A Project Report

on

NETWORK PACKET SNIFFER

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Dr. N. Gayathri
Associate Professor

Submitted By

Ayushi Srivastava
18SCSE1010364

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING / DEPARTMENT OF COMPUTER APPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
MAY, 2022

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"NETWORK PACKET SNIFFER"** in partial fulfillment of the requirements for the award of the B-TECH_submitted to the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of May. Year to Month and Year, under the supervision of Dr. N. Gayathri (Associate Professor), Department of Computer Science and Engineering/Computer Application and Information and Science, of School ofComputing Science and Engineering, Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Ayushi Srivastava, 18SCSE1010364

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. N. Gayathri

Ass. Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Ayushi Srivastava: 18SCSE1010364 has been held on 13th May and his/her work is recommended for the award of B-Tech CSE.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: May, 2022

Place: Greater Noida

Acknowledgment

I would like to express my gratitude to everyone who has assisted me in this attempt. I would not have completed this assignment without their practical help, continual guidance, and encouragement. I owe a huge debt of gratitude to Dr. N. Gayathri, Ma’am, for his real assistance and support in completing this project. I thank the University of Galgotias for providing me with this chance. I also want to express my gratitude to my parents and relatives, who have always been financially and morally supportive of me. Last but not least, I'd like to express my gratitude to all of my friends who assisted me in completing my project report. Any omissions in this brief acknowledgment are not indicative of a lack of gratitude.

Abstract

Packet sniffing is a technique for exploiting each packet as it travels over the network. Network analysis is one of the most difficult issues that network managers encounter. Existing technologies for network traffic analysis provide very little information, which would be massive data if it were all kept for subsequent study, making it impossible to evaluate. The goal of this research is to present a sniffing tool that can collect both IPv4 and IPv6 packets. The suggested tool uses the socket class in Visual Studio to access the collected packets. The first scenario involves capturing packets and identifying the ports, protocols, and packets utilized in IPV4. The second scenario, on the other hand, examines IPV6 in the same way. The number of captured protocols and the utilized ports for both source and destination ports differ from IPv4 to IPv6.

Table of Contents

List of Table

List of Figures

Acronyms

B.Tech.	Bachelor of Technology
M.Tech.	Master of Technology
BCA	Bachelor of Computer Applications
MCA	Master of Computer Applications
B.Sc. (CS)	Bachelor of Science in Computer Science
M.Sc. (CS)	Master of Science in Computer Science
SCSE	School of Computing Science and Engineering

CHAPTER-1

Introduction

Managers and administrators of networks employ a packet sniffing tool to monitor data transported over the network. Unauthorized users might use packet followers to steal information from the network. Packet followers are used for network security and network administration (Xu et al., 2016). A packet analyzer can show a wide range of data delivered over the network, as well as the network traffic (Singh and Kumar, 2018). Packet sniffers are hardware or software devices that can record both incoming and outgoing network traffic, as well as screen and username and password information and other sensitive data (Anu and Vimala, 2017).

A packet sniffer allows you to configure the network interface such that it displays all data sent across the network. Using sniffer tools, data sent or received across the network can be captured for study (Chauhan and Sharma, 2014). A sniffer intercepts data and records packets, comparing them to header formats to identify standardized components and extract needed parts such as used ports, addresses, and so on. There are numerous network analysis tools available to everyone, where the user's intent cannot be determined as to whether it is for good, useful, or destructive purposes (Davis and Clark, 2011). Hackers might employ a program that can collect user credentials, but a network administrator might use the same tool to uncover network information like available bandwidth. Sniffer can also be used to evaluate web filters, firewalls, and client/server connections (Elsen et al., 2015).

Network-related topics now require actual labs to learn and comprehend protocol behavior (Gandhi et al., 2014). To fully comprehend these protocols, we must first construct an adequate environment for testing and confirming their behavior under various conditions. In this paper, we will create an environment that allows us to analyze network components including packet loss and latency. The best approach to change the characteristics of these networks is to use a software tool that allows you to do so quickly and easily; this instrument is a wide-area network emulator (also known as Sniffer) (Oluwabukola et al., 2013; Jaisinghani et al., 2017). Following the programming and implementation of this tool, certain tests will be run to analyze network elements by analyzing and identifying the protocols utilized in transceiver activities. Furthermore, the packet will be thoroughly examined in order to display the key message components by defining the variables' values. This will result in a comprehensive network study to better understand network and user behavior by identifying and collecting statistics on the most often used protocols and data length. The tool described in this paper assists in network analysis and provides a clear live statistic about the network with no wasted storage.

