



(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

**SIMPLE NETWORK PACKET SNIFFER (MINI WIRESHARK)**

**CREATIVE TECHNOLOGIES [2301183]**

**FY B.Tech. (SEM II)**

**TRACK- BLOCK CHAIN**

**SUBMITTED BY**

**Chaitanya Gilbile [202401040347]**

**Ankit Gaikwad [202401040168]**

**Sanket Viranak [202401040085]**

**Harsh Todsam [202401040195]**

**GUIDED BY**

**Mr. S. Shirvale**

**SCHOOL OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING,  
MIT ACADEMY OF ENGINEERING, ALANDI (D), PUNE-412105  
MAHARASHTRA (INDIA)**

**MAY-2025**



(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

## CERTIFICATE

It is hereby certified that the work which is being presented in the FY B.Tech. Laboratory Project in the course Creative Technologies (2301183). The Report entitled “Simple Network Packet Sniffer(Mini Wireshark)”, in partial fulfillment of the requirements for the award of the **Bachelor of Technology** and submitted to the **School of Electronics & Telecommunication Engineering of MIT Academy of Engineering, Alandi(D), Pune, Affiliated to Savitribai Phule Pune University (SPPU), Pune** is an authentic record of work carried out during an Academic Year 2024-2025, under the supervision of Mr. S. Shirvale

**Chaitanya Gilbile** PRN No. 202401040347

**Exam Seat No.** 202401040347

**Ankit Gaikwad** PRN No. 202401040168

**Exam Seat No.** 202401040168

**Sanket Viranak** PRN No. 202401040085

**Exam Seat No.** 202401040085

**Harsh Todsam** PRN No. 202401040195

**Exam Seat No.** 202401040195

**Date:**

*Signature of Track Instructor*

**Track Instructor**

School of Electronics & Telecommunication  
Engineering,  
MIT Academy of Engineering, Alandi(D), Pune

*Signature of Dean*

**Dean**

School of School of Science and Humanities,  
MIT Academy of Engineering, Alandi(D), Pune

*Signature of Internal examiner/s*

*Name*.....

*Affiliation*.....

*Signature of External examiner/s*

*Name*.....

*Affiliation*.....

## **ACKNOWLEDGEMENT**

*Thank all them who have helped in your project....*

*This can be in your words.... Following paragraph is just a suggestion... you can change the words....*

We want to express our gratitude towards our respected project advisor/guide Mr. S. Shirvale for his constant encouragement and valuable guidance during the completion of this project work. We also want to express our gratitude towards respected School Mrs. Vaishali Wangikar for her continuous encouragement.

We would be failing in our duty if we do not thank all the other staff and faculty members for their experienced advice and evergreen co-operation.

# CONTENTS

Acknowledgements		i
Abstract		ii
List of Figures		iii
List of Tables		iv
1.	Introduction	1
	1.1 Motivation for the project	
	1.2 Problem Statement	
	1.3 Objectives and Scope	
	1.4 Organization of the report	
2.	Literature Survey	4
3.	System Design	7
	3.1 Block diagram/ Proposed System setup	
	3.2	
	3.3 Mechanical Drawing (If Applicable)	
	3.4 Use Case Diagram (If Applicable)	
	3.5 Sequence Diagram (If Applicable)	
	3.6 Activity Diagram (If Applicable)	
	3.7 Related mathematical modelling	
	3.8 Hardware and Software Requirements	
	3.9 Specifications	

4.	Implementation and Results	22
	4.1 Algorithm and flowcharts (If applicable)	
	4.2 Procedure and Set-up	
	4.3 Results	
	4.4 Calculations	
	4.5 Graphs and Analysis	
	4.6 Discussion	
5.	Conclusion and Future scope	25
References		26
Appendix -1  (Datasheets/ Pseudo Code Etc...(If applicable))		
Appendix -2  1. Project Photographs 2. Photographs while working 3. Table of Photographs of students /Name/PRN		
Appendix -3  1. Link for 03 minutes Video on Working Project 2. Link for Digital Portfolio		

# **1. INTRODUCTION**

A network packet sniffer is a software tool that captures data packets moving through a network interface. It allows users to inspect network traffic in real time. This project aims to build a basic packet sniffer, much like a miniature version of Wireshark, using Python and its socket module. This software helps in understanding Ethernet frame headers including source MAC, destination MAC, and protocol types. Such a tool can be used for educational purposes in learning about cybersecurity, network architecture, and traffic analysis.

