RESEARCH ARTICLE OPEN ACCESS

Hybrid Intrusion Detection System with Flask Framework

Ms. Veda Swetha S*, Mr. Marimuthu R**

*(Department of Computer Science and Engineering, Dr. M.G.R. Educational and Research Institute, Chennai, India)

** (Assistant Professor, Center Of Excellence in Digital Forensics, Chennai, India)

ABSTRACT

The increasing intricacy of network traffic and the rising frequency of cyberattacks, ensuring effective network security has become a formidable task. Intrusion Detection Systems (IDS) are essential tools for spotting and responding to suspicious activity, yet many traditional IDS approaches fall short when it comes to identifying novel or advanced threats. This study aims to overcome those limitations—specifically, the tendency for high false alarm rates and poor adaptability in dynamic environments. The research proposes a hybrid IDS approach that harnesses the strengths of machine learning algorithms to boost both detection accuracy and real-time responsiveness. By combining Random Forest, Support Vector Machine (SVM), and Neural Network models, and deploying them within a Flask-based web interface, the system offers both analytical power and practical usability. Trained and evaluated using the NSL-KDD dataset, the model undergoes thorough preprocessing and feature selection to enhance performance. The final system not only achieves high accuracy across multiple attack categories but also delivers an intuitive, real-time dashboard for actionable network monitoring.

Keywords:- Intrusion Detection System, Machine Learning, Random Forest, Support Vector Machine, Flask Application, Cybersecurity, Real-Time Threat Detection.

I. INTRODUCTION

The rapid growth of digital technologies has made network security a critical concern. Traditional Intrusion Detection Systems (IDS) are struggling to cope with the complexity of modern cyberattacks. Hybrid Intrusion Detection Systems (HIDS) that integrate multiple machine learning techniques have shown potential to improve detection accuracy [1]. These systems learn from data patterns and can adapt to new attack types, offering a promising solution for modern network security [2].

Hybrid IDS are important because they combine various machine learning models to enhance security by reducing false positives and improving detection rates. As cyber threats evolve, a hybrid approach can adjust to new attack patterns, ensuring more accurate and reliable detection. Integrating such systems with frameworks like Flask makes them scalable and efficient for real-time applications [3].

This research focuses on developing a hybrid IDS using Flask combined with machine learning techniques. It involves analyzing existing hybrid models, selecting the best machine learning algorithms, and building a Flask-based application. The performance of the proposed system will be evaluated using standard datasets to assess its accuracy and efficiency [4].

Various hybrid intrusion detection systems have been developed, such as those by Meryem and El Ouahidi (2021), which combine different learning methods to improve accuracy [5]. Bhagya (2020) also developed a system using Flask and Random Forest for real-time intrusion detection [3]. However, challenges like scalability, false positives, and adaptation to new attacks remain, and this paper aims to address these limitations.

II. LITERATURE REVIEW

Rasmitha et al. [5] introduce a Hybrid Intrusion Detection System (HIDS) developed using the Flask framework, which integrates various machine learning techniques to strengthen the detection capabilities for network intrusions. The model utilizes a combination of classifiers to provide real-time protection. It demonstrates high accuracy and low latency when detecting malicious activities, showcasing its suitability for real-world deployment in dynamic environments. This hybrid approach aims to overcome the limitations of single-classifier models, especially in evolving network conditions.

Karthik [6] presents a Flask-based Network Intrusion Detection System (NIDS) that uses the KDDCUP'99 dataset for both training and evaluation. Built on the Flask framework, the system ensures a lightweight deployment for real-time applications. Karthik's method focuses on machine learning algorithms like Random Forest and Support Vector Machines (SVM) to detect both known and novel intrusions in network traffic. The outcomes emphasize the system's ability to achieve high detection accuracy, particularly in identifying Distributed Denial-of-Service (DDoS) attacks.

Abhiram et al. [7] examine the integration of Machine Learning (ML) techniques with Flask for creating a highly effective Intrusion Detection System. Their approach employs classification algorithms such as Decision Trees and Naive Bayes to identify network intrusions. The combination of Flask and machine learning techniques enables the system to provide real-time predictions and alerts, making it well-suited for practical applications. The approach is evaluated on standard datasets, demonstrating impressive results in both detection accuracy and system efficiency.

Wang [8] investigates an Edge-Based Hybrid Intrusion Detection Framework for Mobile Edge Computing, which merges edge computing with machine learning algorithms to enhance intrusion detection at the network edge. This approach addresses the increasing demand for scalable intrusion detection systems in Mobile Edge Networks (MENs). Wang's framework combines both supervised and unsupervised machine learning models, ensuring efficient detection of various cyberattacks in mobile environments while maintaining accuracy and low latency in detection.

