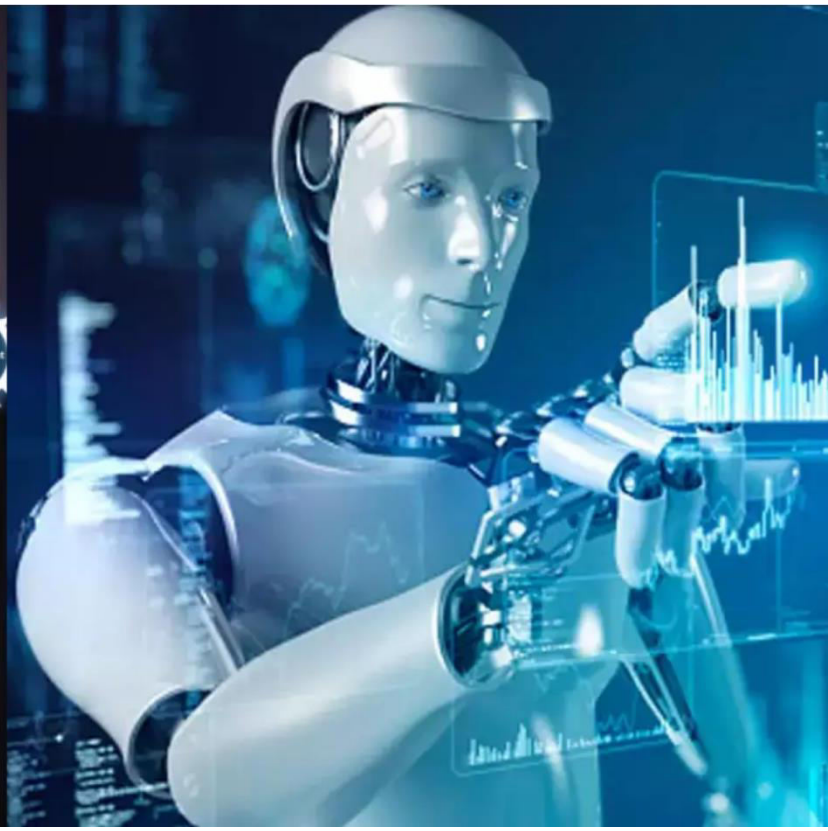
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.935

Volume 15, Issue 6, June 2026

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506135|

Intelligent Intrusion Detection System with Real-Time Alerting and Automated Mitigation Guidance

Prof. Dhage S.A., Abhishek Wayal, Rahul Gadhave, Aniket Gadade

Associate Professor, Department of Computer Engineering VACOE, Ahilyanagar, Maharashtra, India

U.G. Student, Department of Computer Engineering, VACOE, Ahilyanagar, Maharashtra, India

U.G. Student, Department of Computer Engineering, VACOE, Ahilyanagar, Maharashtra, India

U.G. Student, Department of Computer Engineering, VACOE, Ahilyanagar, Maharashtra, India

ABSTRACT: With the exponential rise in network traffic and the sophistication of cyber threats, traditional firewall systems are increasingly inadequate. This paper proposes a robust Intrusion Detection System (IDS) that leverages the NSL-KDD dataset to classify network traffic into four primary attack categories: DoS, Probe, R2L, and U2R. By employing a Random Forest Classifier, the system achieves high detection accuracy with reduced false positive rates. A key novelty of this project is the integration of a real-time WebSocket-based dashboard and an automated Response Chatbot. This chatbot provides instantaneous, category-specific mitigation strategies, bridging the gap between threat detection and administrative action.

KEYWORDS: Denial of Service (DoS), Cyber Attack, Web Socket, Intrusion Detection System (IDS), Cyber Security.