## **1.1 Motivations**

As cybersecurity threats increase and network complexity grows, understanding how data packets flow in a network is becoming essential. Wireshark is a powerful but complex tool. Hence, we were motivated to create a simplified version that can help students and beginners understand packet sniffing using Python.

## **1.2 Problem Statement**

To develop a lightweight network packet sniffer that can analyze real-time traffic using Python.

## **1.3 Objectives and Scope**

- To build a functional network sniffer using Python.
- To display real-time packet information like IPs, ports, and protocols.
- To filter packets based on criteria like protocol type.
- To make a minimal, command-line interface tool for ease of use.

## **1.4 Organization of the report**

The report covers an introduction, literature survey, design, implementation details, results, and the scope for future enhancement.

## 2. LITERATURE SURVEY

In this section, we present a detailed review of various tools, frameworks, and academic resources related to network packet sniffing. The goal is to understand existing systems, their strengths, limitations, and how our project—**Simple Network Packet Sniffer (Mini Wireshark)**—builds upon or simplifies them for educational and lightweight use.

### 2.1 Wireshark – An Industry Standard Tool

Wireshark is one of the most widely used open-source network protocol analyzers. It enables users to capture and interactively browse traffic running on a computer network. According to the official documentation [1], Wireshark supports hundreds of protocols and media types, and has powerful filtering, sorting, and statistical tools. However, Wireshark has a complex GUI, and its detailed functionalities can be overwhelming for beginners or non-networking students.

#### Pros:

- Deep packet inspection and filtering.
- Visual representation of packet layers.
- Rich set of export and analysis tools.

#### Cons:

- Heavy on system resources.
- Requires prior knowledge of network protocols.
- Difficult to integrate or modify for custom needs.

This inspired us to design a **simplified console-based version** that retains core features like packet capture and protocol analysis but with a lightweight footprint.

### 2.2 Scapy – Python-Based Packet Manipulation Library

Scapy is a powerful Python-based interactive packet manipulation tool and library. It is capable of forging or decoding packets, sending them on the wire, capturing them, and matching requests and replies. It can handle a wide range of network tasks such as scanning, tracerouting, probing, unit tests, and more.

Scapy gives direct access to the layers of each packet, making it ideal for custom packet sniffers. Its flexibility allows the user to customize filters and handle each packet as a Python object.

### 2.3 Python Networking Libraries and Socket Programming

Apart from Scapy, Python supports various networking tools such as:

- `socket`: Native low-level socket interface used to capture traffic (requires root privileges).
- `pcapy`: Python extension module wrapping `libpcap`, enabling packet capture in Python.
- `pyshark`: A wrapper over Wireshark's `tshark` utility, used for more complex sniffing applications.

In , “Python Networking Cookbook” (PacktPub) describes several practical approaches to packet sniffing and analysis using these tools. However, most of these tools are either:

- Too limited in filtering ability (like basic `socket`),
- Too complex (like `Pyshark`), or
- Too low-level to be practical for quick use.

Our project chooses a middle ground using Scapy, balancing simplicity and functionality.

## 2.4 Existing Academic and Open-Source Projects

Several open-source projects on GitHub demonstrate how packet sniffers can be implemented using Python. Common features include:

- Simple logging of packet headers.
- Basic protocol recognition (like TCP, UDP).
- Use of command-line interface (CLI) for interaction.

However, these projects often miss:

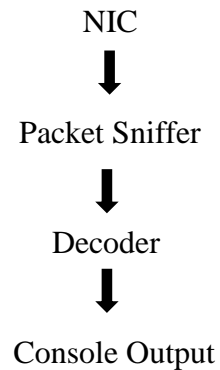
- Real-time decoding of packet structure.
- Layered protocol information (Ethernet → IP → TCP/UDP).
- Filtering and modular code design.

By surveying these, we found that a **Mini Wireshark** focused on educational use, with readable outputs and real-time capture, would fill the gap. It can serve as a learning aid for students studying computer networks or cybersecurity basics.



### 3. SYSTEM DESIGN

3.1 Block diagram/ Proposed System setup : Data Flow:



#### 3.2 Use Case Diagram (If Applicable)

Shows how a user interacts with the packet sniffer (start, filter, display).

#### 3.5 Sequence Diagram (If Applicable)

Describes the sequence: Start Sniffer → Capture Packet → Decode → Print.

