**Basic Network Sniffer using Python**

**Title Page**

**Project Title**: Basic Network Sniffer using Python

|  |  |
| --- | --- |
| |  | | --- | | **Name**: HAJRA KHALID **Submission Date**: 27 july,2025 | |

Table of Contents

1. Introduction
2. Objectives
3. Tools and Environment
4. Theoretical Background
5. System Requirements
6. Implementation (with Code)
7. Sample Output
8. Challenges Faced
9. Learning Outcomes
10. Conclusion
11. References

## 1. Introduction

A network sniffer is a tool that captures and analyzes packets of data that are transmitted over a network. It allows users to inspect the contents of network traffic in real time. This project implements a basic network sniffer using Python, allowing students to gain practical experience in low-level network programming and understanding of packet structures.

## 2. Objectives

* Build a simple network sniffer using Python’s socket module.
* Learn how to work with raw sockets.
* Understand the structure and decoding of IP headers.
* Gain hands-on experience in packet-level data inspection.

## 3. Tools and Environment

* **Programming Language**: Python 3.x
* **Editor**: Visual Studio Code (VS Code)
* **Operating System**: Windows 10/11
* **Modules Used**: socket, struct
* **Permissions**: Must be run as administrator due to raw socket usage

## 4. Theoretical Background

### What is Packet Sniffing?

Packet sniffing involves capturing and logging traffic passing over a network. Sniffers are used in network diagnostics, monitoring, and ethical hacking.

### What is a Raw Socket?

A raw socket allows direct sending and receiving of IP packets without any protocol-specific transport formatting. It’s used when we want full control over the headers.

### What is the IP Protocol?

The Internet Protocol (IP) handles addressing and routing packets across networks. Each IP packet has a header with information like version, source IP, destination IP, TTL, and protocol type.

### What is Promiscuous Mode?

A mode where a network interface captures all packets, not just those addressed to it. This mode is used in full-featured sniffers but may not be enabled in basic raw socket sniffers.

### Structure of an IP Packet Header

* **Version**: IP version (IPv4 = 4)
* **IHL (Header Length)**
* **TTL (Time to Live)**
* **Protocol**: Indicates whether it’s TCP (6), UDP (17), etc.
* **Source IP Address**
* **Destination IP Address**

## 5. System Requirements

* Python 3.x installed
* Administrator access
* Active network interface (Ethernet/Wi-Fi)
* VS Code with Python extension

## 6. Implementation (with Code)

### Code

import socket  
import struct  
  
def parse\_ip\_header(data):  
 ip\_header = struct.unpack('!BBHHHBBH4s4s', data[:20])  
 version\_ihl = ip\_header[0]  
 version = version\_ihl >> 4  
 ihl = version\_ihl & 0xF  
 ttl = ip\_header[5]  
 protocol = ip\_header[6]  
 src\_ip = socket.inet\_ntoa(ip\_header[8])  
 dest\_ip = socket.inet\_ntoa(ip\_header[9])  
  
 print("--- IP Header ---")  
 print(f"Version: {version}")  
 print(f"Header Length: {ihl \* 4}")  
 print(f"TTL: {ttl}")  
 print(f"Protocol: {protocol}")  
 print(f"Source IP: {src\_ip}")  
 print(f"Destination IP: {dest\_ip}")  
  
def sniff():  
 sniffer = socket.socket(socket.AF\_INET, socket.SOCK\_RAW, socket.IPPROTO\_IP)  
 sniffer.bind(("YOUR\_LOCAL\_IP", 0))  
 sniffer.setsockopt(socket.IPPROTO\_IP, socket.IP\_HDRINCL, 1)  
 sniffer.ioctl(socket.SIO\_RCVALL, socket.RCVALL\_ON)  
  
 print("Sniffer started... Press Ctrl+C to stop.")  
 try:  
 while True:  
 raw\_data = sniffer.recvfrom(65565)[0]  
 parse\_ip\_header(raw\_data)  
 except KeyboardInterrupt:  
 print("\nStopping sniffer...")  
 sniffer.ioctl(socket.SIO\_RCVALL, socket.RCVALL\_OFF)  
 sniffer.close()  
  
sniff()

## 7. Sample Output

--- IP Header ---  
Version: 4  
Header Length: 20  
TTL: 128  
Protocol: 6  
Source IP: 192.168.1.101  
Destination IP: 142.250.190.78

## 8. Challenges Faced

* Raw sockets require administrative privileges, which caused initial permission errors.
* IP headers require byte-level parsing using struct.unpack().
* Understanding endian formats and IP packet structures took effort.

## 9. Learning Outcomes

* Gained hands-on experience in low-level socket programming.
* Understood how real-time network data can be intercepted and parsed.
* Improved knowledge of networking protocols and packet analysis.

## 10. Conclusion

This project successfully demonstrates how raw sockets in Python can be used to create a basic network sniffer. It provides foundational insight into packet inspection, which is essential for cybersecurity, diagnostics, and ethical hacking. The project can be extended to support parsing TCP/UDP headers, logging data, or adding a GUI.

## 11. References

* Python Socket Docs: <https://docs.python.org/3/library/socket.html>
* Struct Module Docs: <https://docs.python.org/3/library/struct.html>
* RealPython: <https://realpython.com/python-sockets/>