RESEARCH ARTICLE OPEN ACCESS

NETWORK PACKET SNIFFER

Dr. G. Kalpana[1], Sreemya Gavini[2]
1(Malla Reddy Engineering College for Women (Autonomous), Hyderabad, India
Email: dr.gkalpanamrecw@gmail.com)
2 (Malla Reddy Engineering College for Women (Autonomous), Hyderabad, India
Email : sreemgav.52@gmail.com)

Abstract:

A computer network consists of a group of computers that are linked with one another which share the resources. Data packets are generally used to transfer the data among these nodes. When a large volume of data in a network is divided into smaller sub portions, they are called as data packets. It is an important task to analyse different packets of data when they are shared across a network without any supervision. This is a situation where packet sniffer is used. Packet sniffer is a tool that is usually used in network monitoring to analyse all the incoming and outgoing data packets. The aim of this paper is to develop a packet sniffing tool from scratch using python, networking and cloud knowledge. It also focuses on developing a standalone tool that consumes little memory on the hard disk.

Keywords *– Computer network, Packet sniffer, Network monitoring, Apache Kafka, Amazon MSK, Virtual Private Cloud, Mirrormaker tool*

I. INTRODUCTION

As online services are ruling the current era, security experts are continuously working to find new possibilities to investigate the growing cybercrimes. Transfer via data packets is the most common way in which the large volumes of information can be transferred by the online services over the computer networks. These are groups of bits that travel along a network path usually given by layer 3 (network layer) of the OSI model [1]. These packets are the similar fragments of data at an instant of time and are then reassembled to the original data when all the packets reach the destination.

A data packet usually contains a header and a payload. The header consists of the information related to the packet like source IP address, destination IP address, hop limit, etc. The payload contains a transmitted message (or data) which then combines with the header to form a data packet. A data unit in data link layer of the OSI model is known as a frame, that is a group of data bits from network layer and are encapsulated by the data link layer. In transport layer, segment (or datagram) is the equivalent of a data packet.

Packet sniffing refers to examining each packet as it crosses a network interface. In this process, the incoming data packets are tapped and hence moved to the other network. Here, the data that is sniffed belongs to the other users of a different network. Packet sniffers work with same efficiency in case of both switched and non-switched networks [2]. These sniffers prove to be useful for the network administrators to analyse the various activities like LAN behaviour, spot any troubles, trouble shooting, threat detection, bandwidth management, etc. Transmission control protocol/Internet protocol (TCP/IP) layer tools called packet sniffers are used to inspect data packets moving over networks [3]. In simpler terms, packet sniffer on computer networks is like wiretapping on telephone networks. That means, a packet sniffer acts as a tool that eavesdrops the packets of one network and send them to another network when they are found to be legitimate.

When network packets are efficiently captured and analysed, they can be used for network monitoring, which may even provide convincing evidence to support the security experts' future inquiry [4].

II. LITERATURE SURVEY

An enterprise decided to migrate their self-managed Apache Kafka cluster to Amazon MSK to reduce infrastructure management and focus more on growing business. An asynchronous service to service communication system, or distributed publish-subscribe messaging system, called Apache Kafka allows you to transfer messages from one endpoint to another while handling large amounts of data. Several approaches are used to migrate the data from Apache Kafka to Amazon MSK. Among which creating a private link between the on-premises servers is most popular. A private link creates connectivity among the VPCs to transfer the data securely without exposing the user data to the public internet [5]. Although with the private link, the data will securely be transferred, this attempt to share the data could not transfer all the data packets implying that the created link failed to meet certain requirements. There are a few underlying issues that are responsible for this situation and hence, a new approach is preferred to solve the issue. To come up with the solution for the above situations packet sniffers are developed. The developed tool, as described in my paper, is cost effective and can be modified as per the user environment. This tool provides excellent integration with the front end and makes it convenient for the users to work with.

III. METHODOLOGY

To solve the above problem and to overcome the problem of other sniffer tools in migrating the data from one Virtual Private Cloud (VPC1) to a different Virtual Private Cloud (VPC2), there comes the possibility to develop our own sniffer tool that monitors the incoming and outgoing data packets, over a network interface, which ensures data security [6].