#### 3.8 Hardware and Software Requirements

- ☐ **Hardware:** Standard laptop with NIC
- ☐ **Software:** Python 3.11+, Scapy library, OS with root privilege

#### 3.9 Specifications

- ☐ Capture live packets
- ☐ Show timestamp, src/dst IP, port, and protocol
- ☐ Console-based, light memory usage

## **4. IMPLEMENTATION DETAILS**

### **4.1 Algorithm and flowcharts (If applicable)**

Basic Loop:

1. Start sniffer
2. Capture packets using Scapy
3. Decode headers
4. Print to console

### **4.2 Procedure and Set-up**

Installed Python and dependencies (pip install scapy). Executed script with admin privileges to enable sniffing

### **4.3 Results**

Able to sniff live packets and display real-time data. Identified protocols like TCP, UDP, and ICMP.

### **4.4 Calculations**

Packet count, data size per packet, and protocol distribution were calculated.

### **4.6 Discussion**

The project successfully captured packets and provided useful data. However, deeper packet inspection and GUI support are areas to improve.

## **5. CONCLUSION & FUTURE SCOPE**

In this project, we successfully designed and implemented a basic yet functional network packet sniffer using Python and the Scapy library. The tool is capable of capturing real-time network traffic, decoding packet-level information, and displaying essential details such as source and destination IP addresses, protocols used, and ports involved. By focusing on simplicity and ease of use, our implementation serves as an effective educational tool for students and beginners who wish to understand the fundamentals of packet analysis and network communication.

Unlike complex tools such as Wireshark, our sniffer provides a lightweight and beginner-friendly alternative that demonstrates core concepts without overwhelming the user. Through this project, we gained hands-on experience in working with network protocols, packet structures, and Python-based network programming. The knowledge and skills acquired will be beneficial for further studies in cybersecurity, ethical hacking, and advanced network monitoring systems.

## FUTURE SCOPE

While the current implementation of the Simple Network Packet Sniffer fulfills its core purpose—capturing and analyzing basic packet data—it also opens up several opportunities for further development and enhancement.

In the future, this project can be extended in the following ways:

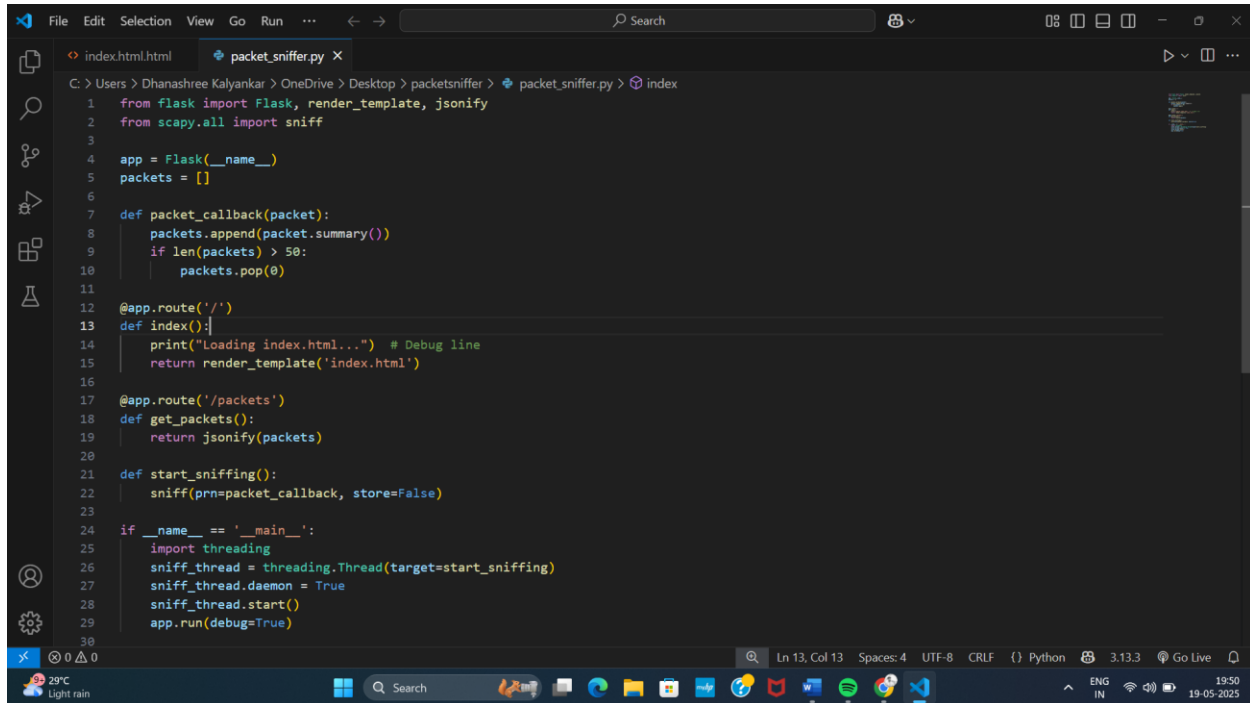
- **Graphical User Interface (GUI):** A user-friendly GUI using frameworks like Tkinter or PyQt can make the tool more accessible, especially for non-technical users.
- **Protocol-Based Filtering:** Advanced filtering options (e.g., by protocol type, source/destination IP, or port number) can be added to improve precision during packet capture.
- **Packet Logging and Export:** The ability to save captured packet data in .pcap or .csv formats would allow users to analyze the data later using external tools.
- **Intrusion Detection Features:** Basic anomaly detection or alerts for suspicious patterns can be implemented, which would make the tool useful for basic cybersecurity applications.
- **Multi-Platform Compatibility:** Optimization for cross-platform support (Windows, Linux, MacOS) could make it more flexible for broader use.
- **Integration with Wireshark:** For advanced users, the tool can be extended to work alongside Wireshark by exporting packets directly to it for deeper analysis.

