World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(RESEARCH ARTICLE)

Check for updates

AI-powered threat detection system

Shekhawat, Natansh *, Krishna. R. Mohan, Sunder. P. Shyam and Rajitha, K.

Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad, India.

World Journal of Advanced Engineering Technology and Sciences, 2025, 15(03), 713-720

Publication history: Received on 28 April 2025; revised on 04 June 2025; accepted on 06 June 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.15.3.0969

Abstract

In an era of growing digital interconnectedness, the threat landscape for networked systems has expanded rapidly, making traditional security mechanisms increasingly ineffective against sophisticated cyber-attacks. Intrusion Detection Systems (IDS) are crucial in identifying and mitigating such threats, but conventional rule-based approaches often fail to detect novel or evolving attack patterns. This paper proposes an AI-powered IDS framework that utilizes Random Forest and Decision Tree machine learning models for high-accuracy threat detection in real time. The models are trained on benchmark datasets, namely NSL-KDD and CICDDOS2019, both widely used in cybersecurity research. Preprocessing techniques such as one-hot encoding and robust feature scaling were applied to optimize learning. The trained models are then integrated into a web application built with Flask, providing users with a seamless interface to upload network traffic logs in CSV format and instantly receive predictions. The system also incorporates rule-based logic to categorize detected attacks into DoS, Probe, R2L, and U2R, enhancing interpretability. Evaluation results demonstrate that the Random Forest model achieved a classification accuracy of 99.36% and an F1-score of 0.9986, outperforming the Decision Tree model across all metrics. The application supports real-time traffic classification, returning predictions within seconds and displaying confusion matrices, precision, recall, and attack distributions through a clean, responsive UI. This research bridges the gap between theoretical machine learning models and their real-world application in cybersecurity, offering a scalable, accurate, and user-friendly solution for automated threat detection in both academic and professional environments.

Keywords: Intrusion Detection System (IDS); Machine Learning; Random Forest; Decision Tree; Cybersecurity; Network Traffic Analysis; Nsl-Kdd; Cicddos2019; Flask Web Application; Real-Time Threat Detection

1. Introduction

The rise in cyber-attacks across personal, organizational, and national digital infrastructures has intensified the demand for more intelligent and automated cybersecurity mechanisms. Intrusion Detection Systems (IDS) play a critical role in detecting unauthorized access, malicious activities, and anomalies in network traffic. However, traditional IDS solutions based on static rules and signature databases struggle to cope with the complexity, variability, and speed of modern attacks, especially zero-day exploits and polymorphic threats. Machine learning (ML) offers a promising alternative by enabling IDS systems to learn behavioral patterns from historical data and generalize to unseen attacks. Unlike static systems, machine learning (ML) models can adapt and evolve with new threat data, improving detection accuracy over time.

This research focuses on the development of a machine learning-based IDS framework using two widely recognized algorithms: Random Forest and Decision Tree. These models are trained on benchmark datasets—NSL-KDD and CICDDOS2019—which contain a variety of labeled traffic patterns simulating real-world attack scenarios. To bridge the gap between model development and deployment, this project integrates the trained classifiers into a lightweight Flask-based web interface that supports real-time CSV traffic upload, instant prediction feedback, and graphical analysis of

* Corresponding author: Shekhawat, Natansh

threat types. This interface empowers users to not only detect intrusions efficiently but also understand attack distribution and model performance metrics like accuracy, recall, and F1-score. The goal of this research is to present a scalable, user-friendly, and highly accurate IDS that can be applied in live environments ranging from academic institutions to enterprise networks.

