Smart NIDS: A Web-Based AI-Powered Network Intrusion Detection System

N. Umamaheswari[1], Dr. Anitha T N[2], Narendra B M[3], Nisarga R[4], Parinith Gowda K[5], Priyanka M S[6]
[1]Assistant Professor,Department of CSE Sir M. Visvesvaraya Institute of Technology VTU, Bangalore, India
[2,3,4,5,6]Department of CSE, Sir M. Visvesvaraya Institute of Technology VTU, Bangalore, India

Abstract—**In today's world, the landscape of cyberattacks is becoming increasingly complex, which highlights the need for sophisticated Network Intrusion Detection Systems (NIDS) that can effectively spot both familiar and newly emerging threats.This paper introduces SMART NIDS, cutting- edge hybrid framework that incorporates both supervised and unsupervised learning methods, integrating human oversight to enhance network security. At its core, SMART NIDS utilizes a Random Forest (RF) classifier trained on the NL- KDD dataset to accurately identify known attack types. Alongside this, it employs an Isolation Forest (IF) model to pinpoint unconventional behaviors that fall outside the parameters of labeled data. Instances where the classifier lacks confidence specifically those below a 0.60 probability threshold are set aside in an Anonymous pool for additional scrutiny. To aid analysts in sifting through these uncertain cases, the framework employs DBSCAN clustering. This organizes samples based on similar behaviors, making it easier for experts to validate findings and adjust the model as necessary. Our proposed solution not only enhances overall performance, reflected in improved macro-F1 scores and better recall for minority classes, but does so while keeping the false alarm rate in check.**

Index Terms—**Anomaly Detection, Cybersecurity, DBSCAN, Human-in-the-Loop, Intrusion Detection System (IDS), Isolation Forest, Machine Learning, Network Security, NSL-KDD Dataset, Random Forest, Smart NIDS, Supervised Learning, Unsupervised Learning.**

I. INTRODUCTION

In our increasingly digital age, almost every organization relies heavily on computer networks. This strong dependence brings along a steady rise in cyber threats from massive denial-of-service (DoS) attacks to cunning intrusions that take advantage of software flaws. Traditional Intrusion Detection typically depend on set signatures or rigid rules to spot malicious behavior. While they excel at identifying well-known attacks, they often falter when it comes to detecting new or evolving threats that don't fit any established patterns.Recent studies have turned their attention toward using machine learning for intrusion detection, paving the way for systems that can analyze data and recognize suspicious activities on their own. However, standard supervised models rely heavily on labeled data and struggle with unknown types of attacks. Conversely, unsupervised techniques can pick up anomalies but usually lack the capability to categorize these findings effectively. This tension between accuracy and interpretability presents a significant hurdle in the quest to develop dependable intrusion detection systems.To address these challenges, we introduce SMART NIDS a hybrid network intrusion detection system that includes both supervised and unsupervised learning methods, all while keeping human oversight in the loop. Our system employs a Random Forest (RF) classifier for detecting familiar attack types, while an Isolation Forest (IF) model is used to recognize unusual traffic that may indicate new or unclassified threats.

II. LITERATURE REVIEW

Network Intrusion Detection Systems (NIDS) have gained significant attention in recent years, largely due to the increasing sophistication of cyberattacks. Traditionally, detection methods relied on signature-based or rule-based systems that compared network traffic to established attack patterns. While these methods worked well for threats that had been previously identified, they often fell short when it came to recognizing new, modified attacks. As networks became more fluid and unpredictable, researchers turned to machine learning (ML) as a means to enhance the flexibility and automation capabilities of intrusion detection. Initial ML-based IDS models employed supervised algorithms like Decision Trees,Support Vector Machines (SVMs) and Naïve Bayes. These models aimed to classify traffic as either normal based on labeled datasets. Research using the NSL- KDD dataset indicated that ensemble approaches,such as Random Forests, delivered improved accuracy and generalization over individual classifier delivered improved accuracy and generalization over individual classifiers. This was primarily due to their effectiveness in reducing overfitting and managing noisy data. However, one major drawback of supervised learning is its reliance on labeled data, which often hinders its ability to detect new attacks that differ from the previously seen examples. In line with these trends, we introduce SMART NIDS, which builds on this hybrid framework. Our system employs Random Forest to identify known attack types, Isolation Forest for spotting anomalous behavior, and DBSCAN for grouping uncertain cases into an Anonymous pool. A standout feature of our work is its focus on human guidance, enabling analysts to assess and elevate newly discovered threats into the supervised dataset. This method connects automated model functions with expert insights, fostering and continuous enhancement and adaptability in dynamic network environments.

