International Journal of Web of Multidisciplinary Studies
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
website: *https://ijwos.com*
Volume 03, Issue 10,October 2026
OPEN ACCESS
E-ISSN : 3049-2424
DOI: https://doi.org/10.71366/ijwos

NETVISOR: A Network Analysis and VPN Detection Tool

Prof. Dattatreya Choudari[1], Manoj Bhadakambi[2], Somesh Kichadi[3], Shilpa Malai[4], Snehal Hoskeri[5]
[1,2,3,4,5] *Department of Artificial Intelligence and Data Science, Angadi Institute of Technology and Management (AITM), Belagavi, india*

Article Info

Article History:
Published: 30 Oct 2026

Publication Issue:
Volume 03, Issue 10
October 2026

Page Number:
539-556

Corresponding Author:
Prof. Dattatreya Choudari

Abstract:
NETVISOR is a network analysis and VPN detection tool built to study network traffic and spot patterns that may point to VPN usage. It relies on packet-level inspection through Wireshark to capture and examine network packets. During the analysis, it looks at key packet characteristics such as source and destination IP addresses, port numbers, communication protocols, general packet information, and encrypted tunnel patterns.
The system pairs this packet analysis with a web-based interface so that users can observe network activity in an easy-to-use environment. The frontend is built with HTML, CSS, and JavaScript, while the Flask framework handles processing on the backend. The backend takes the captured packet information, pulls out the relevant network characteristics, and analyzes them to find suspicious or VPN-associated traffic patterns.
NETVISOR is meant for network administrators, cybersecurity learners, ethical security practitioners, and students who want to understand how networks behave. Instead of making users dig through huge amounts of raw packet data by hand, the system offers a simpler interface for viewing and interpreting traffic. The project shows how packet analysis, network monitoring, backend processing, and web technologies can come together in a practical, security-focused application. The current Phase-1 work covers the problem domain, an analysis of existing systems, the proposed architecture, the requirements, and the methodology. Figures for detection accuracy and performance will be established later, during the implementation and testing phase.
Keywords: Network Analysis, VPN Detection, Network Traffic, Packet Analysis, Flask, Cybersecurity, Network Monitoring, Encrypted Traffic.

1. INTRODUCTION

Computer networks are now a basic part of modern communication and digital services. Organizations, schools and colleges, businesses, and individual users all exchange information over networks every day. As we depend more on network communication, keeping an eye on network traffic and understanding it has become an important part of cybersecurity. Network traffic is made up of packets that devices send to each other, and each packet can carry details such as the source and destination addresses, communication ports, protocols, packet sizes, and timestamps.

Studying these details helps us recognize normal communication patterns and also spot unusual or suspicious behaviour. One of the biggest challenges in network monitoring today is the growing use of encrypted communication and Virtual Private Networks (VPNs). A VPN creates an encrypted tunnel between two endpoints and is commonly used for privacy, secure remote access, and protecting data on untrusted networks. The same encryption, however, makes it hard to inspect the contents of the communication directly. Traditional packet-analysis tools give a lot of technical detail, but they usually expect the user to have solid networking knowledge. Going through large volumes of packet data manually is also very time-consuming.

Copyright © 2026 The Author(s) : This is an open access article under the CC BY license (http://creativecommons.org/licenses/by-nc/4.0/)

NETVISOR is proposed as a network analysis and VPN detection tool that makes this job easier. It captures and analyzes network packets using Wireshark and extracts useful characteristics such as IP addresses, ports, protocols, and patterns linked to encrypted tunnels. The processed information is shown through a web interface built with HTML, CSS, and JavaScript, and a Flask backend takes care of packet processing and the detection logic. The goal is to give users a simple way to watch network activity and get meaningful information about traffic that may be related to a VPN.

2. RELATED WORK

2.1 Review of Existing Systems

Several kinds of existing systems are relevant to NETVISOR:

- **Packet Analysis Tools:** These tools give detailed information about network communication. Wireshark is a well-known example that provides packet capture and inspection features.
- **Network Monitoring Systems:** These systems observe network traffic and report on devices, protocols, connections, and overall network activity.
- **VPN Detection Systems:** These approaches study observable network characteristics to decide whether traffic shows properties that are typical of VPN communication.
- **Security Monitoring Systems:** These platforms can combine network traffic analysis with logging, alerting, and anomaly detection.

Research gaps identified. The review points to several gaps that motivate NETVISOR:

- **Complexity of packet analysis:** Detailed packet information can be hard for beginners to interpret.
- **Need for simplified VPN analysis:** Users may need a dedicated interface for examining traffic that could be related to a VPN.
- **Manual inspection overhead:** Going through large amounts of traffic by hand takes a lot of time.
- **Integration of packet analysis and web presentation:** A simple web interface can make packet information far more accessible.
- **Educational requirement:** Students learning networking and cybersecurity can benefit from a system that shows packet-level analysis in an approachable way.

2.2 Comparative Study

The table below compares NETVISOR with traditional packet-analysis tools and general network-monitoring systems.

Feature	Traditional Packet Analysis	General Network Monitoring	NETVISOR
Packet Capture	Yes	Varies	Yes
Packet-Level Inspection	Yes	Limited / varies	Yes
Protocol Analysis	Yes	Yes	Yes
VPN-Oriented Analysis	Not necessarily simplified	Varies	Main focus
Web Interface	Tool dependent	Usually available	Yes

Feature	Traditional Packet Analysis	General Network Monitoring	NETVISOR
Beginner-Oriented Presentation	Limited	Varies	Intended
Flask Backend	No	Varies	Yes
Educational Use	Possible	Possible	Strong focus

In short, the research gaps listed in Section 2.1 are the main reasons for building NETVISOR: packet analysis is complex for beginners, VPN-related traffic needs a simpler view, manual inspection is slow, a web interface can make packet data easier to understand, and students need a practical way to learn packet-level analysis.