CHAPTER-2

Literature Review

At the transport layer level, Nishanth and Babu (2014) suggested a defense against "hijacking assaults." The web server and client must first establish a logical connection to the TCP layer, which is known as a triple handshake. During this process, the parties synchronize their sequence numbers. Changing the sequence number across the connection is necessary. The notion is that every time a new packet is sent, the sequence number must be recalculated. During a 'hijacking attack,' this makes it more difficult for attackers to forecast or estimate the sequence number. The problem with this strategy is that if an attacker gathers enough packets, he can break the algorithm and change the sequence.

Poonkuntran and Arun (2014) proposed employing honey pots as a way of detecting bogus access points. On wireless networks, the honey pots will be superfluous access points. The honey pot will be utilized to acquire information or evidence, as well as to identify attack patterns used by attackers. Honey pots will be used to lure in intruders and direct them to the network trap system. The honey pots will only detect the false access points, leaving the identification of the legitimate stations undiscovered.

Singh et al. (2012) proposed a wireless network (cross-layer IDS) intrusion detection system that combines the weight value of the received signal strength (RSS) and the time it takes to request and transmit transmission packets. The TT and RSS values are captured and monitored on the server in technology. When the aggregate weight of the station exceeds the specified limit during the detecting procedure, the IDS will sound an alarm. To get an accurate result, the technology relies on the threshold value. When the leg is legitimate silent, this strategy gives false negatives.

Atlas (Qazi et al., 2013) is a method for discovering granular applications from mobile agents through collective outsourcing. The trainer is then forwarded to a control aircraft's automatic learning plane for classification. Mekky et al. (2014) developed an extended application architecture for SDN systems that are aware of the generalization of redirection abstractions, which includes data from 4 to 7. Their implementation adds application logic in converters to boost efficiency. FlowQoS (Seddiki, 2014) is also a reference design for implementing application-based service quality by delegating application identification and QoS tuning to the SDN controller (Tsai et al., 2018).

CHAPTER – 3

Methodology

In TCP/IP networks, a packet is an extremely large data carrier. Active information is broken down and encoded into packets in the packet exchange network. The source nodes deliver packets to the access point with destination and source addresses. After obtaining the packet destination, decryption and aggregation are conducted to extract the required data. We work on a suggestion tool for sniffing and network packet recording in this study. The program collects the packet and analyses its contents in order to identify network obstructions that cause transmission, reception, or work delays in general. The delay in network operation could be caused by packet congestion on a certain port**.** It will be simple for network administrators to propose different approaches and techniques for solving the existing problem following the study and comprehensive analysis of the data. Even if the network appears to be free of difficulties, as the network grows in size and future work progresses, new problems will emerge, hence the major goal of the suggested tool in this project is to analyze network behavior in order to improve performance or solve problems. Based on past research, we believe that developing a tool that works with both IPv4 and IPv6 will be a significant outcome that will immediately aid network management and improvement both now and in the near future, given the introduction of IPv6. The implementation of the sniffing tool is the most important stage of this project. After some coding in the C# computer language, this step will be proposed. To begin, the application displays all of the network's connected devices' IP addresses. The chosen IP address will be used as an instruction to capture the packet and offer some information about it by determining if the packet is being transmitted or received. In addition, the protocol used for this packet will be recognized. In addition to the protocol, fundamental packet information such as checksum, time to leave TTL, data length, source address, a destination address, and port will be investigated.

Phase 1: Data collection and simulations

One of the most significant phases is simulation and data collecting, in which the real situation is recreated and traffic data is gathered for the project needs of both IP versions 4 and 6. Instead of Wireshark, the simulation will use a packet generator to capture the optimal traffic for pattern analysis. The pattern analysis results will be incorporated into a new sniffing tool in a novel method, allowing the new sniffing tool to monitor and record such traffics.

Phase 2: Design and Implementation

This phase focuses on the design of the new Sniffing tool, which is a continuation of the previous phase. Starting with the specification of system requirements that meet the demands of the users and progressing via software engineering diagrams, implementation of provided requirements into source code, and software compilation. Finally, network simulation will be used to test the new sniffing tool.

This phase focuses on the design of the new Sniffing tool, which is a continuation of the previous phase. Starting with the specification of system requirements that meet the demands of the users and progressing via software engineering diagrams, implementation of provided requirements into source code, and

software compilation. Finally, network simulation will be used to test the new sniffing tool.

The network's weakness may not be fully interrupted, but it's worth noting that there's a lot of pressure on one of the network's ports, which causes congestion, which causes the network to weaken or slow down, resulting in poor performance and inefficiency.