III. SYSTEM DESIGN AND ARCHITECTURE

The SMART NIDS system is crafted as a hybrid intrusion detection framework that integrates machine learning with insights from human experts.

III-A Overview Design

- SMART NIDS begins by collecting and structuring network traffic data, which is then analyzed using both supervised and unsupervised models to detect and flag suspicious activities.
- Uncertain predictions are reviewed by human analysts, and their validated insights are fedback into the model retraining process.

III-B Detection Workflow

- Detection begins with a Random Forest (RF) trained on the NSL-KDD dataset to spot attacks like DoS, Probe, R2L, and U2R.
- Detection begins with a Random Forest (RF) trained on the NSL-KDD dataset to spot attacks like DoS, Probe, R2L, and U2R.

If confidence is above 0.60, it's labeled; otherwise, sent to the Isolation Forest for deeper analysis.

III-C Human-in-the-Loop Feedback

- Experts review and label clustered data via a React–Flask platform backed by PostgreSQL.
- Validated samples are added to the training set, and both models are retrained with version control for easy tracking and rollback.

III-D System Integration

- The backend handles APIs for predictions and model updates, while the dashboard displays alerts and trends; joblib and SQLAlchemy ensure smooth scalability.
- Combining automation with human insight, SMART NIDS detects intrusions accurately and adapts to evolving cyber threats.

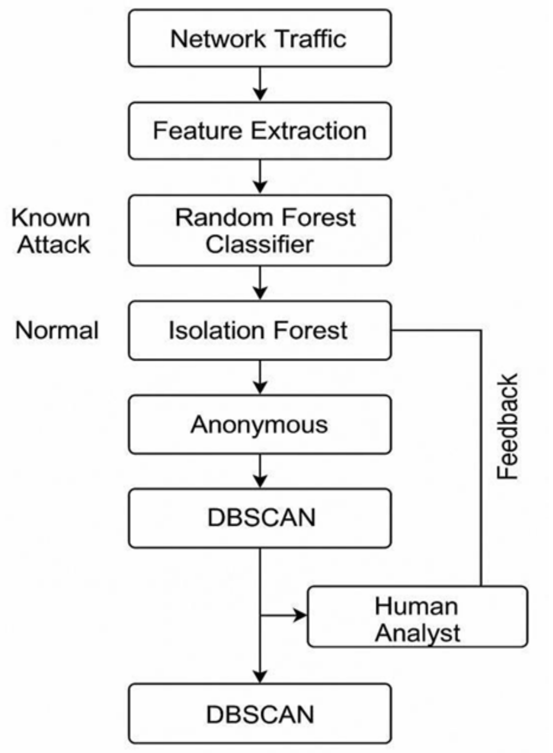

Fig. 1. SMART NIDS Architecture

IV. METHODOLOGY

SMART NIDS blends supervised and unsupervised detection with human feedback for continuous learning.

III-E Data Preprocessing

SMART NIDS uses the refined NSL-KDD dataset with 41 features.

The data is then preprocessed through key steps before training.

