Machine Learning Report

Final Project

Tri-Layer Intrusion Detection System (TLIDS): Advanced Anomaly Detection with a CNN-LSTM Cascade

Felix Delali Adigbli
Emmanuel Chibua
College of Engineering
Northeastern University
Toronto, ON
Adigbli.f@northeastern.edu
Chibua.e@northeastern.edu

Abstract

The continuous digitization of our modern society presents substantial hurdles for intrusion detection systems (IDS), which were once considered an effective example of a security monitoring system. The quantity of data that must be monitored is increasing beyond what can be handled by a single system, and the diversity and quantity of linked devices are contributing to the constant appearance of new dangers that are missed by current systems. There is a need for scalable intrusion detection system (IDS) necessary to identify unknown, zero-day attack.

In this final project, we extended our assignment four project Hybrid Neural Network for Intrusion Detection: Unveiling Anomalies in Network Traffic with a CNN-LSTM Fusion by introducing two new stages in the anomaly detection process. The project is dubbed **Tri-Layer Intrusion Detection System (TLIDS): Advanced Anomaly Detection with a CNN-LSTM Cascade**.

1 Introduction

Due to the constant digitization of the modern world, our world today is constantly exposed to an increased risk of cybersecurity threats. Not only does the increasing quantity and diversity of linked devices increase the attack surface for malicious actors, but it also has an adverse effect on the potential outcomes if an attack is successful. The increasing generated load

on existing security monitoring systems is exceeding single system capabilities and challenging their scalability to detect threats in near real-time. Traditional security mechanisms such as a firewall are effective at detecting specific types of attacks but are unable to detect unknown or more advanced attacks. Intrusion detection systems (IDS) are often deployed as a second line of defense and are an example of a security monitoring system capable of detecting known as well as unknown and more sophisticated attacks. Currently proposed IDS solutions often rely on a single machine learning model for either attack detection or classification. In this our final project, hierarchical intrusion detection system was developed.

1.1 Objective

The primary aims of this project are:

1. **Develop machine learning model** for hierarchical intrusion detection performing both binary and multiclass detection for detecting known and zero-day attacks.
2. **Model Deployment**: Deploy the model as service for used by external systems using Docker.
3. **Develop Web interface** for simulating attack requests to the deployed model.
4. **Develop Dashboard** for the model's predictions based on the simulated attacks.

2 Implementation

2.1 Dataset

We are using Canadian Institute for Cybersecurity (CIC) Intrusion detection evaluation dataset (CIC-IDS2017) in this project. The dataset can be found here. CICIDS2017 dataset contains benign and the most up-to-date common attacks, which resembles the true real-world data. It also includes the results of the network traffic analysis using CICFlowMeter with labeled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols and attack.

2.2 Exploratory Analysis

2.2.1 Distribution of Anomaly in the Dataset

An initial exploratory analysis of the dataset was conducted to understand the distribution of anomaly. The figure below illustrates the anomaly distribution:

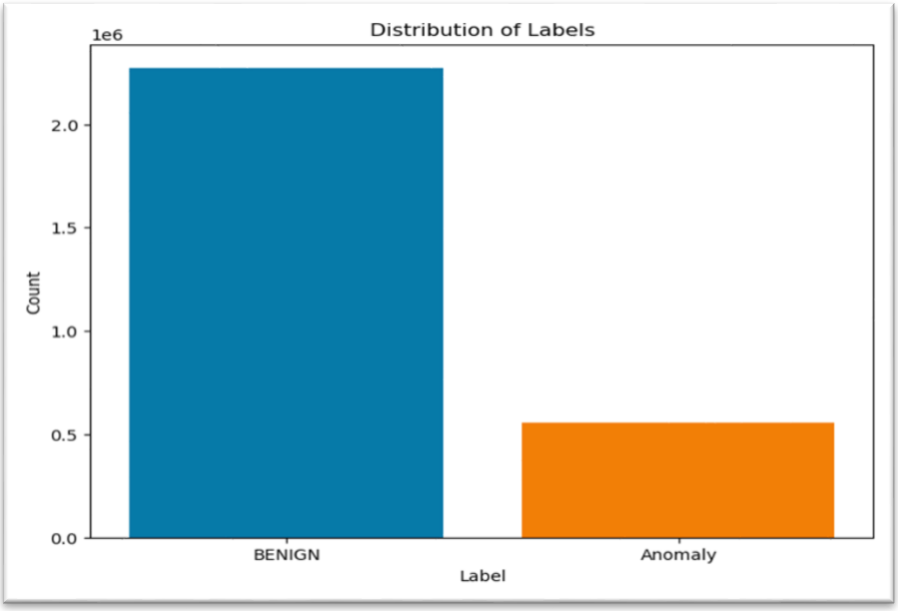


Fig: Distribution of Anomaly

This visualization presents a clear disparity in the attack labels, serving as a foundational understanding of the dataset's composition before delving into more complex analysis.

Correlation heatmap of the dataset and pair plot:

The graphs show some features are highly correlated. Many of the pair plots show a strong positive linear relationship indicating that many of the variables in the dataset are positively correlated with each other. There are also some potential outliers. Any points that fall far away from the main cloud of points could be considered anomalies or outliers. In the plots where we see a clear positive trend, any points that do not follow this trend might be outliers.