2.3 Summary of Literature Review

The review shows that network packet analysis and traffic monitoring are well-established practices in cybersecurity. At the same time, detailed packet-analysis environments often demand technical knowledge, and VPN-related traffic adds another layer of difficulty because the communication is encrypted. NETVISOR tackles this problem at a project scale by bringing together Wireshark-based packet inspection, feature extraction, VPN-oriented analysis, Flask backend processing, and a web interface.

3. SYSTEM ANALYSIS

3.1 Existing System

Most existing network-analysis approaches rely on packet-analysis tools, network-monitoring platforms, or manual inspection. Wireshark, for example, provides extensive packet information and is very useful for troubleshooting and security analysis. The catch is that users must understand networking concepts and interpret the captured data on their own. VPN traffic makes things even harder, because encryption and tunneling can hide the application-level contents of the communication.

3.2 Limitations of the Existing System

Limitation	Description
Complexity	Packet-level information can be difficult for beginners to understand.
Manual Analysis	Users may have to inspect a very large number of packets by hand.
Technical Knowledge	Using these tools effectively requires good networking knowledge.
VPN Interpretation	Encrypted traffic is hard to interpret.
Information Overload	Packet captures can contain a huge amount of information.
Lack of Simplified Presentation	Raw packet information is usually not presented in a way that suits a specific application.

3.3 Proposed System

NETVISOR is a network-analysis and VPN-detection application designed to make captured network traffic easier to analyze. The system captures packets through Wireshark and extracts their important characteristics. The Flask

backend then processes these characteristics and applies the project's detection logic to find patterns that may indicate VPN usage. Finally, the processed information is presented through a web interface.

Major Components

1. **Packet Capture Module:** Obtains network packets using packet-analysis functionality.
2. **Packet Analysis Module:** Examines the packet information and extracts the relevant fields.
3. **Feature Extraction Module:** Extracts characteristics such as source IP, destination IP, source port, destination port, protocol, and tunnel- or encryption-related patterns.
4. **VPN Detection Module:** Analyzes the extracted characteristics to identify traffic patterns associated with VPN usage.
5. **Flask Backend:** Handles requests and coordinates packet analysis and detection.
6. **Web Interface:** Gives users an accessible way to view the analysis results.

3.4 Methodology

1. **Network and Endpoint Data Collection:** NETVISOR collects information from connected devices and network traffic using endpoint monitoring agents and gateways. This includes details such as devices, IP addresses, ports, protocols, packets, and general traffic information.
2. **Data Processing and Feature Extraction:** The collected network data is cleaned and converted into useful information. Important features such as packet count, byte count, connection duration, source port, and destination port are extracted for further analysis.
3. **Network Activity Monitoring:** The system continuously monitors network activity and connected devices to learn what normal communication looks like. This gives administrators better visibility over the network and helps them notice unusual activity.
4. **Threat Detection:** NETVISOR uses predefined detection rules to identify common security threats. It can detect activities such as port scanning, brute-force attempts, DNS tunneling, data exfiltration, and suspicious beaconing behavior.
5. **Anomaly Detection Using Machine Learning:** The system uses the Isolation Forest algorithm to find unusual network traffic patterns. It studies selected network-flow features and flags activity that differs significantly from normal behavior.
6. **Risk Analysis and Event Correlation:** Detected security events are analyzed and correlated to understand how important they are. Based on the type and severity of the events, the system assigns risk levels so administrators can prioritize threats.
7. **Centralized Dashboard and Visualization:** Processed information, network status, connected devices, traffic activity, and security alerts are all shown on one web dashboard. This lets administrators monitor and understand the overall condition of the network easily.
8. **Alert and Security Monitoring:** When suspicious activity is detected, the system records the security event and shows the relevant details on the dashboard. This helps administrators identify potential threats quickly and respond appropriately.

4. MLOPS INFRASTRUCTURE AND TECHNOLOGY JUSTIFICATION

The proposed MLOps infrastructure for NETVISOR is designed to support the future integration, training, evaluation, deployment, and monitoring of a machine-learning-based VPN traffic detection model. It follows a modular pipeline: network traffic data is turned into features, used to develop a model, deployed as a detection service, and then monitored continuously.

4.1 Data Collection Layer

Network packets captured with Wireshark are the main source of traffic information. From them, relevant packet-level and flow-level features can be extracted and stored to build a labelled dataset containing both VPN and non-VPN traffic. Potential features include:

- Source IP
- Destination IP
- Source port
- Destination port
- Protocol
- Packet size
- Packet rate
- Flow duration
- Inter-arrival time
- Traffic volume
- Other flow-level characteristics

Sensitive information should be handled with care, and packet payloads that are not needed should not be kept.

4.2 Data Processing and Feature Engineering

The collected traffic data has to be prepared before a model can be trained on it. The preprocessing stage can include:

- Removing invalid records
- Handling missing values
- Converting categorical network information into numerical features
- Scaling features where required
- Removing redundant features
- Creating flow-level statistical features
- Splitting the data into training, validation, and testing sets

This stage makes sure the machine-learning model always receives consistent and meaningful input.

4.3 Model Development Layer

The processed dataset can be used to train and compare several machine-learning algorithms. The candidate models are:

- Random Forest
- Support Vector Machine (SVM)
- XGBoost
- Logistic Regression
- Neural Network

The final model should be chosen based on how well it actually performs in experiments, rather than assuming a particular algorithm in advance. Evaluation can use:

- Accuracy
- Precision

- Recall
- F1-score
- Confusion matrix

For VPN detection, precision and recall deserve special attention, because both false VPN alerts and missed VPN traffic can reduce how useful the system is.