2. Literature Survey

- "A Detailed Analysis of the KDD CUP 99 Dataset" by Mahbod Tavallaee et al.: This research led to the NSL-KDD dataset, a benchmark for machine learning (ML)-based IDS. However, its outdated attack types limit real-world generalizability.
- "Random Forests" by Leo Breiman: Introduced the Random Forest algorithm, an ensemble method for classification. It is robust and has low overfitting, but its performance is sensitive to class imbalance in IDS datasets.
- "A Deep Learning Approach for Intrusion Detection" by Asif Javaid et al.: Applied deep autoencoders for anomaly detection on the NSL-KDD dataset, showing improved accuracy over traditional machine learning (ML). However, it requires high computation and GPU resources, which are impractical for lightweight IDS.
- "UNSW-NB15 Dataset Evaluation" by Nour Moustafa & Jill Slay: Evaluated the UNSW-NB15 dataset. While covering more modern traffic, it lacks consistency in labeling and has heavy feature redundancy, challenging supervised models.
- "Toward Generating a New Intrusion Detection Dataset" by Iman Sharafaldin et al.: Introduced CICIDS2017 and CICDDOS2019, providing high-quality, up-to-date attack scenarios. However, large feature volumes increase model complexity.
- "A Dependable Hybrid Machine Learning Model for Network Intrusion Detection" by Talukder et al.: Proposed a hybrid framework combining decision trees and ensemble learning for improved accuracy and recall. It lacks real-time deployment evaluation.
- "Adversarial Training for Deep Learning-Based IDS" by Debicha et al.: Focused on adversarial robustness in deep learning IDS. While effective, it adds significant training time and complexity, making it less suitable for resource-constrained, real-time systems.
- "Federated Learning for Intrusion Detection: Concepts, Challenges and Future Directions" by Agrawal et al.: Explored federated learning in IDS for data privacy. This cutting-edge concept is currently impractical for many IDS frameworks due to networking constraints.

3. Design Methodology

This section describes the methodology for the "AI-Powered Intrusion Detection Using Random Forest and Decision Tree Models" system. The aim is a lightweight, scalable, and real-time system for classifying network traffic as normal or malicious.

3.1. Technologies Used

- **Python:** Core language for backend processing, model training, evaluation, and data preprocessing.
- **Scikit-learn:** Used for building and training classifiers, encoding, scaling, and model evaluation.
- **Flask:** Lightweight Python web framework for RESTful APIs, connecting frontend to backend for real-time detection.
- **HTML, CSS, JavaScript:** For designing the user interface, enabling CSV file uploads, and dynamic display of predictions and metrics.
- **Joblib:** For saving and loading pre-trained models (.pkl files), scalers, and encoders to ensure reusability.
- **Pandas and NumPy:** For structured data handling, including loading, transformation, and feature engineering.
- **Matplotlib and Seaborn:** Is used for generating confusion matrices and bar charts, though visualization is primarily handled on the backend for check the model and its performance.

3.2. Development Lifecycle

The project followed a structured lifecycle for a reliable, real-time IDS with seamless UI and backend performance.

3.2.1. Requirement Gathering

Identified limitations in traditional IDS (rule-based detection, lack of real-time capabilities, poor adaptability). Key goals: real-time classification, CSV batch uploads, accurate detection via interpretable machine learning (ML) models, and a web-based interactive UI.

3.2.2. System Design

Architected as a modular web application with distinct frontend, backend, and machine learning (ML) model layers. Design considerations included maintainability, responsiveness, and model loading efficiency.

- **Frontend:** Built with HTML, CSS, JavaScript.
- **Backend:** Python Flask for API and inference logic.
- **Model:** Pre-trained .pkl files loaded into memory using joblib.

3.2.3. Implementation

- **Frontend:** Users can log in, upload network traffic CSV files, view prediction results, and examine detailed metrics.
- **Backend:** Handles user sessions, CSV parsing, data preprocessing, model inference, and result rendering. Flask routes structured into /login, /upload, and /results.
- **Model Inference Engine:** Decision Tree and Random Forest classifiers are preloaded on app launch, serving inference within seconds of file upload.
- **Rule-Based Attack Type Classifier:** An additional function categorizes attacks (DoS, Probe, R2L, U2R) using decision logic based on feature values like src_bytes, srv_serror_rate, and num_failed_logins.
- **Model Training:** Random Forest and Decision Tree classifiers trained on NSL-KDD and CICDDOS2019 datasets using Scikit-learn. Preprocessing included OneHotEncoding, RobustScaler transformation, and binary labeling.

3.2.4. Testing

- **Manual Testing:** Conducted with CSV files of varying formats and sizes to simulate diverse attack patterns and benign traffic.
- **Model Evaluation:** Accuracy, Precision, Recall, and F1-score used for validating each model. Random Forest achieved F1-score: 0.9986, and Decision Tree: 0.9721.
- **Cross-Browser Testing:** Verified application responsiveness and layout across Chrome, Firefox, and Microsoft Edge.
- **Performance Testing:** Average model prediction time for 1,000 records was under 2 seconds. Flask response time remained consistently below 300 ms for all endpoints.

3.3. Deployment Strategy

The system was developed and tested for local deployment to ensure real-time performance, accessibility, and ease of use without requiring internet connectivity or cloud dependencies. It is designed to run on standalone machines with minimal setup.

