Task 1: Basic Network Sniffer

Introduction

In today's interconnected world, cybersecurity professionals must possess a deep understanding of how data flows through a network. At the core of this understanding lies packet analysis, which enables security analysts to inspect and interpret the low-level details of network communications. This task aimed to provide a practical foundation in packet analysis by building a custom network sniffer tool using Python and the Scapy library.
The goal of this exercise was not only to develop a functional tool capable of capturing and analyzing network traffic but also to understand the structure and behavior of packets as they traverse the network. By dissecting real-time packets at the IP, TCP, UDP, and ICMP levels, this project allowed me to explore core networking concepts such as protocols, ports, payloads, and headers.

Through this task, I became familiar with how legitimate network traffic is structured, and how anomalies or malicious activity might stand out. This knowledge is fundamental for roles such as penetration testers, network administrators, and threat hunters who must continuously monitor, analyze, and secure networks against evolving threats.

Objective

The objective of this task was to develop a simple yet effective packet sniffer tool in Python using the scapy library. The tool captures real-time network traffic, analyzes each packet's structure, and displays important details such as:
- Source and destination IP addresses
- Protocol types (TCP, UDP, ICMP)
- Source and destination ports
- Raw payload (when available)

This task helps in understanding the basics of networking, packet structures, and how data flows through a network.

Skills Demonstrated

- Python scripting for networking tools
- Real-time packet capture and analysis
- Use of scapy for protocol-level inspection
- Understanding of TCP/IP, UDP, ICMP protocols
- Ethical considerations of packet sniffing

TOOLS USED

Tools and Technologies Used

Tool/Library	Purpose
Python	Programming language used for building the sniffer
Scapy	Python library used for capturing and analyzing network packets
Command Prompt / Terminal	Execution environment
Windows OS (with Wi-Fi interface)	Host system for packet capture

Task Description

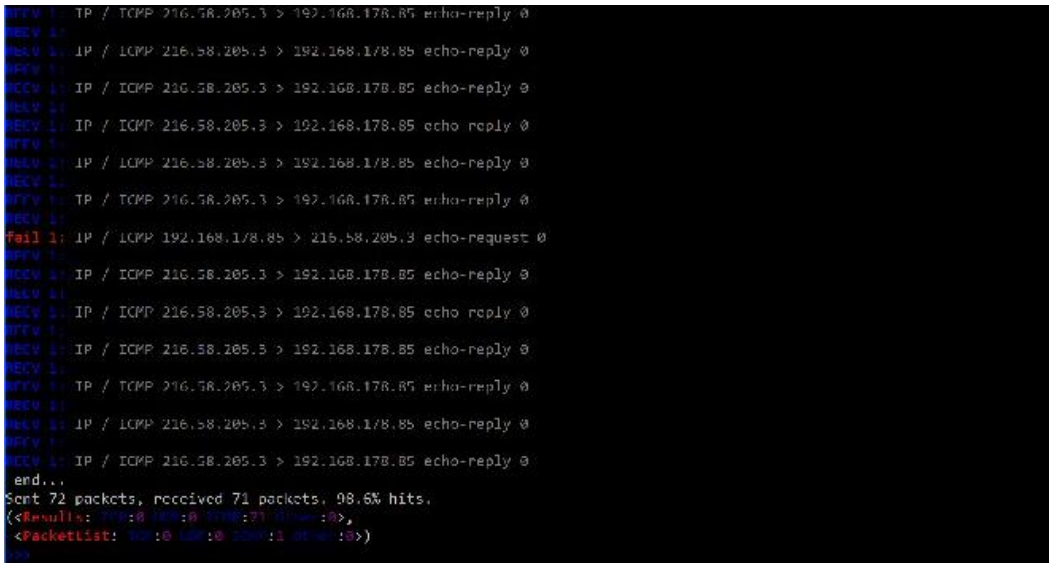
I developed a Python-based packet sniffer that:
1. Captures packets live from a specified network interface.
2. Analyzes each packet for IP layer, and extracts protocol-specific data.
3. Prints useful information such as:
   - Source and Destination IP
   - Protocol (TCP/UDP/ICMP)
   - Source and Destination Ports (for TCP/UDP)
   - ICMP Type and Code (for ICMP)
   - Payload (when available)

Code Implementation

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

def analyze_packet(packet):
    print("\n--- Packet Captured ---")

    if IP in packet:
        ip_layer = packet[IP]
        print(f"Source IP: {ip_layer.src}")
        print(f"Destination IP: {ip_layer.dst}")
        print(f"Protocol: {ip_layer.proto}")

        if TCP in packet:
            print("Layer: TCP")
            print(f"Source Port: {packet[TCP].sport}")
            print(f"Destination Port: {packet[TCP].dport}")

        elif UDP in packet:
            print("Layer: UDP")
            print(f"Source Port: {packet[UDP].sport}")
            print(f"Destination Port: {packet[UDP].dport}")

        elif ICMP in packet:
            print("Layer: ICMP")
            print(f"Type: {packet[ICMP].type}")
            print(f"Code: {packet[ICMP].code}")

        if packet.haslayer(Raw):
            print("Payload:")
            print(packet[Raw].load)
    else:
        print("Non-IP Packet")

