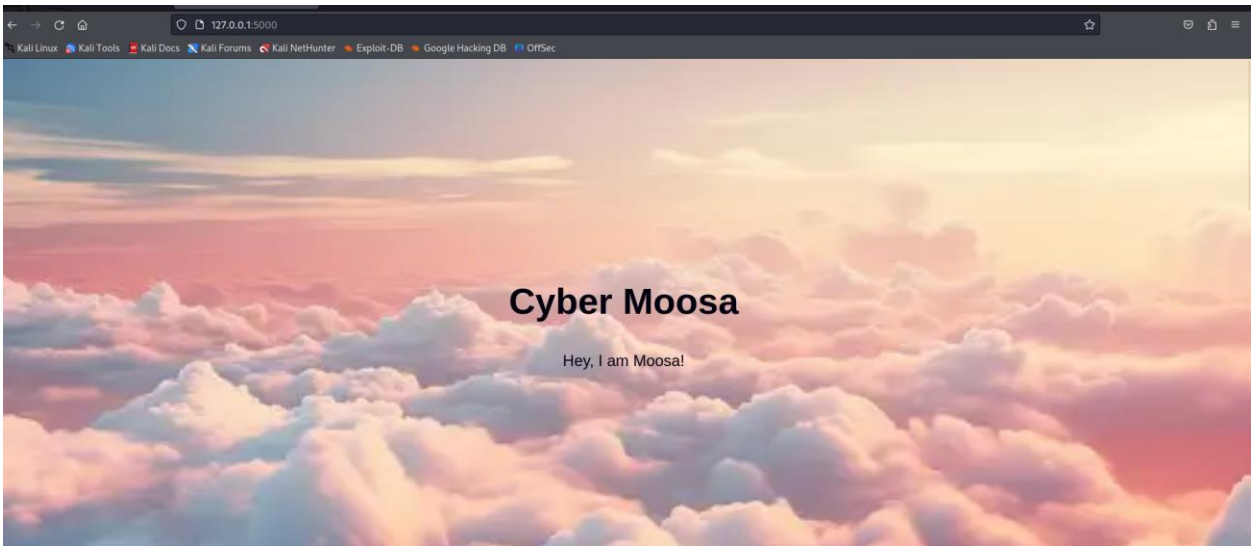
Packet Sniffer Using Python with a Web-Based Interface

Web Name:

CYBER MOOSA



- **Project Title**: Packet Sniffer Using Python with a Web-Based Interface
- **Submitted By**: Muhammad Moosa Ahmed [22k-4703]
- **Date of Submission**: 2-DEC-2024

Abstract

This project, *Packet Sniffer Using Python with a Web-Based Interface*, aims to develop a tool for capturing and analyzing network packets. The application features a user-friendly web interface, allowing users to initiate and terminate packet sniffing operations seamlessly. By integrating Python's robust networking libraries with a web-based design, the project provides a practical and accessible tool for real-time network analysis.

The tool is designed to support security professionals, developers, and network administrators in monitoring traffic and identifying potential vulnerabilities. It bridges the gap between advanced technical capabilities and usability by offering features such as packet capture, detailed inspection of data packets, and easy-to-navigate controls. The web-based interface enhances accessibility, making it suitable for diverse user groups, including beginners.

Introduction

Problem Statement

Network monitoring is essential for ensuring security and maintaining performance. Existing tools can be complex or lack an intuitive interface, making them less accessible for entry-level users.

Motivation

To bridge the gap between complexity and usability, this project introduces a packet sniffer with a web-based interface. The goal is to provide a visually appealing, easy-to-use tool for real-time packet analysis.

Tools and Technologies Used

- **Programming Language**: Python
- **Framework**: Flask for web development
- **Libraries**: scapy for packet capturing, HTML/CSS for UI design
- **Tools**: Kali Linux, web browsers for testing