Abdallah et al. [9] propose a Hybrid Intrusion Detection System (HIDS) that blends both supervised and unsupervised learning techniques for identifying network intrusions. Their system integrates k-means clustering and Support Vector Machines (SVM) to enhance detection accuracy and minimize false positives. When tested on standard benchmark datasets, the system outperforms traditional models, particularly in detecting complex and novel attacks.

III. PROPOSED METHODOLOGY:

This research proposes a hybrid Intrusion Detection System (IDS) that uses Random Forest and Linear Regression to classify various network attacks, including DoS, Probe, R2L, and U2R [2], [11]. To address data imbalance issues, the system will utilize techniques like oversampling and synthetic data generation for fair class representation [10].

The models will be trained on labeled datasets such as KDDCup'99 and integrated into a Flask-based web application for real-time monitoring and visualization [7], [4]. Administrators will access detection results through a user-friendly dashboard to ensure timely responses [5].

Based on recent studies, hybrid systems are more effective in adapting to evolving threats [1], [3]. This design ensures scalability, accuracy, and adaptability, offering a robust security solution.

By leveraging the advantages of different detection methods, hybrid models are able to more accurately identify complex and sophisticated attack patterns. They also help in minimizing false positive rates, which enhances the overall dependability of the system.

Fig 3.3: Hybrid Intrusion Detection System Architecture with Flask

MODULE DESCRIPTION:

The proposed Hybrid IDS consists of several interconnected modules, each carefully designed to ensure accurate detection, efficient monitoring, and effective threat management. Every module plays a vital role in the data processing pipeline, contributing to the system's overall effectiveness and reliability.

- Data Collection Module
- Data Preprocessing Module
- Feature Engineering Module
- Data Balancing Module
- Model Building Module
- Model Training Module
- Evaluation Module
- Flask Web Application Module
- Visualization Module
- Real-Time Monitoring Module

Data Collection: This module is responsible for systematically collecting raw data from various sources, such as network logs, sensors, databases, APIs, and external monitoring tools. The gathered data forms the foundation for comprehensive analysis, feature extraction, and model training. Sources can include packet captures, server logs, firewall logs, and external threat intelligence feeds.

Data Preprocessing: This module focuses on cleaning and preparing the collected data by handling missing values, correcting errors, normalizing feature scales, and removing outliers. Proper preprocessing improves data quality and ensures that subsequent analytical processes work with reliable and consistent datasets. Noise reduction, duplicate removal, and format conversion are also carried out to standardize the inputs.

Feature Engineering Module: In this module, relevant features are extracted, selected, and transformed from the preprocessed data. The goal is to identify significant attributes, create new features from existing ones, handle categorical variables, and scale numeric features to optimize the model's ability to learn patterns and detect anomalies accurately. Techniques like PCA and feature selection algorithms may be used to refine the dataset.

Data Balancing: This module addresses the issue of imbalanced class distribution in the dataset by using techniques like oversampling of minority classes, under sampling of majority classes, and synthetic data generation methods such as SMOTE. This ensures a fair representation of all attack types, preventing model bias and enhancing detection accuracy. Balanced datasets help ensure that rare but critical attack instances are not overlooked.

Model Building Module: This module involves selecting and implementing machine learning algorithms, specifically Random Forest and Linear Regression, to build predictive models capable of accurately classifying network traffic as either normal or malicious. Hyperparameters are fine-tuned to optimize model performance, and ensemble techniques may be used to combine predictions for better accuracy.

Model Training Module: Using labeled datasets, this module trains the predictive models, adjusting model parameters to maximize detection accuracy. Extensive training ensures that the models can generalize well to unseen data and detect complex attack patterns. Techniques such as cross-validation and early stopping are applied to avoid overfitting.

Evaluation Module: This module evaluates the trained models by assessing their performance against labeled datasets. Metrics like accuracy, precision, recall, and F1-score are used to evaluate how well the models can detect various attack types.

This module ensures that the models are reliable and robust for real-world deployment.

Flask Web Application Module: This module focuses on developing an interface and user-friendly web interface using the Flask framework. It enables administrators to access real-time IDS insights, manage alerts, and control system configurations conveniently.

Visualization Module: This module is responsible for generating dynamic and informative visual representations of the IDS results, including charts, graphs, and dashboards. It presents detected attack types, model performance metrics, and network traffic patterns in an easy-to-understand format. Visualization improves situational awareness and aids in informed decision-making.

Real-Time Monitoring Module: This module enables continuous, live monitoring of network traffic, feeding data directly to the IDS models for immediate analysis and detection. It allows the system to respond quickly to emerging threats, ensuring proactive network security management. Automated alerting and logging mechanisms are integrated to assist administrators in making timely interventions.

Machine Learning Models Overview:

Random Forest: Random Forest is a powerful ensemble technique that builds several decision trees and uses them collectively to categorize network behavior. It combines the results from individual trees to improve classification accuracy, increase robustness against overfitting, and handle large datasets with high-dimensional feature spaces. The feature importance scores provided by Random Forest help identify which attributes are most influential in predictions.

