IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

# Wi-Fi Packet Sniffer-Web Based Dashboard

*Analyzing and Monitoring Wi-fi Traffic*

A. Rohith[1], C. Sai Charanya[2], D. Sony Sri[3], K. Vishnu Vardhan[4], M. Naveen Kumar[5]
[12345] *Undergraduate Students*
*Hyderabad Institute of Technology and Management, Medchal, Hyderabad, Telangana*

***Abstract:*** This project presents a web-based Wi-Fi packet sniffer and analyser built with Python, Flask, and Scapy. It captures and analyses real-time network traffic over a wireless interface. The system tracks packets sent through the Wi-Fi network and retrieves key details, including source and destination IP addresses, protocol type, packet length, and, when available, hostname information. Unlike traditional command-line or standalone GUI tools like Wireshark, this tool provides a user-friendly dashboard in your browser for viewing and managing network activity as it occurs. Users can start, stop, and filter packet captures directly from the web interface, without needing to use technical commands.

The tool finds the active Wi-Fi interface automatically and sorts network traffic into common protocols like TCP, UDP, ICMP, HTTP, and HTTPS, making it easier to read and analyse. It also lets users search and filter by specific IP addresses or protocols, and export data as a CSV file for offline use or reports. By combining packet analysis with an interactive web interface, the project makes network monitoring more accessible, lightweight, and efficient. It is a helpful resource for students, network administrators, and cybersecurity learners to see real-world packet flow, spot unusual activity, and better understand how networks work.

***Index Terms*** - Wi-Fi Packet Sniffer, Flask Framework, Scapy, Network Traffic Analysis, Cybersecurity Monitoring

## I. INTRODUCTION

In today's rapidly evolving digital landscape, network monitoring and packet analysis play a vital role in maintaining security, optimising performance, and understanding data transmission across computer networks. While conventional tools such as Wireshark provide advanced packet inspection capabilities, they often require installation, intricate configuration, and technical expertise — creating barriers for new or non-technical users. To overcome these limitations, the Web-Based Wi-Fi Packet Sniffer and Analyser was developed as a lightweight, browser-accessible solution that enables real-time packet capture, inspection, and visualisation over Wi-Fi networks. Built using Python, Flask, and Scapy, the system offers an intuitive web dashboard that allows users to view, filter, and export live packet data, including protocol type, source and destination IP addresses, packet size, and hostnames, without relying on command-line interfaces.

By incorporating automated Wi-Fi interface detection and intelligent protocol classification, the proposed system simplifies the process of monitoring and analysing network traffic. Its ability to display, interpret, and export captured data in an interactive format makes it both a practical tool for cybersecurity professionals and an educational resource for students learning about network communication and data flow. Moreover, this project bridges the gap between traditional packet sniffing tools and modern web technologies by delivering cross-device accessibility without installation overhead. Leveraging Python's robust networking capabilities alongside Flask's lightweight web framework, it offers a responsive, real-time analysis experience directly through the browser. Designed with a balance of simplicity and depth, the system empowers beginners to visualise how data traverses networks while still offering the analytical power required by advanced users for detailed traffic inspection and forensic investigations.