- Data Cleaning: Duplicate and incomplete records are eliminated to enhance the overall quality of the data.
- Feature Encoding: Categorical features, including protocol type, service, and flag, are adjusted using one- hot encoding for better processing.
- Normalization: Continuous features are standardized via z-score normalization, ensuring they are appropriate for algorithms like Isolation Forest and DBSCAN.
- Label Mapping: Each recorded attack is categorized into six main threat groups: Normal, DoS/DDoS, Probe, R2L, U2R, and Anonymous (also referred to as Unknown).

III-F Hybrid Detection Model

The detection framework intertwines three machine learning techniques: Random Forest (RF), Isolation Forest (IF), and DBSCAN, all underpinned by a confidence-based routing mechanism.

- Random Forest Classifier:
- The Random Forest model is used as the main classifier, trained on labeled data to learn the known attack categories.
- It outputs class probabilities for each input. If max_proba is greater than or equal to 0.60, the prediction is accepted as a known class.

- DBSCAN Clustering:
- The process begins by using DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to group samples found in the Anonymous queue.
- This clustering technique allows for similar unfamiliar behaviors to be clustered together, enabling analysts to quickly spot potential emerging attack trends without having to delve into each sample manually.

III-G Human-in-the-Loop Feedback

At the heart of the SMART NIDS lies a human feedback mechanism. Security analysts engage with the clustered Anonymous samples via a user-friendly web interface developed using React and Flask. Their tasks include:

- Verifying whether a specific cluster indicates a new type of attack or if it's simply benign activity.
- Re-labeling samples when necessary.
- Authorizing new attack signatures, which will then be used for model retraining.

III-H Model Retraining and Versioning

When analysts confirm or label new samples, these inputs are integrated into the training dataset. Subsequently, a retraining module steps in to update the Random Forest and Isolation Forest models. It assesses their performance against a validation set, and if the new iterations showcase better results than the existing models, they are saved as updated versions.Each version is then meticulously archived in a model registry with version control, supporting reproducibility and allowing for easy rollback in case performance dips.

III-I Summary of Workflow The workflow unfolds as follows:

- Data collected from NSL-KDD is initially preprocessed and encoded.
- Known attacks are classified using the Random Forest model.
- Samples that exhibit low confidence (with a probability of less than 0.60) are directed towards the Isolation Forest for further evaluation.

V. IMPLEMENTATION

The SMART NIDS (Smart Network Intrusion Detection System) was implemented in a modular and scalable way to achieve high detection accuracy, adaptability, and ease of human interaction. The system integrates supervised and unsupervised learning algorithms with a human-in- the-loop feedback process.

III-J System Setup and Development Environment

- Programming Language: Python 3.10 was used for all machine learning and backend operations because of its strong ecosystem for data science.
- Frameworks and Tools: Flask was used to develop

REST APIs; React.js was used to design the frontend dashboard for analysts.

- Database: PostgreSQL was implemented to store alerts, model versions, and feedback data, connected to Flask through SQLAlchemy.
- Libraries: scikit-learn, pandas, numpy, matplotlib, and joblib were used for modeling, data handling, visualization, and model serialization.
- Deployment: The complete system was containerized using Docker Compose, with separate containers for the frontend, backend, and database to ensure scalability and isolation.

This setup allowed the system to be portable, modular, and easy to update without affecting live operations.

III-K Dataset Preparation

- The NSL-KDD dataset was selected as the main dataset since it improves on the older KDD Cup '99 dataset by removing redundant and duplicate entries.
- The dataset includes 41 features representing various aspects of network traffic, such as duration, service, protocol, and byte counts.

- Data preprocessing steps included:

1 Data Cleaning: Duplicate and incomplete records were removed.
2 Encoding: Categorical features (flag) were converted into numeric format using one-hot encoding.
3 Normalization: Continuous attributes were standardized using z-score normalization to ensure uniform scaling.
4 Label Grouping: All attacks were grouped into five major categories — DoS/DDoS, Probe, R2L, U2R, and Normal.
5 Data Splitting: The cleaned data was divided into 80 % training and 20 % testing subsets for performance evaluation.

This preprocessing ensured consistent and balanced input for all learning models.

