Basic Python Network Sniffer

Objective

Build a Python program to capture and analyze network traffic packets using libraries like scapy or socket.

Key Features

- Capture network traffic packets
- Analyze captured packets to understand structure and content
- Learn how data flows through a network and understand protocols
- Display source/destination IPs, protocols, and payloads

Libraries Used

- scapy: Powerful Python-based packet manipulation tool
- socket: Standard Python library for low-level network interactions

Code Example (Using scapy)

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


def packet_callback(packet):
    if IP in packet:
        ip_layer = packet[IP]
        print(f"\n[+] Packet: {ip_layer.src} -> {ip_layer.dst}")
        print(f"    Protocol: {ip_layer.proto}")


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

        if Raw in packet:
            print("    Payload:")
            try:
                print(f"    {packet[Raw].load.decode('utf-8', errors='ignore')}")
            except:
                print("    [Payload not decodable]")

print("Starting packet capture... Press Ctrl+C to stop.")
sniff(prn=packet_callback, count=10)
```

Usage Tips

- Run the script with administrator/root privileges
- Use virtual environments to manage dependencies
- You can capture indefinitely by removing the 'count=10' parameter

Learning Outcome

This project helps understand how network packets are structured and transmitted, the role of protocols, and how to analyze data packets in a real-world scenario.