ISSN: 2349-6002

AI-Based Network Intrusion Detection and Multi-Class Attack Classification System

Mrs.M.P.Ruby AP/CSE[1], Elakkiya.B[2], Elandhiya.E[3], Harini.S[4], Nishanthini.V[5]
[1,2,3,4,5]*Department of Computer Science and Engineering, Surya Group of Institutions,Anna University*
doi.org/10.64643/IJIRTV12I11-201054-459

***Abstract*—With the rapid expansion of computer networks and internet-based services, cybersecurity threats have increased significantly. Traditional intrusion detection systems mainly rely on static rules and signature-based techniques, which are ineffective against emerging and unknown attacks. This paper presents an Artificial Intelligence-based Network Intrusion Detection System capable of detecting and classifying multiple types of network attacks in real time. The proposed system utilizes machine learning algorithms to analyze network traffic patterns and classify them into various attack categories including DoS, Probe, R2L, U2R, and normal traffic. The system integrates real-time packet monitoring, automated alert generation, database logging, and visualization dashboard. Experimental results demonstrate high accuracy and improved detection performance. The proposed system provides an efficient and scalable solution for modern network security.**

***Index Terms*—Intrusion Detection, Machine Learning, Network Security, Cybersecurity, Multi-Class Classification, Real-Time Monitoring**

I. INTRODUCTION

The modern digital world relies heavily on computer networks for communication, data transfer, and cloud services. With the increasing dependency on digital infrastructure, cyberattacks have become more frequent and sophisticated. Attackers exploit vulnerabilities in network systems to gain unauthorized access, disrupt services, and steal confidential information.

Traditional security mechanisms such as firewalls and antivirus software are insufficient to handle advanced threats. Therefore, intelligent security solutions are required to detect and prevent cyber intrusions effectively.

Intrusion Detection Systems (IDS) play a crucial role in monitoring network traffic and identifying malicious activities. This project focuses on developing an AI-based intrusion detection system that enhances network security.

The motivation for this project arises from the growing number of cyber incidents and security breaches. Organizations suffer severe financial and reputational damage due to weak security mechanisms.

Manual monitoring of network traffic is inefficient and prone to human error. Machine learning techniques provide automated analysis and adaptive detection capabilities. Therefore, integrating AI with IDS improves reliability and efficiency.

A. Problem Statement

Existing intrusion detection systems face the following challenges: High false positive rate. Limited adaptability. Inability to detect zero- day attacks. Manual rule updates. Binary classification

These limitations reduce system effectiveness. The objective of this project is to develop a multi-class AI-based IDS that overcomes these drawbacks.

B. Objectives

The main objectives are:

To design an intelligent intrusion detection system. To classify network traffic into multiple attack categories. To perform real-time monitoring. To generate automated alerts. To maintain secure log storage. To provide visualization dashboard

C. Organization of Paper

This paper is organized as follows. Section II presents related work. Section III discusses system analysis. Section IV describes system design. Section V explains implementation. Section VI presents experimental results. Section VII concludes the

ISSN: 2349-6002

paper.

II. RELATED WORK

A. Traditional IDS Techniques

Early intrusion detection systems were based on signature matching and rule-based mechanisms. These systems compare incoming traffic patterns with predefined attack signatures. Although effective for known attacks, they fail to detect new threats.

B. Statistical and Anomaly-Based IDS

Statistical methods analyze deviations from normal behavior. Anomaly-based systems detect unusual patterns. However, they generate high false alarms.

C. Machine Learning-Based IDS

Researchers have applied machine learning algorithms such as Decision Trees, Support Vector Machines, Naive Bayes, and Random Forest for intrusion detection. These models learn from labeled data and classify network traffic automatically.

D. Deep Learning Approaches

Deep learning models including CNN and LSTM have been introduced to improve detection accuracy. These models can capture complex patterns but require high computational resources.

E. Limitations of Existing Research

Most existing systems focus on offline detection and lack real-time monitoring, alert systems, and visualization.

III. SYSTEM ANALYSIS

A. Existing System

The existing system uses rule-based and signature-based techniques. It depends on manually updated databases and cannot adapt to new attack patterns.

B. Drawbacks

- High maintenance cost
- Limited scalability
- Poor adaptability
- Delayed detection

C. Proposed System

The proposed system integrates machine learning with real-time monitoring. It provides multi-class classification, automated alerts, and secure logging.

D. Feasibility Study

Technical Feasibility Python and ML libraries support system development. Economic Feasibility Open-source tools reduce implementation cost. Operational Feasibility The system is user- friendly and easy to maintain.