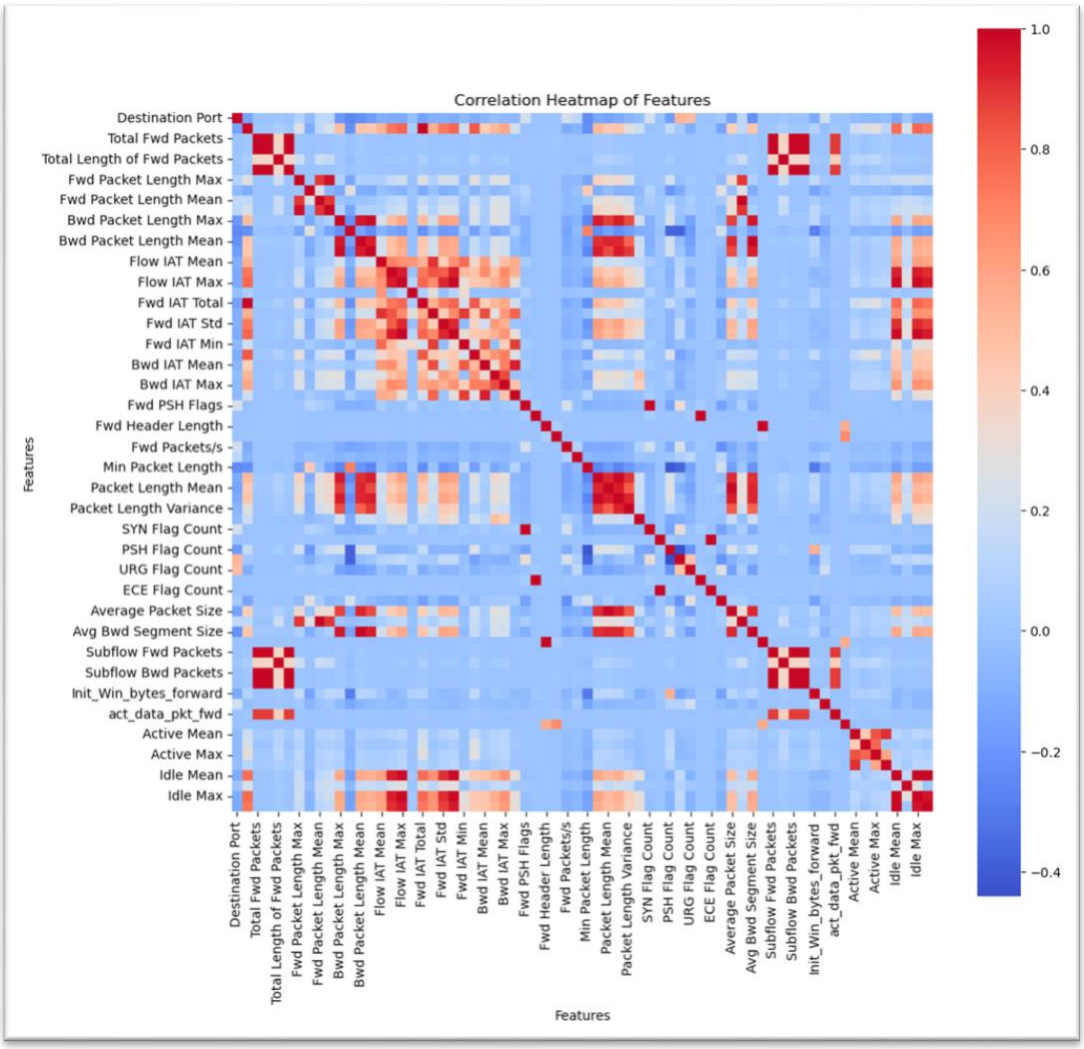


Fig: Correlation heatmap of features

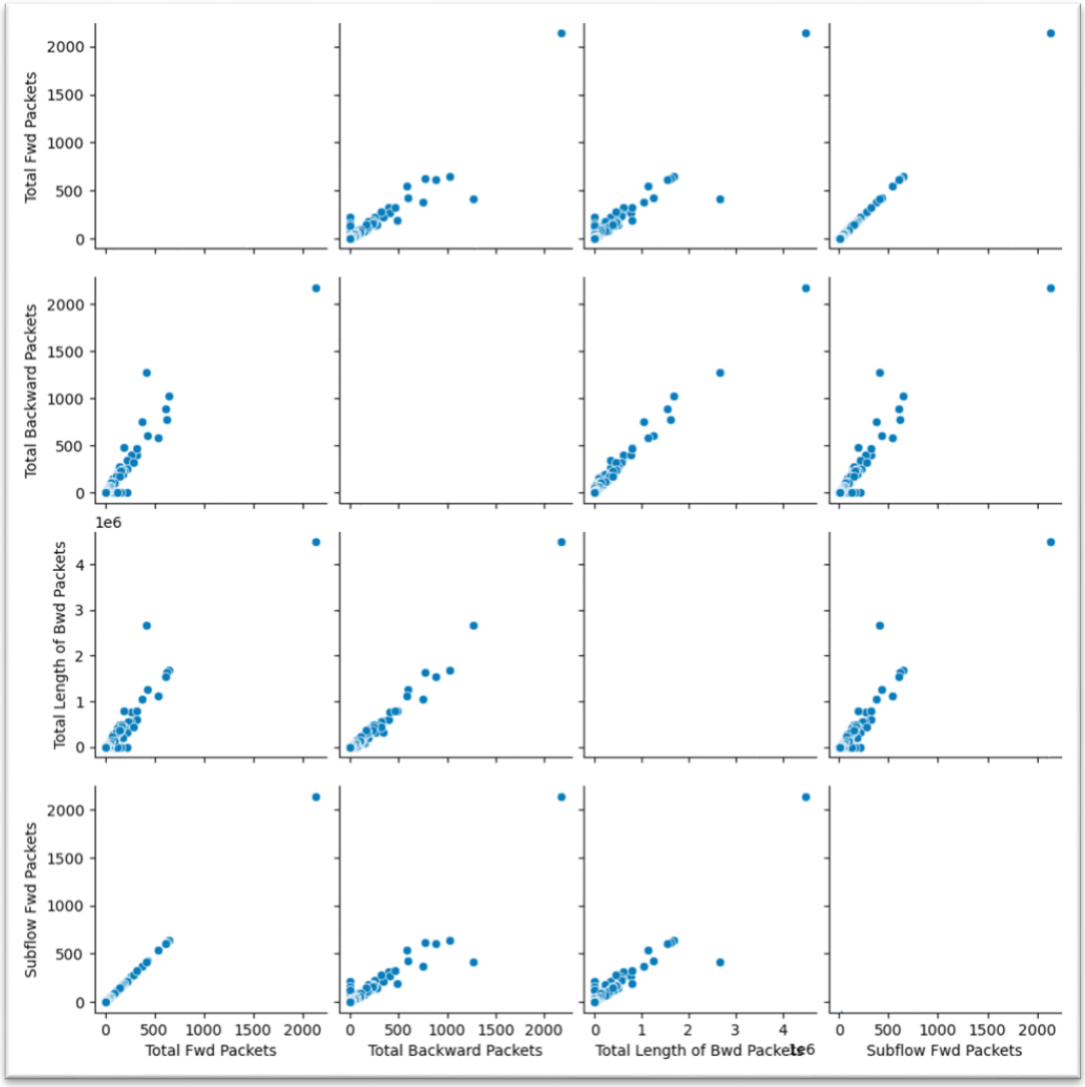


Fig. Pair plot of some feature.

2. Methodology

2.1 Data Handling:

The dataset used in this project comprises network traffic data, which includes features such as packet sizes, transmission rates, protocol types, and flags indicating whether the traffic is normal or anomalous. The data handling process involved several steps to ensure that the dataset was clean, properly formatted, and suitable for training the hybrid neural network model.

Data Cleaning: The first step was to clean the data by removing any irrelevant features that do not contribute to the anomaly detection task. Features that were constant across the dataset were dropped, as they do not provide useful

information for distinguishing between normal and anomalous traffic. Features that has very large values or infinite values were dropped as well.

Handling Missing Values: The dataset was checked for missing values, and any instances with missing data were handled appropriately.

Ensuring Correct Data Types: Each feature in the dataset was examined to ensure that its data type was appropriate according to the feature description. Numerical features were represented as float or integer types, while categorical features were represented as object or category types.

Standardization: To ensure that all numerical features were on a similar scale, standardization was applied using the **StandardScaler** from scikit-learn. This scaler removes the mean and scales the features to unit variance, which is important for models that are sensitive to the scale of the input features, such as the CNN and LSTM components of the hybrid model.

Encoding: Categorical features were encoded using label encoding, which converts each category into a unique integer value. This is necessary for the model to process categorical data, as neural networks require numerical input.

Train-Test Split: After preprocessing, the dataset was split into training and testing sets to evaluate the model's performance on unseen data. A typical split ratio used was 70% for training and 30% for testing. The training set was used to train the model, while the testing set was used to assess its generalization ability and performance in detecting anomalies.

By following these data handling steps, the dataset was transformed into a clean, standardized, and encoded format suitable for training and evaluating the hybrid neural network model for intrusion detection.

2.1.3 Model Selection and Development

2.1.3.1 Architecture of Tri-Layer Intrusion Detection System

The 3-stage hierarchical architecture in the project consists of three stages with each a distinct characteristic and objective. The figure below represents the overall architecture of the system.