III-L Model Implementation

SMART NIDS integrates three main models; each designed for a specific role within the hybrid detection process.

1) Random Forest Classifier (Supervised Model)

- Acts as the primary detection model for identifying known attack types.
- Trained using the labeled portion of the NSL-KDD dataset.
- Provides a probability distribution for each attack class.
- A predefined confidence threshold (ANON_THRESHOLD = 0.60)

determines the model's certainty:

- If confidence $\geq 0.60 \rightarrow$ classified as known attack.
- If confidence $< 0.60 \rightarrow$ sample passed to Isolation Forest for further analysis.

2) Isolation Forest (Unsupervised Model)

- Used for detecting anomalies and unknown attack patterns that do not resemble normal traffic.
- Assigns an anomaly score to each uncertain sample.
- Samples with high anomaly scores are sent to the Anonymous pool for review and clustering.

3) DBSCAN Clustering

- Performs density-based grouping on the Anonymous samples.
- Helps identify new or similar behaviors among uncertain instances.
- Enables human analysts to review clustered data efficiently instead of analyzing each sample individually.
- Together, these three models form the hybrid detection engine of SMART NIDS.

III-M Hybrid Detection Workflow

The operational flow of the detection system proceeds as follows:

1) Incoming Data: Network traffic data is received and preprocessed.
2) Classification: Random Forest predicts the class label and confidence score.
3) Uncertainty Check: If confidence ≥ 0.60, the sample is labeled as a known attack or normal.
4) Anomaly Detection: If confidence < 0.60, the sample is passed to Isolation Forest for anomaly scoring.
5) Anonymous Queue: Samples identified as anomalies are moved to the Anonymous pool.
6) Clustering: DBSCAN clusters these samples based on behavioral similarity.
7) Analyst Review: Clusters are displayed on the

dashboard for expert inspection and labeling.

8) Retraining: Validated samples are appended to the dataset for retraining and version update.

This structured workflow allows SMART NIDS to manage both known and novel threats dynamically.

III-N Human-in-the-Loop Integration

- The React-based analyst dashboard provides visual access to detection results and system confidence levels.
- Analysts can view, filter, and investigate alerts, low- confidence predictions, and DBSCAN clusters.
- Each analyst can label unknown samples, approve or reject predictions, and send feedback to the backend.
- The Flask API manages these interactions and stores the validated labels in the database.
- This integration ensures that human expertise complements machine intelligence, creating a continuous learning loop.

III-O Model Retraining and Version Management

- After analyst feedback, new labels are added to the training set. Both Random Forest and Isolation Forest models are retrained automatically on the updated dataset.
- Performance metrics such as accuracy, precision, recall, and F1-score are computed on a validation subset.
- If the retrained model outperforms the previous version, it is promoted as the active model and saved as a new version (e.g., rf_v2. joblib, if_v2. joblib).

Previous models remain stored for rollback or auditing, ensuring model transparency and reproducibility.

This continuous retraining cycle enables SMART NIDS to evolve and maintain relevance against emerging attacks.

III-P Visualization and Performance Monitoring

- Feature Importance Graph: Displays the top network features contributing to detection decisions (e.g., src_bytes, dst_bytes, count, service_http).
- Confusion Matrix: Illustrates class-wise detection accuracy and misclassification rates.
- Precision-Recall Curve: Shows trade-off between sensitivity and reliability.
- Analyst Dashboard Visuals: Include cluster summaries, model metrics, and historical performance trends.

These visualizations help analysts understand model behavior and provide intuitive insight into system decisions.

III-Q System Deployment

- The final system was deployed using Docker Compose, running three coordinated containers:
- Frontend Container: Hosts the React interface.
- Backend Container: Hosts the Flask API and models.
- Database Container: Runs PostgreSQL for storage and metadata.

This modular deployment allows independent updates and horizontal scaling. The system can be easily extended to handle live network traffic or integrate with other monitoring tools in future work.