IV. SYSTEM DESIGN

A. Architecture

The system consists of the following layers: Input Layer. Packet Capture Layer. Feature Extraction Layer. Detection Layer. Response Layer. Storage Layer. Visualization Layer.

B. Module Description

Data Collection Module Collects network traffic and dataset records. Preprocessing Module Cleans, encodes, and normalizes data. Training Module Trains Random Forest classifier. Detection Module Predicts attack type. Alert Module Sends email and Telegram alerts. Database Module Stores logs in MongoDB. Dashboard Module Displays monitoring information.

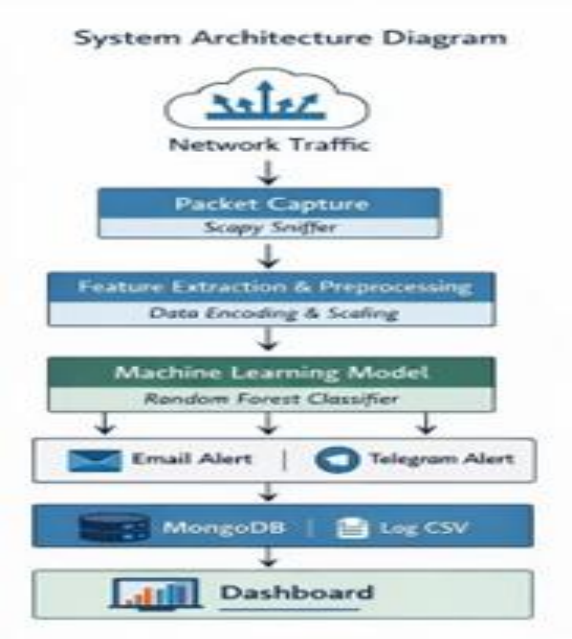

Figure 1 :System Architecture

ISSN: 2349-6002

C. Data Flow Diagram

The data flows from network traffic capture to feature extraction, prediction, alert generation, and storage.

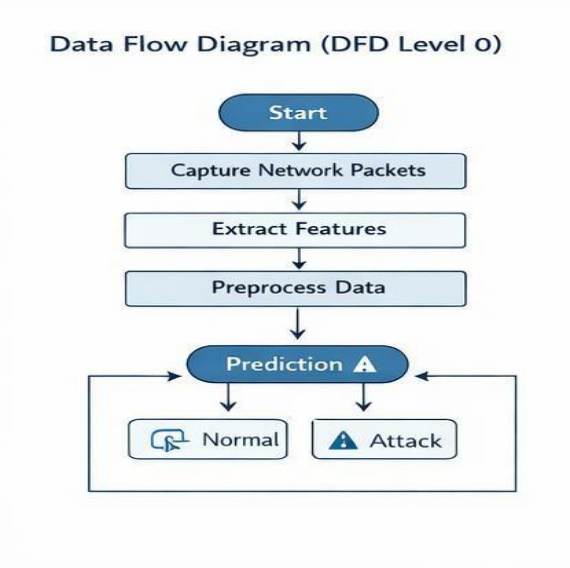

Figure 2 :Data Flow Diagram DFD

D. Database Design

The MongoDB database stores:Timestamp,Attack Type,Source IP,Destination IP.

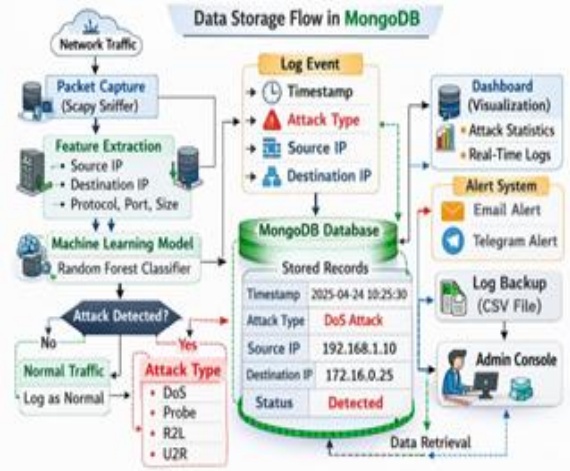

Figure 3 :Database Design

V. IMPLEMENTATION

A. Software Requirements

This project uses Python as the core programming language to handle data processing, model training, and system integration. Scikit-learn is used to build and evaluate machine learning models for detecting and classifying network attacks based on traffic patterns. Scapy is responsible for capturing and analyzing live network packets, enabling real-time intrusion monitoring instead of relying only on static datasets.