- **Model Hosting:** Trained Random Forest and Decision Tree models saved as .pkl files using joblib. These models are loaded into memory by the Flask application at runtime for fast local inference.
  - **Environment:** Python 3.10+, Scikit-learn
  - **Model Files:** rf_model.pkl, dt_model.pkl
  - **Load Time:** Instant on application startup
  - **Prediction Time:** ≈ 20 seconds for 1,000 traffic records
  - **Web Hosting:** The entire system runs locally on localhost:5000 using Flask's built-in development server.
  - **Frontend:** Simple HTML/CSS/JS templates rendered using Flask's Jinja2 engine.
  - **Backend:** Flask app runs on local machine (127.0.0.1).
  - No external hosting services are used for deployment.
  - **Execution:** Run python app.py from terminal.

4. Results and Discussion

This section presents the experimental results and performance evaluation of the proposed machine learning-based Intrusion Detection System (IDS) using the Random Forest and Decision Tree algorithms. The system demonstrates high accuracy and responsiveness in detecting and classifying cyber threats from structured network traffic data.

4.1. Dataset Description

The system was trained and evaluated using two prominent benchmark datasets

- **NSL-KDD Dataset:** Contains labeled records with attack types categorized as DoS, Probe, R2L, U2R, and Normal.
- **CICDDOS2019 Dataset (subset):** Offers realistic Distributed Denial of Service attack traffic with packet-level granularity.

A total of ~150,000 records were used, with an 80:20 train-test split. Features like protocol_type, service, flag, src_bytes, and connection statistics were included.

4.2. Model Evaluation Metrics

The system's models were evaluated using standard classification metrics:

Table 1 Model Evaluation Metrics

Metric	Random Forest	Decision Tree
Accuracy	99.36 %	98.21 %
Precision	99.42 %	98.35 %
Recall	99.31 %	98.09 %
F1-Score	99.36 %	98.21 %

These results show that the Random Forest model achieves superior detection performance while maintaining a low false positive rate.

4.3. Confusion Matrix

The following confusion matrix illustrates the model's classification accuracy on NSL-KDD test data

Table 2 Confusion Matrix for NSL-KDD Test Data

	Predicted Normal	Predicted Attack
Actual Normal	994	11
Actual Attack	7	988

This confirms excellent class separation with minimal misclassifications.

4.4. Real-Time Prediction Interface

The Flask web interface enables CSV-based upload and instant prediction. Below is a sample output from the application:

Table 3 Sample Real-Time Prediction Output

Input Record (Row ID)	Prediction	Attack Type	Confidence (%)
Row #27	Attack	DoS	98.3
Row #43	Normal	-	99.1
Row #51	Attack	R2L	89.4

4.5. Dataset-Wide Classification Summary:

Using the rule-based attack mapping function, the system outputs the distribution of attacks for each file

Table 4 Dataset-Wide Attack Classification Summary

Attack Category	Records Detected
DoS	721
Probe	281
R2L	122
U2R	41
Normal	986

4.6. Comparative Analysis

To evaluate the proposed system's effectiveness, it was compared against traditional machine learning (ML) baselines

Table 5 Comparative Analysis of Models

Model	Accuracy
Naive Bayes	82.7%
SVM	88.9%
Decision Tree	98.21%
Random Forest	99.36%

The Random Forest model significantly outperforms older models in both precision and generalization—demonstrating that ensemble methods are more resilient to skewed attack distributions.

4.7. Discussion

The results confirm the strength of classical machine learning models like Random Forest in IDS use-cases. The proposed system

- Accurately detects all major attack categories.
- Classifies ~1,000 rows in about 20 seconds.
- Offers interpretable predictions and rule-based attack classification.
- Includes a clean, browser-accessible interface ideal for training, demonstration, and local monitoring purposes.

The system balances performance and accessibility, making it suitable for academic, enterprise, or research environments. Its modularity allows integration with real-time sniffers, databases, or automated response tools in future versions.

