**Project Title: Network Sniffer  
Submitted By: Muhammad Ismail Awan , Ahan Ali  
Roll Number: 22K-4225 , 22K-4180  
Department: Computer Science  
Date: 6-05-2025**

**2. Introduction**

The internet has grown to become a vital part of communication in today's digital world. Monitoring network activity is important for security, troubleshooting, and performance analysis. This project, titled **"Network Sniffer"**, aims to develop a Python-based tool that captures, analyzes, and displays information from raw network packets.

The sniffer operates at the IP layer and supports protocol-level parsing for **ICMP**, **TCP**, and **UDP** packets. The application captures real-time data being transmitted over a network and decodes it into a human-readable format.

**3. Objective**

* To build a basic packet sniffer using Python.
* To analyze packet headers of different protocols (IPv4, ICMP, TCP, UDP).
* To extract useful information such as IP addresses, port numbers, flags, TTL, etc.
* To learn low-level networking using raw sockets in Python.

**4. Technologies Used**

* **Programming Language:** Python
* **Modules:** socket, struct, textwrap
* **Platform:** Windows (with administrator privileges)
* **Tools:** Command Prompt, Python 3.x

**5. Code Explanation**

**5.1. Socket Creation**

python

CopyEdit

conn = socket.socket(socket.AF\_INET, socket.SOCK\_RAW, socket.IPPROTO\_IP)

conn.bind(('192.168.43.200', 0))

conn.ioctl(socket.SIO\_RCVALL, socket.RCVALL\_ON)

* Raw socket created to receive low-level packets.
* Bound to a local IP address.
* Promiscuous mode enabled to capture all packets.

**5.2. IPv4 Header Parsing**

python

CopyEdit

def ipv4\_packet(data):

...

* Extracts version, header length, TTL, protocol, source and destination IP.

**5.3. ICMP, TCP, UDP Parsing**

* Each protocol has a dedicated function that decodes its header fields.
* **ICMP:** type, code, checksum.
* **TCP:** ports, sequence, acknowledgment, flags.
* **UDP:** ports and length.

**5.4. Main Loop**

python

CopyEdit

while True:

raw\_data, addr = conn.recvfrom(65536)

* Continuously listens for new packets.
* Delegates parsing based on the protocol type.
* Displays formatted results on the console.

**5.5. Complete Code**import socket

import struct

import textwrap

PROTOCOLS = {

    1: 'ICMP',

    6: 'TCP',

    17: 'UDP'

}

def ipv4\_packet(data):

    version\_header\_length = data[0]

    version = version\_header\_length >> 4

    header\_length = (version\_header\_length & 15) \* 4

    ttl, proto, src, target = struct.unpack('! 8x B B 2x 4s 4s', data[:20])

    return version, header\_length, ttl, proto, ipv4(src), ipv4(target), data[header\_length:]

def ipv4(addr):

    return '.'.join(map(str, addr))

def icmp\_packet(data):

    icmp\_type, code, checksum = struct.unpack('! B B H', data[:4])

    return icmp\_type, code, checksum, data[4:]

def tcp\_segment(data):

    (src\_port, dest\_port, sequence, acknowledgment, offset\_reserved\_flags) = struct.unpack('! H H L L H', data[:14])

    offset = (offset\_reserved\_flags >> 12) \* 4

    flag\_urg = (offset\_reserved\_flags & 32) >> 5

    flag\_ack = (offset\_reserved\_flags & 16) >> 4

    flag\_psh = (offset\_reserved\_flags & 8) >> 3

    flag\_rst = (offset\_reserved\_flags & 4) >> 2

    flag\_syn = (offset\_reserved\_flags & 2) >> 1

    flag\_fin = offset\_reserved\_flags & 1

    return src\_port, dest\_port, sequence, acknowledgment, flag\_urg, flag\_ack, flag\_psh, flag\_rst, flag\_syn, flag\_fin, data[offset:]

def udp\_segment(data):

    src\_port, dest\_port, size = struct.unpack('! H H 2x H', data[:8])

    return src\_port, dest\_port, size, data[8:]

def main():

    try:

        # Create a raw socket for IPv4 packets

        conn = socket.socket(socket.AF\_INET, socket.SOCK\_RAW, socket.IPPROTO\_IP)

        # Bind to all interfaces

        conn.bind(('192.168.0.197', 0))