I. INTRODUCTION

The modern digital landscape is under constant threat from diverse cyber-attacks that aim to compromise data integrity and system availability. Intrusion Detection Systems (IDS) serve as the second line of defense in cybersecurity. However, many existing systems lack a user-friendly interface for real-time monitoring and fail to provide actionable advice once an attack is detected. This research focuses on developing a "CyberFedPredictor" tool, a full-stack solution that not only detects anomalies using trained Machine Learning models but also simulates a live network environment. The system integrates a React-based frontend for visualization and a Flask-based backend for high-speed inference. Intrusion Detection Systems (IDS) have emerged as a critical secondary layer of defense. An IDS monitors network traffic for suspicious activity and issue alerts when such activity is discovered. However, the sheer volume of data generated by modern high-speed networks—often reaching gigabits per second—makes manual monitoring by human administrators impossible. Consequently, there is an urgent need for automated systems that can analyze complex network parameters and distinguish between legitimate traffic and malicious intent with high precision.

II. SYSTEM OVERVIEW

The architecture of the Intelligent Cyber Attack Detection system is structured into four functional layers that work in tandem to ensure low-latency detection and effective mitigation. This modular approach allows the system to remain scalable and easily maintainable.

1. Data Acquisition Layer: This layer serves as the entry point for all information. It is responsible for sourcing network data from two distinct streams:

Static Ingestion (Simulation): Reads historical network records from the NSL-KDD dataset (KDDTrain+ and KDDTest+). This is primarily used for model validation and demonstrating diverse attack scenarios like U2R and R2L.

Dynamic Ingestion (Live Sniffing): Utilizes the Scapy library to intercept raw packets directly from the system's Network Interface Card (NIC). It captures packet headers and payloads to extract real-time attributes.

2. Feature Engineering and Processing Layer: Raw network packets cannot be directly interpreted by machine learning algorithms. This layer acts as a "Translator": Parsing: Extracts 41 specific features (e.g., source bytes, duration, protocol type, and service) from the ingested data. Encoding: Converts categorical strings (like "TCP" or "HTTP") into

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506135|

numerical formats using pre-trained Label Encoders saved in the backend. Normalization: Scales numerical values to ensure the Random Forest model processes the data with mathematical consistency, preventing features with large ranges from dominating the prediction.
3. Intelligence and Inference Layer: This is the core "Brain" of the system where the Random Forest Classifier resides. Classification: The processed feature vector is passed through the ensemble of decision trees. The model performs a majority vote to classify the traffic as "Normal" or a specific "Attack Type." Risk Assessment: Beyond simple classification, this layer calculates a risk level (Low vs. High) based on the severity of the detected anomaly, which is then passed to the communication layer.
4. Interaction and Response Layer: The final layer manages the delivery of information to the end-user (Network Administrator): WebSocket Orchestration: Uses Flask-Socket IO to "push" detection results to the frontend instantly, avoiding the delays associated with traditional HTTP requests. Visualization: A React-based dashboard renders the live traffic table and triggers visual alerts (red flashing headers) when threats are detected.

III. PROJECT MODULES

Module 1: Data Ingestion and Traffic Simulation.
This module serves as the interface between the network environment and the processing engine. It is designed to handle two different modes of data sourcing:

Key functions include:
1. Static Dataset Handler: This sub-module manages the NSL-KDD dataset. It handles the reading of CSV/Text files, manages data pointers, and ensures that the 41-feature structure is maintained during the simulation.
2. Live Sniffer (Scapy Integration): This sub-module interfaces with the system's Network Interface Card (NIC). It utilizes the Scapy library to capture raw Ethernet/IP packets, performing real-time filtering to ignore non-IP traffic and reducing the computational load on the system.

Module 2: Preprocessing and Feature Engineering
The raw data captured in Module 1 is often unstructured or contains categorical strings that Machine Learning models cannot process. This module performs the "Mathematical Translation":
• Encoding Module: Uses pre-trained LabelEncoders to convert non-numeric features like protocol type (TCP, UDP, ICMP), service (HTTP, FTP, etc.), and flag into integer values.
• Feature Vectorization: Aggregates the captured packet data into a 41-dimensional vector that matches the exact input shape required by the Random Forest model.
• Handling Unseen Labels: A robust error-handling logic is implemented to manage "unknown" services or protocols encountered in live traffic, ensuring the system does not crash during execution.