## II. LITERATURE SURVEY

[1] Real-Time Network Packet Capture and Analysis Using Scapy and Python—S. Sharma et al. This paper discusses the development of a Python-based network packet analysis system utilizing the Scapy library for real-time packet capture and filtering. The study emphasizes Scapy's flexibility in handling multiple network protocols, allowing users to analyze live traffic and detect potential anomalies. It highlights how Scapy can be integrated into lightweight monitoring tools that do not require complex configurations like Wireshark. The researchers demonstrate how such tools can be extended for educational and cybersecurity applications, making them suitable for students and professionals to understand network behavior effectively. This approach supports modular design, enabling easy integration with visualization and reporting frameworks, which aligns closely with the goals of our project. [2] Web-Based Network Traffic Monitoring System Using Flask Framework— R. Kumar and A. Singh. This study focuses on building a web-based network traffic monitoring system using Python's Flask framework. The system allows real-time visualization of network packets through a web dashboard, enabling users to view protocol information, packet size, and data transfer rates. The authors highlight that combining Flask with background packet sniffing scripts provides a scalable and interactive way to monitor network performance without relying on traditional command-line tools. The research further suggests that web-based systems improve usability and accessibility, especially in remote and cross-platform environments. This concept directly influenced our project's design of a browser-accessible Wi-Fi packet analyser.[3]A Comparative Study of Packet Sniffing Tools for Network Analysis" — M. Patel et al. This paper evaluates different packet sniffing tools such as Wireshark, tcpdump, and Ettercap, focusing on their performance, usability, and accuracy in capturing and analyzing network traffic. The study reveals that while traditional tools provide comprehensive analysis, they often require installation, root access, and advanced technical knowledge. The authors suggest that modern approaches should emphasize web integration and automation to make network analysis more accessible to non-expert users. This insight reinforces our project's motivation to develop a web-based packet sniffer that combines powerful backend packet capture with a simple, user-friendly interface.[4] Design and Implementation of Lightweight Network Intrusion Detection Using Python — K. Verma and N. Gupta. The authors present a lightweight intrusion detection system developed in Python, designed to detect malicious traffic patterns using packet inspection and filtering techniques. The research demonstrates that real-time packet capture using Python's networking libraries can efficiently detect threats without heavy resource consumption. Their proposed framework shows how modular packet processing can be extended into security analysis or web-based dashboards. This study contributes to our project's goal of integrating real-time monitoring with potential future extensions for security and anomaly detection. [5] Implementation of Web-Based Network Packet Analyzer for Real-Time Data Visualization — P. Das and R. Nair. This paper presents a web-based packet analyser system designed to capture, process, and visualise live network traffic data in real time. The proposed model integrates backend packet capture modules with a web interface, enabling users to monitor active connections, bandwidth usage, and communication protocols from any browser. The authors emphasize the system's usability in educational and research environments, as it simplifies network monitoring without requiring deep networking expertise or command-line operations. The implementation leverages Python, JavaScript, and Flask for communication between backend and frontend, demonstrating how lightweight web frameworks can deliver powerful data visualisation tools. This study supports our project's objective of developing a browser-accessible Wi-Fi packet sniffer with real-time visualisation and user-friendly controls.[6] Occupancy Estimation Using Low-Cost Wi-Fi Sniffers— Paolo Galluzzi, Edoardo Longo, Alessandro E. C. Redondi, Matteo Cesana (2019). The system uses Wi-Fi management packet capture and sends data to a **web-based dashboard**. This demonstrates Wi-Fi sniffing + web visualization.[7] A Real-Time Streaming System for Customized Network Traffic Logging—A.T. Costin et al (2023). This focuses on packet capture and streaming to backends, which is useful for your real-time capture + dashboard work.

## III.EXISTING METHOD

The existing approaches to network monitoring largely rely on traditional tools such as Wireshark, tcpdump, and Ettercap. Although these tools provide comprehensive packet-level analysis, they also present several limitations. They often require manual configuration, administrative privileges, and substantial technical expertise, which makes them less accessible to non-expert users. Furthermore, these applications are typically standalone desktop-based solutions that lack web accessibility, thereby restricting capabilities such as real-time visualisation and remote monitoring.

In addition, conventional tools generally do not offer modern features such as user authentication, simplified deployment, or interactive browser-based dashboards. As a result, despite their analytical power, these systems are not well-suited for scenarios that demand ease of use, accessibility, and real-time, web-enabled network

analysis. The proposed project addresses these gaps by providing a lightweight, web-based packet sniffing and analysis platform designed for both simplicity and functionality.

## IV. PROPOSED METHOD

The proposed methodology for the Web-Based Wi-Fi Packet Sniffer and Analyser focuses on developing a secure, real-time network monitoring system capable of capturing, analysing, and visualising wireless packets through an interactive web dashboard. The system is designed to provide both accessibility and technical depth, allowing users to monitor Wi-Fi traffic seamlessly through a browser. The process begins with packet capture and data acquisition, where the Scapy library in Python is used to capture raw network packets from a Wi-Fi interface operating in promiscuous mode. This configuration allows the system to detect all traffic types, including TCP, UDP, ARP, and ICMP packets. Captured packets are filtered based on protocol identifiers and port numbers to extract essential attributes such as IP addresses, MAC addresses, protocols, and packet lengths. The system also supports live capture mode, enabling continuous monitoring of network behaviour and the identification of unusual or potentially malicious activities in real time.

In the packet analysis and classification phase, each captured packet is examined to determine its protocol type, communication pattern, and traffic behaviour. The header and payload are parsed to extract key details such as source and destination addresses, TTL (Time-To-Live), packet size, and checksum values. A set of detection rules is implemented to identify suspicious activities like port scanning, excessive transmission rates, or malformed packets, which could indicate possible security threats. The analysed data is stored in a structured database (SQLite or MySQL) to ensure persistence and organised access. Database indexing and schema optimisation techniques are applied to enhance query performance during live monitoring. The web-based dashboard is developed using the Flask framework for the backend, which handles client requests, manages API communication, and delivers real-time packet information to the frontend. The frontend interface, built with HTML, CSS, JavaScript, and visualisation libraries such as Chart.js or Plotly.js, provides an intuitive and interactive view of captured traffic. Users can explore network activity through dynamic tables, charts, and graphs, while AJAX or WebSocket integration ensures real-time updates without manual page refreshes.

