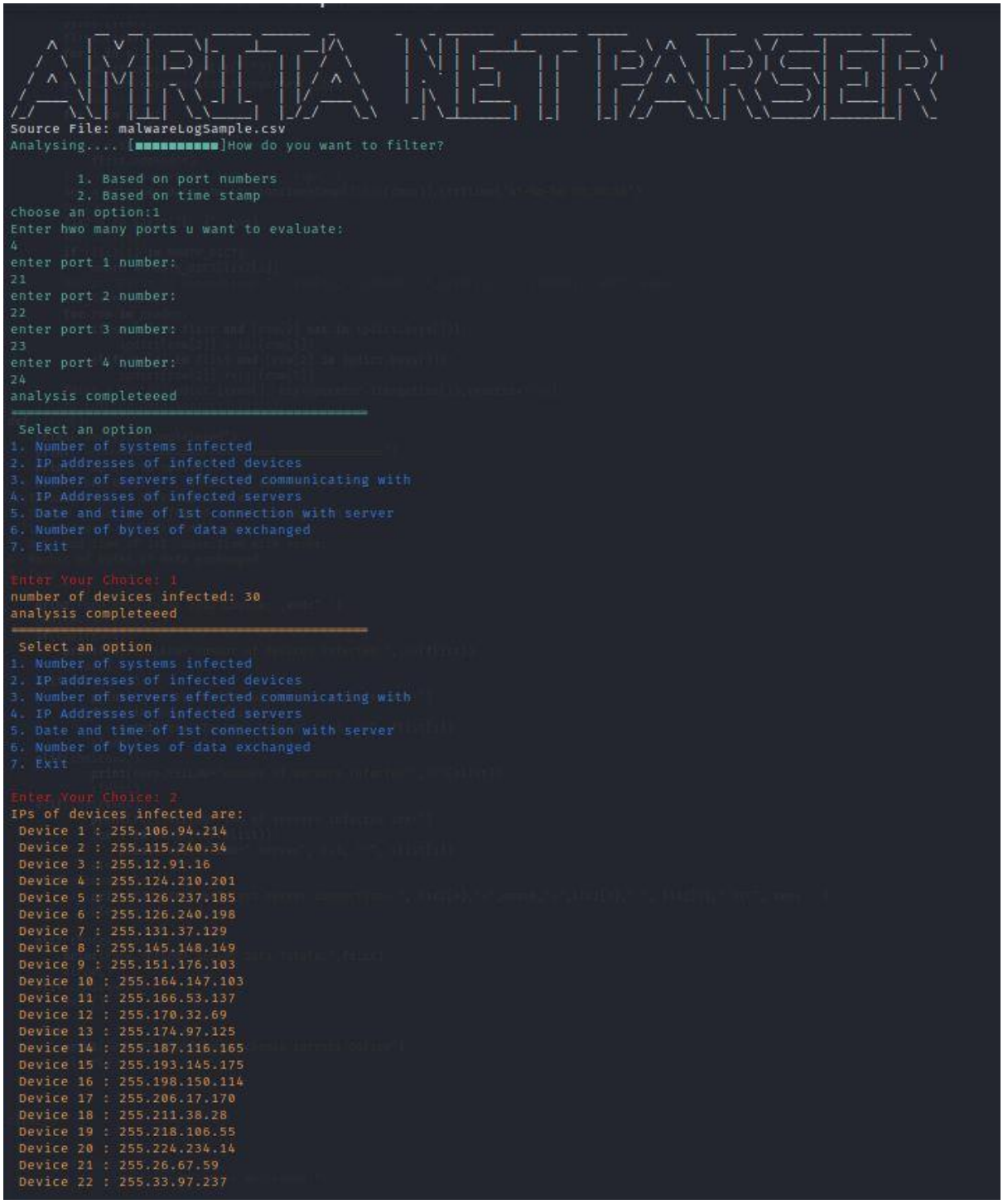
Next, the script prompts the user to choose between two filtering options: filtering based on port numbers or based on time stamps. If the user chooses port numbers, the script prompts the user to enter the number of ports they want to evaluate, and then asks the user to input the port numbers. It then iterates over the rows of the log file and adds the IP address and source port to their respective lists if the row's port number matches one of the user's selected ports.