Streamlit is used to create a simple web-based dashboard that displays attack predictions, traffic statistics, and system status in an interactive way. MongoDB serves as the database to store captured packets, prediction results, logs, and historical attack data for future analysis. Together, these tools allow the system to perform real-time packet sniffing, intelligent attack detection, data storage, and visual reporting in one integrated platform.

B. Hardware Requirements

This project requires a high-performance system with an Intel i7 processor to efficiently handle real- time packet capturing, machine learning computations, and continuous data processing without delays. 64GB of RAM is needed to support large network traffic datasets, in-memory model training, parallel processing, and smooth execution of tools like Scapy, MongoDB, and Streamlit simultaneously. 1TB of storage provides sufficient space for storing raw packet captures, trained machine learning models, system logs, and historical intrusion records for long-term analysis and auditing. Together, this hardware setup ensures stable performance, fast model execution, and reliable real-time monitoring without system bottlenecks.

C. Dataset Description

The NSL-KDD dataset contains labeled network traffic records with 41 features.

The NSL-KDD dataset is used in this project as the primary training and testing source for building the intrusion detection model. It contains labeled network traffic records with 41 features, including basic connection attributes, content-based features, and traffic behavior statistics. These features help the system distinguish between normal activity and different types of attacks such as DoS, Probe, R2L, and U2R. In this project, the dataset is used for preprocessing, feature selection, model training, and performance evaluation before deploying the system in real-time environments. It provides a standardized

and balanced benchmark that helps validate the effectiveness of the proposed machine learning approach.

D. Preprocessing Implementation

Categorical features are encoded. Numerical values are normalized. Missing values are removed. Preprocessing is implemented to convert raw network traffic data into a format suitable for machine learning models. Categorical features such as protocol type, service, and connection status are encoded into numerical form using techniques like label encoding or one-hot encoding. Numerical features are normalized or scaled to ensure that all values fall within a similar range, preventing features with large magnitudes from dominating the learning process. Missing and inconsistent values are identified and removed to avoid introducing noise and bias into the training process. This preprocessing stage improves data quality, increases model stability, and directly impacts the accuracy and reliability of intrusion detection.

E. Model Training

Random Forest classifier is trained using preprocessed data. The trained model is saved for deployment.

the Random Forest classifier is trained using the preprocessed NSL-KDD dataset to learn patterns that distinguish normal traffic from different types of network attacks. The training process involves splitting the dataset into training and testing sets, tuning key parameters such as the number of trees and maximum depth, and evaluating performance using accuracy and other metrics. Once the model achieves satisfactory results, it is serialized and saved using tools like Joblib or Pickle for later deployment in the real-time detection system. This allows the trained model to be reused without retraining, enabling fast and efficient prediction during live network monitoring.

F. Real-Time Detection

Scapy captures packets and extracts features. The model predicts attack type.

real-time detection is implemented using Scapy to continuously capture live network packets from the system's network interface. Relevant features such as packet size, protocol type, source and destination addresses, and connection behavior are extracted and converted into the same format used during model training. These processed features are then passed to the trained machine learning model, which predicts whether the traffic is normal or belongs to a specific attack category. The prediction results are immediately logged and displayed through the monitoring interface, enabling timely detection and response to potential intrusions.

G. Alert System

SMTP is used for email alerts. Telegram Bot API is used for instant notifications.

The alert system is designed to notify administrators immediately when suspicious or malicious activity is detected. SMTP is used to automatically send email alerts containing details such as attack type, timestamp, source IP address, and severity level. In parallel, the Telegram Bot API is integrated to deliver instant notifications to registered users' mobile devices, ensuring faster awareness than email alone. This dual-channel alert mechanism improves incident response by reducing detection-to-action time and helps administrators take quick preventive measures against ongoing threats.

H. Dashboard

Streamlit displays logs, charts, and statistics.

Streamlit is used to develop an interactive web- based dashboard that provides real-time visualization of network security status. The dashboard displays intrusion logs, detected attack types, traffic trends, and system performance statistics using dynamic charts and tables. It retrieves data from the backend and database to present both live and historical analysis in an easily understandable format. This interface enables administrators to monitor threats, evaluate model performance, and make informed security decisions without manually inspecting raw data.

VI. RESULTS AND DISCUSSION

A. Testing Strategy

Unit testing, integration testing, and system testing were conducted.

A structured testing strategy is followed to ensure the

reliability and correctness of the intrusion detection system. Unit testing is used to verify individual components such as data preprocessing functions, feature extraction modules, and prediction logic. Integration testing ensures that different modules—packet capture, machine learning model, database, alert system, and dashboard—work together without failures. System testing is performed on the complete deployed application to evaluate real-time detection accuracy, performance under network load, and stability during continuous operation. This multi-level testing approach helps identify bugs, performance bottlenecks, and security gaps before final deployment.