To maintain system security, authentication mechanisms are implemented to restrict access to authorised users only. Sensitive information, including credentials, is encrypted using the AES algorithm. The system also adheres to the principle of least privilege, operating with limited permissions to reduce security risks in case of a breach.

The testing and validation phase includes functional testing of individual modules, as well as performance evaluation based on parameters like packet loss, processing speed, and CPU utilisation. The system's accuracy and stability are compared against established tools such as Wireshark to ensure reliability and efficiency.

Finally, in the deployment and maintenance stage, the application is hosted either on a local or cloud-based server, offering easy web access for multiple users. The system's modular architecture supports future enhancements, such as automated alert mechanisms, advanced filtering options, and integration with intrusion detection systems for expanded analytical capabilities.

### 4.1 ADVANTAGES OF THE PROPOSED METHOD

1. Real-time network monitoring
2. Web-based accessibility
3. User-friendly dashboard interface
4. Protocol-based filtering and analysis
5. Lightweight and efficient performance
6. Secure user authentication system
7. Data export in CSV format
8. Cross-platform compatibility
9. Useful for education and research
10. Easy future extensibility and upgrades

## V. FLOWCHART

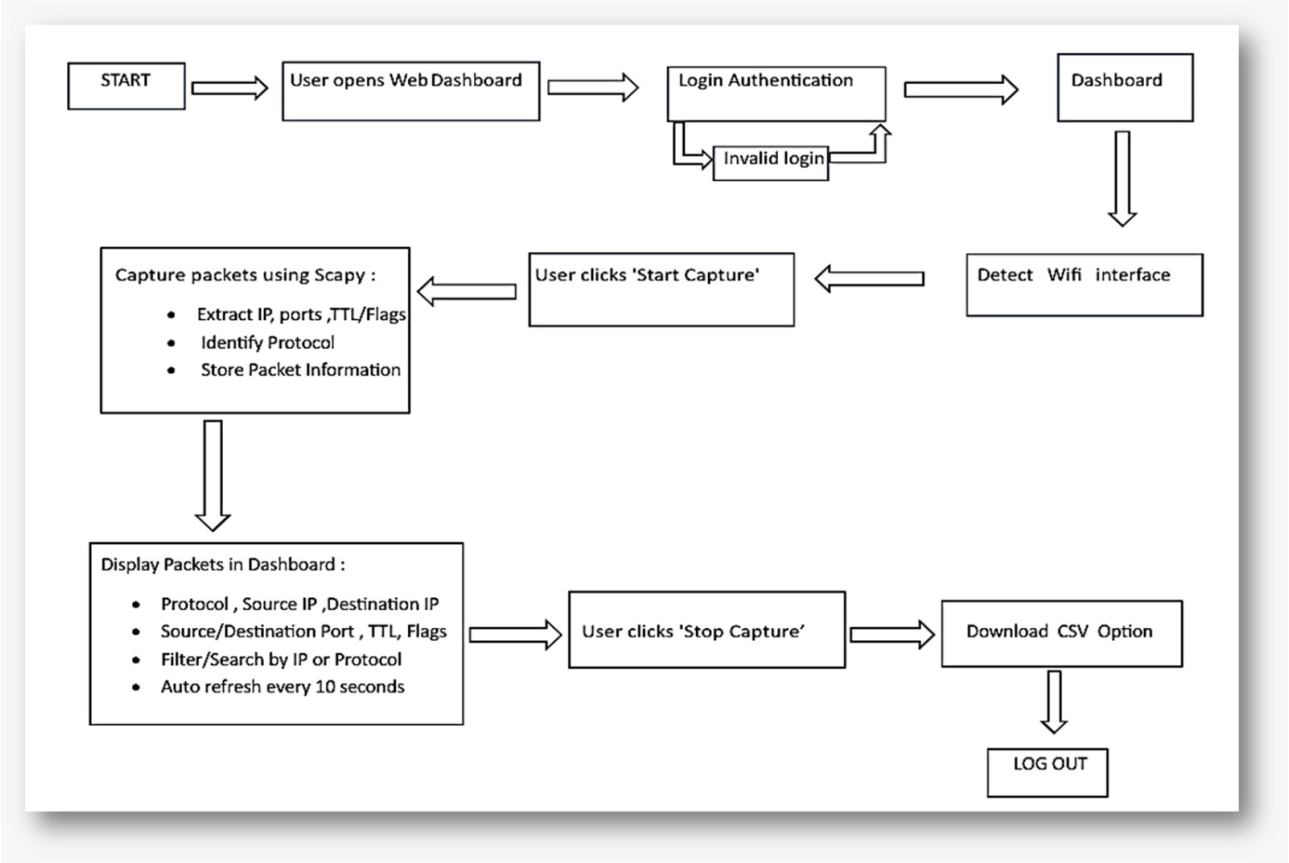



Fig.1

## VI. WORKING MODEL OF PROJECT

The working model of the Web-Based Wi-Fi Packet Sniffer and Analyser follows a systematic and secure workflow, beginning with user interaction through a browser-based dashboard. This dashboard serves as the central control interface, allowing users to initiate packet captures, monitor live network activity, and analyse captured data in real time.