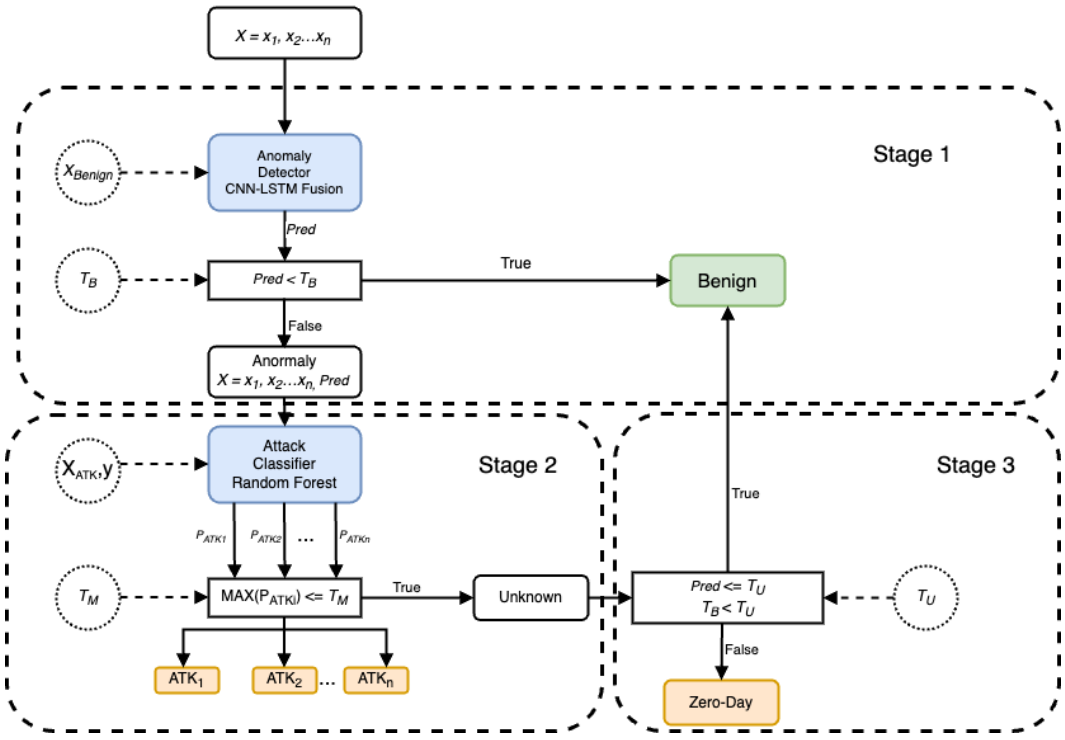


Figure: Tri-Layer Intrusion Detection System architecture

The vector X represent the input to the system and is sent on first stage. The first stage consists of an anomaly detector that outputs an anomaly score, *Pred*, specifically a high value indicates a high probability that X is not benign and thus an intrusion. If this anomaly score is lower than a threshold T_B, the model is confident enough to predict the sample as benign and no further processing is necessary. However, if the anomaly score is higher than T_B, the sample is forwarded to the second stage where an attack classifier predicts if the sample belongs to one of the known attack classes (ATK_i). If the attack classifier fails to match the sample to a known attack class with a certainty higher than a threshold T_M, the sample is sent to the third and last stage. The stage does not introduce another classifier but rather reuses the anomaly score *Pred* outputted by the Stage 1. In case the anomaly score is lower than the threshold T_U, the output of the first layer is corrected by eventually classifying the sample as benign, on the contrary, the sample is predicted as a zero-day attack if the anomaly score is higher than T_U. The main goal of the stage 3 is to reduce the number of false positives, namely benign traffic being classified as an attack while providing the ability to effectively detect zero-day attacks.

2.1.3.2 Algorithms

a. Anomaly Detection: Hybrid CNN-LSTM Fusion model

The first stage relies on an anomaly detector to filter the malicious samples from benign traffic. We used our CNN-LSTM Fusion model from assignment 4. The hybrid CNN-LSTM model represents a more advanced approach to

anomaly detection in network traffic, leveraging the strengths of both Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. With the spatial feature extraction capabilities of CNNs and the temporal pattern recognition strengths of LSTMs, the model aims to accurately identify normal and malicious activities.

b. Attack classifier: Random Forest Classifier

The second stage classifies forwarded suspicious samples from the first stage using a multiclass classifier to a known attack class. Supervised machine learning models are well suited to learn this representation from labeled data. for the stage 2, we used Random Forest Classifier.

The Random Forest classifier is an ensemble learning model used for both classification and regression tasks, which operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes (classification) of the individual trees. It is particularly effective for multiclass detection offering high accuracy and robustness, especially when dealing with complex and high-dimensional data.

2.1.3 Evaluation Metrics

Stage 1 and stage 2 Models were assessed based on accuracy, precision, and recall to ensure a comprehensive evaluation of their performance on anomaly detection and attack type classification. The model at each stage were evaluated independently.

3 Results

3.1 Model Performance Analysis

The overall performance of the system is measured by measuring the performance of the anomaly detector (stage 1) and the attack classifier (stage 2).