System Design

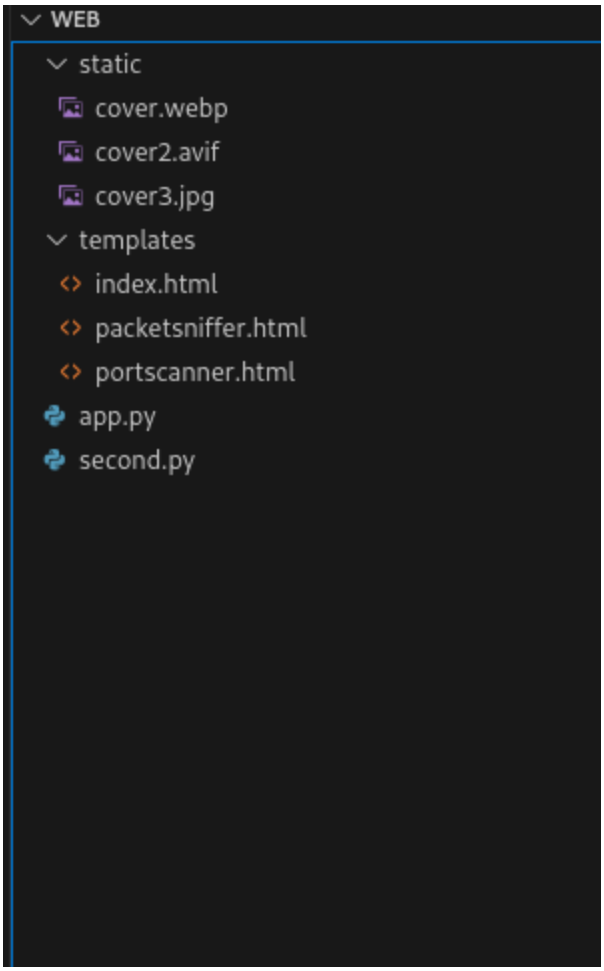
Architecture Overview

The system follows a client-server model:

- The backend, written in Python, manages packet capturing and processing.
- The frontend, built with HTML and CSS, provides an interface for user interaction.
- Flask connects the frontend and backend.

Python Scripts

1. **app.py**:
    - Acts as the main entry point for the application.
    - Handles routing and communication between the web interface and packet sniffing functionalities.
2. **second.py**:
    - Contains supporting logic for packet analysis and other secondary functionalities.


WEB
static
cover.webp
cover2.avif
cover3.jpg
templates
index.html
packetsniffer.html
portscanner.html
app.py
second.py

Features and Functionality

Main Pages

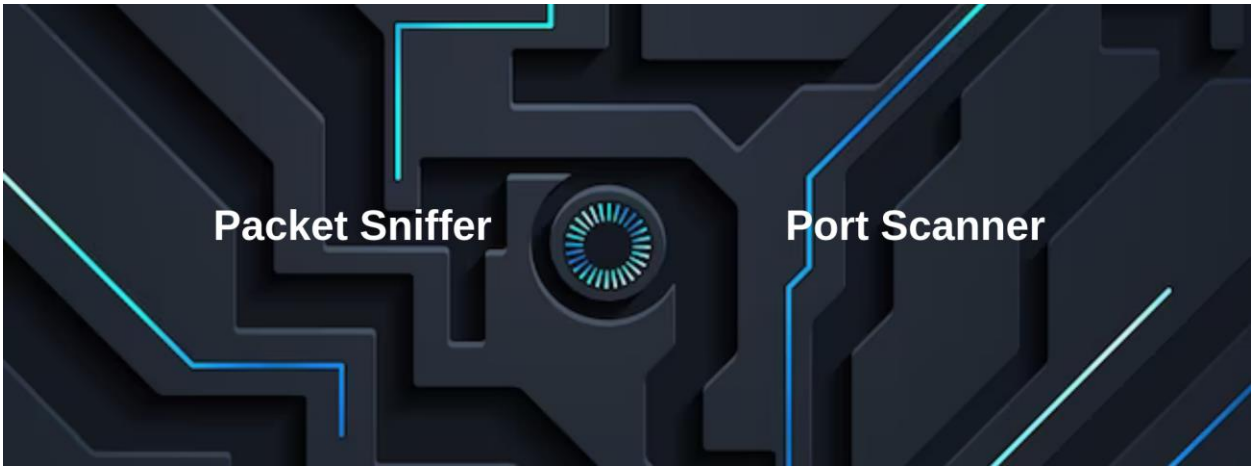
1. index.html (Landing Page)

- Purpose: Welcome page of the web interface.
- Includes navigation links to the packet sniffer and other tools.