### 6.1UserAuthentication

When the user opens the web application, they are first directed to a login page. The system verifies credentials at the server level, ensuring that only authorised users can access the main dashboard. Invalid credentials prompt an error message, maintaining proper access control and security.

### 6.2InterfaceDetectionandSelection:

After successful login, the system automatically detects available Wi-Fi interfaces (such as wlan0 or wlan1). The user selects the desired interface for monitoring.

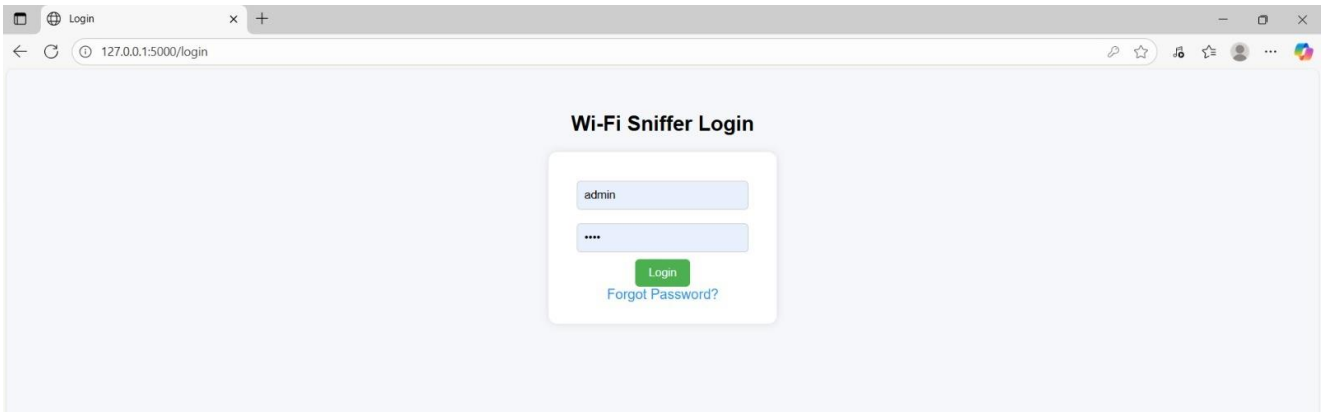



Fig.2

**6.3 Packet Capture Initialisation** Once the user clicks "Start Capture," the backend, powered by the Scapy library in Python, begins capturing live packets. This process runs on a separate thread to keep the web interface responsive during continuous data capture.

**6.4DataExtractionandProcessing:**

The system extracts important packet details such as source and destination IP addresses, port numbers, protocols (TCP, UDP, ICMP, ARP, DNS, HTTP, etc.), TTL values, flags, and packet sizes. The extracted packets are temporarily stored in structured memory buffers or CSV format for efficient handling.

**6.5LiveDataDisplayandVisualization**

The captured packets are displayed dynamically on the web dashboard in a tabular view, showing each packet's details, including protocol, source, destination, and timestamp. Users can filter or search packets by IP address, port, or protocol. The dashboard updates automatically in real time using AJAX or WebSocket, without needing manual refreshes.