4.4 Model Versioning

Each trained model should be versioned along with the following details:

- Training dataset version
- Feature configuration
- Algorithm
- Hyperparameters
- Evaluation metrics
- Training date

Keeping this information makes it possible to reproduce and compare earlier versions of the model.

4.5 Model Serving Layer

Once a model has been validated, the selected model can be exposed through the Flask backend. The workflow would be:

Captured Traffic → Feature Extraction → Trained Model → VPN Prediction → Flask API → Web Interface

The Flask service can receive the extracted traffic features and return a prediction such as:

- VPN detected
- Non-VPN traffic
- Prediction probability or confidence, if the selected model supports it

4.6 Technology Justification

The technologies chosen for NETVISOR were picked for how well they suit network traffic analysis, backend development, web-based presentation, and future machine-learning integration.

Technology	Purpose	Justification
Python	Backend, data processing, and ML	Python has a strong ecosystem for network analysis, data processing, and machine learning, and it works easily with Flask and ML libraries.
Wireshark	Packet capture and inspection	Wireshark gives detailed packet-level visibility and supports the analysis of network protocols and traffic characteristics.
Flask	Backend / API	Flask is lightweight and well suited for connecting the packet-processing and ML components to the web interface.

Technology	Purpose	Justification
HTML	Frontend structure	HTML provides the basic structure of the NETVISOR web interface.
CSS	UI styling	CSS lets the interface present network information in a clear and organized way.
JavaScript	Frontend interaction	JavaScript can be used for dynamic result updates, user interaction, and dashboard functionality.
Scikit-learn	Machine learning	Scikit-learn offers commonly used classification algorithms and evaluation metrics that suit VPN traffic classification.
Pandas	Dataset processing	Pandas is a good fit for cleaning, transforming, and analyzing structured network traffic datasets.
NumPy	Numerical processing	NumPy provides the efficient numerical operations needed for feature processing and model development.
Joblib	Model serialization and storage	A trained model can be saved and loaded for deployment, so it does not need to be retrained each time the application starts.
Git	Version control	Git lets us track source code, configuration, and development changes in an organized way.
Database / File Storage	Traffic and result storage	A suitable storage layer can keep processed traffic information, model metadata, and historical analysis results wherever they are needed.

4.7 Technology Alternatives Considered

Requirement	Selected Technology	Possible Alternative	Reason for Selection
Packet Analysis	Wireshark	tcpdump	Wireshark provides detailed packet inspection and is easier to use for visual analysis and development.
Backend	Flask	FastAPI / Django	Flask has a lightweight architecture that suits the current application.
Programming	Python	Java	Python integrates much better with data science and ML libraries.

Requirement	Selected Technology	Possible Alternative	Reason for Selection
ML	Scikit-learn	TensorFlow / PyTorch	Scikit-learn is appropriate for classical classification models and tabular traffic features.
Frontend	HTML / CSS / JavaScript	React	A conventional web stack is enough for the current project; React can be introduced later if the dashboard becomes more complex.
Version Control	Git	Manual backups	Git tracks development changes systematically.

5. IMPLEMENTATION DETAILS

5.1 Implementation Overview

NETVISOR is implemented as a web-based network analysis and VPN detection system. The implementation brings together network packet capture, packet-level inspection, traffic feature extraction, VPN-oriented analysis, a Flask backend, and a web-based frontend.

The system is designed to receive network traffic information, process the captured packet data, identify the relevant network characteristics, and show the analysis results through an interactive interface.

The major implementation components are:

1. Network Packet Capture
2. Packet Inspection
3. Network Feature Extraction
4. Traffic Analysis
5. VPN Detection
6. Flask Backend
7. Web Interface
8. Result Presentation

The overall implementation follows this flow:

Network Traffic → Packet Capture → Packet Inspection → Feature Extraction → Traffic Analysis → VPN Detection → Flask Backend → Web Interface → Results