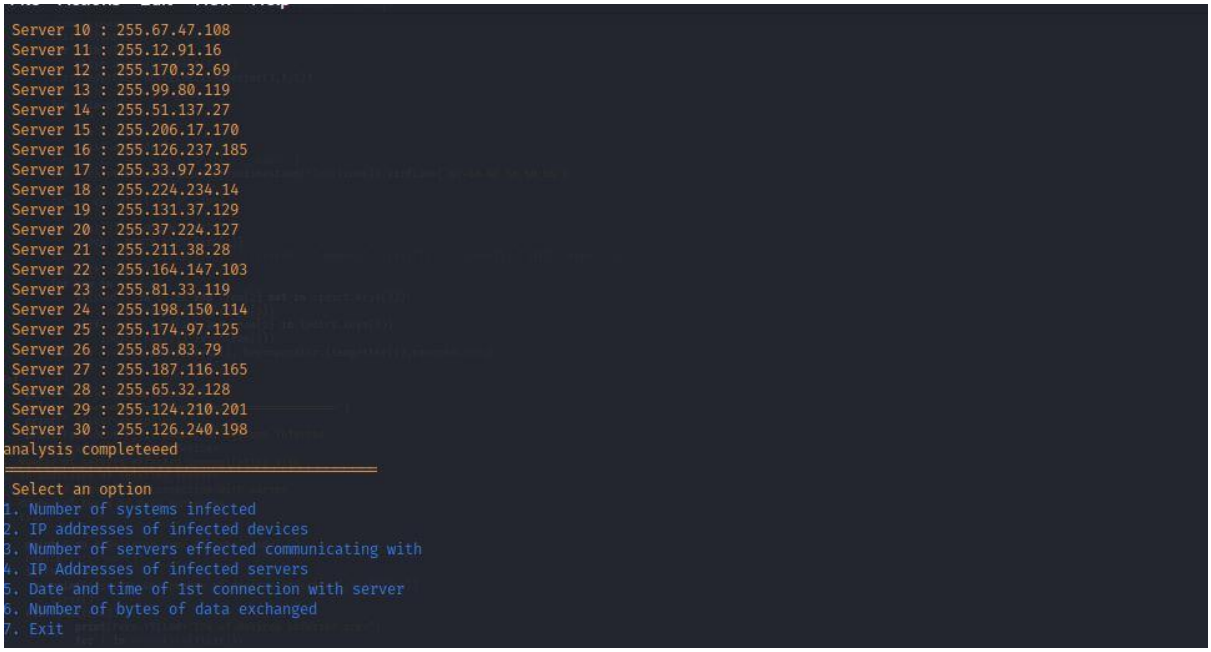
If the user chooses time stamps, the script prompts the user to enter the start and end time in epoch time format. It then iterates over the rows of the log file and adds the IP address and source port to their respective lists if the row's time stamp falls within the user's selected time range.

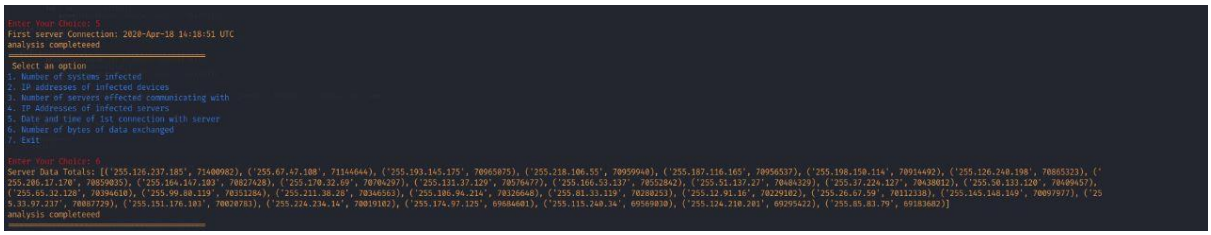
Once the IP address and source port lists are populated, the script converts the IP addresses into a sortable format, sorts them by IP address, and converts them back to their original format. Finally, the script prints the results of the analysis.