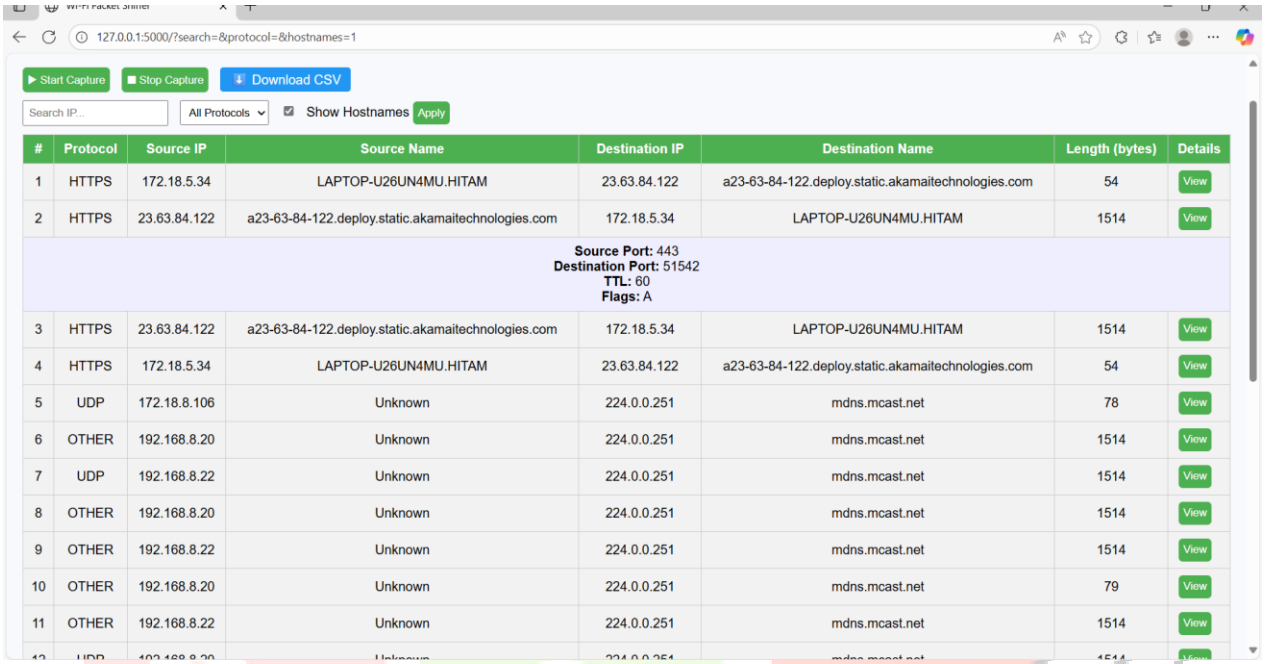


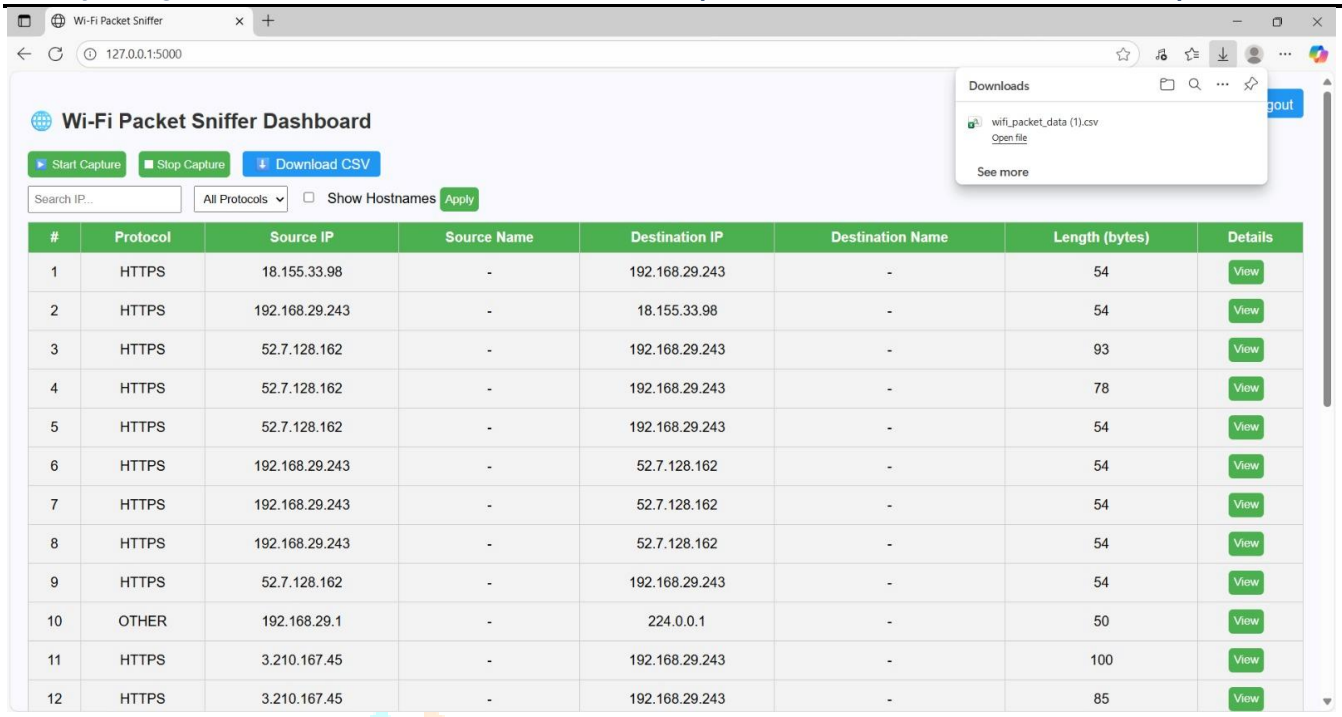

Fig.3

**6.6Stopping Capture and Data Export**

When the user clicks "Stop Capture," the sniffing process terminates gracefully. The user can then download the captured data as a CSV file for offline analysis or documentation.