Phase 3: Tool Evaluation

The study of traffic monitoring and capture from prior simulations will be explained in the project analysis. The new sniffer program, like the other two, will save whatever it collects in a log file. Depending on the settings, the log file is located in the application folder or the home folder. This file contains information about the type of traffic/packets that pass through an IPv4 or IPv6 network, as well as the source and destination of the traffic. The log file's output can determine whether the new sniffing instruments can monitor or collect the traffic as proposed. If not, this utility needs to be updated.

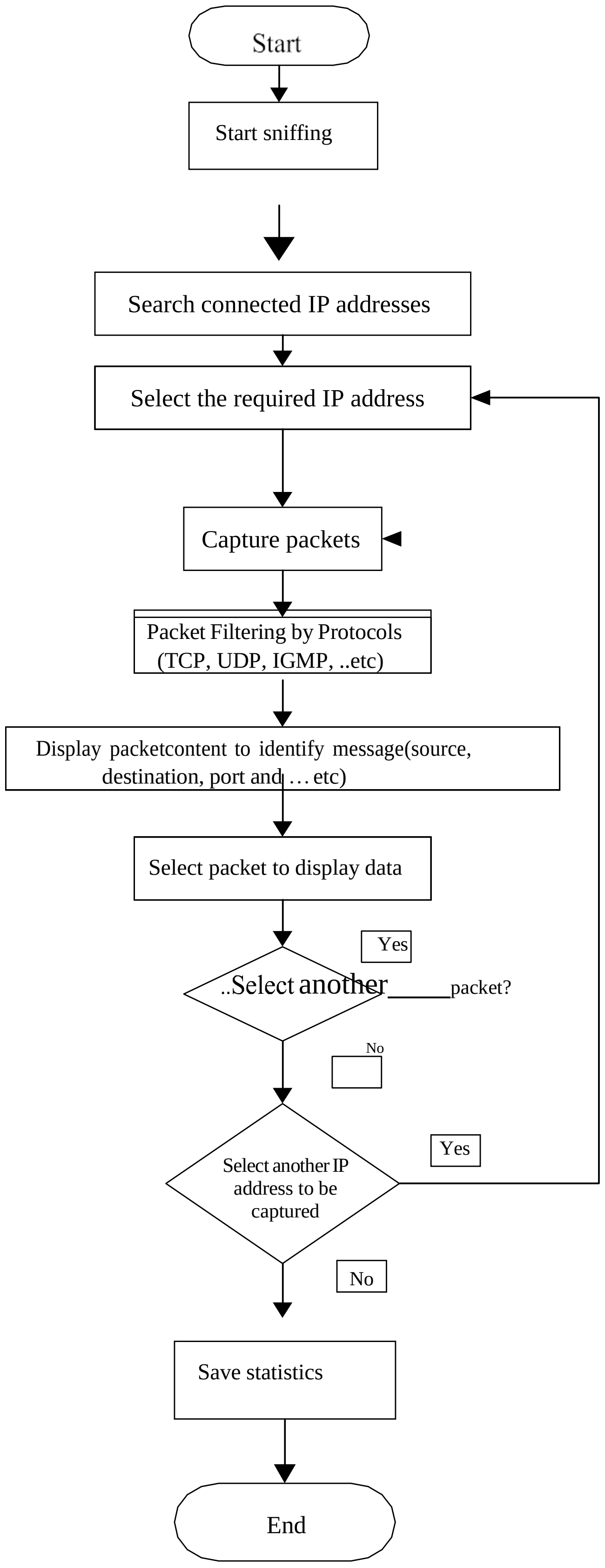
CHAPTER – 3

Proposed system

The new sniffer tool must be able to de-capsulate both IPv4 and IPv6 data on a network. The procedure of smelling steps is depicted in Figure 1. The process begins by conducting a complete search of all devices connected to the network and putting them in a list for the user to choose from, as illustrated in Figure 1. Following the retrieval of IP addresses for all devices, whether version 4 or 6, the required IP address is chosen, followed by the buffer size, which is the number of packets collected. The data is subsequently analyzed in order to present information about each packet. Source IP address, source port, destination IP address, destination port, and protocol will all be required and very crucial in network analysis. This information will allow network analysts to gain a better understanding of the network. The proposed sniffing tool then provides various statistics for displaying information in user-friendly interfaces.