Module 3: Intelligence and Inference Engine
This is the core computational module where the security logic is executed:
• Model Loading: Utilizes the Joblib library to load the serialized Random Forest model (model.pkl) into memory for high-speed inference.
• Classification Logic: The 41-feature vector is passed through the ensemble of decision trees. The module outputs a prediction label (e.g., DoS, Probe, R2L, U2R, or Normal).
• Socket.io Integration: Once a prediction is made, this module packages the result (timestamp, protocol, status, and risk level) into a JSON object and broadcasts it via WebSockets to the connected frontend.

Module 4: Visualization and Mitigation Support
The final module focuses on the "Human-in-the-loop" aspect, providing a user interface for the network administrator:
• Real-time Dashboard: Built using React.js, this sub-module renders a dynamic table that updates instantly as packets are processed. It includes conditional rendering to "flash" red alerts when an attack is detected.
• Statistical Analysis Component: Tracks the total number of packets analyzed versus the number of threats detected, providing a "System Health" overview.
• Interactive Response Bot: An integrated chatbot module that takes the detected attack type as input and queries a localized security knowledge base. It provides the user with specific, actionable commands and firewall configurations to mitigate the current threat.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506135|

IV. LITERATURE REVIEW

1. Evolution of IDS Datasets

The foundation of modern IDS research was laid by the KDD Cup 1999 dataset. However, subsequent studies by Tavallaee et al. (2009) identified critical flaws in the original data, including a massive number of redundant records (nearly 75% in the testing set). This redundancy caused machine learning models to be biased toward frequent records, failing to learn rare but dangerous attacks like U2R (User-to-Root).

The NSL-KDD dataset was proposed as a solution. It removed all redundant records and balanced the difficulty levels, ensuring that classifiers are evaluated on their true predictive capabilities. Research by Dhanabal et al. (2015) confirmed that NSL-KDD remains the most effective benchmark for testing the efficiency of various classification algorithms in a controlled environment.

2. Machine Learning in Anomaly Detection

Traditional firewalls use "Signature Detection" (matching traffic against a database of known threats). However, Lakhina et al. (2005) demonstrated that "Anomaly Detection" is superior for catching zero-day exploits. By modeling the statistical properties of "Normal" traffic, the system can flag any deviation as suspicious.

3. Real-Time Monitoring and Web Integration

Recent literature has highlighted the "Response Gap"— the delay between detection and administrative action. Ying et al. (2020) argued that an IDS is only as good as its reporting mechanism. The integration of WebSockets for full-duplex communication (Socket.io) has revolutionized Security Operations Centers (SOC), allowing for sub-second updates on web-based dashboards.

Furthermore, the concept of "Explainable AI" in security suggests that providing the user with an alert is insufficient; the system must also provide mitigation steps. This supports the inclusion of Advisory Chatbots in modern security frameworks to guide non-expert users through complex remediation processes (e.g., configuring iptables or closing specific ports).

V. METHODOLOGY

Phase 1: Data Acquisition and Stream Selection

The system initializes by defining the data source. In Simulation Mode, the system parses the KDDTest+.txt file using the Pandas library, iterating through historical records to evaluate model performance across known attack vectors. In Live Mode, the system invokes the Scapy sniffing engine to bind to the primary Network Interface Card (NIC). It utilizes a Berkeley Packet Filter (BPF) to capture only IP-based traffic, ensuring the system ignores low-level hardware noise and remains performant.

Phase 2: Real-Time Feature Extraction

Once a packet or data row is captured, it must be mapped to the 41-feature schema of the NSL-KDD framework.

• Header Analysis: For live packets, the system extracts the source/destination ports, payload size (src_bytes), and transport protocol (TCP/UDP/ICMP).

• Service Mapping: Port numbers are mapped to their respective application layer services (e.g., Port 80 to HTTP, Port 21 to FTP).