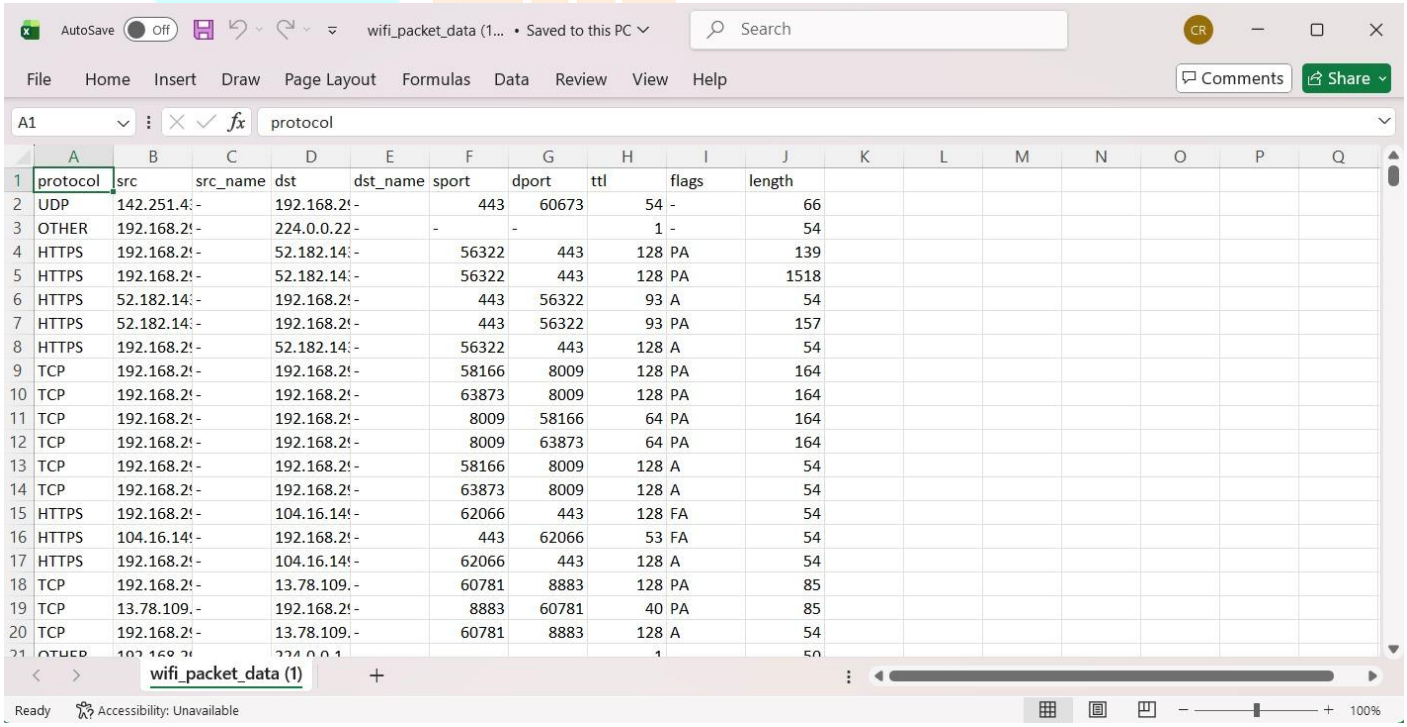
**6.7Logout and Session Termination:**

Finally, the user can log out of the application, which ends the active session and clears temporary or sensitive data to maintain system security and privacy.




| # | Protocol | Source IP | Source Name | Destination IP | Destination Name | Length (bytes) | Details |
|---|---|---|---|---|---|---|---|
| 1 | HTTPS | 18.155.33.98 | - | 192.168.29.243 | - | 54 | View |
| 2 | HTTPS | 192.168.29.243 | - | 18.155.33.98 | - | 54 | View |
| 3 | HTTPS | 52.7.128.162 | - | 192.168.29.243 | - | 93 | View |
| 4 | HTTPS | 52.7.128.162 | - | 192.168.29.243 | - | 78 | View |
| 5 | HTTPS | 52.7.128.162 | - | 192.168.29.243 | - | 54 | View |
| 6 | HTTPS | 192.168.29.243 | - | 52.7.128.162 | - | 54 | View |
| 7 | HTTPS | 192.168.29.243 | - | 52.7.128.162 | - | 54 | View |
| 8 | HTTPS | 192.168.29.243 | - | 52.7.128.162 | - | 54 | View |
| 9 | HTTPS | 52.7.128.162 | - | 192.168.29.243 | - | 54 | View |
| 10 | OTHER | 192.168.29.1 | - | 224.0.0.1 | - | 50 | View |
| 11 | HTTPS | 3.210.167.45 | - | 192.168.29.243 | - | 100 | View |
| 12 | HTTPS | 3.210.167.45 | - | 192.168.29.243 | - | 85 | View |