In this paper, we discuss how an own sniffer tool is can be developed using concepts of Packet Tracer, Cloud knowledge and the layers of TCP/IP Protocol (namely, Application, Transport, Network, Link layers).

Tools and Modules

Python is used as key programming language to implement the entire tool. Visual Studio is used as the preferrable IDE. As a part of implementation,

- PacketSniffer is imported from the core module
- OutputToScreen is imported from the output module
- OS module
- Packet module
- Time module
- other iteration tools are chiefly used.

IV. IMPLEMENTATION

After successful establishment of the connection, packet sniffer tool is placed between the servers which taps all the packets and then exports the legitimate packets to the other network interface.

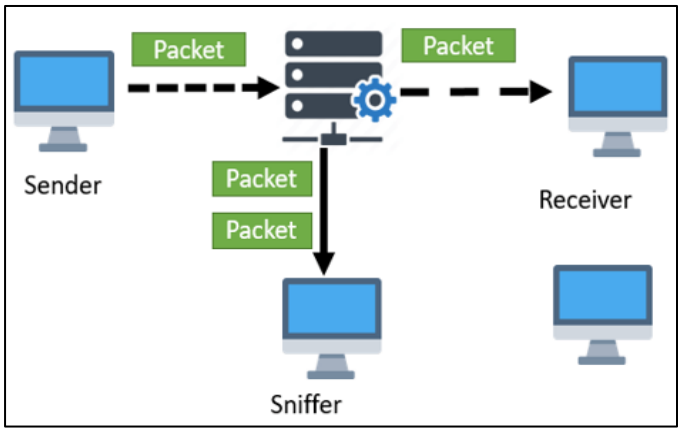

Fig 2. Function of Packet Sniffer

This tool is implemented on AWS. Amazon VPCs and Apache Kafka are connected to each other through a central hub and a transit gateway [7] is used to establish this connection. As per the client requirements, third party tools cannot be used to analyse the incoming and outgoing data packets. Hence, by using the interface of Visual Studio and programming language Python, a tool using Mirrormaker is developed to meet the requirements of the client and the provided environment. Mirrormaker tool initially sets up the connectivity between two different servers using the below configuration. Slowly all the data packets transfer from server to the other we simultaneously receive acknowledgment of the sent frame along with frame number.

```
{
"name": "server1",
"connector.class": "org.apache.kafka.connect.mirror.MirrorCheckpointConnector",
"clusters": "prim, sec",
"source.cluster.bootstrap.servers": "kafka.us-west-2.amazonaws.com:9092",
"source.cluster.security.protocol": "PLAINTEXT",
"source.cluster.sasl.mechanism": "PLAIN",
"target.cluster.alias": "sec",
"target.cluster.bootstrap.servers":
"target.cluster.security.protocol": "PLAINTEXT",
"target.cluster.sasl.mechanism": "PLAIN",
"tasks.max": "3",
"key.coverter": " org.apache.kaka.connect.converters.ByteArrayConverter",
"value.coverter": " org.apache.kaka.connect.converters.ByteArrayConverter",
"replication.factor": "3",
"checkpoints.topic.replication.factor": "3",
"emit.checkpoints.interval.seconds": "20",
"emit.checkpoints.enabled": "true",
"sync.group.offsets.enabled": "true",
"sync.group.offsets.interval.seconds": "10"
}
```


Fig 3. Established connection of Server 1


```
{
"name": "server2",
"connector.class": "org.apache.kafka.connect.mirror.MirrorHeartBeatConnector",
"clusters": "prim, sec",
"source.cluster.alias": "sec",
"source.cluster.bootstrap.servers":
"source.cluster.security.protocol": "PLAINTEXT",
"source.cluster.sasl.mechanism": "PLAIN",
"target.cluster.sasl.jaas.config": "org.apache.kafka.common.security.plain.Plainlogin",
"target.cluster.alias": "sec",
"target.cluster.security.protocol": "PLAINTEXT",
"target.cluster.sasl.mechanism": "PLAIN",
"target.cluster.sasl.jaas.config": "org.apache.kafka.common.security.plain.Plainlogin",
"tasks.max": "3",
"key.coverter": " org.apache.kaka.connect.converters.ByteArrayConverter",
"value.coverter": " org.apache.kaka.connect.converters.ByteArrayConverter",
"replication.factor": "3",
"heartbeats.topic.replication.factor": "3",
"emit.heartbeats.interval.seconds": "20",
"emit.heartbeats.enabled": "true",
"sync.group.offsets.enabled": "true",
"sync.group.offsets.interval.seconds": "10"
}
```