	Anomaly Detector	Attack Classifier
Accuracy	0.997588396	0.998374120
Precision	0.994432211	0.998231491
Recall	0.993327260	0.998374120

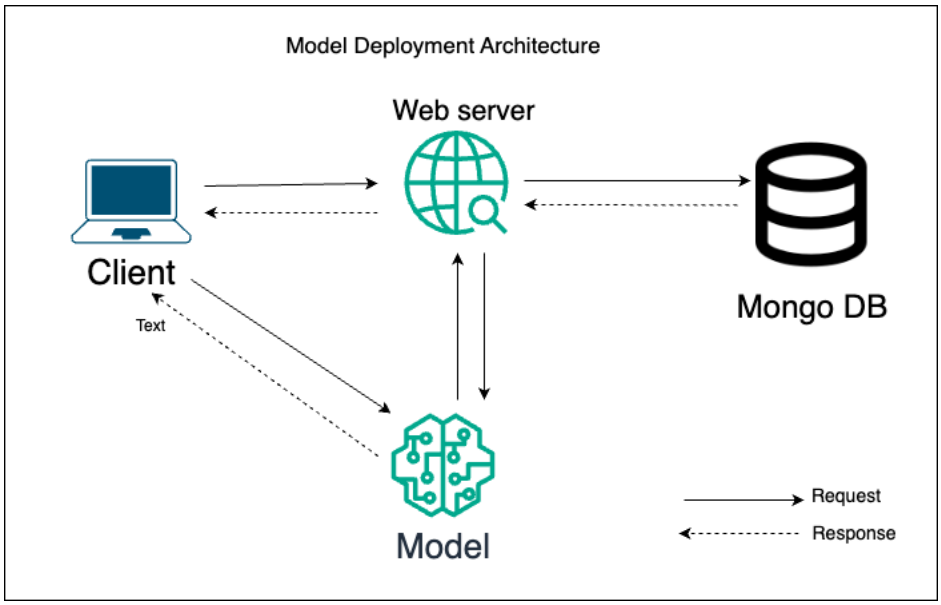
High precision and recall for both stage 1 and stage indicate that the system is exceptionally reliable in detecting true threats without over-flagging benign activities as threats. Such metrics signify the robustness of TLIDS in real-world scenarios, ensuring minimal disruption from false alarms while maintaining security integrity.

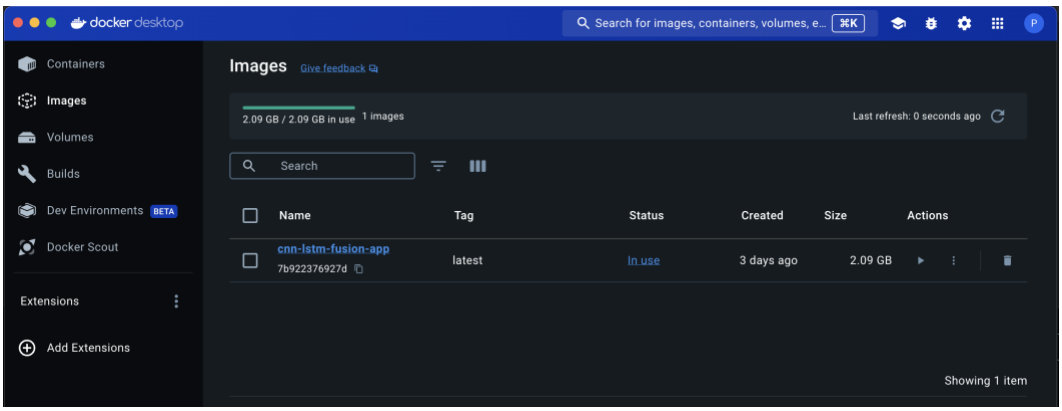
4 Model deployment.

The **Tri-Layer Intrusion Detection System (TLIDS)** model was deployed as a containerized service using Docker, with Flask serving a RESTful API for anomaly detection in network traffic. The general architecture of the model deployment is below.

4.1 Deployment Steps

1. **Container Creation**: Used **python:3.9-slim** to build a Docker image.
2. **Dependencies**: Installed necessary packages via **requirements.txt**.
3. **Model Integration**: Loaded serialized Keras (CNN-LSTM fusion) and scikit-learn model (Random Forest classifier) along with a scaler.
4. **API Provision**: Configured Flask to expose the **/predict** endpoint on port 8050.