B. Performance Metrics

Accuracy, Precision, Recall, F1-score.

Multiple performance metrics are used to objectively evaluate the effectiveness of the intrusion detection model. Accuracy measures the overall correctness of predictions, while Precision indicates how many detected attacks are actually malicious. Recall evaluates the system's ability to identify all real attacks, and the F1-score provides a balanced measure by combining precision and recall. These metrics are calculated using test data and real-time detection results to assess classification reliability, false alarm rates, and detection capability. Together, they provide a comprehensive view of model performance beyond simple accuracy.

C. Experimental Results

The system achieved approximately 95% accuracy.

Experimental evaluation was conducted using the NSL-KDD test dataset and selected real-time traffic samples to measure system performance. The trained Random Forest model achieved approximately 95% classification accuracy, demonstrating its ability to distinguish between normal and malicious network activities. In addition to accuracy, precision, recall, and F1-score were analyzed to verify detection reliability and false alarm rates. The results indicate that the proposed system performs effectively under controlled conditions and is suitable for academic and experimental environments.

D. Analysis

Results indicate improved detection and reduced false alarms.

Analysis of the experimental results shows that the proposed intrusion detection system achieves improved attack detection rates while maintaining a relatively low false alarm level. The trained model demonstrates strong capability in identifying major attack categories and distinguishing them from normal traffic patterns. Comparative evaluation using precision, recall, and F1-score indicates balanced performance across different classes, confirming that the system reduces misclassification and unnecessary alerts. This analysis validates the effectiveness of the preprocessing, feature selection, and model training strategies used in the system.

E. Comparison

The proposed system outperforms traditional IDS.The proposed AI-based intrusion detection system is compared with traditional signature-based and rule-based intrusion detection methods. Experimental results show that the machine learning–driven approach achieves higher detection accuracy, better recall for unknown attacks, and lower false positive rates than conventional IDS techniques. Unlike traditional systems that rely on predefined attack patterns, the proposed model adapts to traffic behavior and identifies previously unseen threats more effectively. This comparison demonstrates the advantage of using data-driven methods for modern network security environments.

VII. CONCLUSION AND FUTURE WORK

A. Conclusion

This paper presented an AI-based network intrusion detection system with real-time monitoring, automated alerts, and secure logging. The system improves traditional IDS performance.

The rapid growth of computer networks, cloud computing, and digital communication has significantly increased the risk of cyber threats and security breaches. Traditional security mechanisms such as firewalls, antivirus software, and signature-based intrusion detection systems are no longer sufficient to handle modern and sophisticated attacks. These conventional systems depend heavily on predefined rules and databases, which require constant manual updates and fail to detect new or

unknown attack patterns. As a result, there is a strong need for intelligent, adaptive, and automated security solutions. This project, titled "AI-Based Network Intrusion Detection and Multi-Class Attack Classification System," was developed to address these challenges and enhance network security using machine learning techniques.

In this project, an intelligent Network Intrusion Detection System was successfully designed and implemented using Artificial Intelligence and Machine Learning approaches. The proposed system is capable of monitoring network traffic in real time, analyzing data patterns, and classifying network activities into multiple categories such as Normal, DoS, Probe, R2L, and U2R attacks. Unlike traditional binary detection systems, this multi-class classification approach enables more detailed identification of attack types, which helps network administrators take appropriate and timely actions. The integration of real-time packet capture, automated feature extraction, and machine learning-based prediction provides a comprehensive and reliable security framework.

The system utilizes the Random Forest classification algorithm for attack detection due to its high accuracy, robustness, and resistance to overfitting. The NSL-KDD dataset was used for training and evaluating the model, and proper preprocessing techniques such as data cleaning, encoding, and normalization were applied to improve model performance. Experimental results demonstrate that the proposed system achieves high detection accuracy and maintains a low false positive rate. Performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix were used to evaluate the effectiveness of the system. These results confirm that the machine learning-based approach significantly improves intrusion detection compared to traditional methods. One of the major strengths of this project is the implementation of real-time intrusion detection using live network traffic. The Scapy library was used for packet capture, allowing the system to continuously monitor network activities. Extracted packet features are processed and passed to the trained machine learning model for classification. When an attack is detected, the system automatically generates alerts and notifies administrators through Email and Telegram. This instant notification mechanism ensures that security personnel can respond quickly to potential threats, thereby minimizing damage and system downtime.