2. packetsniffer.html (Packet Sniffer Page)

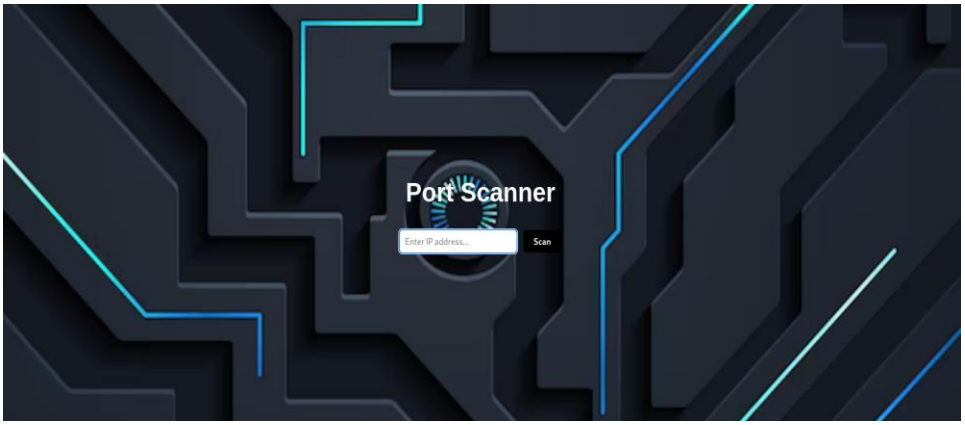
- Purpose: Provides packet sniffing functionality.

- Features:
    - **Start Button**: Begins capturing network packets.
    - **Stop Button**: Terminates the packet capture.
    - Displays captured packets with detailed information.
    - Clicking on a packet reveals more detailed analysis.



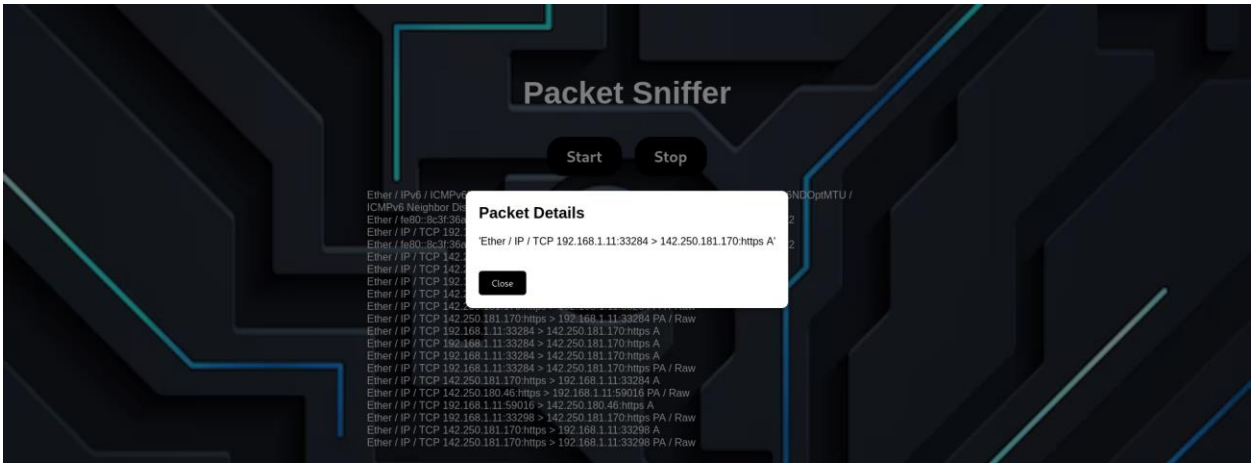
3. portscanner.html (Port Scanner Page/ extra feature for my linkdin)

- Purpose: (Optional if implemented) Provides a port scanning tool.
- Includes input fields for scanning target IP addresses



System Overview:

Packet Sniffer UI:



Challenges

Problems Faced

1. Integrating Python’s scapy library with Flask.
2. Handling real-time updates on the web interface.
3. Debugging network-related errors during packet sniffing.

Solutions

- Used asynchronous features to enhance real-time performance.
- Employed browser developer tools to debug and optimize UI.

Conclusion

Summary

This project successfully demonstrates a packet-sniffing application with a web-based interface. By combining Python's powerful networking libraries with a simple and intuitive UI, the tool makes network analysis accessible to users.

Future Enhancements

1. Add support for filtering packets by protocol (e.g., HTTP, TCP, UDP).
2. Implement export options for captured data.
3. Extend functionalities to include port scanning and network mapping.