        # Enable promiscuous mode to capture all packets

        conn.ioctl(socket.SIO\_RCVALL, socket.RCVALL\_ON)

        print("Packet sniffer started. Press Ctrl+C to stop...")

        while True:

            raw\_data, addr = conn.recvfrom(65536)

            version, header\_length, ttl, proto, src, target, data = ipv4\_packet(raw\_data)

            print('\nIPv4 Packet:')

            print(f'Version: {version}, Header Length: {header\_length}, TTL: {ttl}')

            print(f'Protocol: {PROTOCOLS.get(proto)}, Source: {src}, Target: {target}')

            if proto == 1:

                icmp\_type, code, checksum, data = icmp\_packet(data)

                print('ICMP Packet:')

                print(f'Type: {icmp\_type}, Code: {code}, Checksum: {checksum}')

            elif proto == 6:

                src\_port, dest\_port, sequence, acknowledgment, flag\_urg, flag\_ack, flag\_psh, flag\_rst, flag\_syn, flag\_fin, data = tcp\_segment(data)

                print('TCP Segment:')

                print(f'Source Port: {src\_port}, Destination Port: {dest\_port}')

                print(f'Sequence: {sequence}, Acknowledgment: {acknowledgment}')

                print('Flags:')

                print(f'URG: {flag\_urg}, ACK: {flag\_ack}, PSH: {flag\_psh}, RST: {flag\_rst}, SYN: {flag\_syn}, FIN: {flag\_fin}')

            elif proto == 17:

                src\_port, dest\_port, length, data = udp\_segment(data)

                print('UDP Segment:')

                print(f'Source Port: {src\_port}, Destination Port: {dest\_port}, Length: {length}')

    except PermissionError:

        print("Error: This script requires administrative privileges to capture raw packets. Please run Command Prompt as Administrator and try again.")

    except KeyboardInterrupt:

        print("\nStopping packet sniffer...")

    finally:

        try:

            # Disable promiscuous mode

            conn.ioctl(socket.SIO\_RCVALL, socket.RCVALL\_OFF)

            conn.close()

        except:

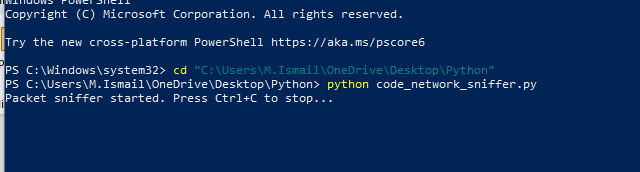
            pass

if \_\_name\_\_ == '\_\_main\_\_':

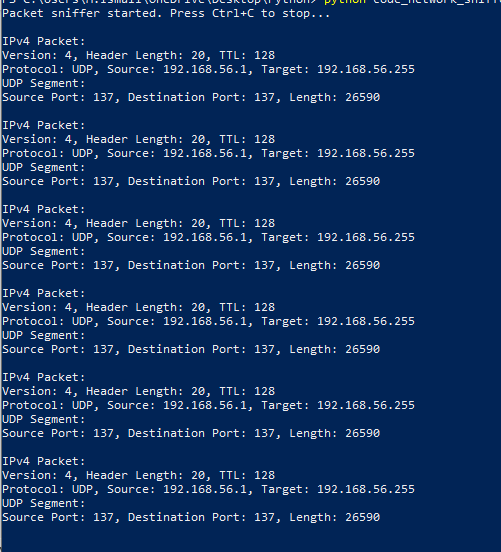
    main()

**6. Sample Outputs**

**Screenshot 1: Sniffer Start**



**Screenshot 2: IPv4 Packet Captured**



**Screenshot 3: TCP Packet Details**

A blue screen with white text

AI-generated content may be incorrect.

**7. Conclusion**

The **Network Sniffer** project demonstrates how low-level network monitoring can be achieved using Python's socket module and raw sockets. It successfully decodes IPv4, ICMP, TCP, and UDP packets and provides insights into the packet contents. This tool is a foundational step for building more advanced applications like intrusion detection systems or custom firewalls.

**8. Limitations and Future Work**

* Requires admin/root privileges to run.
* Only works on specific OS platforms (Windows in this case).
* Limited to basic protocol parsing.

**Future Enhancements:**

* GUI interface for visualization.
* IPv6 support.
* Packet filtering by protocol or IP.
* Logging captured data to file.