

Kafka-based Intrusion Detection System

Chandan M S¹, Soujanya B K², Suman S³, Mohammad Fawaz Beejady⁴ and Roshan B S⁵

¹⁻⁵School of CSE, REVA University, Bengaluru, India

chandanms1312@gmail.com, sbk923@gmail.com, sgrsuman90@gmail.com, mohammadfawazbeejady@gmail.com, roshanmaari15@gmail.com

Abstract—The growing complexity and sophistication of cyber-attacks necessitate the development of robust intrusion detection systems for effective network security. It is also essential for businesses to identify and deal with these threats in real time to ensure operational continuity, secure sensitive data, prevent financial loss and safeguard their reputation in an increasingly competitive and fast-paced business environment. In this paper, we develop a Kafka-based Intrusion Detection System that offers real-time anomaly detection by utilizing Apache Kafka and machine learning models to detect and classify connection attempts as anomalous or not. This system employs Apache Kafka as a central messaging platform, which has efficient data streaming capabilities, fault tolerance, and scalability. It also enables the system to transition from traditional batch processing to real time intrusion detection. We apply the anomaly-based approach to develop an IDS engine that leverages historic network data, attack patterns, and a trained machine learning model to identify peculiar inputs. Random Forest Classifier model is employed for threat detection. The proposed system displays a proficient ability to recognize suspicious patterns while also generating Slack notifications to the concerned channel and log the event to the terminal.

Index Terms— Intrusion Detection, Apache Kafka, Random Forest Classifier, API, Optuna, Anomaly Detection.

I. INTRODUCTION

An intrusion detection system (IDS) is a security mechanism mainly centered on the observation and analysis of network events. It plays a pivotal role in protecting information systems against malicious access attempts and cyber threats. In essence, an IDS acts as a crucial component in the defense-in-depth strategy providing an additional layer of security beyond traditional preventive measures. It operates by analyzing network traffic patterns, system logs and other relevant data to detect deviations from normal behavior, indicative of potential security breaches. The implementations of this system are broadly categorized into two main types:

A. System that harnesses signatures

This category hinges on pre-established signs/patterns of already determined malicious activities. The system identifies a match between observed network behavior and a stored signature and informs the user. Although effective in dealing with known threats, these models may struggle with recognizing the ever-changing and new attack vectors.

B. Systems that recognize anomalies

anomaly-based IDS focuses on establishing a baseline of standard/regular behavior and flagging any departures

as possible anomaly. This method is well-suited for detecting previously unknown or sophisticated attacks but requires a robust understanding of what constitutes “normal” within a specific network environment.

Our system detects anomalies in a computer network using collected network data and a trained machine learning algorithm and provides real-time alerts to users through slack API. Apache Kafka serves as a central messaging system for smooth flow of data. Kafka is an event streaming platform. It is used to transmit and receive “messages” (which represent a data stream) [1]. In case of an IDS, this stream of events refers to the network information collected from a network monitor or logging device. A machine learning model is trained on the collected network data and is responsible for detecting anomalies in the incoming data. This approach utilizes classification-type machine learning models. Broadly, two types of classification models are used in IDS:

1. Binary Classifiers analyze the received data and predict if the connection attempt is malicious or not (representing two classes), as done in [2].

2. Multiclass classifiers not only predict if the connection attempt is malicious or not; they also predict the type of attack the target system may be under, as done in [3].

A “producer” Kafka topic is created using confluent-Kafka package that processes and transmits the network data [4]. A “consumer” Kafka topic created using confluent- Kafka package that receives data from the “transactions” topics, and applies the trained machine learning model to detect anomalies. Once an intrusion attempt is detected, the information regarding said attempt is published to “Anomalies” topic. This information is logged to the terminal of the system hosting the Kafka brokers. Then a python script subscribed to this Kafka topic, alerts a pre-configured user’s slack channel with data of the network connection attempt. The system described in the paper uses an API, specifically Slack is, implemented in a python script to alert the user of the network intrusion attempt [5].