Linear Regression: While Linear Regression is primarily used for regression tasks, it is applied in this IDS to identify patterns and relationships within network traffic data. Although Logistic Regression or other specialized classifiers are generally more effective for classification tasks in intrusion detection, Linear Regression serves as a baseline for comparative analysis and highlights linear patterns within the data.

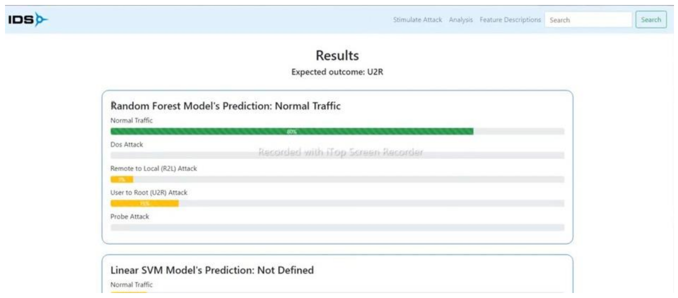
IV. FINDINGS:

In this project, a web-based Intrusion Detection System (IDS) was created using the Flask web framework. The system was built and evaluated using the NSL-KDD dataset, with the goal of detecting a variety of cyberattack types, including Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R) attacks.

To ensure smooth execution, the application follows a clearly defined setup process through the Command Prompt. The user begins by navigating to the project folder, switching to the correct drive, activating a designated Conda environment, defining the Flask application environment variable, and finally launching the Flask server. This organized approach helps maintain a consistent and reliable method for running the real-time web interface.

Fig 4.1: Running intrusion detection system with flask

As part of the evaluation, a U2R attack was simulated using specific input features such as duration, protocol_type, service, flag, src_bytes, dst_bytes, logged_in, wrong_fragment, same_dst_host_count, and same_srv_count.



Results

Expected outcome: U2R

Random Forest Model's Prediction: Normal Traffic

Normal Traffic

Dos Attack

Remote to Local (R2L) Attack

User to Root (U2R) Attack

Probe Attack

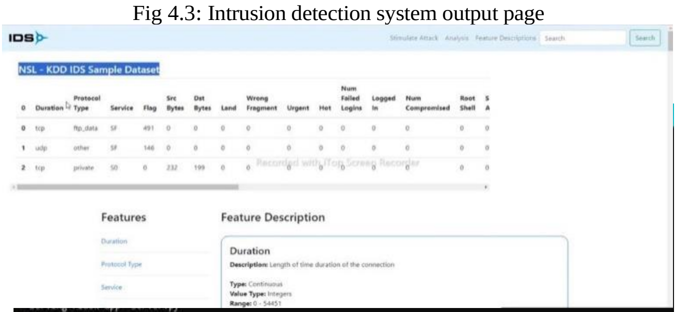
Linear SVM Model's Prediction: Not Defined

Normal Traffic

Fig 4.2: Intrusion detection system input page

When this data was submitted for prediction, the Random Forest model correctly labeled the traffic as "Normal", while the Linear SVM misclassified it, indicating a "DoS Attack".

Fig 4.3: Intrusion detection system output page



NSL - KDD IDS Sample Dataset

0	Duration	Protocol Type	Service	Flag	Src Bytes	Dst Bytes	Land	Wrong Fragment	Urgent	Hot	Num Failed Logins	Logged In	Num Compromised	Root Shell	S A
0	tcp	ftp_data	SF	491	0	0	0	0	0	0	0	0	0	0	0
1	udp	other	SF	146	0	0	0	0	0	0	0	0	0	0	0
2	tcp	private	S0	0	232	199	0	0	0	0	0	0	0	0	0