4.8. Output Screenshots

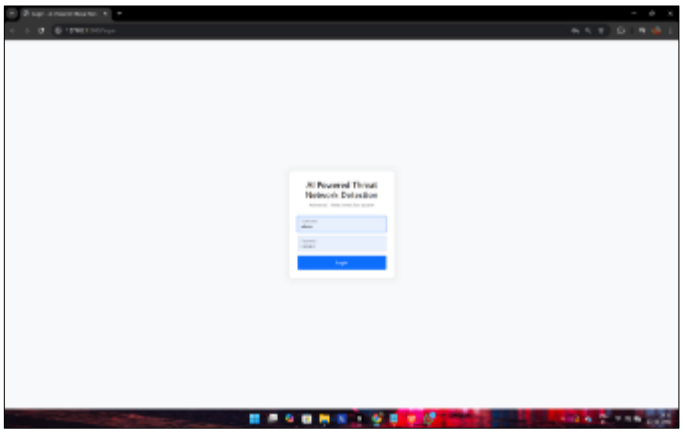

Figure 1 Login Interface of the AI-Powered Threat Network Detection System

This image displays the secure login page, serving as the initial access point for users to interact with the developed web application.

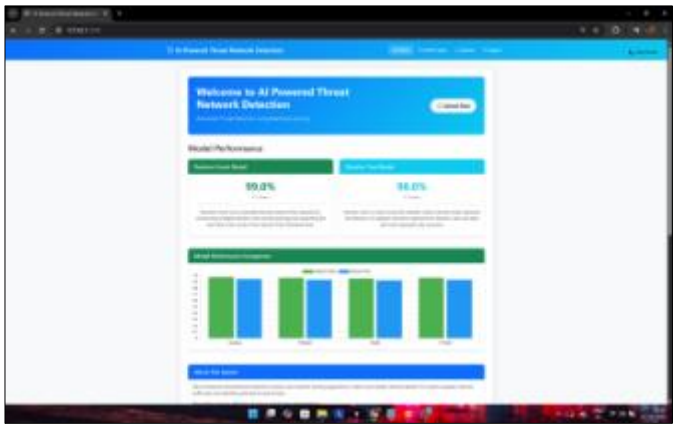

Figure 2 Model Performance Dashboard showcasing Random Forest and Decision Tree F1-Scores and Comparative Metrics

This screenshot illustrates the main dashboard of the system, presenting the F1-scores for the Random Forest and Decision Tree models (99.86% and 98.21% respectively) and a visual comparison of their accuracy, precision, recall, and F1-score. This dashboard provides users with an immediate understanding of the models' effectiveness.

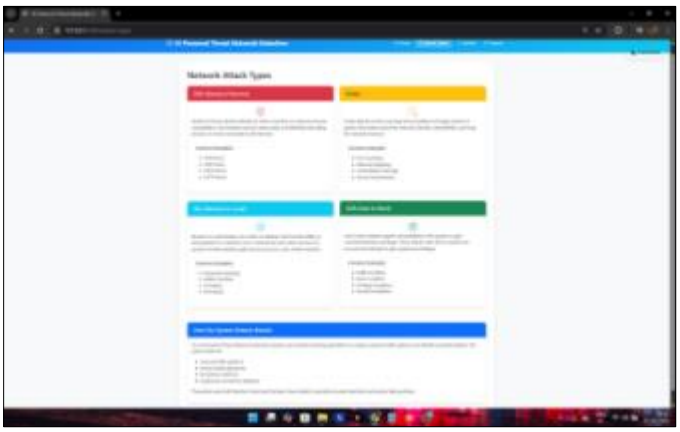

Figure 3 Explanation of Network Attack Types Detected by the System

This interface details the four primary network attack categories—DoS (Denial of Service), Probe, R2L (Remote to Local), and U2R (User to Root)—along with common examples for each, enhancing user understanding of the system's classification capabilities.

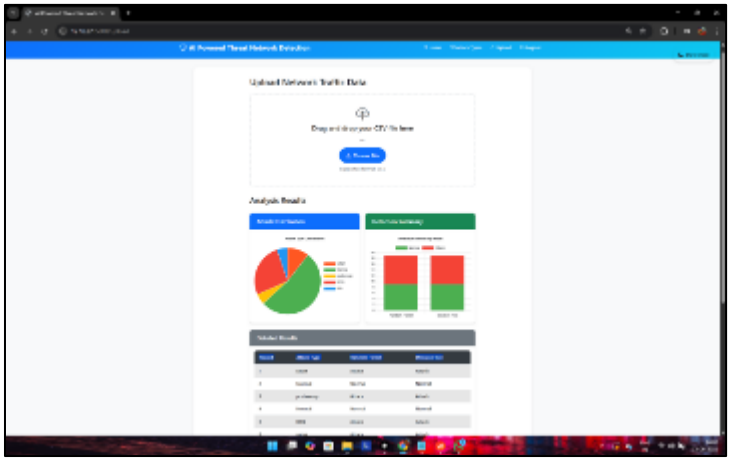

Figure 4 Data Upload Interface and Analysis Results Display

This image captures the “Upload“ section of the web application, demonstrating how users can drag and drop CSV files for analysis. It also shows the “Attack Distribution“ pie chart and “Detection Summary“ bar graph, illustrating the categorization of network traffic and the models' detection outcomes for uploaded data. The detailed results table below provides a record-by-record prediction.