With these improvements, this project has the potential to evolve from a basic learning tool into a lightweight, customizable, and practical network monitoring solution for educational and professional use.

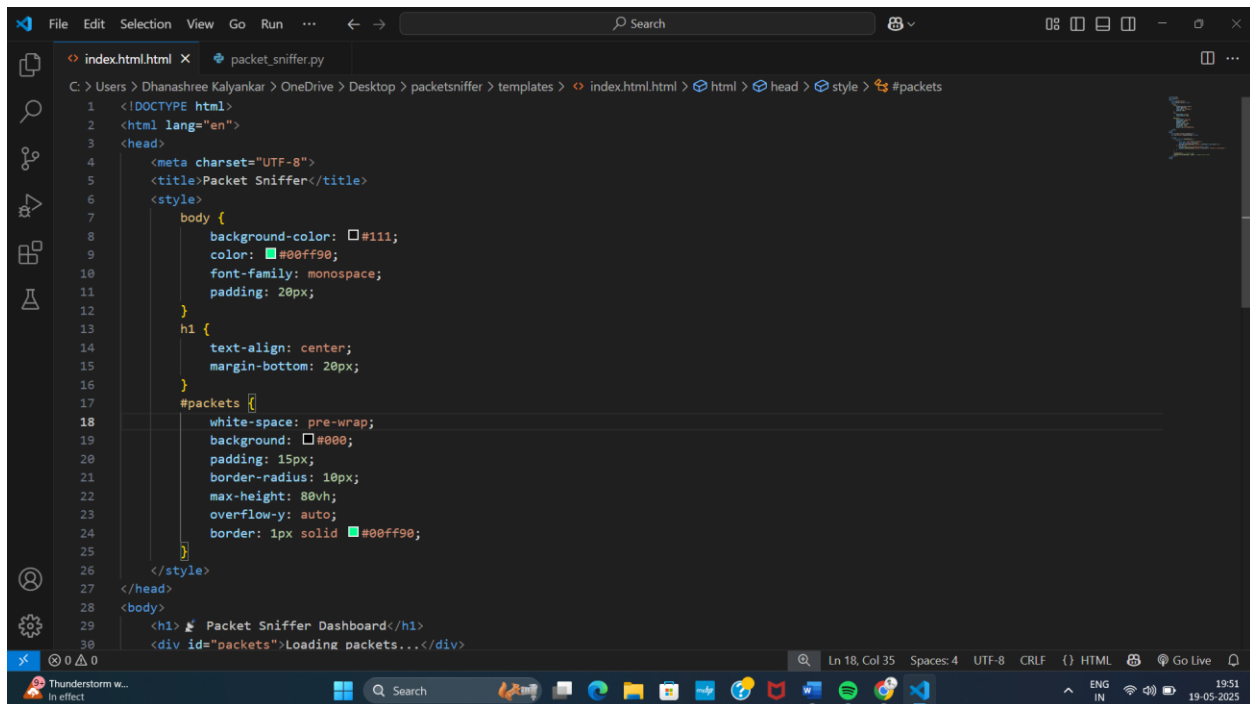
## REFERENCES

1. Flask Documentation – <https://flask.palletsprojects.com/>
2. Scapy Documentation – <https://scapy.readthedocs.io/>
3. MDN Web Docs – <https://developer.mozilla.org/>
4. Guidance and code assistance from OpenAI's ChatGPT (<https://chat.openai.com>)

