CODE
ALPHA

Cyber Security Internship(15sep - 15oct)

DEPARTMENT: CS

NETWORK PACKET SNIFFER

Course Code: CS 4151

Course Name: Cyber Security

MINI PROJECT REPORT

20240918

VIKRAM JAT

CONTENTS

OBJECTIVE

Network packet sniffer or simply packet sniffer is a packet analyzer software that monitors all network traffic. The proposed project is implemented in Python programming language, and using this application admin of the system can capture network packet and analyze data received/sent from/to the network.

Developed as a desktop application, packet sniffer facilitates web-based monitoring of network packets which are traveling over the system network. The primary data captured by this software is the packets source and destination addresses.

In this article, the project has been briefly discussed explaining its scope, features, and system specifications.

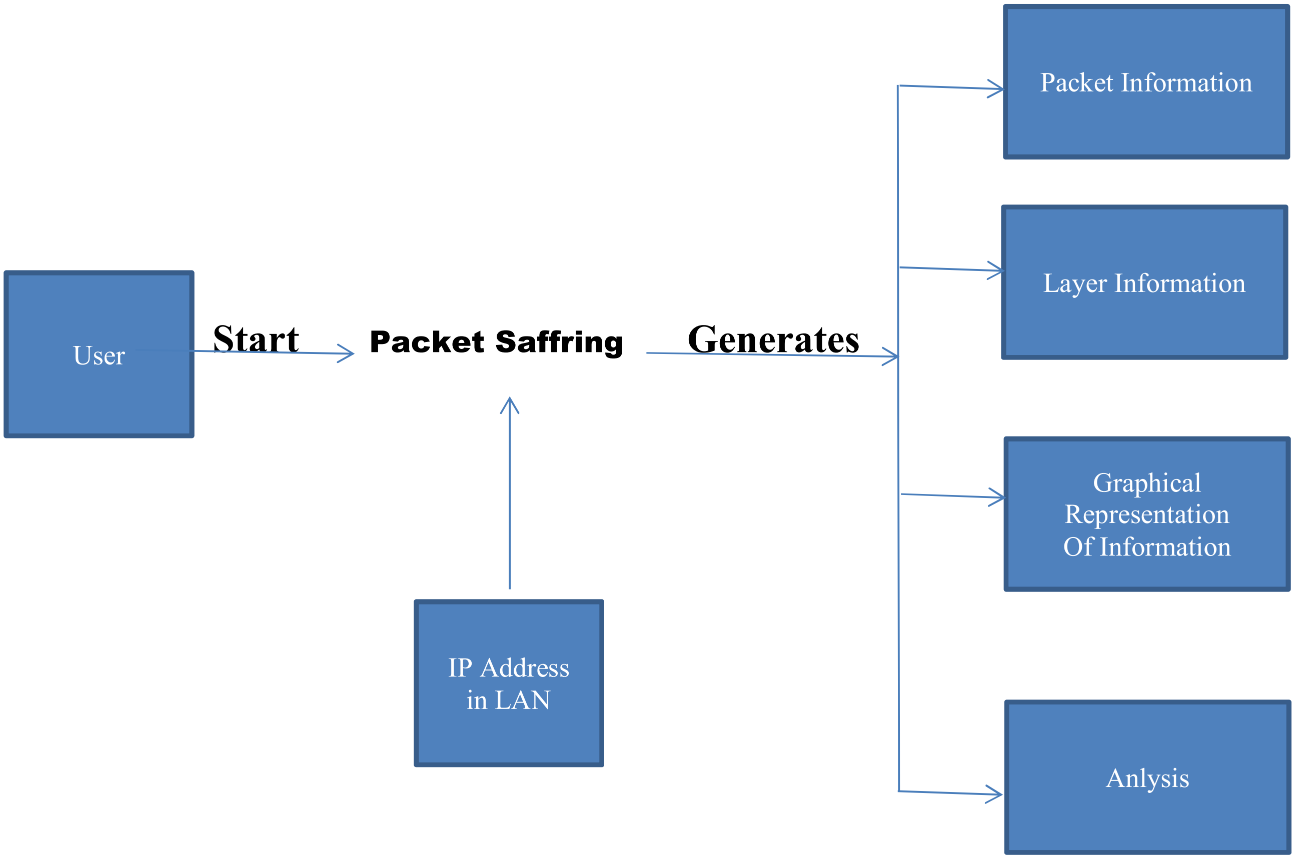
PROBLEM STATEMENT

Network packet sniffer is simply a web-based application that monitors all traffic over a network. Unlike other standard network hosts that only track traffic sent particularly to them, this software captures each packet, eventually decoding and analyzing its data as the data streams flow across the system network.

This project, developed in PYTHON, shows mainly two things:

1. how real-time network connection behavior can be modeled as chromosomes
2. how the parameters in genetic algorithm can be defined in this respect.