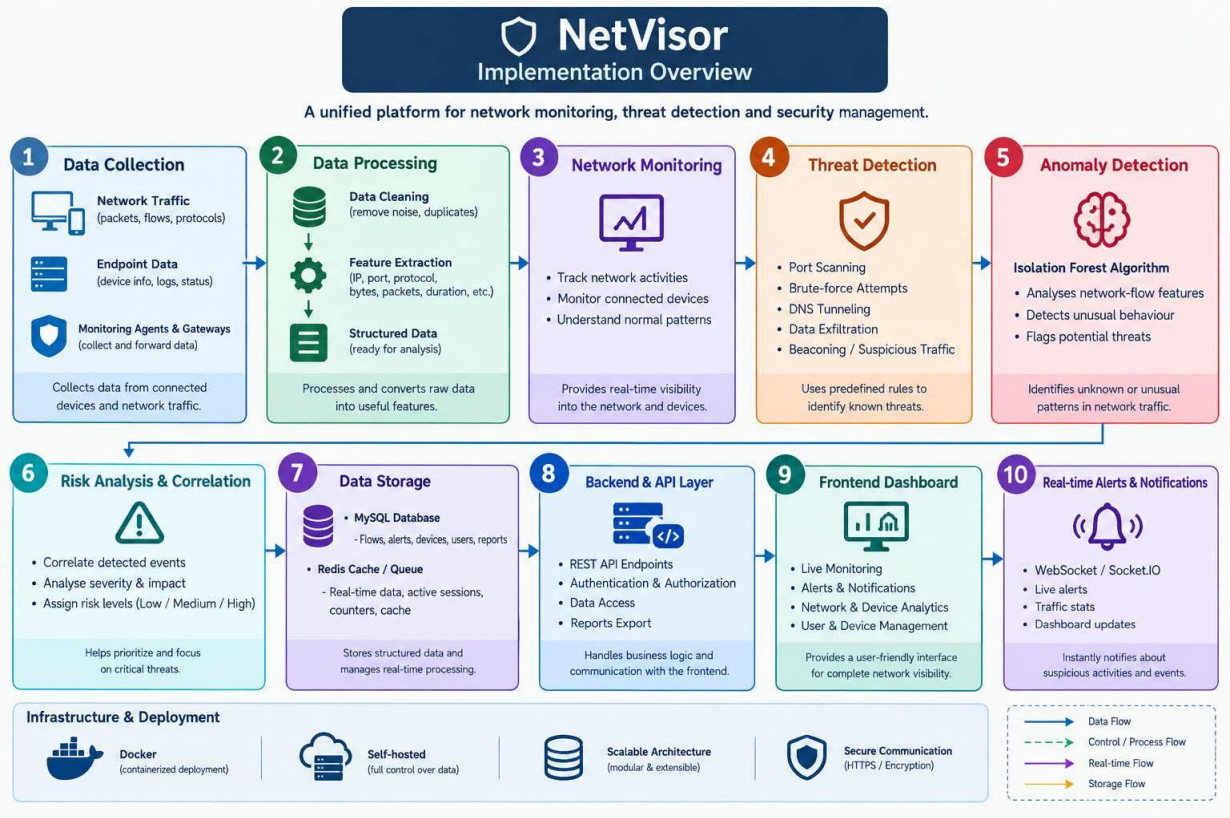


Figure 1: NETVISOR implementation overview

5.2 Network Packet Capture

The first stage of NETVISOR is capturing network packets from the available network interface. Wireshark is used for packet-level inspection and capture. Whenever network communication takes place, packets are generated between the local system and remote devices or services. These captured packets provide the raw information needed for the later stages of analysis.

At this stage the system can observe characteristics such as:

- Source IP address
- Destination IP address
- Source port
- Destination port
- Protocol
- Packet information
- Traffic patterns

The captured traffic becomes the input for the analysis component.

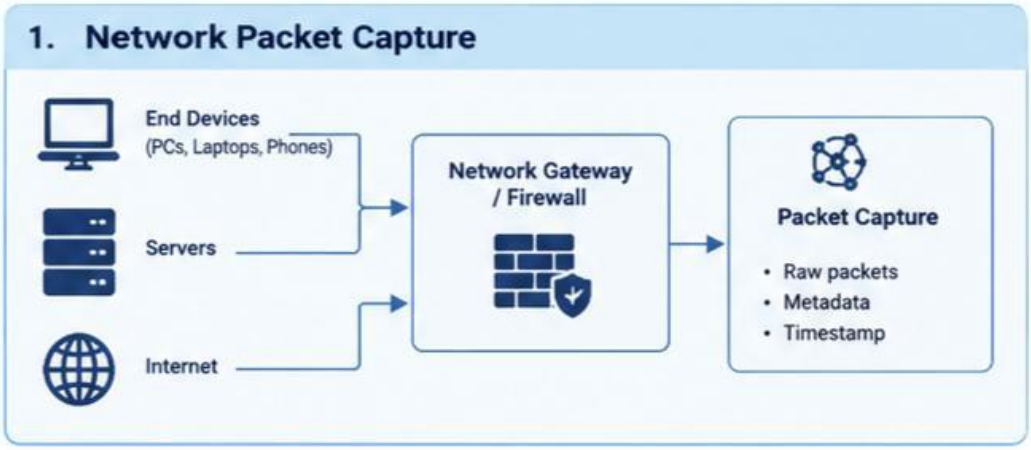


Figure 2: Network packet capture

5.3 Packet Inspection

Once the packets are captured, the system inspects them at the packet level. Every packet holds information about the communication taking place between network endpoints. The implementation focuses on pulling out the useful network information automatically, so the user does not have to inspect every packet manually. In this way, the inspection stage converts raw packet data into structured information that the application can process.

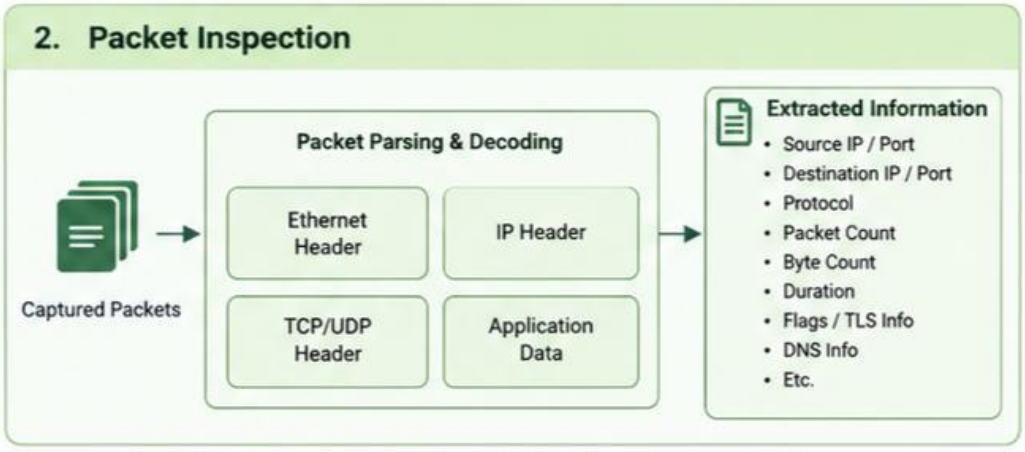


Figure 3: Packet inspection

Packet Inspection Flow:

Captured Packet → Packet Header Inspection → Source/Destination Identification → Protocol Identification → Port Identification → Feature Extraction

5.4 Network Feature Extraction

Feature extraction is a key part of NETVISOR, because VPN-oriented traffic analysis needs meaningful network characteristics to work with. The implementation considers the following features:

- **Source IP Address:** Identifies the endpoint from which the packet originates.
- **Destination IP Address:** Identifies the endpoint that receives the packet.
- **Source Port:** Gives information about the connection the packet comes from.
- **Destination Port:** Gives information about the service or connection that receives the packet.
- **Protocol:** Identifies the type of network communication associated with the packet.
- **Traffic Characteristics:** The system can use the available packet-level characteristics to study communication patterns linked to encrypted or tunnelled traffic.

These extracted characteristics are then passed on to the traffic-analysis component.

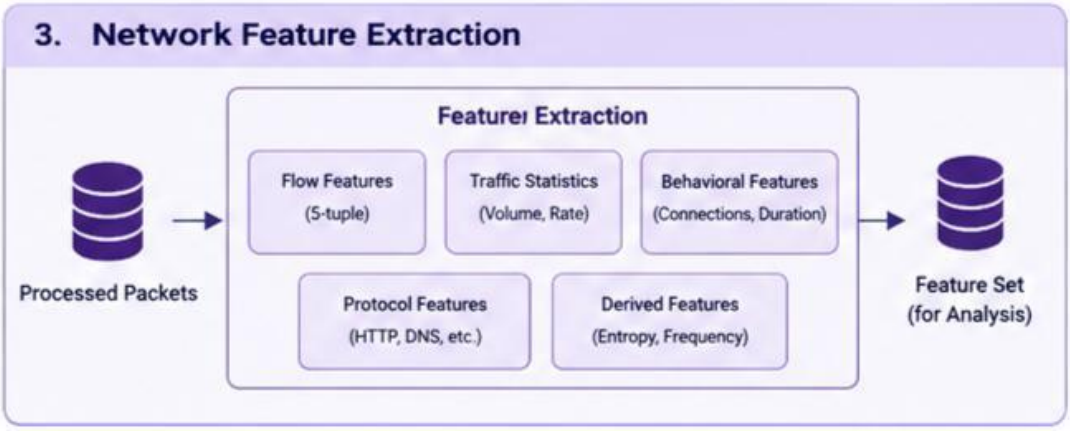


Figure 4: Network feature extraction

5.5 Traffic Analysis

The traffic-analysis component studies the extracted packet information to understand what the observed network communication looks like. The analysis is based only on the information that is available from the captured packets. The system examines:

- Network endpoints
- Ports and protocols
- Packet characteristics
- Communication patterns
- Encrypted traffic characteristics

The purpose of this stage is to turn raw packet information into meaningful network-analysis information.



Figure 5: Traffic analysis

5.6 VPN Detection

VPN detection is one of the major functions of NETVISOR. A VPN sets up an encrypted communication tunnel between a client and VPN infrastructure. Since the contents of encrypted communication cannot normally be read directly, the system concentrates on traffic characteristics that can be observed from outside.

The VPN-detection component analyzes the available packet information and looks for patterns that may indicate VPN or encrypted-tunnel traffic. The detection process can be represented as:

Captured Traffic → Feature Extraction → Traffic Characteristics → VPN-Oriented Analysis → VPN Detection Result

The output can show whether the observed traffic contains characteristics associated with VPN usage. This result should be treated as an indication from the analysis rather than an absolute guarantee, because different VPN technologies can produce different traffic characteristics.

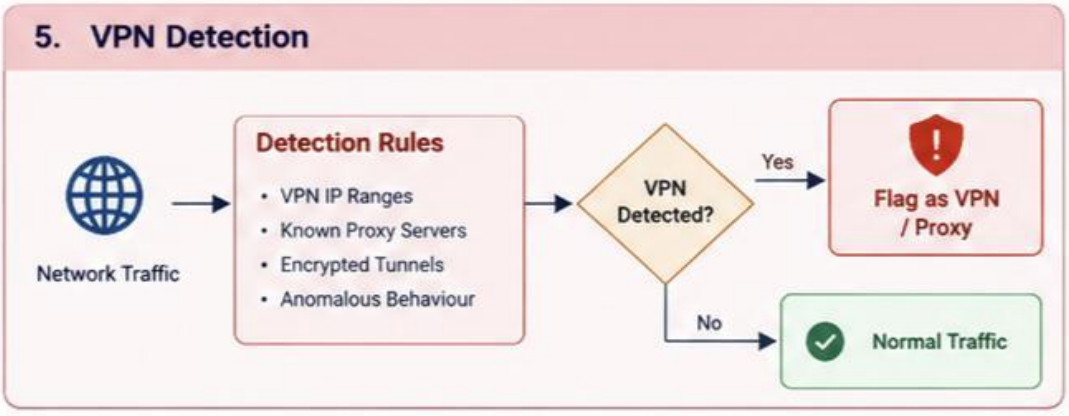


Figure 6: VPN detection

5.7 Flask Backend Implementation

Flask is the backend framework used in NETVISOR. It connects the network-analysis functionality with the frontend. The backend is responsible for:

- Receiving requests from the frontend
- Managing the analysis process
- Processing network information
- Connecting the packet-analysis functionality with the rest of the application
- Returning the analysis results to the frontend

In other words, the backend acts as the central processing layer of the whole system.

