Basic Network Sniffer – CodeAlpha Cyber Security Internship

Introduction

A network sniffer is a tool that captures and analyzes packets traveling through a network. It helps security professionals understand data flow, identify potential threats, and debug issues. This project demonstrates a Basic Packet Sniffer in Python using the scapy library.

Objectives

- Capture live network packets
- Extract useful information (Source/Destination IPs, Protocol, Payload)
- Learn basics of packet structures and protocols
- Save captured packets for later analysis in Wireshark

Methodology

1. Install required Python library (scapy)
2. Write a Python program using sniff() to capture packets
3. Process each packet using a callback function
4. Optionally save packets to a .pcap file for Wireshark analysis
5. Test the program in real-time

Code Implementation

Basic Network Sniffer

```
from scapy.all import sniff

def packet_callback(packet):
    print(packet.summary())

print("Starting Packet Sniffer... Press CTRL+C to stop.")
sniff(prn=packet_callback, count=20)  # Capture 20 packets
```

Additional Features:

- Filter by Protocol (TCP Only)

```
sniff(filter="tcp", prn=packet_callback, count=10)
```

- Save Packets for Wireshark

```
from scapy.all import wrpcap
packets = sniff(count=50)
wrpcap("captured_packets.pcap", packets)
```

- Continuous Live Capture

```
sniff(prn=packet_callback, store=False)
```

▯ Sample Output

```
Ether / IP / TCP 192.168.1.5:49832 > 142.250.182.14:https S
Ether / IP / UDP 192.168.1.5:56892 > 8.8.8.8:domain
Ether / IP / ICMP 192.168.1.5 > 192.168.1.1 echo-request
```

✓Conclusion

The Basic Network Sniffer project shows how Python and Scapy can capture and analyze network packets. It provides hands-on exposure to network monitoring and can be extended with filtering, alerting, or integration into Intrusion Detection Systems (IDS).