Our packet sniffer copies packets from the kernel buffer into a buffer formed when a live capture session is started at the user level. The buffer only handles one packet at a time for application processing before copying the next packet into it. The novel method used in the creation of our packet sniffer is to boost performance by employing our proposed tool to share buffer space between kernel and application space. As illustrated in Algorithm 1, the sniffing operation is carried out in real time, providing real-time statistics on the active ports and protocols. Because it saves storage and time, live mode sniffing is more efficient than traditional technique

Figure 1: Flowchart


Start
Start sniffing
Search connected IP addresses
Select the required IP address
Capture packets
Packet Filtering by Protocols (TCP, UDP, IGMP, ..etc)
Display packetcontent to identify message(source, destination, port and … etc)
Select packet to display data
Yes
..Select another packet?
No
Yes
Select another IP address to be captured
No
Save statistics
End

```
ALGORITHM 1: Capture an IP address
To begin capturing, enter the desired IP address and buffer size.
Captured packets as output.
Start
Step 1: Capture all packets connected to the selected IP address (both received and delivered).
Step 2: In the list view, list all of the collected packets.
Step 3: Use the sniffer tool's list view to determine the key contents of the captured packet (such as the source IP address, destination IP address, port, and packet size) by comparing the captured packet with the header of the packet.
Step 4: Breaks if the amount of captured packets equals the buffer size; else, proceed to step 3.
Step 5: Temporarily store all of the recorded packets in the buffer and continue live sniffing.
End
```

CHAPTER – 4

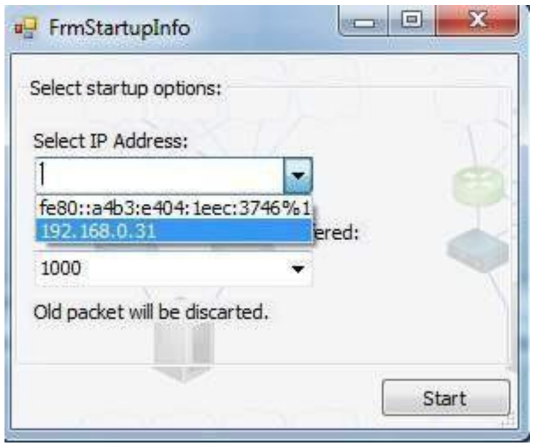
Results and Discussion

Two scenarios will be tested in this section to discuss the results received while using the proposed tool.

Scenario A (IPV4)

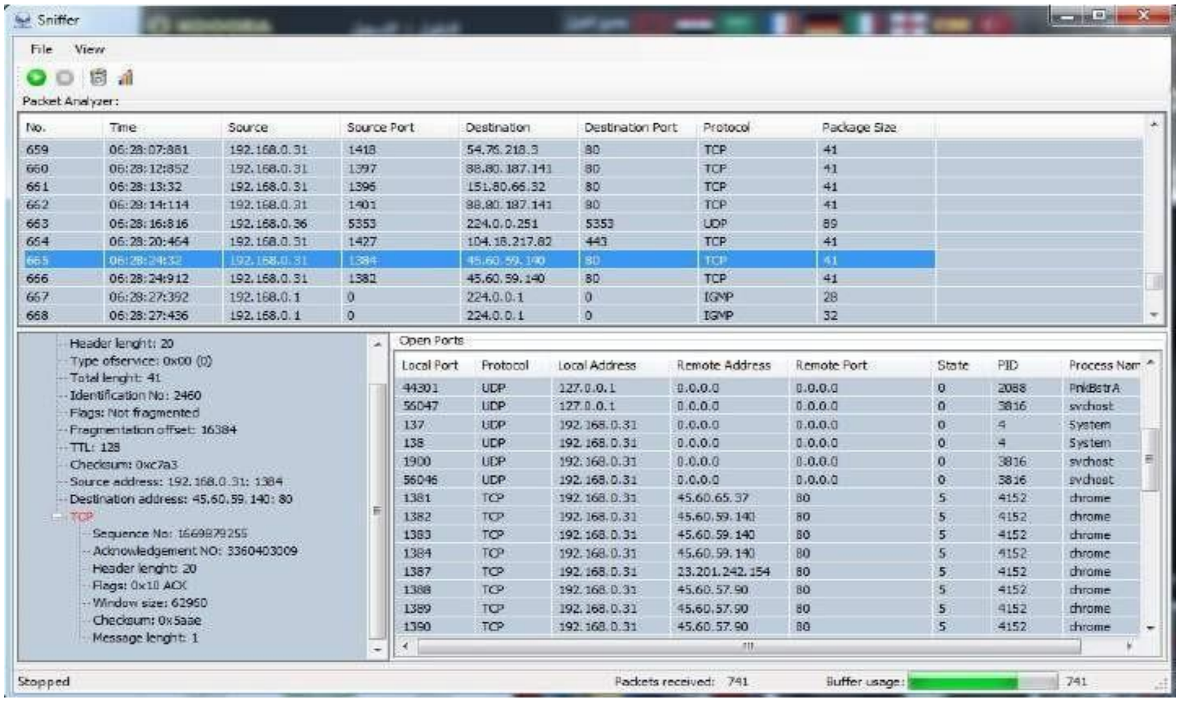
IP address version 4 is tested in this scenario by scanning all IP addresses linked to the network and then specifying an address to begin the capturing and sniffing operation. All network-related addresses are scanned and listed in a drop-down list for selection, as illustrated in Figure 2.