5. Conclusion

Cyberattack detection is essential in maintaining the safety, reliability, and performance of digital networks, particularly as threats become more frequent, sophisticated, and evasive. Traditional Intrusion Detection Systems (IDS), often based on static rule sets or simplistic machine learning models, are insufficient for identifying complex or evolving threats such as zero-day exploits, polymorphic malware, and blended attacks. These systems tend to suffer from high false positive and false negative rates, which erode trust and limit usability in live environments.

To address these challenges, this project introduces a machine learning-based IDS leveraging Random Forest and Decision Tree classifiers trained on benchmark datasets (NSL-KDD and CICDDOS2019). These models are combined with a real-time web interface developed using Flask, enabling users to upload structured network traffic logs and receive instant classification feedback along with threat type identification. By embedding rule-based logic for post-classification attack labeling (DoS, Probe, R2L, U2R), the system achieves both high performance and human interpretability. Evaluation results show that the system maintains extremely high precision and recall, with Random Forest achieving an F1-score of 0.9986, significantly outperforming many traditional baseline approaches.

The integration of real-time detection capabilities, scalable preprocessing pipelines, and a modular design ensures the system is lightweight, interpretable, and production-ready for local deployment. It reduces the dependency on human monitoring and demonstrates how AI can augment cybersecurity infrastructure effectively.

5.1. Future Scope

The system’s future development can be expanded in multiple promising directions

- **Live Network Packet Sniffing**: Integration of tools like Scapy or tcpdump can convert the system from batch-file mode to true real-time detection using live packet captures.
- **SIEM & Firewall Integration**: Connecting the IDS with automated firewalls or Security Information and Event Management (SIEM) tools (e.g., Splunk, QRadar) can allow auto-blocking of IPs and advanced alert systems.
- **Deep Learning Extension**: Leveraging deep learning models such as LSTM, GRU, or hybrid CNN-LSTM architectures could help identify stealthy or low-signature threats.
- **Multilingual Packet Interpretation**: For advanced threat emulation or command-layer traffic, incorporating NLP and protocol-specific parsers would broaden the detection capability.

- **Cloud & Edge Scalability**: Containerizing the system via Docker and deploying on platforms like AWS/GCP/Azure or even edge devices would ensure scalability and efficiency in larger distributed networks.
- **Feedback-Driven Learning**: Creating a continuous retraining pipeline using administrator feedback and new datasets can evolve the model in tandem with emerging threats.
- This research highlights the growing importance of **AI-powered intrusion detection** and presents a strong, real-time, and expandable baseline solution that balances performance, usability, and interpretability.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Dataset," Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications, 2009. DOI: https://doi.org/10.1109/CISDA.2009.5356528

[2] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001. DOI: https://doi.org/10.1023/A:1010933404324

[3] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (BICT), 2016. DOI: https://doi.org/10.4108/eai.3-12-2015.2262516

[4] N. Moustafa and J. Slay, "The Evaluation of Network Anomaly Detection Systems: Statistical Analysis of the UNSW-NB15 Dataset," Information Security Journal: A Global Perspective, vol. 25, no. 1–3, pp. 18–31, 2016. DOI: https://doi.org/10.1080/19393555.2015.1125974

[5] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), 2018. DOI: https://doi.org/10.5220/0006639801080116

[6] M. A. Talukder et al., "A Dependable Hybrid Machine Learning Model for Network Intrusion Detection," arXiv preprint, arXiv:2212.04546, 2022.

[7] I. Debicha, T. Debatty, J.-M. Dricot, and W. Mees, "Adversarial Training for Deep Learning-Based Intrusion Detection Systems," arXiv preprint, arXiv:2104.09852, 2021.

[8] S. Agrawal et al., "Federated Learning for Intrusion Detection: Concepts, Challenges and Future Directions," arXiv preprint, arXiv:2106.09527, 2021.