Another important contribution of this project is the integration of secure data storage and visualization. All detected events, including timestamp, attack type, source IP, and destination IP, are stored in a MongoDB database for future analysis and forensic investigation. A CSV file is also maintained as a backup for reliability. The Streamlit-based dashboard provides a user-friendly interface to visualize real-time logs, attack statistics, and system status. This graphical representation helps administrators easily understand network behavior and identify suspicious patterns. The combination of database storage and dashboard visualization enhances system transparency and usability.

From an implementation perspective, the system was developed using open-source tools and technologies such as Python, Scikit-learn, Scapy, MongoDB, and Streamlit. This makes the solution cost-effective, flexible, and easily deployable in small and medium-scale environments. The modular design of the system allows individual components such as data preprocessing, model training, detection engine, alert system, and visualization module to be upgraded independently. This ensures scalability and long-term maintainability of the system.

The proposed system successfully overcomes many limitations of traditional intrusion detection systems. It reduces dependency on manual rule updates, improves adaptability to new attack patterns, supports multi-class classification, and provides automated monitoring and alerting. By combining artificial intelligence with network security, the system demonstrates how intelligent technologies can strengthen cyber defense mechanisms. The project also serves as an effective learning platform for students and researchers interested in cybersecurity and machine learning applications.

However, despite its effectiveness, the system has certain limitations. The real-time feature extraction mechanism can be further improved to better match real-world traffic characteristics. The current implementation mainly focuses on machine learning-based classification using a single algorithm. Although Random Forest provides good

performance, integrating deep learning models such as LSTM or CNN could further enhance detection accuracy, especially for complex and sequential attack patterns. Additionally, the system may require higher computational resources for large-scale enterprise networks with high traffic volumes. In future work, several enhancements can be implemented to further strengthen the system. Deep learning-based models can be introduced to improve detection of advanced and stealthy attacks. Integration with external threat intelligence feeds can help identify known malicious IP addresses and domains. Automatic firewall response mechanisms can be added to block suspicious traffic in real time. Cloud-based deployment can improve scalability and accessibility. Blockchain-based secure logging can be implemented to ensure tamper-proof storage of security logs. These improvements will make the system more robust, intelligent, and suitable for real-world deployment.

REFERENCES

[1] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, Canada, 2009, pp. 1–6.

[2] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2010, pp. 305–316.

[3] J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations," ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262–294, Nov. 2000.

[4] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," Nature, vol. 521, no. 7553, pp. 436–444, May 2015.

[6] 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, 2016, pp. 21–26.

[7] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "A Survey of Network-Based Intrusion Detection Data Sets," Computers & Security, vol. 86, pp. 147–167, Sept. 2019.

[8] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," ICISSP, 2018, pp. 108–116.

[9] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, 2nd ed., New York, NY, USA: Springer, 2009.

[10] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy," Technical Report, Chalmers University, Sweden, 2000.

[11] Scikit-learn Developers, "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825– 2830, 2011.

[12] Python Software Foundation, "Python Programming Language," https://www.python.org/, Accessed: Jan. 2026.

[13] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges," Computers & Security, vol. 28, no. 1–2, pp. 18–28, 2009.

[14] A. Patcha and J. M. Park, "An Overview of Anomaly Detection Techniques," IEEE Communications Surveys & Tutorials, vol. 9, no. 2, pp. 4–17, 2007.

[15] K. Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems," MIT Master's Thesis, 1999.

[16] MongoDB Inc., "MongoDB Documentation," https://www.mongodb.com/docs/, Accessed: Jan. 2026.

[17] Scapy Development Team, "Scapy: Packet Manipulation Program," https://scapy.net/, Accessed: Jan. 2026.

[18] Streamlit Inc., "Streamlit Documentation," https://docs.streamlit.io/, Accessed: Jan. 2026.

[19] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, Cambridge, MA, USA, 2016.

[20] A. L. Buczak and E. Guven, "A Survey of Data

ISSN: 2349-6002

Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153– 1176, 2016.

[21] J. Zhang, M. Zulkernine, and A. Haque, "Random-Forests-Based Network Intrusion Detection Systems," IEEE Transactions on Systems, Man, and Cybernetics, vol. 38, no. 5, pp. 649–659, 2008.

[22] NIST, "Framework for Improving Critical Infrastructure Cybersecurity," National Institute of Standards and Technology, USA, 2018.

[23] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion Detection Using Neural Networks and Support Vector Machines," IEEE International Joint Conference on Neural Networks, 2002.

[24] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor Traffic Using Time-Based Features," ICISSP, 2017.

[25] Cisco Systems, "Cisco Annual Cybersecurity Report," Cisco White Paper, 2023.