Fig.4

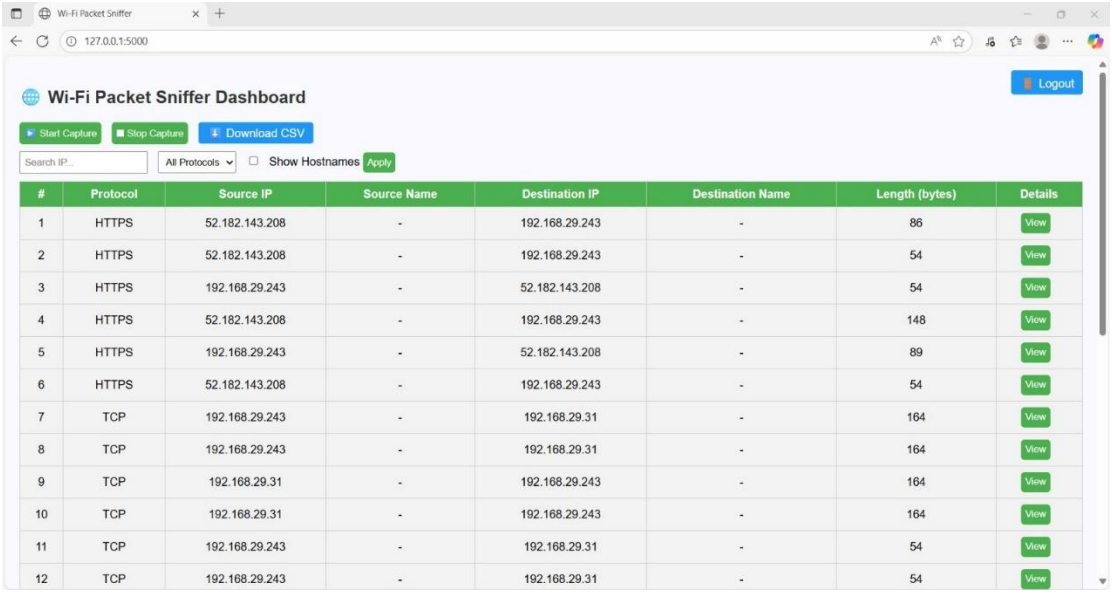



| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | protocol | src | src_name | dst | dst_name | sport | dport | ttl | flags | length |
| 2 | UDP | 142.251.4 | - | 192.168.2 | - | 443 | 60673 | 54 | - | 66 |
| 3 | OTHER | 192.168.2 | - | 224.0.0.22 | - | - | - | 1 | - | 54 |
| 4 | HTTPS | 192.168.2 | - | 52.182.14 | - | 56322 | 443 | 128 | PA | 139 |
| 5 | HTTPS | 192.168.2 | - | 52.182.14 | - | 56322 | 443 | 128 | PA | 1518 |
| 6 | HTTPS | 52.182.14 | - | 192.168.2 | - | 443 | 56322 | 93 | A | 54 |
| 7 | HTTPS | 52.182.14 | - | 192.168.2 | - | 443 | 56322 | 93 | PA | 157 |
| 8 | HTTPS | 192.168.2 | - | 52.182.14 | - | 56322 | 443 | 128 | A | 54 |
| 9 | TCP | 192.168.2 | - | 192.168.2 | - | 58166 | 8009 | 128 | PA | 164 |
| 10 | TCP | 192.168.2 | - | 192.168.2 | - | 63873 | 8009 | 128 | PA | 164 |
| 11 | TCP | 192.168.2 | - | 192.168.2 | - | 8009 | 58166 | 64 | PA | 164 |
| 12 | TCP | 192.168.2 | - | 192.168.2 | - | 8009 | 63873 | 64 | PA | 164 |
| 13 | TCP | 192.168.2 | - | 192.168.2 | - | 58166 | 8009 | 128 | A | 54 |
| 14 | TCP | 192.168.2 | - | 192.168.2 | - | 63873 | 8009 | 128 | A | 54 |
| 15 | HTTPS | 192.168.2 | - | 104.16.14 | - | 62066 | 443 | 128 | FA | 54 |
| 16 | HTTPS | 104.16.14 | - | 192.168.2 | - | 443 | 62066 | 53 | FA | 54 |
| 17 | HTTPS | 192.168.2 | - | 104.16.14 | - | 62066 | 443 | 128 | A | 54 |
| 18 | TCP | 192.168.2 | - | 13.78.109. | - | 60781 | 8883 | 128 | PA | 85 |
| 19 | TCP | 13.78.109. | - | 192.168.2 | - | 8883 | 60781 | 40 | PA | 85 |
| 20 | TCP | 192.168.2 | - | 13.78.109. | - | 60781 | 8883 | 128 | A | 54 |

Fig.5

This structured workflow ensures that the proposed system offers a secure, efficient, and user-friendly environment for real-time monitoring and analysis of wireless network traffic, bridging the gap between traditional packet sniffing tools and modern web-based solutions.