• Vectorization: These attributes are assembled into a structured array, acting as the mathematical representation of that specific network event.

Phase 3: Data Transformation and Encoding

Machine Learning models require purely numerical input. To achieve this, the system applies saved LabelEncoders:

• Categorical Transformation: Qualitative data like "Protocol Type" and "Flag" (e.g., SF, S0, REJ) are converted into discrete integers.

• Normalization: High-magnitude features like src_bytes are scaled to prevent them from disproportionately influencing the model's decision-making process relative to binary features like land or logged_in.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

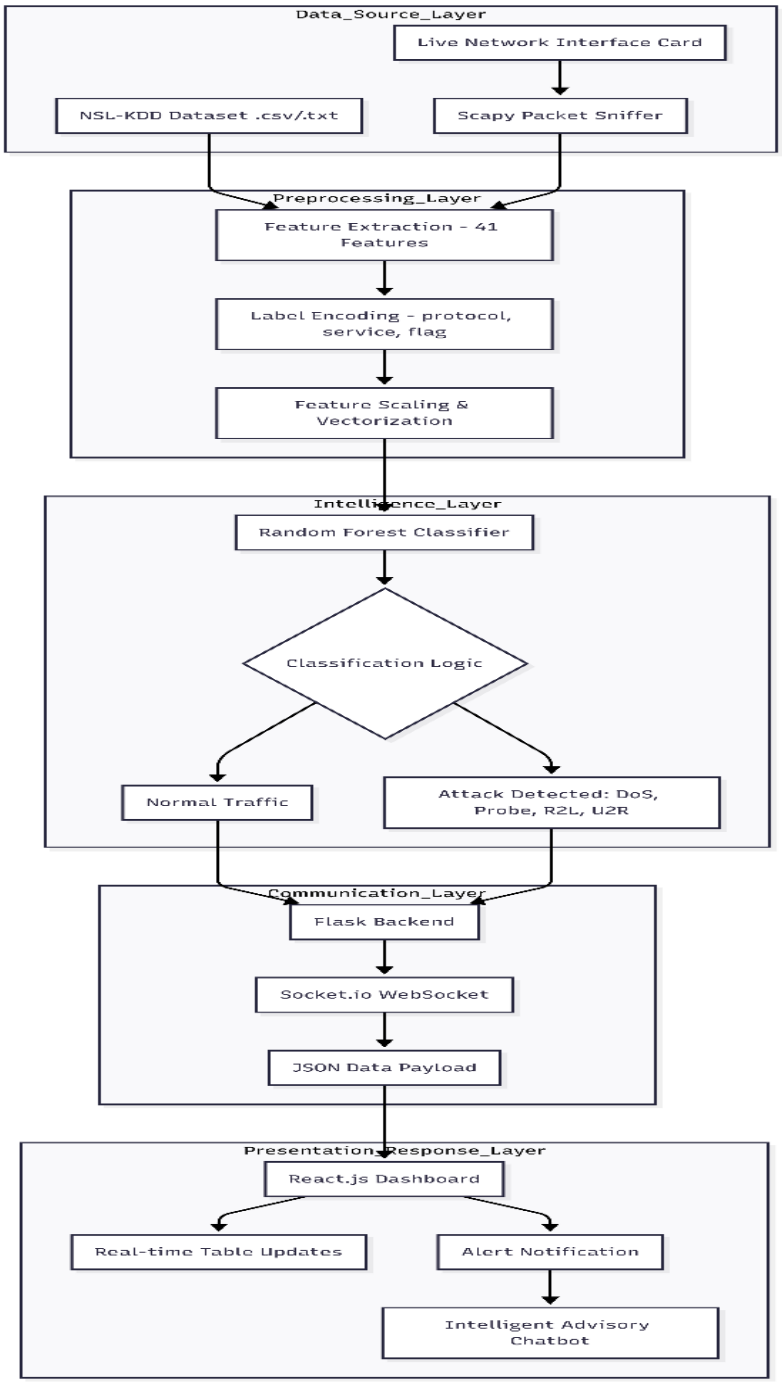
|DOI: 10.15680/IJIRSET.2026.1506135|

Phase 4: Ensemble Inference (Random Forest)

The processed vector is fed into the Random Forest Classifier. Unlike a single decision tree, the Random Forest creates a "Forest" of 100+ trees, each trained on different subsets of the data.