print("Starting packet capture... Press Ctrl+C to stop.")
sniff(prn=analyze_packet, iface="Wi-Fi", count=10)
```

Screenshots

Scapy installation



Code

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

def analyze_packet(packet):
    print("\n--- Packet Captured ---")

    if IP in packet:
        ip_layer = packet[IP]
        print(f"Source IP: {ip_layer.src}")
        print(f"Destination IP: {ip_layer.dst}")
        print(f"Protocol: {ip_layer.proto}")

        if TCP in packet:
            tcp_layer = packet[TCP]
            print("Layer: TCP")
            print(f"Source Port: {tcp_layer.sport}")
            print(f"Destination Port: {tcp_layer.dport}")

        elif UDP in packet:
            udp_layer = packet[UDP]
            print("Layer: UDP")
            print(f"Source Port: {udp_layer.sport}")
            print(f"Destination Port: {udp_layer.dport}")

        elif ICMP in packet:
            icmp_layer = packet[ICMP]
            print("Layer: ICMP")
            print(f"Type: {icmp_layer.type}")
            print(f"Code: {icmp_layer.code}")

        if packet.haslayer(Raw):
            print("Payload:")
            try:
                print(packet[Raw].load.decode('utf-8', errors='ignore'))  # safer decode
            except:
                print(packet[Raw].load)
```

```
    else:
        print("Non-IP Packet")

# Replace 'Wi-Fi' with your actual interface name if needed
print("Starting packet capture... Press Ctrl+C to stop.")
sniff(prn=analyze_packet, iface="Wi-Fi", count=10)

```

OUTPUT

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL> & C:/Users/DELL/AppData/Local/Programs/Python/Python313/python.
Starting packet capture... Press Ctrl+C to stop.

--- Packet Captured ---
Source IP: 192.168.1.164
Destination IP: 13.89.179.13
Protocol: 6
Layer: TCP
Source Port: 64446
Destination Port: 443

--- Packet Captured ---
Source IP: 13.89.179.13
Destination IP: 192.168.1.164
Protocol: 6
Layer: TCP
Source Port: 443
Destination Port: 64445

--- Packet Captured ---
Source IP: 13.89.179.13
Destination IP: 192.168.1.164
Protocol: 6
Layer: TCP
Source Port: 443
Destination Port: 64447
Payload:
bHrx1\#GxrfP8▯%M[}k#UWlZKfX14d[tAy
3HwV.)"|a=

--- Packet Captured ---
Source IP: 13.89.179.13
Destination IP: 192.168.1.164
Protocol: 6
Layer: TCP
Source Port: 443
Destination Port: 64447
Payload:
9G7{Zfg8~c|Œ<PW;HdhN
```

```
--- Packet Captured ---
Source IP: 192.168.1.164
Destination IP: 13.89.179.13
Protocol: 6
Layer: TCP
Source Port: 64447
Destination Port: 443

--- Packet Captured ---
Non-IP Packet

--- Packet Captured ---
Non-IP Packet

--- Packet Captured ---
Non-IP Packet

--- Packet Captured ---
Non-IP Packet
--- Packet Captured ---
Non-IP Packet
Non-IP Packet


--- Packet Captured ---
Non-IP Packet

--- Packet Captured ---
Non-IP Packet


--- Packet Captured ---
Non-IP Packet
Non-IP Packet

--- Packet Captured ---
Non-IP Packet
```

Sample Output Explanation

Packet Captured ---

Source IP: 13.89.179.13

Destination IP: 192.168.1.164

Protocol: 6

Layer: TCP

Source Port: 443

Destination Port: 64447

Payload:

bHrx1\#GxrfP8 ·%M[}k#UWlZKfX14d[tAy

3Hwv.)"|a=

Learning Outcome

- Gained hands-on experience with **packet-level network analysis**.
- Learned how to differentiate between TCP, UDP, and ICMP packets.
- Understood how raw payloads are embedded within packets.
- Learned about **ethical boundaries** of sniffing and the importance of using this skill responsibly.

Ethical Consideration

This tool was tested only on my own network interface and did not involve unauthorized access to any third-party network. Packet sniffing should always be done with proper permissions and legal compliance.

Conclusion

The packet sniffer tool built during this task served as a valuable learning platform for understanding the inner workings of network traffic. It enabled real-time capture and inspection of packets, providing insights into how protocols like TCP, UDP, and ICMP function at the transport and network layers. By analyzing key attributes such as source/destination IPs, ports, protocols, and raw payloads, I developed the ability to interpret and explain what happens during common network operations like HTTPS requests, pings, or data transfers.

Beyond technical implementation, this task reinforced the importance of ethical boundaries in cybersecurity. Packet sniffing has legitimate use cases in network diagnostics and security monitoring, but it also poses privacy and legal concerns if misused. Throughout this project, ethical principles were strictly followed by working only on my own network and systems.

This experience has significantly strengthened my understanding of network fundamentals and packet-based communication. It has also laid the groundwork for more advanced projects involving intrusion detection, traffic filtering, or even malware detection through traffic pattern analysis. Overall, this task was an essential step in my journey to becoming a proficient cybersecurity analyst, equipped with both theoretical knowledge and practical skills in network forensics and traffic analysis.