The main contributions of this work include:

- Exhaustive evaluation of various classification models, including Random Forest, Decision Tree, KNN, logistic regression, SVM, Bernoulli Naïve Bayes. We choose Random Forest Classifier due its promising performance across various metrics.
- Defining an objective function and utilizing the same as a parameter for a hyperparameter optimization function. The output of said function would in turn, be used to provide parameters for the Random Forest Model.
- Employing trained model in an actual anomaly detection scenario by incorporating it in a Kafka based streaming system, rather than just displaying the output of the machine learning algorithm.

The remainder of the paper is organized as follows: Section II conducts the literature survey. Section III delves deeper by describing this system’s components in detail, as well the workings of the same. The results obtained from the evaluation of the system are detailed in Section IV. Finally, Section V serves as the conclusion to the paper, while also listing possible improvements to be implemented in the future. References are cited at last.

II. RELATED WORKS

In 2022 L. Wirz, R. Tanthanathewin, A. Ketphet and S. Fugkeaw, proposed an architecture for an intrusion detection mechanism designed and developed on the cloud using that makes use of Apache Kafka and Spark Streaming [6]. The focus is to detect several network attacks like pattern injection issues, irregular scripting attempts etc. by constructing an engine that learns to recognizing patterns of known attacks. The system can efficiently identify malicious patterns with high accuracy, produce SMS messages, and store the event in a GCP Bucket.

In 2021, J. Jose Diaz Rivera, T. Ahmed Khan, W. Akbar, M. Afaq and W. -C. Song proposed a NIDS built using NetFlow data from switches and Apache Kafka as streaming platform. The anomaly detection was performed by the combination of an unsupervised machine learning algorithm and a supervised algorithm [7]. It also consists of a monitoring system built using ELK stack that collects historical data for further refining of classification process. An ensured data transfer approach to safeguard Kafka’s data from privacy leakage proposed by Hanyi Zhang et al. in 2021 serves as an influence on our proposed system [8]. In 2021, M. Ouhssini, K. Afdel, M. Idhammad and E. Agherrabi described a decentralized anomaly identification model that resides in the cloud. The proposed system is built on the cloud network architecture, and uses decision tree classifier for anomaly detection [9].

Our approach is informed by the model proposed by M. Choubisa, R. Doshi, N. Khatri and K. Kant Hiran in 2022 [10]. This paper presents a robust IDS model that utilizes the random forest classifier algorithm in the

initial setup, and improves it by employing a random feature selection method. Random forest classifier is a powerful algorithm in machine learning that creates numerous decision trees while training. Every single one of these trees independently classify instances and the final prediction is determined through a majority vote. To improve the model robustness, feature selection is employed, where a set of features are dynamically and stochastically selected during the implementation phase. This model is evaluated on the NSL-KDD dataset and has yielded favorable result.

Several datasets were considered and evaluated to guarantee the effectiveness and generalization of our system [11] [12]. The Reno Intrusion Dataset (also known as UNR-IDD) was developed by Tapadhir Das, A Hamdan, R M Shukla, S Sengupta, and E. Arslan in 2023. This dataset consists of data obtained from network port statistics, which provides a better control of network information when compared to traditional datasets [13]. In a 2009 research paper, Mahbod Tavallae, et al. introduced a new dataset named NSL-KDD. This was developed to address certain issues identified in the KDDCUP'99 dataset, which is one of the most widely used datasets for network intrusion detection systems [14]. The dataset contains few variables with high skewness. In 2017, M Ring et al. published the CIDDS-002 dataset. This dataset is developed by simulating a small business setting in the OpenStack and capturing the network traffic in the NetFlow format [15]. UNSW-B15 is another dataset used for evaluation of our proposed system. It was published by Moustafa et al. in 2015. This dataset's raw network packets were generating using the IXIA PerfectStorm tool at UNSW [16]. A similar dataset named NF-UNSW-NB15 was developed by Mohanad Sarhan et al. in 2020. It contains NetFlow features from four benchmark datasets [17]. A multilayer dataset named USB-IDS-1, developed by Marta Catillo et al. in 2021 was also considered for evaluation. It consists of both network traffic information as well as application-level information [18].

While these systems showcase various robust implementations of IDS, they also contain certain flaws. Some of these studies employ outdated datasets for evaluation of their models. Although widely used, these datasets do not accurately reflect the current network environment. Therefore, the effective of models/architectures built on these datasets might be compromised. Additionally, many of these approaches fail to device a mean to alert relevant personnel upon threat identification, which is a critical aspect for any cybersecurity system.