Fig 4. Established connection of Server 2

V. RESULTS

```
┌──(venv)─(kali㉿kali)-[~/Desktop/packet-sniffer]
└─$ sudo ./venv/bin/python3 packet_sniffer/sniffer.py

[>>>] Packet Sniffer initialized. Waiting for incoming data. Press Ctrl-C to abort...
```

Fig 5. Initialized packet sniffer

This is the first stage when the packet sniffer is initialized and in a waiting state to receive the incoming packets.

```
        Traffic Class: 0x000
    [+] Unknown Protocol
    [+] TCP .......................443 -> 39582
        Sequence Number: 4277651313
        ACK Number: 4093463841
        Flags: 0x018 > PSH ACK
        Window Size: 265
        Checksum: 0x7b33
        Urgent Pointer: 0
[>] Frame #8 at 13:55:14:
    [+] Ethernet ......08:00:27:be:20:60 -> dc:d9:ae:71:d1:a0
        Interface: all
        Frame Length: 86
        Epoch Time: 1657302914.8955085
    [+] IPv6 2804:d55:5e76:8e00:76e6:c308:b5d8:c47 -> 2800:3f0:4001:833::200a
        Traffic Class: 0x000
    [+] Unknown Protocol
    [+] TCP .....................39582 -> 443
        Sequence Number: 4093463841
        ACK Number: 4277651352
        Flags: 0x010 > ACK
        Window Size: 501
        Checksum: 0xb233
        Urgent Pointer: 0
```

Fig 6. Transfer of Frame 8

Here, the data is transferred in the form of frames by listing all the necessary details of the transmitted frame like sequence number, acknowledgement number, flags used, check sum, frame length, etc. Also, the address of IPv6 is also mentioned in the output window. Currently frame no. 8 is being transferred.

```
        Traffic Class: 0x000
    [+] Unknown Protocol
    [+] TCP .......................80 -> 49640
        Sequence Number: 674558763
        ACK Number: 2541341194
        Flags: 0x010 > ACK
        Window Size: 261
        Checksum: 0xfed7
        Urgent Pointer: 0
[>] Frame #27 at 13:55:17:
    [+] Ethernet ......dc:d9:ae:71:d1:a0 -> 08:00:27:be:20:60
        Interface: all
        Frame Length: 86
        Epoch Time: 1657302917.7270625
    [+] IPv6 ....2800:3f0:4001:824::2003 -> 2804:d55:5e76:8e00:76e6:c308:b5d8:c47
        Traffic Class: 0x000
    [+] Unknown Protocol
    [+] TCP .......................80 -> 49638
        Sequence Number: 2369001901
        ACK Number: 1778493724
        Flags: 0x010 > ACK
        Window Size: 261
        Checksum: 0xdef
        Urgent Pointer: 0
```

Fig 7. Transfer of Frame no. 27

At this point, frame no. 27 is being transferred with simultaneously receiving sequence number and the respective acknowledgement number of the frame.

```
        Frame Length: 86
        Epoch Time: 1657302925.8406196
    [+] IPv6 ......2804:d55:5e76:8e00::1 -> 2804:d55:5e76:8e00:a00:27ff:febe:2060
        Traffic Class: 0x000
    [+] Unknown Protocol
    [+] ICMPv6 ......2804:d55:5e76:8e00::1 -> 2804:d55:5e76:8e00:a00:27ff:febe:2060
        Control Message Type: 135 (Neighbor Solicitation)
        Control Message Subtype: 0
        Header Checksum: 0x130a
[>] Frame #53 at 13:55:25:
    [+] Ethernet ......08:00:27:be:20:60 -> dc:d9:ae:71:d1:a0
        Interface: all
        Frame Length: 78
        Epoch Time: 1657302925.840882
    [+] IPv6 2804:d55:5e76:8e00:a00:27ff:febe:2060 -> 2804:d55:5e76:8e00::1
        Traffic Class: 0x000
    [+] Unknown Protocol
    [+] ICMPv6 2804:d55:5e76:8e00:a00:27ff:febe:2060 -> 2804:d55:5e76:8e00::1
        Control Message Type: 136 (Neighbor Advertisement)
        Control Message Subtype: 0
        Header Checksum: 0x2fff
^C[!] Aborting packet capture...

┌──(venv)─(kali㉿kali)-[~/Desktop/packet-sniffer]
└─$
```