Figure 2: Selection of desired IP address



The number of IP addresses on the network, as indicated in Figure 2. The size of the buffer, which saves captured packets, should be specified after selecting the required IP address. In other words, the buffer size is equal to the number of packets collected. Pressing the start button in the capturing process after selecting the desired basic information such as IP address and buffer size will display the system's main screen with the beginning of the data for this IP address gradually displayed on the screen as new packets are captured automatically, as shown in Figure 3.

Figure 3: Sniffer's main interface with packet capture results



No.	Time	Source	Source Port	Destination	Destination Port	Protocol	Package Size
659	06:28:07:881	192.168.0.31	1418	54.76.218.3	80	TCP	41
660	06:28:12:852	192.168.0.31	1397	88.80.187.141	80	TCP	41
661	06:28:13:32	192.168.0.31	1396	151.80.66.32	80	TCP	41
662	06:28:14:114	192.168.0.31	1401	88.80.187.141	80	TCP	41
663	06:28:16:816	192.168.0.36	5353	224.0.0.251	5353	UDP	89
664	06:28:20:464	192.168.0.31	1427	104.18.217.82	443	TCP	41
665	06:28:24:32	192.168.0.31	1384	45.60.59.140	80	TCP	41
666	06:28:24:912	192.168.0.31	1382	45.60.59.140	80	TCP	41
667	06:28:27:392	192.168.0.1	0	224.0.0.1	0	IGMP	28
668	06:28:27:436	192.168.0.1	0	224.0.0.1	0	IGMP	32

```
Header lenght: 20
Type ofservice: 0x00 (0)
Total lenght: 41
Identification No: 2460
Flags: Not fragmented
Fragmentation offset: 16384
TTL: 128
Checksum: 0xc7a3
Source address: 192.168.0.31: 1384
Destination address: 45.60.59.140: 80
TCP
    Sequence No: 1669879255
    Acknowledgement NO: 3360403009
    Header lenght: 20
    Flags: 0x10 ACK
    Window size: 62960
    Checksum: 0x5aae
    Message lenght: 1
```

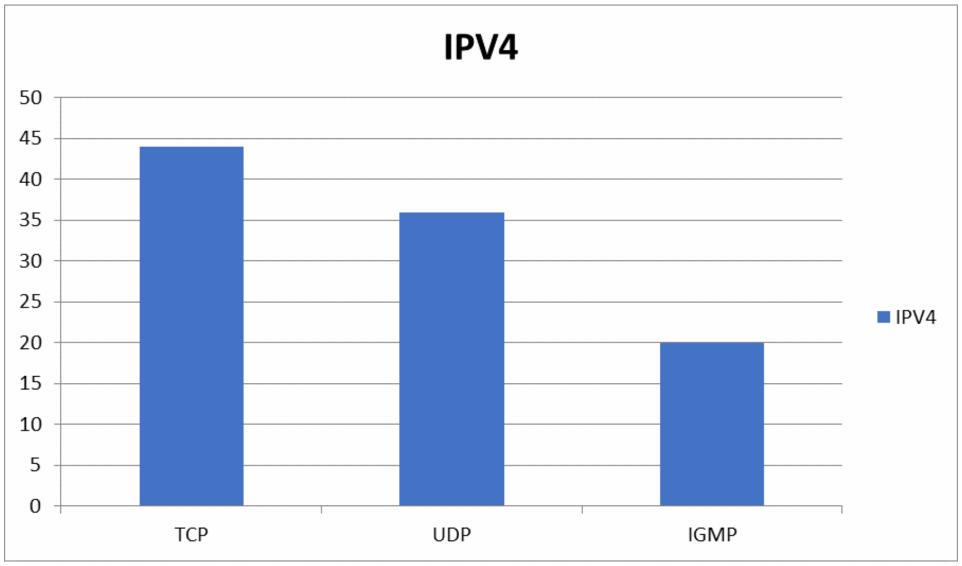
Local Port	Protocol	Local Address	Remote Address	Remote Port	State	PID	Process Nam
44301	UDP	127.0.0.1	0.0.0.0	0.0.0.0	0	2088	PnkBstrA
56047	UDP	127.0.0.1	0.0.0.0	0.0.0.0	0	3816	svchost
137	UDP	192.168.0.31	0.0.0.0	0.0.0.0	0	4	System
138	UDP	192.168.0.31	0.0.0.0	0.0.0.0	0	4	System
1900	UDP	192.168.0.31	0.0.0.0	0.0.0.0	0	3816	svchost
56046	UDP	192.168.0.31	0.0.0.0	0.0.0.0	0	3816	svchost
1381	TCP	192.168.0.31	45.60.65.37	80	5	4152	chrome
1382	TCP	192.168.0.31	45.60.59.140	80	5	4152	chrome
1383	TCP	192.168.0.31	45.60.59.140	80	5	4152	chrome
1384	TCP	192.168.0.31	45.60.59.140	80	5	4152	chrome
1387	TCP	192.168.0.31	23.201.242.154	80	5	4152	chrome
1388	TCP	192.168.0.31	45.60.57.90	80	5	4152	chrome
1389	TCP	192.168.0.31	45.60.57.90	80	5	4152	chrome
1390	TCP	192.168.0.31	45.60.57.90	80	5	4152	chrome