III. METHODOLOGY

Our proposed system uses Machine Learning classification models for detection intrusion. These models need to be trained on a dataset to learn the patterns and relationships within the data. However, some of these datasets have labels with fewer samples (called tail classes), leading to improper fitting of machine learning models. Oversampling or under sampling can be done to address this issue but oversampling increases dataset size and complexity while Under sampling decreases size and hence accuracy. Methods like these may incur higher computational expenses. Other limitations of these datasets are their heavy dependence on the flow-level statistics which may constrain the adaptability of the anomaly detection model. It is therefore imperative to use a dataset with port-level statistics and fewer problems. Therefore, we identified some datasets which are UNR-IDD, UNSW-NB15, NF-UQ-NIDS and NSL-KDD. For the next step, we had to choose a classification model for our system. After careful consideration, we evaluated the performance of various classification models, including Random Forest, Decision Tree, Gradient Boosting algorithm, K-Nearest Neighbor, Bernoulli Naïve Bayes etc.

Table. 1 showcases the performance of these model with different accuracy metrics. The Random Forest classifier exhibits exceptional performance with high accuracy, precision, and recall on both train and test subsets. The metrics score keeps on reducing for all other remaining classifiers. The ROC Curves (Receiver Operating Characteristic Curve) and the Precision Recall Curves is also shown in Fig.4 and Fig.5 respectively. The Random Forest model outperformed others not only in conventional performance metrics but also displayed superior performance in precision- recall curves and ROC curves, validating its robustness and effectiveness across various evaluation criterion. Hence, Random Forest was the most ideal choice for our model. RF is a type of classifier that combines multiple decision trees to improve accuracy. This is done by utilizing several decision trees and adjusting their parameters such as tree magnitude, smallest possible size of the node, and the quantity of features used for splitting each node. It is possible to save the forest generated by this model for future references. It also helps eliminate overfitting problems. Accuracy metrics are also automatically generated by the RF. When creating individual trees in the random forest, the determination for optimal node to be split on is done by randomization. However, RF may produce some noisy trees, which may negatively influence the algorithm's accuracy and lead to incorrect decisions for new samples. After it is done, we modify the dataset as needed, before the processing stage. Then, we divide the dataset for training and test sets. Next, we have model training.

First step is to load the dataset. Then we apply pre-processing techniques on our dataset. First pre-processing was data imbalance. We used SMOTE (Synthetic Minority Over-sampling Technique) to mitigate this problem [19]. Some features had categorical variables in the dataset and were managed using LabelEncoder method. Feature selection using Recursive Feature Elimination (RFE) was employed to enhance model performance and interpretability [20]. Additionally, data splitting and scaling were conducted to ensure model stability and efficiency. It specifically refers to split of dataset into test and train for model training and independent evaluations.