In addition to the main logic, the script also defines a function for animating a progress bar and prints a colourful header at the beginning of the output.

OUTUT OF THE PROGRAM:







CONCLUSION:

Amrita netparse is a highly specialized tool designed to analyze and provide detailed reports on malware attacks. It utilizes a range of filters, including those based on timestamp and port number, to identify and generate reports on various types of malware attacks. Users can provide a CSV file and specify the filter types they want to use to get the desired report.

There are many network parsing tools available, ranging from command-line based tools like Tcpdump and Tshark to GUI-based tools like Wireshark and NetworkMiner, and more advanced tools like Bro and Suricata. Each tool has its unique features and capabilities, making them suitable for different use cases. Therefore, it is crucial to understand the strengths and limitations of different network parsing tools and select the appropriate tool for a specific task. Overall, network parsing tools play a crucial role in network security and forensic investigations and are an asset for security professionals.

One of the main advantages of Amrita netparse is that it can be integrated into Kali, an operating system commonly used for penetration testing and ethical hacking. As a command-line tool, it offers a straightforward interface that makes it easy for users to access and use the various features.

The increasing prevalence of cyber-attacks in today's world makes it imperative for individuals and organizations to have access to sophisticated tools like Amrita netparse. Such tools help users to stay informed and aware of the latest malware threats and to take proactive steps to protect their systems and data.

In conclusion, Amrita netparse is a powerful and reliable tool that provides an effective way to analyze and report on malware attacks. With its range of filters and customizable features, it offers a comprehensive solution for individuals and organizations looking to protect themselves against cyber threats. Its integration with Kali and easy-to-use interface makes it a must-have tool for anyone concerned about the security of their digital assets.

Overall, the script provides a basic functionality to filter and extract information from log files, making it easier to analyze and interpret the data. However, there is scope for improvement such as handling errors more gracefully, providing more options for filtering and displaying more detailed information from the log files.

REFERENCES:

i. "Practical Malware Analysis" by Michael Sikorski and Andrew Honig, published by No Starch Press in 2012.
ii. "The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory" by Michael Hale Ligh, Andrew Case, Jamie Levy, and AAron Walters, published by Wiley in 2014.
iii. "Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code" by Michael Hale Ligh, Steven Adair, Blake Hartstein, and Matthew Richard, published by Wiley in 2010.
iv. "Windows Malware Analysis Essentials" by Victor Marak, published by Packt Publishing in 2015.
v. "The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler" by Chris Eagle, published by No Starch Press in 2011.
vi. "Network Forensics: Tracking Hackers Through Cyberspace" by Sherri Davidoff and Jonathan Ham, published by Prentice Hall in 2012.
vii. "Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems" by Chris Sanders, published by No Starch Press in 2017.
viii. "The Practice of Network Security Monitoring: Understanding Incident Detection and Response" by Richard Bejtlich, published by No Starch Press in 2013.
ix. "Investigating Network Intrusions and Cybercrime: A Cybersecurity Professional's Guide" by Scott J. Roberts, published by Wiley in 2018.

WEB REFERENCES:

1. Tcpdump: https://www.tcpdump.org/
2. Wireshark: https://www.wireshark.org/
3. Bro: https://www.bro.org/
4. Zeek: https://zeek.org/
5. Suricata: https://suricata-ids.org/
6. NetworkMiner: https://www.netresec.com/?page=NetworkMiner
7. Snort: https://www.snort.org/
8. Tshark: https://www.wireshark.org/docs/man-pages/tshark.html