Stopped   Packets received: 741   Buffer usage: 741

As illustrated in Figure 3, the primary interface will be separated into three main areas, each of which displays distinct results in addition to the existence of auxiliary conditions such as a status bar, which contains a progress bar that represents the number of packets in the buffer. The upper portion, dubbed the 'packet

analyzer,' is the most important of these three sections. Each of the eight columns in the packet analyzer list view has a different function: (No, Time, Source IP address, Source port, Destination IP address, Destination port, Protocol, Package size). The information is organized into rows. Each row represents a collected packet, with simple and basic information displayed as previously mentioned. The number of packets captured is flexible, but it is fully determined by the buffer value given in the first interface before you begin capturing. When you press once on any of the collected packets, the program's second sub-interface is activated. This interface contains more detailed information on the selected packet, which is critical for system engineers who are analyzing network data. For example, if the IP address version is 4 or 6, this information is useful.
There is also information on the protocol utilized by this packet in either transmitting or receiving it. Because this packet uses the TCP protocol, there will be more information about it, such as sequence number, acknowledgment number, header length, flags, checksum, and message length. Changing the kind of protocol used by this packet changes this information. TCP, UDP, ICMP, and IGMP are the four types of protocols that our suggested system recognizes and analyses. Because the protocol utilized is TCP, which is a trustworthy protocol, there is acknowledgment in this scenario. When you click on a different packet, the data in this sub-interface of the comprehensive information about the captured packet changes to the information of the newly selected packet.

Figure 4: Statistics analyzer for IPV4



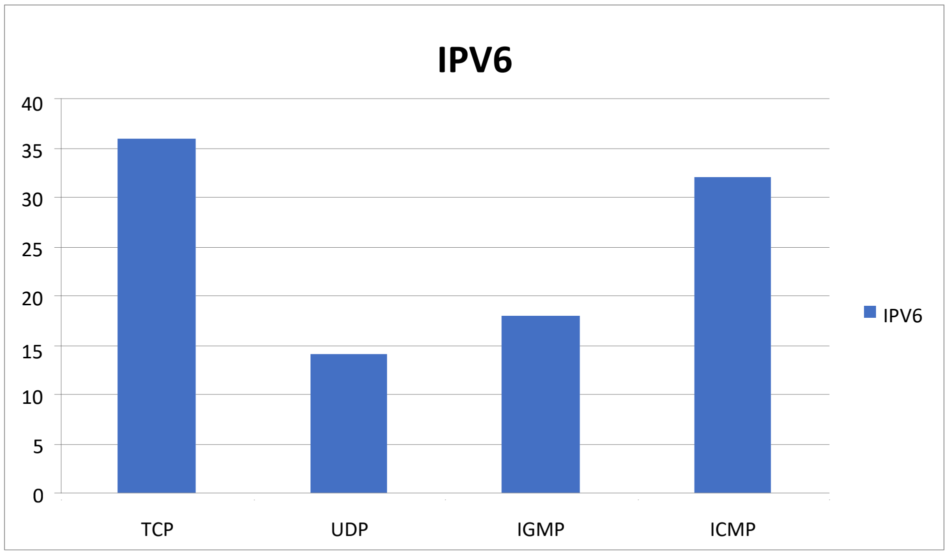
The system also shows the most frequently used ports in the network by providing the most often used source port. This is only for the most often used source ports and the most frequently used destination ports from the sample used to open the statistics pane. This only applies to destination ports within the network packet sample until the statistics window is opened. This data is critical for analysing the network and determining the pressure areas on each transmission or receiving port that could cause network congestion. Figure 4 shows three intercepted protocols that were found within the captured sample and estimated to be 1,000 packets each. TCP, UDP, and IGMP are the protocols in question, with TCP having the highest percentage and being depicted in blue, followed by the UDP protocol. The IGMP protocol was the least commonly used in this sample of network packets.