Fig 8. Aborting packet sniffer

This is the final stage where frame no. 53 is successfully transferred and hence all the data is transferred. At this point, the packet sniffer is ready to terminate by aborting the further packet capture.

VI. CONCLUSION

A crucial part of the layered networking concept is played by network packet sniffers. Network sniffers examine and analyse streams of data packets moving between network nodes as well as between networked computers and the Internet to perform

their function. Packet sniffer gives us the advantages of doing traffic analysis, troubleshooting and network traffic monitoring. It is user-friendly and can easily adopt the new changes. It takes less memory storage because we can easily export the captured data to a database. Sniffing is possible on switched and non-switched networks. By this research we can conclude that the packet sniffer is very good to be used in intrusion detection [8]. Network packets are often the only source of information regarding what happened during a web behavior. The potential of packets in delivering forensic evidence has been described, and the limitations have been underlined, because using packet assessment in network forensics [9] differs from using it in other application areas. The comprehensive review provided in this paper aids the reader in comprehending the key steps in packet analysis and also the particular needs of network monitoring. This can be employed while creating tools and algorithms for packet analysis.

VII. FUTURE SCOPE

As the system is used, the user needs will inevitably vary because they are not constant. We are unable to create a product (or tool) that can adequately meet all of the user's intended needs. Some of the future enhancements that can be made to this proposed system are:

- It is possible to update the proposed system that can be adaptable to the desired environment in case the new technology advances.
- In data security point of view, emerging technologies can be used to improve data security and the tool can be modified as per the requirement.
- There is a chance to extend this tool across the internet as it currently works as a standalone tool.

ACKNOWLEDGMENT

The paperwork was done at Department of Computer Science and Engineering with specialisation in Artificial Intelligence & Machine Learning, at the Malla Reddy Engineering College for Women, (Autonomous), Hyderabad, India.

REFERENCES

[1]. Leslie F. Sikos, Packet analysis for network forensics: A comprehensive survey, Forensic Science International: Digital Investigation, Volume 32, 2020, 200892, ISSN 2666-2817.

[2]. Patel, Nimisha & Patel, Rajan & Patel, Dhiren. (2009). Packet Sniffing: Network Wiretapping. IEEE International Advance Computing Conference (IACC 2009) Patiala, India, 6–7 March 2009. 2691-2696.

[3]. Wang, Shaoqiang & Xu, DongSheng & Yan, Shiliang. (2010). Analysis and application of Wireshark in TCP/IP protocol teaching. 10.1109/EDT.2010.5496372.

[4]. **S.** Ansari, S. G. Rajeev and H. S. Chandrashekar, "Packet sniffing: a brief introduction," in IEEE Potentials, vol. 21, no. 5, pp. 17-19, Dec. 2002-Jan. 2003, doi: 10.1109/MP.2002.1166620.

[5]. Aljumaily, Mustafa & Kodituwakku, Angel. (2018). Final Project Virtual Private Network (VPN) Lab. 10.13140/RG.2.2.17464.24323.

[6]. Buch, Rachana & Shah, Sachi. (2018). NETWORK SNIFFERS AND TOOLS IN CYBER SECURITY.

[7]. https://d1.awsstatic.com/analyst-reports/AWS_Transit_Gateway

[8]. J. Yang, Y. Zhang, R. King and T. Tolbert, "Sniffing and Chaffing Network Traffic in Stepping-Stone Intrusion Detection," 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2018, pp. 515-520, doi: 10.1109/WAINA.2018.00137.

[9]. Sikos, Leslie. (2020). Packet Analysis for Network Forensics: A Comprehensive Survey. Digital Investigation. 32C. 10.1016/j.fsidi.2019.200892.