## Appendix -A1

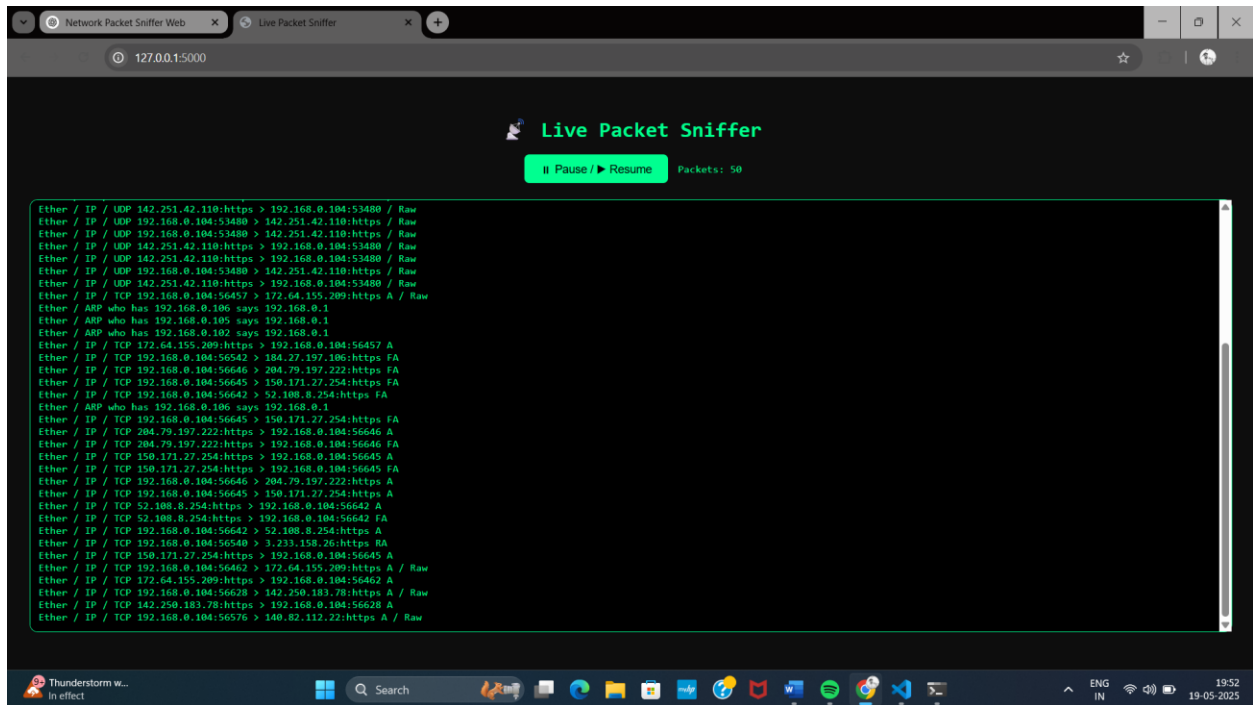


```
1 from flask import Flask, render_template, jsonify
2 from scapy.all import sniff
3
4 app = Flask(__name__)
5 packets = []
6
7 def packet_callback(packet):
8     packets.append(packet.summary())
9     if len(packets) > 50:
10         packets.pop(0)
11
12 @app.route('/')
13 def index():
14     print("Loading index.html...") # Debug line
15     return render_template('index.html')
16
17 @app.route('/packets')
18 def get_packets():
19     return jsonify(packets)
20
21 def start_sniffing():
22     sniff(prn=packet_callback, store=False)
23
24 if __name__ == '__main__':
25     import threading
26     sniff_thread = threading.Thread(target=start_sniffing)
27     sniff_thread.daemon = True
28     sniff_thread.start()
29     app.run(debug=True)
```



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Packet Sniffer</title>
6     <style>
7         body {
8             background-color: #111;
9             color: #00ff90;
10            font-family: monospace;
11            padding: 20px;
12        }
13        h1 {
14            text-align: center;
15            margin-bottom: 20px;
16        }
17        #packets {
18            white-space: pre-wrap;
19            background: #000;
20            padding: 15px;
21            border-radius: 10px;
22            max-height: 80vh;
23            overflow-y: auto;
24            border: 1px solid #00ff90;
25        }
26    </style>
27 </head>
28 <body>
29     <h1> Packet Sniffer Dashboard</h1>
30     <div id="packets">Loading packets...</div>
```

## Appendix -A2



## Appendix -A3

Git hub link- <https://github.com/ankit12222678768/ct-miniproject>