Furthermore, port 1901 is the most often used source port. However, in the network packets recorded, port 443 was the most often used destination port. These statistics' percentages and clear numbers are all distinct and conflicting. They differ from one network to the next and from time to time, depending on how the network is being used at the time. As a result, all protocols may appear, or some may be lost as a result of usage. The proportions can change dramatically. Also, depending on network usage and sample size collected from network packets, the most common source port and most frequent destination port will almost likely differ.

Scenario B (IP v6)

Table 1 provides a sample of statistics from IPv6 network packets. These statistics' percentages and clear numbers are all distinct and conflicting. Where they differ from network to network and from time to time depending on how the network is being used at the time, so that all protocols may appear or be absent for some users, and ratios may be significantly different. Also, depending on network usage and sample size collected from network packets, the most common source port and most frequent destination port will almost likely differ.

Figure 5: Statistics analyzer for IPV6



The buffer size chosen for this situation was 2,000 collected packages in the sample. Within this sample, we notice the emergence of four protocols: TCP, UDP, ICMP, and IGMP, which have been emphasised by the system and employed in both transmission and receiving network operations. Figure 5 depicts the use of the TCP and ICMP protocols, with the ICMP being represented by red and the TCP being represented by light blue. While the use of the IGMP protocol, which is shown in dark blue, was slightly higher than the use of the UDP protocol, which is shown in yellow.

For the selected sample of IPv6 network packets recorded, the most common source port was 1742, and the most common destination port was 80.

TABLE 1: PACKET ANALYZER FOR IPV6

No.	*Time*	*Source*	*Source port*	*Destination*	*Destination port*	*Protocol*	*Package size*
62	06:56:38:772	fe80::a4b3:e404:1eec:3746	1900	2001:e68:5414::0001:22cf	1900	IGMP	74
63	06:56:38:793	fe80::a4b3:e404:1eec:3746	1900	2001:a18:25:11::40	1900	UDP	74
64	06:56:38:854	2001:4860:4860::8888	1901	2001:4860:40::8114	1901	ICMP	90
65	06:56:38:915	fe80::a4b3:e404:1eec:3746	1901	2001:4860:40::8114	1901	IGMP	90
66	06:56:38:974	fe80::a4b3:e404:1eec:3746	1901	2001:4860:40::8114	1901	TCP	90
67	06:56:39:35	2001:4860:4860::8888	1740	2001:4860::25cc:877	80	TCP	133
68	06:56:39:93	fe80::a4b3:e404:1eec:3746	1740	2001:a18:25:11::40	80	ICMP	133
69	06:56:39:153	fe80::a4b3:e404:1eec:3746	1740	2001:4860::25cc:877	80	TCP	133
70	06:56:39:214	2001:4860:4860::8888	1741	2001:a18:25:11::40	80	ICMP	255
71	06:56:39:275	fe80::a4b3:e404:1eec:3746	1741	2001:e68:5414::0001:22cf	80	IGMP	255
72	06:56:39:335	fe80::a4b3:e404:1eec:3746	1741	2001:e68:5414::0001:22cf	80	UDP	255
73	06:56:39:395	2001:4860:4860::8888	1742	2001:a18:25:11::40	80	TCP	180
74	06:56:39:455	fe80::a4b3:e404:1eec:3746	1742	2001:e68:5414::0001:22cf	80	UDP	180

CHAPTER – 5

Conclusion

Finally, we may see a significant variation in the apparent results of IPv4 and IPv6 network packets. The outcomes of each scenario were examined and analyzed independently in order to determine the lessons learned from each scenario, and then a conversation about the scenarios' outcomes was held with some. There was a difference in the outcomes between the two scenarios. The variation in values and protocols employed accounts for this disparity. Because the ICMP protocol was not caught in Scenario A, only three protocols were captured: TCP, UDP, and IGMP. In Scenario B, however, four protocols were utilized and recorded: TCP, UDP, ICMP, and IGMP. It was discovered that the protocols utilized in Scenario B for IP version 6 were not visible and did not exist in IP version 4. The ICMP protocol, for example, appeared to be a distinction between the two cases. Aside from the protocols employed, the ports in the two cases differed, or more accurately, the most commonly used ports in the network. This discrepancy is attributable to a number of factors that have a direct impact on the results, including the collected IP packet (version 4 or 6) and the sample size. Furthermore, the network used at the time of system development has a significant impact on the results, both in terms of protocols collected and through the most commonly used network ports.