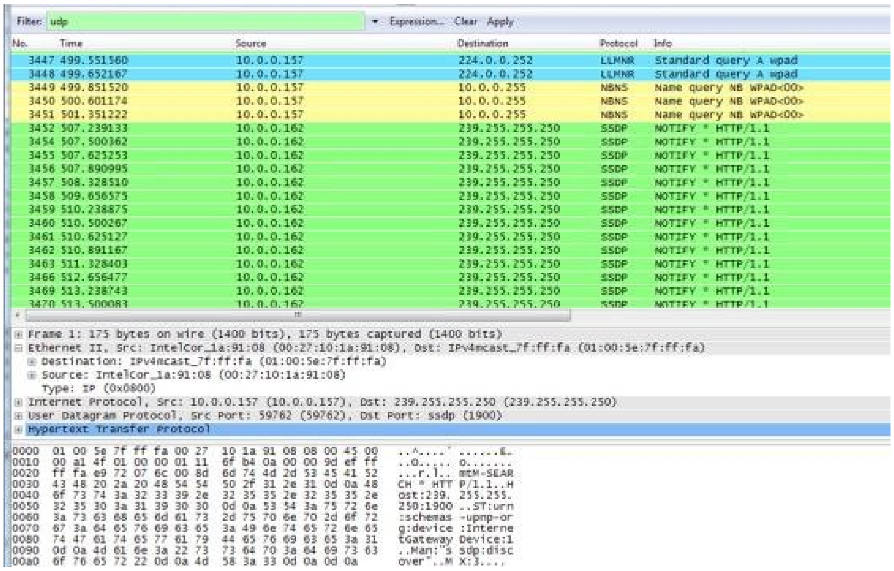
DATAFLOW DIAGRAM



SAMPLE CODING

```
from scapy.all import sniff, IP, TCP, UDP, ICMP


def packet_callback(packet):
    if packet.haslayer(IP):
        ip_layer = packet.getlayer(IP)
        print(f"Source IP: {ip_layer.src} -> Destination IP: {ip_layer.dst}")

        if packet.haslayer(TCP):
            tcp_layer = packet.getlayer(TCP)
            print(f"TCP Packet: Src Port: {tcp_layer.sport}, Dst Port: {tcp_layer.dport}")

        elif packet.haslayer(UDP):
            udp_layer = packet.getlayer(UDP)
            print(f"UDP Packet: Src Port: {udp_layer.sport}, Dst Port: {udp_layer.dport}")

        elif packet.haslayer(ICMP):
            icmp_layer = packet.getlayer(ICMP)
            print(f"ICMP Packet: Type: {icmp_layer.type}, Code: {icmp_layer.code}")

        print("-" * 50)

def start_sniffing(interface=None):
    print("Starting packet capture...")
    sniff(iface=interface, prn=packet_callback, store=0)

if __name__ == "__main__":
    network_interface = None  # You can specify the network interface here, e.g., 'eth0', 'wlan0'
    start_sniffing(network_interface)
```

SCREEN SHOT:



Filter: udp

No.	Time	Source	Destination	Protocol	Info
3447	499.551560	10.0.0.157	224.0.0.252	LLMNR	Standard query A wpad
3448	499.652167	10.0.0.157	224.0.0.252	LLMNR	Standard query A wpad
3449	499.851520	10.0.0.157	10.0.0.255	NBNS	Name query NB WPAD<00>
3450	500.601174	10.0.0.157	10.0.0.255	NBNS	Name query NB WPAD<00>
3451	501.351222	10.0.0.157	10.0.0.255	NBNS	Name query NB WPAD<00>
3452	507.239133	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3454	507.500362	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3455	507.625253	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3456	507.890995	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3457	508.328510	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3458	509.656575	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3459	510.238875	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3460	510.500267	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3461	510.625127	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3462	510.891167	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3463	511.328403	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3466	512.656477	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3469	513.238743	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1
3470	513.500083	10.0.0.162	239.255.255.250	SSDP	NOTIFY * HTTP/1.1

```
⊞ Frame 1: 175 bytes on wire (1400 bits), 175 bytes captured (1400 bits)
⊟ Ethernet II, Src: IntelCor_1a:91:08 (00:27:10:1a:91:08), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
  ⊞ Destination: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
  ⊞ Source: IntelCor_1a:91:08 (00:27:10:1a:91:08)
    Type: IP (0x0800)
⊞ Internet Protocol, Src: 10.0.0.157 (10.0.0.157), Dst: 239.255.255.250 (239.255.255.250)
⊞ User Datagram Protocol, Src Port: 59762 (59762), Dst Port: ssdp (1900)
⊞ Hypertext Transfer Protocol
```

```
0000  01 00 5e 7f ff fa 00 27  10 1a 91 08 08 00 45 00   ..^....' ......E.
0010  00 a1 4f 01 00 00 01 11  6f b4 0a 00 00 9d ef ff   ..O..... o.......
0020  ff fa e9 72 07 6c 00 8d  6d 74 4d 2d 53 45 41 52   ...r.l.. mtM-SEAR
0030  43 48 20 2a 20 48 54 54  50 2f 31 2e 31 0d 0a 48   CH * HTT P/1.1..H
0040  6f 73 74 3a 32 33 39 2e  32 35 35 2e 32 35 35 2e   ost:239. 255.255.
0050  32 35 30 3a 31 39 30 30  0d 0a 53 54 3a 75 72 6e   250:1900 ..ST:urn
0060  3a 73 63 68 65 6d 61 73  2d 75 70 6e 70 2d 6f 72   :schemas -upnp-or
0070  67 3a 64 65 76 69 63 65  3a 49 6e 74 65 72 6e 65   g:device :Interne
0080  74 47 61 74 65 77 61 79  44 65 76 69 63 65 3a 31   tGateway Device:1
0090  0d 0a 4d 61 6e 3a 22 73  73 64 70 3a 64 69 73 63   ..Man:"s sdp:disc
00a0  6f 76 65 72 22 0d 0a 4d  58 3a 33 0d 0a 0d 0a      over"..M X:3....
```