Backend Flow:

Frontend Request → Flask Backend → Packet Analysis → VPN Detection → Processed Result → Flask Response → Frontend

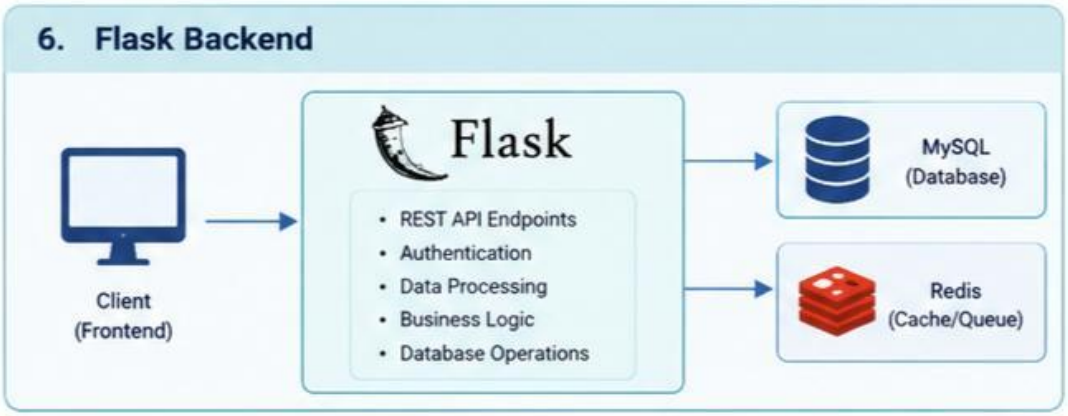


Figure 7: Flask backend

5.8 Web Interface Implementation

The NETVISOR web interface is developed using HTML, CSS, and JavaScript.

HTML provides the basic structure of the application interface. It is used to organize:

- Network information

- Packet information
- Analysis results
- Detection status
- User controls

CSS gives the interface a structured and readable look. It controls:

- Layout
- Spacing
- Text presentation
- Components
- Tables
- Result sections

JavaScript handles the interaction between the user and the application. It can be used to:

- Send requests to the Flask backend
- Receive analysis results
- Update the displayed information
- Refresh the network-analysis data
- Provide interactive controls

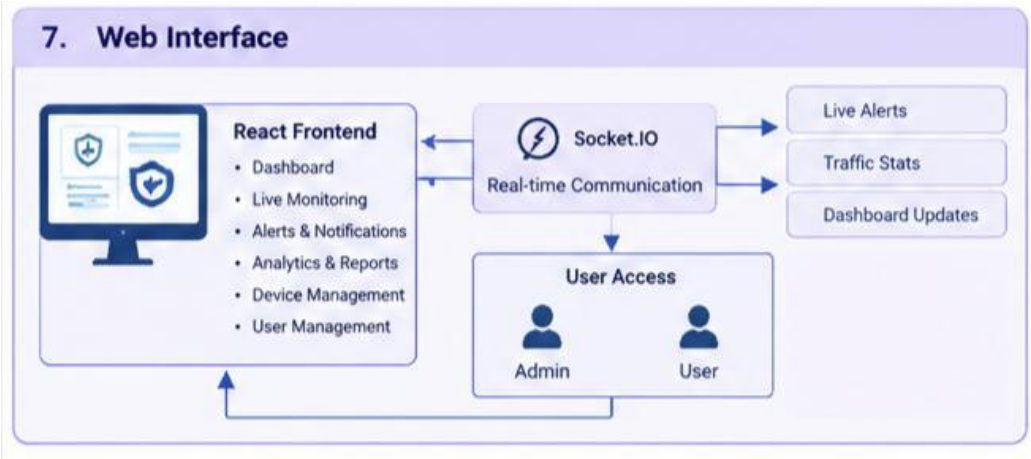


Figure 8: Web interface

6. RESULTS AND DISCUSSION

6.1 Results

1. **Network Visibility:** Provides a centralized view of network traffic and connected devices.
2. **Device Monitoring:** Displays the detected devices and their current network activity.
3. **Traffic Analysis:** Shows network traffic patterns, active flows, and protocol usage.
4. **Threat Detection:** Identifies suspicious activities such as port scans, brute-force attempts, and DNS tunneling.
5. **Anomaly Detection:** Uses Isolation Forest to identify unusual network behavior.
6. **Security Alerts:** Displays detected security events along with their severity levels.
7. **Risk Assessment:** Helps analyze detected events and prioritize potentially serious threats.
8. **Network Visualization:** Provides graphical views of traffic, devices, and network relationships.

9. **Real-Time Monitoring:** Displays live network statistics, alerts, and system activity.
10. **Reports and Logs:** Keeps security events and system logs for monitoring, analysis, and reporting.



Figure 9: NETVISOR dashboards (1 of 3)