CHAPTER – 6

REFERENCE

Anu, P. and Vimala, S. (2017) ‘A survey on sniffing attacks on computer networks’, 2017 International Conference on Intelligent Computing and Control (I2C2), pp.1–5 [online] http://ieeexplore.ieee.org/document/8321914/ (accessed 21 May 2019).

Chauhan, D. and Sharma, S. (2014) ‘A survey on next generation internet protocol IPV6’, International Journal of Electronics and Electrical Engineering, Vol. 2, No. 2, pp.143–146.

Davis, J.J. and Clark, A.J. (2011) ‘Data pre-processing for anomaly based network intrusion detection: a review’, Comput. Secur., Vol. 30, Nos. 6–7, pp.353–375.

Elsen, L. et al. (2015) ‘goProbe: a scalable distributed network monitoring solution’, 2015 IEEE International Conference on Peer-to-peer Computing (P2P), pp.1–10, Boston, MA, USA.

Gandhi, C., Suri, G., Golyan, R., Saxena, P. and Saxena, B. (2014) ‘Packet sniffer – a comparative study’, International Journal of Computer Networks and Communications Security, Vol. 2, No. 5 pp.179–187.

Jaisinghani, D., Naik, V., Kaul, S.K. and Roy, S. (2017) ‘Sniffer-based inference of the causes of active scanning in WiFi networks’, 2017 Twenty-third National Conference on Communications (NCC), Chennai, pp.1–6.

Mekky, H. et al. (2014) ‘Application-aware data plane processing in SDN’, in ACM Workshop Hot Topics Network, pp.13–18, Chicago, IL, USA.

Nishanth, N. and Babu, S.S. (2014) ‘Sequence number alteration by logical transformation (SALT): a novel method for defending session hijacking attack in mobile ad hoc network’, International Journal of Computer and Communication Engineering, Vol. 3, No. 5, pp.338–342. Oluwabukola, O., Awodele, O., Ogbonna, C., Chigozirim, A. and Anyaehie, A. (2013) ‘A packet sniffer (PSniffer) application for network security in Java’, Proceedings of the Informing Science and Information Technology Education Conference, Informing Science Institute, pp.389–400. Poonkuntran, S. and Arun, A.M. (2014) ‘Study of Honeypots: analysis of WiFi Honeypots and Honeypots tools’, AENSI Journals on Advances in Natural and Applied Sciences, Special Edition, Vol. 8, No. 17, pp.48–59. 244 R.F. Albadri

Qadeer, M.A., Iqbal, A., Zahid, M. and Siddiqui, M.R. (2010) ‘Network traffic analysis and intrusion detection using packet sniffer’, 2010 Second International Conference on Communication Software and Networks, Singapore, Singapore.

Qazi, Z.A. et al. (2013) ‘Application-awareness in SDN’, ACM SIGCOMM Comput. Commun. Rev., Vol. 43, No. 4, pp.487–488.

Seddiki, M.S. (2014) ‘FlowQoS: QoS for the rest of us’, in Workshop Hot Topics Softw. Defined Network. pp.207–208, Chicago, IL, USA.

Singh, J., Gupta, S. and Kaur, L. (2012) ‘A cross-layer based intrusion detection technique for wireless networks’, International Arab Journal of Information Technology, Vol. 9, No. 3, pp.201–207.

Singh, R. and Kumar, S. (2018) ‘A comparative study of various wireless network monitoring tools’, 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar, India, pp.379–384.

Tsai, P-W. et al. (2018) 'Network monitoring in software-defined networking: a review', IEEE Systems Journal, Vol. 12, No. 4, pp.3958–3969.

Xu, J., Liu, W. and Zeng, K. (2016) 'Monitoring multi-hop multi-channel wireless networks: online sniffer channel assignment', 2016 IEEE 41st Conference on Local Computer Networks (LCN), Dubai, pp.579–582.

Yang, J., Zhang, Y., King, R. and Tolbert, T. (2018) 'Sniffing and chaffing network traffic in stepping-stone intrusion detection', 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), Krakow, Poland, pp.515–520.

Zhao, Z., Huangfu, W. and Sun, L. (2012) 'NSSN: a network monitoring and packet sniffing tool for wireless sensor networks', 2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC), Limassol, Cyprus, pp.537–542.