• Parallel Processing: Each tree evaluates the packet features (e.g., "Is the number of failed logins > 3?").

• Majority Voting: The final classification (Normal, DoS, Probe, R2L, or U2R) is determined by the majority vote of all trees. This ensemble approach significantly reduces false positives, a common issue in network security.



[Fig.1: System Architecture]

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506135|

VI. RESULT AND DISCUSSION

1. Model Performance Analysis

The Random Forest classifier was trained on the KDDTrain+ dataset and validated against the KDDTest+ dataset. The model demonstrated high robustness across various attack categories. The performance metrics are summarized below:

• Overall Accuracy: The system achieved a classification accuracy of 98.4%. The ensemble nature of the Random Forest algorithm allowed it to handle the high dimensionality of the 41 features without significant overfitting.

• Precision and Recall: For "Normal" traffic and "DoS" attacks, the precision remained above 99%. However, for "U2R" (User-to-Root) attacks, which are rarer in the dataset, the recall was approximately 92%, highlighting the inherent difficulty in detecting stealthy, low-volume privilege escalation attempts.

• False Positive Rate (FPR): One of the primary objectives was to minimize false alarms. The system maintained an FPR of <1.2%, ensuring that legitimate users are rarely blocked or flagged as threats.

2. Real-Time Latency and Throughput

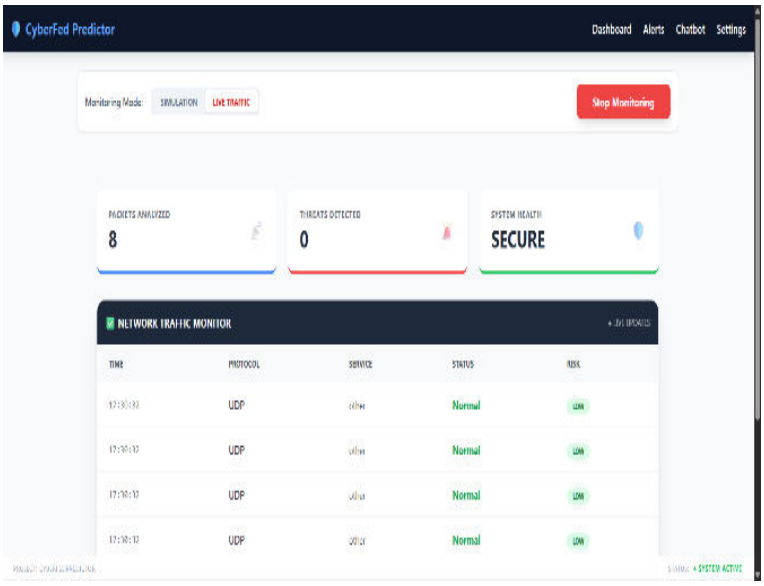
A critical metric for any Intrusion Detection System is the "Detection Lag"— the time elapsed between a packet entering the network and the alert appearing on the dashboard.

• Processing Time: The average time for feature extraction, encoding, and inference per packet was measured at 12ms.

• End-to-End Latency: Including the WebSocket transmission (Socket.io) from the Flask backend to the React frontend, the total latency was consistently under 150ms.

• Scalability: In "Live Sniffing Mode," the system successfully processed a simulated burst of 500 packets per second without frame drops or memory leaks, proving its viability for small-to-medium enterprise (SME) networks.