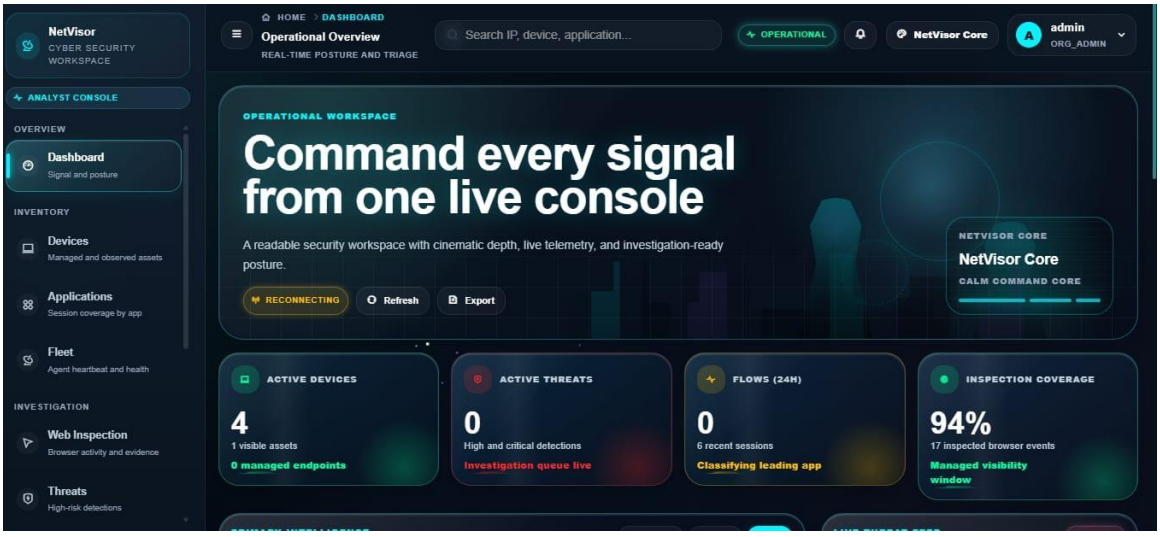


Figure 10: NETVISOR dashboards (2 of 3)

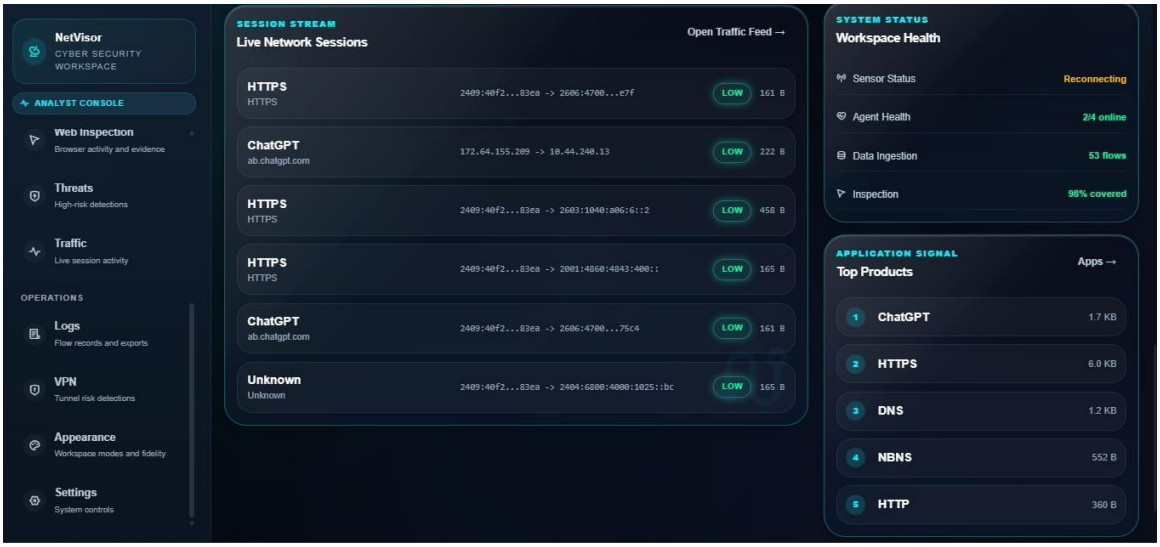


Figure 11: NETVISOR dashboards (3 of 3)

6.2 Discussion

The NETVISOR framework shows that it is feasible to combine packet-level network analysis with VPN-oriented traffic detection inside a web-based application. Wireshark provides the packet-level visibility, Python and Flask provide the processing and application layers, and HTML, CSS, and JavaScript provide an accessible interface for presenting the results.

The current implementation lays the foundation for further development. In particular, the extracted network characteristics can later be used to build a labelled dataset and train machine-learning models for automated VPN traffic classification. Future evaluation should focus on collecting representative VPN and non-VPN traffic, engineering suitable flow-level features, comparing several classification algorithms, and measuring their performance with standard evaluation metrics.

Overall, NETVISOR is a practical starting point for network monitoring and VPN-oriented traffic analysis, and it leaves plenty of room for improvement in automation, machine learning, real-time visualization, and detection accuracy.

7. COMPARISON WITH EXISTING APPROACHES

Existing network-analysis approaches mainly focus on packet capture, protocol inspection, or general network monitoring. Tools such as Wireshark give very detailed packet-level information, but interpreting it generally needs technical knowledge and manual analysis.

NETVISOR builds on this approach by combining packet-level inspection with VPN-oriented traffic analysis and a web-based presentation layer. The aim is not to replace established packet-analysis tools, but to offer a more accessible, integrated analysis environment.

Traditional Packet Analysis

Traditional packet-analysis tools provide detailed information about network communication. They let users inspect packet headers, protocols, addresses, ports, and other packet characteristics. However, the user generally has to interpret the captured information manually.

NETVISOR improvement: NETVISOR uses packet-level information as its foundation and adds an application layer on top. This layer organizes the relevant network characteristics and provides VPN-oriented analysis through a web interface.

Therefore, the two approaches compare as follows:

Traditional approach:

Packet Capture → Manual Inspection → Manual Interpretation

NETVISOR:

Packet Capture → Feature Extraction → Traffic Analysis → VPN Detection → Web Presentation

8. LIMITATIONS

1. **Network Dependency:** The system needs access to network traffic and endpoint data to monitor effectively.
2. **Limited Encrypted Traffic Visibility:** Some encrypted communications may not give enough information for a complete analysis.
3. **False Alerts:** Anomaly-based and rule-based detection may sometimes flag normal activity as suspicious.
4. **Resource Usage:** Continuous monitoring and data processing can use a lot of system resources, especially in large networks.
5. **Scalability:** Very large networks with high traffic volumes may need additional computing resources.
6. **Detection Limitations:** New or highly sophisticated attack patterns may not always be caught by the existing rules and models.