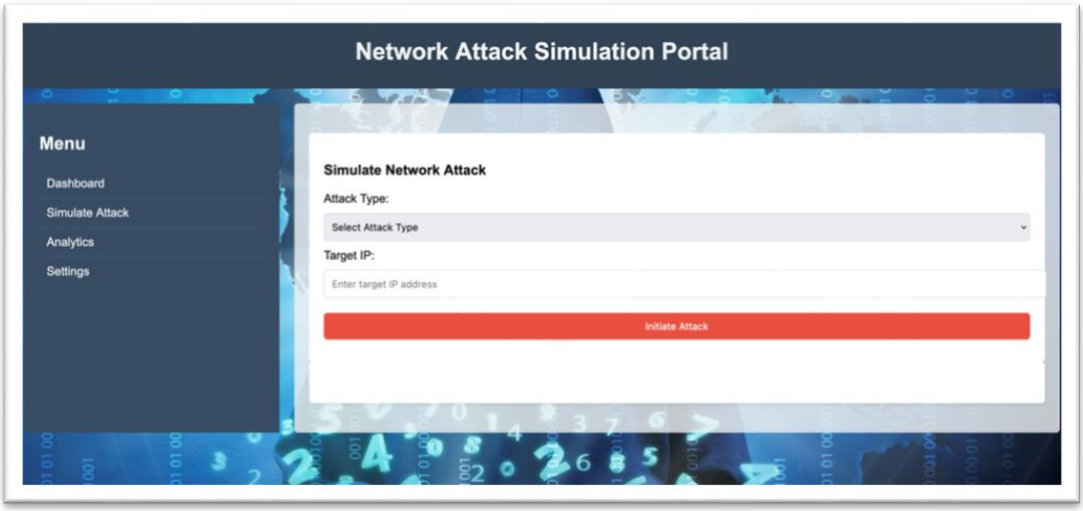




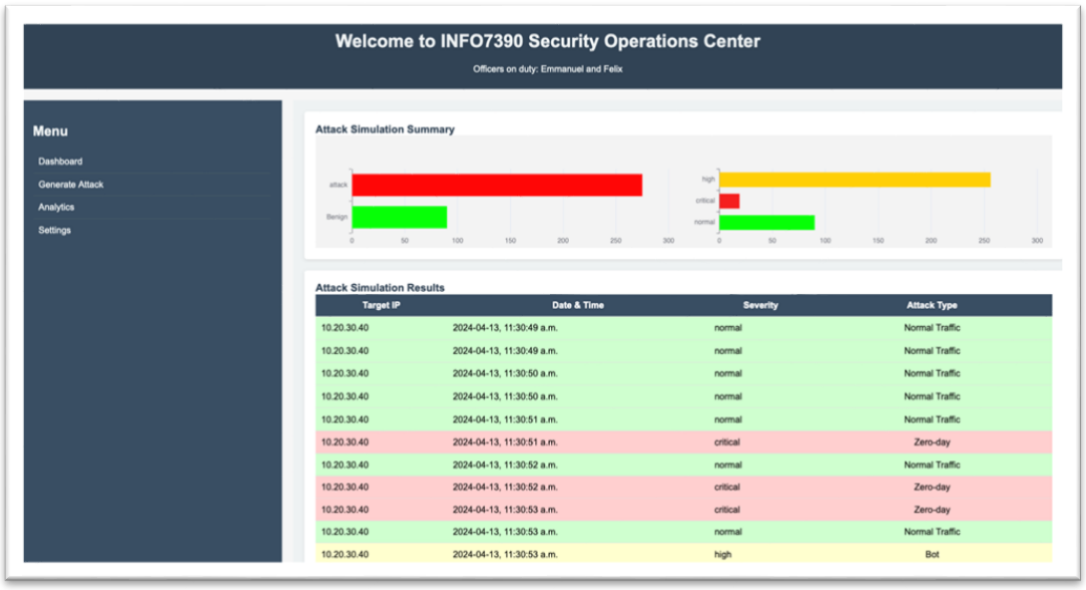
4.2 Model Operational Testing

As a critical component of the deployment, operational testing was conducted to validate the functionality and reliability of the TLIDS as follows:

1. **Web Portal Initialization**: A dedicated web portal was developed. This portal served as the interface for initiating tests, providing a user-friendly environment for test execution.