After pre-processing, we first defined an objective function with certain hyperparameters for optimizing our Random Forest model for better accuracy. `Rf_max_depth` parameter is for the maximum depth of each tree. `Rf_max_feat` represents the largest number of features to be taken into consideration when looking for the best node to split in decision trees in the random forest and `rf_n_est` for the number of trees in the model ensemble. We then use an optuna loop to suggest different values for hyperparameters [21]. This is done to find a tree with maximum accuracy. For maximum accuracy, we set the function, `optuna.create_study(direction='maximize')`. Then we have to select the number of trials in the function as `study_rf.optimize(objective, n_trials=30)`. Here, `objective` is the value returned from the objective function. We asked the optuna loops to do 30 trials. After the process is completed, `study_rf` object contains information about the trials conducted during the optimization, with the best set of hyperparameters found and their corresponding accuracies. Last step is to train the model on the training dataset with hyperparameters from `study_rf` object. Then, the model is evaluated using functions such as `rf_score()` for mean accuracy, `roc_auc_score()` to calculate area under the curve score, `roc_curve()` to plot the ROC curve, `predict()` to obtain the predicted class labels for a given set of input features, `auc()` to compute the area under a curve, `precision_recall_curve()` function precision-recall curves, `predict_proba()` for predicting probabilities of each class for a given set of input features, `accuracy_score()` to compute the accuracy of a classification model, `precision_score()` to compute the precision (positive predictive value), `matthews_corrcoef()` function to compute the Matthews correlation coefficient (MCC), `confusion_matrix()` is a function to compute a confusion matrix for the classification model and `feature_importances()` to assess the importance of each feature in the model's predictions. Now, we have our model ready, and the system can begin its flow. The system's flow begins with network data sourced from transactions as shown in Fig.1, which is then organized into Kafka brokers' "transaction" topics. Following this, an anomaly detection model scrutinizes the data, identifying anomalies that are subsequently directed to another Kafka broker. Finally, an alerting script processes these anomalies, triggering timely alerts for potential cybersecurity threats, and ensuring an effective response mechanism. Fig. 3 depicts the workflow of the proposed model. The process begins when producer initializes a connection. The "Transactions" topic receives and processes the network data and sends the data to the anomaly detection model. The anomaly detection script employs a pre-trained machine learning model to detect a potential intrusion attempt. If that is the case, then that network data is broadcast to the "Anomalies" topic. An alerting script is subscribed to the anomalies topic. It processes the incoming network data and triggers the real-time alert, notifying the slack account of the concerned user by utilizing the Slack API service. This sequence ensures that the system can efficiently process incoming transactions, identify potential intrusion attempts, and promptly alert users about potential intrusion attempts on the network.

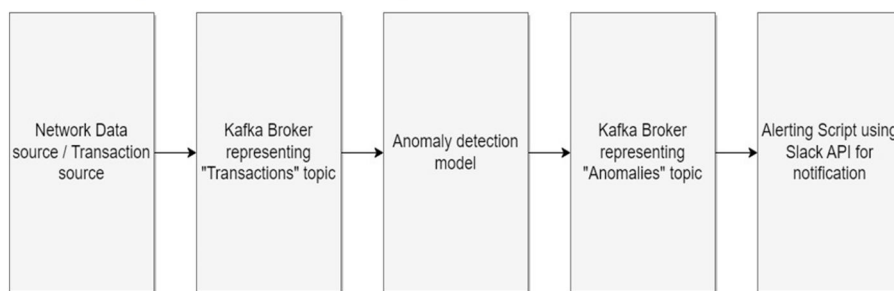


Fig.1 Block Diagram of the Entire System Flow

IV. RESULTS

In the process of selecting the ideal model for the project, a thorough evaluation was conducted on various algorithms using the NSL-KDD dataset. This dataset served as a critical benchmark for gauging the performance of various models, allowing us to make the decision of choosing the right model as per our project requirements. In order to analyze the performance of the model, various measures including precision, recall, accuracy, and F1-