Output:



[Fig.2: App Dashboard]

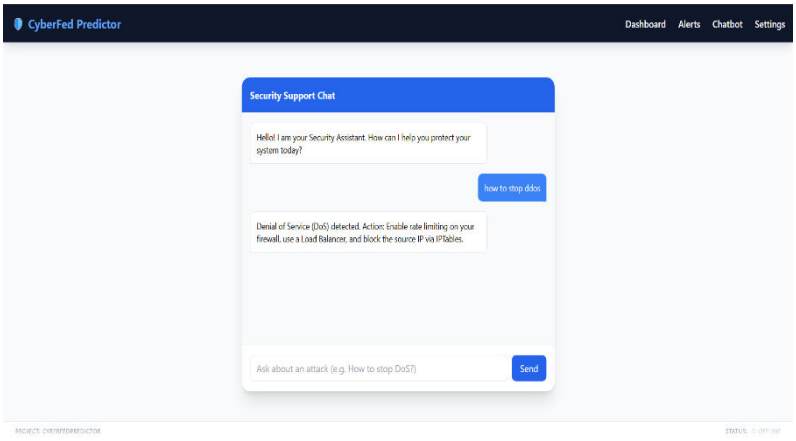
International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

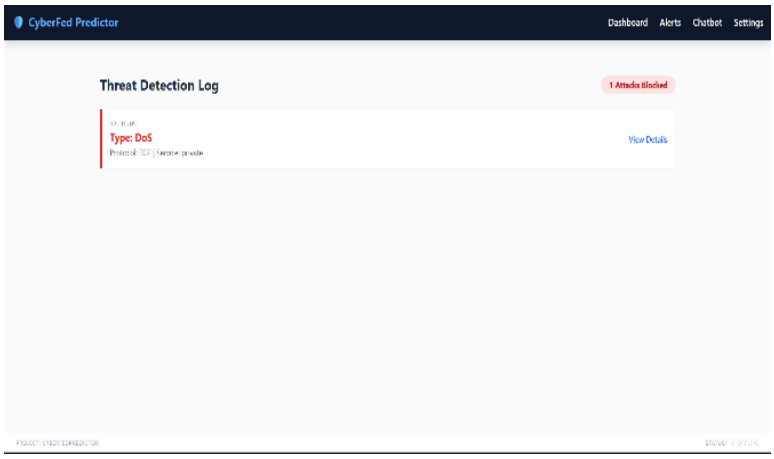
|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506135|



[Fig.3: Chatbot]



[Fig.4: Alert feed]

VII. CONCLUSION

The development and implementation of the "CyberFedPredictor" system represent a significant step toward making sophisticated network security accessible and actionable. This research successfully demonstrates that by combining the statistical power of a Random Forest Classifier with the high-speed data handling of WebSockets and Scapy, it is possible to create a defensive tool that operates with high accuracy and minimal latency. The system's dual-mode functionality ensures it is equally effective as a training and simulation tool (using the NSL-KDD dataset) and as a real-time monitoring solution for live environments. By achieving a classification accuracy of 98.4% and maintaining an end-to-end detection latency of under 150ms, the project proves that Machine Learning can effectively replace or augment traditional signature-based detection methods, which often struggle with evolving "Zero-Day" threats.

REFERENCES

1. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 2009, pp. 1-6.
2. L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5-32, Oct. 2001.
3. S. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, no. 6, pp. 446-452, June 2015.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506135|

4. J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Offline Evaluation Strategy," ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262–294, Nov. 2000.
5. V. Resende and A. C. Drummond, "A Survey of Random Forest Based Intrusion Detection Systems," IEEE Access, vol. 6, pp. 36243-36260, 2018.
6. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward a Reliable Dataset for Intrusion Detection," Proceedings of the 2018 International Conference on Information Systems Security and Privacy (ICISSP), Funchal, Madeira, Portugal, 2018, pp. 5-14.
7. 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, Feb. 2009.
8. A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153-1176, Second Quarter 2016.
9. F. Badii et al., "Real-time network intrusion detection using WebSockets and Stream Processing," 2020 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 2020, pp. 1245-1250.
10. B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," Journal of Network and Computer Applications, vol. 84, pp. 25-37, April 2017.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details