2. **Data Sampling**: The web application interfaced with a MongoDB database, automatically retrieving sample data that represents typical network traffic events. This step ensured that the model was tested with diverse data reflecting potential real-world scenarios.
3. **Feature Generation**: Upon extraction, the portal processed the data to create the appropriate feature set needed for model input, mirroring the preprocessing steps the model expects in a production environment.
4. **API Interaction**: With the features prepared, the web portal sent JSON-formatted requests to the TLIDS service's ‘’**/predict”** endpoint, simulating the way external systems would interact with the service.
5. **Response Handling and Storage**: Responses from the service, which included the classification results (benign, known attack, or zero-day attack), were automatically captured by the web portal and stored back in the MongoDB database for record-keeping and further analysis.
6. **Result Visualization**: To make sense of the results and to monitor the performance of the IDS service, the results are visualized on the dashboard. The dashboard provided real-time visualization of the results, offering insights into the distribution of detected anomalies and the frequency of various attack types. This visualization aided in identifying any patterns or irregularities in the classification results.



Conclusion

The "**Tri-Layer Intrusion Detection System (TLIDS): Advanced Anomaly Detection with a CNN-LSTM Cascade**" represents a significant advancement in network security technology. The implementation of a hierarchical machine learning model, backed by the analysis of the CIC-IDS2017 dataset, has resulted in a robust system capable of detecting known and zero-day attacks with high accuracy.

The deployment as a containerized service using Docker ensures flexibility, scalability, and ease of integration into existing infrastructures. The operational testing through a dedicated web portal and MongoDB interaction not only validates the functionality and reliability of the TLIDS but also demonstrates its capability to process real-world data effectively.

The integration of a real-time visualization dashboard enhances the system's function by providing immediate insights into the network's security status. This enables timely detection and response to potential threats, fortifying the network against a wide array of intrusion tactics.

In summary, the TLIDS project successfully achieves its objectives of developing and deploying a intrusion detection service. The high performance of the models across various metrics, including accuracy, precision, recall is indicative of the system's potential to significantly improve network security posture. Future work could focus on expanding the feature set, optimizing model parameters, and exploring real-time adaptive learning to maintain efficacy against the constantly evolving landscape of cyber threats.

Reference:

Choi, J. Lee, T. Kwon, K. Kim, Y. Choi and J. Song, "An Easy-to-use Framework to Build and Operate AI-based Intrusion Detection for In-situ Monitoring," *2021 16th Asia Joint Conference on Information Security (AsiaJCIS)*, Seoul, Korea, Republic of, 2021, pp. 1-8, doi: 10.1109/AsiaJCIS53848.2021.00011.

GETMAN A.I., GORYUNOV M.N., MATSKEVICH A.G., RYBOLOVLEV D.A., NIKOLSKAYA A.G. Deep Learning Applications for Intrusion Detection in Network Traffic. *Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS)*. 2023;35(4):65-92. (In Russ.) https://doi.org/10.15514/ISPRAS-2023-35(4)-3

GORYUNOV M.N., MATSKEVICH A.G., RYBOLOVLEV D.A. Synthesis of a Machine Learning Model for Detecting Computer Attacks Based on the CICIDS2017 Dataset. *Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS)*. 2020;32(5):81-94. (In Russ.) https://doi.org/10.15514/ISPRAS-2020-32(5)-6

M. Verkerken *et al.*, "A Novel Multi-Stage Approach for Hierarchical Intrusion Detection," in *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3915-3929, Sept. 2023, doi: 10.1109/TNSM.2023.3259474.

Sharafaldin, A. H. Lashkari, S. Hakak and A. A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," *2019 International Carnahan Conference on Security Technology (ICCST)*, Chennai, India, 2019, pp. 1-8, doi: 10.1109/CCST.2019.8888419.