7. **Agent Dependency:** Managed endpoint monitoring depends on the NETVISOR agent being installed and running properly.
8. **Configuration Requirement:** The system needs proper configuration of agents, gateways, the database, and the monitoring services.
9. **Real-Time Processing Limitations:** Very high traffic volumes may cause delays in processing and displaying results.
10. **Model Dependency:** The performance of Isolation Forest depends on the quality and characteristics of the network-flow data given to it.

9. CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

NETVISOR, a network analysis and VPN detection tool, offers a practical way to analyze network traffic and identify characteristics that may point to VPN usage. The system uses Wireshark for packet-level inspection and capture, and it concentrates on extracting important network characteristics such as source and destination IP addresses, ports, protocols, and traffic patterns associated with encrypted tunnels.

A Flask-based backend carries out the analysis, while HTML, CSS, and JavaScript make up the web-based interface that presents the results. The project shows how packet-level network information can be organized and analyzed through one integrated application, so users no longer have to interpret large amounts of packet data by hand.

NETVISOR can be useful for cybersecurity learners and network administrators who want to understand network communication, packet characteristics, and VPN-oriented traffic analysis. That said, VPN detection is still a challenging problem, since encryption and differences between VPN technologies make traffic classification difficult. For this reason, the current system should be seen as a network-analysis and VPN-detection support tool, not as a guaranteed way of identifying every VPN connection.

The architecture also provides a base for further work, especially machine-learning-based traffic classification and more advanced network monitoring capabilities.

9.2 Future Scope

NETVISOR can be improved in several directions to strengthen its detection ability, usability, scalability, and automation.

Machine-Learning-Based VPN Detection

A major future enhancement is adding machine-learning algorithms for automated VPN traffic classification. A labelled dataset of VPN and non-VPN traffic can be created from the extracted network features, and different models can then be trained and compared, such as:

- Random Forest
- Support Vector Machine
- XGBoost
- Neural Networks

The best-performing model can be integrated with the Flask backend for real-time VPN classification.

Real-Time Monitoring Dashboard

The system can be extended with a real-time dashboard that displays:

- Live packet count
- Network traffic volume
- Protocol distribution
- Active connections

- Source and destination information
- VPN detection status
- Traffic trends

This would make NETVISOR much more useful for continuous network monitoring.

Advanced Feature Engineering

Future versions can extract additional flow-level features such as:

- Packet size
- Packet frequency
- Flow duration
- Inter-arrival time
- Upload and download volume
- Number of packets per flow

These features can help a future machine-learning model do a better job of telling VPN traffic apart from normal encrypted traffic.

Support for Multiple VPN Technologies

The detection component can be expanded to analyze traffic from different VPN protocols and configurations. This would make NETVISOR applicable in a wider range of network environments.

Alert Mechanism

NETVISOR can also be given an alert mechanism that notifies the user when potentially VPN-related or unusual traffic is detected. Possible alert methods include:

- Dashboard notifications
- System notifications
- Email alerts

Historical Traffic Analysis

Future versions can store processed network-flow information securely so that users can study past network behaviour. This could enable:

- Daily traffic reports
- VPN usage trends
- Protocol statistics
- Historical comparisons
- Anomaly identification

All of this would make complex network information easier to interpret.

ACKNOWLEDGEMENTS

We sincerely thank the management of Angadi Institute of Technology & Management, Belagavi, for providing the facilities and support we needed to complete our project. We are grateful to Prof. Sagar Birje, Head of the Department of Artificial Intelligence and Data Science, for his valuable support and encouragement.

We are especially thankful to our project guide, Prof. Dattatreya Choudhari, for his continuous guidance, technical suggestions, and encouragement throughout the development of NETVISOR: Network Analysis and VPN Detection Tool.

References

[1] C. M. Bishop, "Pattern Recognition and Machine Learning," Springer, 2006.

[2] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, Vol. 12, pp. 2825–2830, 2011.

[3] W. Richard Stevens and K. Fall, "TCP/IP Illustrated, Volume 1: The Protocols," 2nd Edition, Addison Wesley, 2011.

[4] W. Stallings, "Network Security Essentials: Applications and Standards," 6th Edition, Pearson, 2017.

[5] W. Sun, Y. Zhang, J. Li, C. Sun, and S. Zhang, "A Deep Learning-Based Encrypted VPN Traffic Classification Method Using Packet Block Image," Electronics, Vol. 12, No. 1, 2023.

[6] Z. Diao et al., "EC-GCN: An Encrypted Traffic Classification Framework Based on Multi-Scale Graph Convolution Networks," Computer Networks, Vol. 224, 2023.

[7] Y. Xu, J. Cao, K. Song, Q. Xiang, and G. Cheng, "Fast Traffic: A Lightweight Method for Encrypted Traffic Fast Classification," Computer Networks, 2023.

[8] Y. Liu, X. Wang, B. Qu, and F. Zhao, "ATVITSC: A Novel Encrypted Traffic Classification Method Based on Deep Learning," IEEE Transactions on Information Forensics and Security, Vol. 19, pp. 9374–9389, 2024.

[9] J.-T. Park, C.-Y. Shin, U.-J. Baek, and M.-S. Kim, "Fast and Accurate Multi-Task Learning for Encrypted Network Traffic Classification," Applied Sciences, Vol. 14, No. 7, 2024.

[10] Y. S. Razooqi and A. Pekar, "Binary VPN Traffic Detection Using Wavelet Features and Machine Learning," 2025.