Features

Duration

Protocol Type

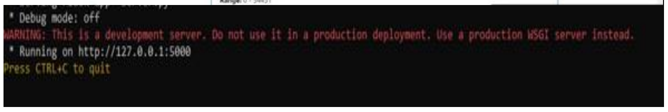
Service

Feature Description

Duration

Description: Length of time duration of the connection

Type: Continuous
Value Type: Integers
Range: 0 - 54451

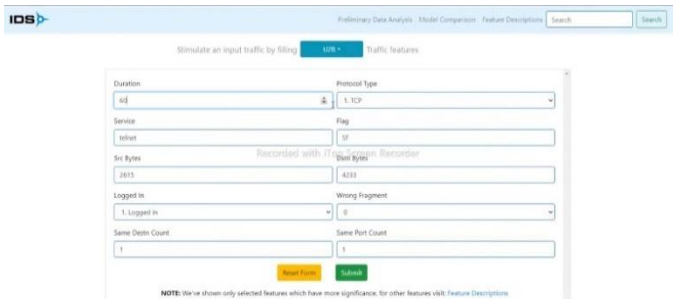


```
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Figure 4.4 illustrates a table presenting key network features like Protocol Type, Service, and Duration. Beneath the table, a descriptive section explains each feature in detail—including its name, data type, value type, and typical range—providing clarity on the dataset attributes.

Fig 4.4: NSL – KDD IDS sample dataset and features description

The results from these experiments confirm the efficiency of hybrid machine learning models in identifying



Stimulate an input traffic by filling U2R Traffic features

Duration: 60	Protocol Type: 1. TCP
Service: telnet	Flag: SF
Src Bytes: 2615	Dstn Bytes: 4233
Logged In: 1. Logged in	Wrong Fragment: 0
Same Destn Count: 1	Same Port Count: 1

NOTE: We've shown only selected features which have more significance, for other features visit: Feature Descriptions

both common and previously unseen attacks. The integration of the NSL-KDD and CIC-IDS2017 datasets has contributed significantly to the system's reliability across various threat types. Additionally, the Flask-powered interface enables users to monitor and analyze network traffic in real time with ease.

V. CONCLUSION:

In conclusion, this project developed an Intrusion Detection System (IDS) using Random Forest and Linear Regression algorithms to classify network attacks, including DoS, Probe, R2L, and U2R. The system addressed dataset imbalances and provided real-time monitoring via a Flask web interface, effectively reducing false positives and improving accuracy.

The IDS successfully identified both known and novel attack patterns, demonstrating the effectiveness of hybrid machine learning models, as noted by Maseno et al. [1] and Rababah [2]. Real-time monitoring, as explored by Bhagya [4], Talha [5], and Rasmitha [6], enabled prompt responses to emerging threats.

The combination of machine learning and Flask proves to be a reliable solution for network security. Future improvements, incorporating feedback and data updates, will ensure the IDS remains effective against evolving threats, with potential enhancements in file integrity monitoring, as suggested by Sulaiman & Hilmi [14] and Sal.

VI. REFERENCE

[1]. Maseno, E. M., Wang, Z., & Xing, H. (2022). A Systematic Review on Hybrid Intrusion Detection Systems. https://www.hindawi.com/journals/scn/2022/9663052

[2]. Rababah, B. (2020). Hybrid Model for Intrusion Detection Systems Using Machine Learning Techniques. http://arxiv.org/abs/2003.08585

[3]. Meryem, A., & El Ouahidi, B. (2021). Hybrid Intrusion Detection System Using Machine Learning. https://www.magonlinelibrary.com/doi/abs/10.1016/S1353-4858(20)30056-8

[4]. Bhagya, - (2023). Network Intrusion Detection Using Flask and Random Forest. https://github.com/Bhagya-06/Network-Intrusion-Detection

[5]. Talha, N. (2023). Machine Learning Model for Network Intrusion Detection System Using Flask. https://github.com/Talha-Nazir13/Machine-Learning-Model-for-Network-Intrusion-detection-System-using-Flask

[6]. Rasmitha, - (2023). Hybrid Intrusion Detection System Using Flask. https://github.com/Rasmitha05/Project-HIDS

[7]. Karthik, - (2023). Flask-Based NIDS Web Application Using KDDCUP'99 Dataset. https://github.com/karthik003/nids-flask-webapp

[8]. Abhiram, D. (2023). Intrusion Detection System Using ML and Flask. https://github.com/abhiramdvs/Intrusion-Detection-System-using-ML-Flask

[9]. Wang, H. (2021). An Edge-Based Hybrid Intrusion Detection Framework for Mobile Edge Computing. https://link.springer.com/article/10.1007/s40747-021-00498-4

[10]. Abdallah, A. (2020). A Hybrid Intrusion Detection System Using Supervised and Unsupervised Learning. https://www.sciencedirect.com/science/article/pii/S1877050920304091

[11]. Zhang, Y. (2020). Hybrid Intrusion Detection System Based on Machine Learning. https://www.mdpi.com/2079-9292/9/1/173

[12]. Yousef, R. (2022). NetGuard: A Random Forest Approach to Intrusion Detection Using Flask. https://ijsart.com/netguard-a-random-forest-approach-to-network-intrusion-detection-using-flask-89686

[13]. Alotaibi, E. (2021). Hybrid Intrusion Detection Framework for Cloud Networks. https://link.springer.com/article/10.1007/s10723-021-09552-9

[14]. Sulaiman, N. S., & Hilmi, M. A. (2024). Avoiding Data Loss and Corruption Towards File Integrity Monitoring. https://tatiuc.edu.my/ijset/index.php/ijset/article/view/222

[15]. Salman, A. (2022). File Integrity Checkers: Functionality, Attacks, and Protection. https://www.researchgate.net/publication/361179928