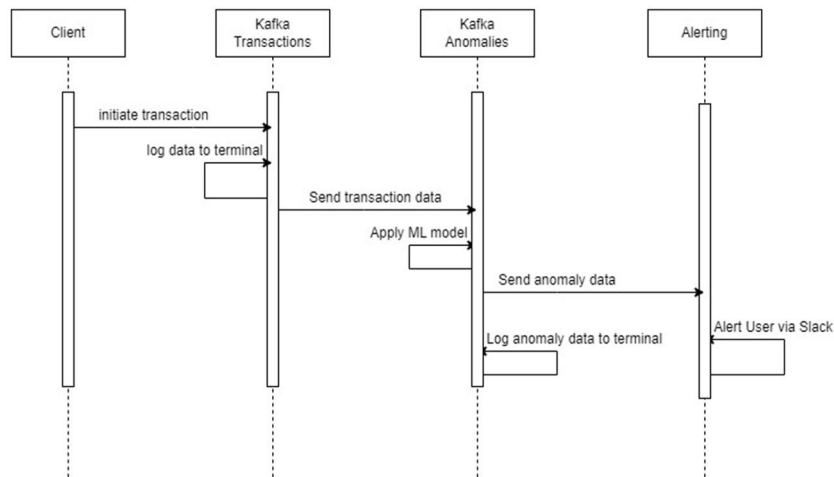


Fig.2 Sequence Flow of the Model

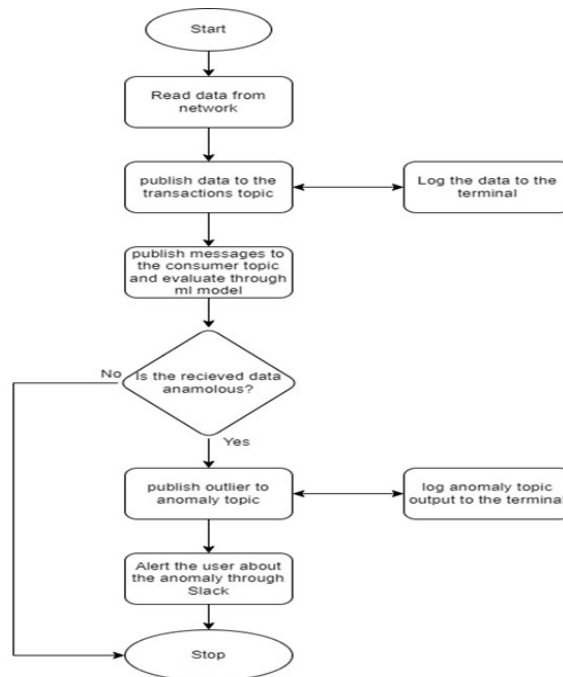


Fig.3 Flowchart of the Entire Process

score were considered such as accuracy, precision, recall and F1 score were considered, which provided a nuanced and understanding of their performances. Table 1 illustrates the percentile performance of various algorithms tested with the NSL-KDD dataset. The Random Forest classifier exhibits exceptional performance with high accuracy, precision, and recall scores. Decision Tree classifier algorithm displays slightly lower performance compared to the Random Forest classifier, albeit still achieving high accuracy, precision, recall, and F1 score on both training and test datasets. The Gradient Boosting classifier exhibits comparable performance to the Decision Tree classifier but slightly lower than the Random Forest classifier. Similarly, the metrics score keeps on reducing for all other remaining classifiers making Random Forest classifier, the most ideal choice for our model.

After the rigorous model selection process, the selected model was evaluated to understand the level of performance of the model (as well as the whole system) across various aspects crucial to the project's objectives. The following metrics were used to analyze the performance of our system. The system yielded favorable results in the experiments. The performance evaluation metrics considered are as follows:

TABLE I. PERFORMANCE OF VARIOUS CLASSIFICATION MODELS WITH NSL-KDD

Algorithm	Train Score	Test Score	Accuracy Score	Precision Score	Recall Score	F1 Score
RF	99.98	99.59	99.59	99.66	99.58	99.62
DT	99.99	99.44	99.44	99.48	99.48	99.48
SKGB	99.52	99.21	99.21	99.19	99.33	99.26
KNN	99.36	98.02	98.02	98.28	98.03	98.15
SVM	96.76	96.53	96.53	97.07	96.5	96.78
LG	94.18	93.89	93.89	95.37	93.39	94.37
BNB	89.38	89.48	89.48	95.05	86.66	90.66