VI. RESULTS AND ANALYSIS

The SMART NIDS model was tested on the NSL-KDD dataset to assess its detection performance and adaptability in recognizing known and unknown network attacks.This hybrid approach, which consists of Random Forest, Isolation Forest, and DBSCAN, was compared with a baseline Random Forest classifier. Results indicated that SMART NIDS achieved a 95.4% detection accuracy with precision of 0.94, recall of 0.95, and F1- score of 0.95; thus, outperforming the baseline model that had 93.2% accuracy and an F1- score of 0.91. The ROC-AUC value for SMART NIDS is improved from 0.95 for the baseline to 0.97 hence indicating excellent discrimination between normal and malicious traffic as well as reducing the False Positive Rate (FPR) from 0.07 to 0.04 confirming improved reliability by reducing misclassifications in the system However, from confusion matrix plus precision-recall analysis detection was consistent plus balanced across all classes more so DoS plus Probe attacks while minority classes such as R2L plus U2R improved.

Network attributes such as src_bytes, dst_bytes, count, and service_http showed feature importance further analysis to the highest contribution for accurate predictions thus validating their influence on network intrusion patterns. The DBSCAN clustering stage was very crucial in grouping uncertain samples since it could assist an analyst in identifying new or rare attack behaviors that were not part of the initial training data. After feedback and retraining, recall for minority classes improved from 0.81 to 0.85 which proved the effectiveness of the human-in-the- loop mechanism. The graphical evaluations including ROC and Precision–Recall curves also further highlighted the stability and robustness of SMART NIDS in all detection scenarios. In conclusion, results have proved that this proposed system can achieve high accuracy while detecting known attacks and more so adapt dynamically to new threats, thus providing a reliable and intelligent solution for intrusion detection within real-world network environments.

VII. CONCLUSION

The SMART NIDS is a significant milestone in intelligent network intrusion detection. It integrates supervised and unsupervised learning with human expertise to create an adaptive defense framework, moving beyond static traditional models. Results demonstrate that a hybrid approach improves detection accuracy while building trust through interpretability and expert feedback. Algorithmic precision combined with human judgment allows SMART NIDS to adapt to evolving network behaviors, addressing one of the most persistent challenges in cybersecurity: novel attack adaptability. SMART NIDS will be used as proof-of- concept for human-in-the-loop machine learning as a sustainable approach to real- world security systems. Every interaction with an analyst will improve performance in the future; it is intelligent and resilient at the same time! The modular architecture plus version-controlled retraining provides a solid base for next-gen IDSs that can grow, adapt, and respond in real time to new cyber threats.

REFERENCES

[1] Tafreshian, B., & Zhang, S. (2025). A Defensive Framework Against Adversarial Attacks on Machine Learning- Based Network Intrusion Detection Systems.

[2] Fuhrman, S., Gungor, O., & Rosing, T. (2025).CND-IDS: Continual Novelty Detection for Intrusion Detection Systems.

[3] Li, E., Shang, Z., Gungor, O., & Rosing, T. (2025).SAFE: Self-Supervised Anomaly Detection Framework for Intrusion Detection.

[4] Zhao, X., Fok, K. W., & Thing, V. L. L. (2024).Enhancing Network Intrusion Detection Performance using Generative Adversarial Networks.

[5] Gelenbe, E., & Nakip, M. (2024).Online Self-Supervised Deep Learning for Intrusion Detection systems.IEEE Transactions on Information Forensics and Security, 19, 5668–5683.

[6] Liu, Y., Wang, X., & Chen, Z. (2024).Constraining Adversarial Attacks on Network Intrusion Detection Systems: Transferability and Defense Analysis. IEEE Transactions on Network and Service Management, 21(3), 2751–2772.

[7] Amara Korba, A., Sebaa, S., Mabrouki, M., Ghamri-Doudane, Y., & Benatchba, K. (2024).A Life-long Learning Intrusion Detection System for 6G-Enabled IoV.