## VII. RESULT &CONCLUSION




| # | Protocol | Source IP | Source Name | Destination IP | Destination Name | Length (bytes) | Details |
|---|---|---|---|---|---|---|---|
| 1 | HTTPS | 52.182.143.208 | - | 192.168.29.243 | - | 86 | View |
| 2 | HTTPS | 52.182.143.208 | - | 192.168.29.243 | - | 54 | View |
| 3 | HTTPS | 192.168.29.243 | - | 52.182.143.208 | - | 54 | View |
| 4 | HTTPS | 52.182.143.208 | - | 192.168.29.243 | - | 148 | View |
| 5 | HTTPS | 192.168.29.243 | - | 52.182.143.208 | - | 89 | View |
| 6 | HTTPS | 52.182.143.208 | - | 192.168.29.243 | - | 54 | View |
| 7 | TCP | 192.168.29.243 | - | 192.168.29.31 | - | 164 | View |
| 8 | TCP | 192.168.29.243 | - | 192.168.29.31 | - | 164 | View |
| 9 | TCP | 192.168.29.31 | - | 192.168.29.243 | - | 164 | View |
| 10 | TCP | 192.168.29.31 | - | 192.168.29.243 | - | 164 | View |
| 11 | TCP | 192.168.29.243 | - | 192.168.29.31 | - | 54 | View |
| 12 | TCP | 192.168.29.243 | - | 192.168.29.31 | - | 54 | View |

Fig.6

The developed Wi-Fi Packet Sniffer Dashboard successfully captures and displays real-time network traffic data through an interactive web interface. As shown in the dashboard, packets transmitted across the Wi-Fi network are efficiently analysed and presented with essential details such as protocol type, source and destination IP addresses, packet length, and additional information accessible via the "View" button. The system allows users to start and stop live capture, filter packets by protocol, search by IP address, and even download data as a CSV file for further analysis. The integration of Flask, Scapy, and AJAX/WebSocket ensures smooth, real-time updates without manual refreshes, providing an intuitive and responsive user experience.

In conclusion, the project demonstrates the effectiveness of combining Python-based packet sniffing with a web-based visualisation interface. It achieves the core objectives of real-time monitoring, data filtering, and secure access while maintaining a lightweight and user-friendly design. Compared to traditional tools like Wireshark, this system offers greater accessibility, ease of use, and remote monitoring capability through a browser-based dashboard. Overall, the Wi-Fi Packet Sniffer Dashboard serves as a practical and educational tool for understanding network behaviour, detecting anomalies, and enhancing cybersecurity awareness in wireless environments.

## VIII. REFERENCES

[1] "Packet Sniffer Cyber Security Tool" – Vishal Injewar et al.
Describes building a packet sniffer tool; useful for implementation reference.
[2] "Network Packet Sniffing and Monitoring" – IJAEM Journal (India)
Explains packet capture, protocol analysis, and monitoring methods.
[3] "A Study of Packet Sniffer Tools" – Comparative Analysis (India)
Compares Wireshark, Tcpdump, and other sniffers for performance and features.
[4] "Intrusion Detection System for Detecting Wireless Attacks" – S.C. Sethuraman et al.
Focuses on detecting wireless network attacks using packet-level data.
[5] "A Survey on Intrusion Detection System for Wireless Network" – Ajita Mishra et al.
Reviews wireless IDS approaches and network security monitoring techniques.
[6] A. K. Yadav and M. R. Singh, "Development of a Python-Based Network Packet Analyser for Real-Time Monitoring," International Journal of Advanced Research in Computer Science (IJARCS), vol. 14, no. 2, pp. 220–226, 2023. P. Jain and
[7] S. N. Rao, "Design and Analysis of Real-Time Web-Based Network Traffic Monitoring Tool Using Flask and Socket.io," International Journal of Engineering Research & Technology (IJERT), vol. 12, no. 5, pp. 312–318, 2024.
[8] T. George and A. Mathew, "A Study on Network Packet Sniffing and Analysis Tools," International Journal of Computer Science Trends and Technology (IJCST), vol. 10, no. 3, pp. 45–51, 2022.