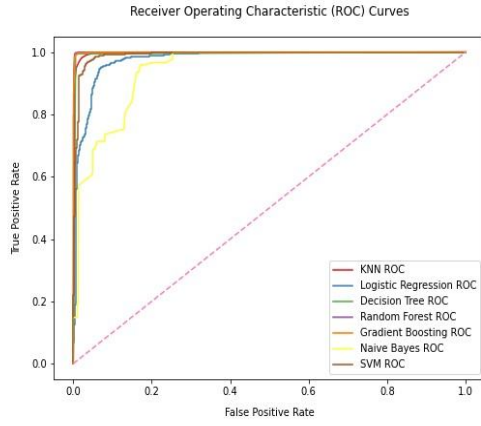


Fig. 4 Receiver Operator Characteristics (ROC) Curve

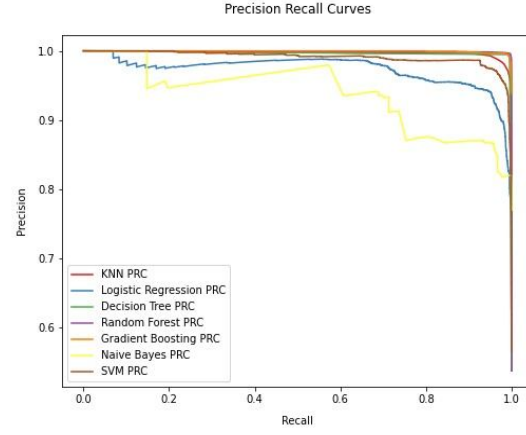


Fig.5 Precision-Recall (PR) Curve for Various Models

TABLE II. PERFORMANCE EVALUATION

Dataset	Accuracy Score	Precision Score	FPR	TPR	MCC
UNSW_NB15-Multiclass	0.899	0.715	0.0000045	0.681	0.860
UNSW_NB15-Binary	0.998	0.999	0.000002	0.999	0.990
NSL-KDD	0.996	0.996	0.0026	0.947	0.991
NF-UQ-NIDS	0.993	0.992	0.000	0.994	0.999
UNR-IDD	0.998	0.999	0.000	0.993	0.999

A. Accuracy

This metric calculates the ratio of correctly identified samples to the total number of samples in the dataset.

$$Accuracy = \frac{\text{Samples correctly classifies in the test data}}{\text{Number of samples in test data}}$$

B. False Alarm Rate (FAR)

It refers to the frequency at which the model incorrectly identifies true samples as false. It is given by

$$FAR = \frac{FP}{(TN + FP)}$$

C. Matthews Correlation Coefficient (MCC):

It is a measure of correlation between observed and predicted binary classifications. It is calculated as,

$$MCC = \frac{(TN \times TP) - (FN \times FP)}{\sqrt{(FP + TP) \times (FN + TP) \times (TN + FP) \times (TN + FN)}}$$

TN → True negative, FN → False negative, FP → False positive, TP → True positive.

V. CONCLUSION

In this paper, an anomaly-based intrusion detection system has been designed and developed. A thorough and detailed review of related works has been accomplished, and the benefits and flaws of each model has been explored. To address the limitations, our research proposes an IDS that leverages contemporary datasets, incorporates effective intrusion detection techniques, and integrates robust alerting mechanisms. An improved implementation of Random Forest Classifier is utilized for anomaly detection. Clusters of Apache of Kafka brokers are employed for real-time data streaming. Also, Slack API has been used to alert concerned personnel in case of threat identification. The performance evaluation of the proposed system, including Accuracy score, Precision score, False Positive Rate, True Positive Rate, Matthews Correlation Coefficient, has been conducted across various datasets, and the system has shown favorable performance. By adopting a holistic approach that combines machine learning algorithms, real-time data analysis, and proactive alerting systems, we aim to enhance the accuracy, efficiency, and adaptability of intrusion detection in modern network environments. For future developments, one significant upgrade to consider would be the ability to accurately determine the nature of the attack the network is under. Also, integrating automated response to threat detection, like blocking the specific IP's would be a proactive measure that would bolster the effectiveness of the system and mitigate potential risks efficiently.

REFERENCES

- [1] "Apache Kafka Documentation." Apache Kafka, kafka.apache.org/documentation/. Accessed: 17-Dec-2023.
- [2] S. -H. Chuang, R. -C. Yang and S. -D. Wang, "Network Intrusion Detection System with Stream Machine Learning in Fog Layer and Online Labeling in Cloud Layer," *2021 International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)*, Yilan County, Taiwan, 2021, pp. 53-59, doi: 10.1109/ICEIB53692.2021.9686445.
- [3] P. Toupas, D. Chamou, K. M. Giannoutakis, A. Drosou and D. Tzovaras, "An Intrusion Detection System for Multi-class Classification Based on Deep Neural Networks," *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, Boca Raton, FL, USA, 2019, pp. 1253-1258, doi: 10.1109/ICMLA.2019.00206.
- [4] T. P. Raptis and A. Passarella, "On Efficiently Partitioning a Topic in Apache Kafka," *2022 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Piraeus, Greece, 2022, pp. 1-8, doi: 10.1109/CITS55221.2022.9832981.
- [5] "Slack API Documentation." Slack, api.slack.com/. Accessed: 20-Feb-2024.
- [6] L. Wirz, R. Tanthanathewin, A. Ketphet and S. Fugkeaw, "Design and Development of A Cloud-Based IDS using Apache Kafka and Spark Streaming," *2022 19th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Bangkok, Thailand, 2022, pp. 1-6, doi: 10.1109/JCSSE54890.2022.9836264.
- [7] J. Jose Diaz Rivera, T. Ahmed Khan, W. Akbar, M. Afaq and W. -C. Song, "An ML Based Anomaly Detection System in real-time data streams," *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA, 2021, pp. 1329-1334, doi: 10.1109/CSCI54926.2021.00270.
- [8] H. Zhang, L. Fang, K. Jiang, W. Zhang, M. Li and L. Zhou, "Secure Door on Cloud: A Secure Data Transmission Scheme to Protect Kafka's Data," *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, Hong Kong, 2020, pp. 406-413, doi: 10.1109/ICPADS51040.2020.00061.
- [9] M. Ouhssini, K. Afdel, M. Idhammad and E. Agherrabi, "Distributed intrusion detection system in the cloud environment based on Apache Kafka and Apache Spark," *2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS)*, Fez, Morocco, 2021, pp. 1-6, doi: 10.1109/ICDS53782.2021.9626721.
- [10] M. Choubisa, R. Doshi, N. Khatri and K. Kant Hiran, "A Simple and Robust Approach of Random Forest for Intrusion Detection System in Cyber Security," *2022 International Conference on IoT and Blockchain Technology (ICIBT)*, Ranchi, India, 2022, pp. 1-5, doi: 10.1109/ICIBT52874.2022.9807766.
- [11] Ankit Thakkar, Ritika Lohiya, "A Review of the Advancement in Intrusion Detection Datasets", *Procedia Computer Science*, Volume 167, 2020, Pages 636-645, ISSN 1877-0509
- [12] Khraisat, A., Gondal, I., Vamplew, P. et al. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecur* 2, 20 (2019). <https://doi.org/10.1186/s42400-019-0038-7>
- [13] T. Das, O. A. Hamdan, R. M. Shukla, S. Sengupta and E. Arslan, "UNR-IDD: Intrusion Detection Dataset using Network Port Statistics," *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2023, pp. 497-500, doi: 10.1109/CCNC51644.2023.10059640.
- [14] 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, doi: 10.1109/CISDA.2009.5356528..

- [15] Ring, M, S Wunderlich, D Grödl, D Landes, and A Hotho. "Creation of Flow-Based Data Sets for Intrusion Detection." *Journal of Information Warfare* 16, no. 4 (2017): 41–54. <https://www.jstor.org/stable/26504117>.
- [16] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," *2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, ACT, Australia, 2015, pp. 1-6, doi: 10.1109/MilCIS.2015.7348942.
- [17] Sarhan, M., Layeghy, S. & Portmann, M. Towards a Standard Feature Set for Network Intrusion Detection System Datasets. *Mobile Netw Appl* 27, 357–370 (2022). <https://doi.org/10.1007/s11036-021-01843-0>
- [18] M. Catillo, A. Del Vecchio, L. Ocone, A. Pecchia and U. Villano, "USB-IDS-1: a Public Multilayer Dataset of Labeled Network Flows for IDS Evaluation," *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Taipei, Taiwan, 2021, pp. 1-6, doi: 10.1109/DSN-W52860.2021.00012.
- [19] G. A. Pradipta, R. Wardoyo, A. Musdholifah, I. N. H. Sanjaya and M. Ismail, "SMOTE for Handling Imbalanced Data Problem : A Review," *2021 Sixth International Conference on Informatics and Computing (ICIC)*, Jakarta, Indonesia, 2021, pp. 1-8, doi: 10.1109/ICIC54025.2021.9632912.
- [20] X. -w. Chen and J. C. Jeong, "Enhanced recursive feature elimination," *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, Cincinnati, OH, USA, 2007, pp. 429-435, doi: 10.1109/ICMLA.2007.35.
- [21] Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19). Association for Computing Machinery, New York, NY, USA, 2623–2631. <https://doi.org/10.1145/3292500.3330701>