Accredited Ranking SINTA 2
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online on: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 6 No. 2 (2022) 322 - 329 ISSN Media Electronic: 2580-0760

Model-Based Feature Selection for Developing Network Attack Detection and Alerting System

Yuri Prihantono[1], Kalamullah Ramli[2]
[1,2]Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia
[1]yuri.prihantono@ui.ac.id, [2]kalamullah.ramli@ui.ac.id

Abstract

The use of the Intrusion Detection Systems (IDS) still has unresolved problems, namely the lack of accuracy in attack detection, resulting in false-positive problems and many false alarms. Machine learning is one way that is often utilized to overcome challenges that arise during the implementation of IDS.. We present a system that uses a machine learning approach to detect network attacks and send attack alerts in this study. The CSE-CICIDS2018 Dataset and Model-Based Feature Selection technique are used to assess the performance of eight classifier algorithms in identifying network attacks in order to determine the best algorithm. The resulting XGBoost Model is chosen as the model that provides the highest performance results in this comparison of machine learning models, with an accuracy rate of 99 percent for two-class classification and 98.4 percent for multi-class classification.

Keywords: Machine Learning, Feature Selection, IDS, Snort, ELK Stack

1. Introduction

Intrusion Detection System (IDS) is a security perimeter that monitor and analyze activities on computers and networks to detect anomalies or attacks [1]. For more than 20 years, several detection approaches have been developed, but there are still two fundamental difficulties that have not been fully solved: the quantity and quality of output (alarms or alerts) from IDS [2]. When it comes to the quantity of alerts, the problem is that the IDS frequently creates too many, which overwhelms the user. As a result, many alarms are unmanageable by the analyst/operator, resulting in the IDS not being used to its full potential. Then, in terms of alarm quality will decrease because normal network behavior continues to change and new attacks also keep appearing, as a result, IDS will lose alarms on actual attacks and report too many false alarms in the normal behavior [2], [3].

Machine learning is one way that is often utilized to overcome challenges that arise during the implementation of IDS. Several machine learning implementations on IDS were performed in previous studies to acquire the highest accuracy value on IDS. The performance of machine learning algorithms to predict attacks has been enhanced by using several techniques including dataset selection, feature selection, parameter optimization, and various other techniques.

Research conducted by Bisyron Wahyudi, et al. produces an intrusion detection model that includes a machine learning approach and is implemented in a real network. They experimented with feature selection using a dataset from KDD. The testing findings produced the best classification accuracy without integrating the content features of KDD, and the developed model may operate well in a real network. Experiments on test data reveal that the SVM technique delivers the greatest results, with a 93.4 percent accuracy rate for two-class classification (normal and attack) and 86.8% for multi-class classification [4].

In other studies, Qusyairi Ridho et al. built the ensemble model with the CSE-CICIDS2018 dataset, decision trees, logistic regression, and gradient boosting selecting 23 of the 80 original dataset features using Spearman's rank correlation. The model they proposed has a final accuracy of 98.8% and a relatively low detection time [5].

Then, Syed Ali Raza Shah et al. also did a feasibility analysis of a machine learning technique that may be applied on Snort IDS by evaluating three different datasets using Weka. The best method is SVM, which

Accepted: 01-04-2022 | Received in revised: 25-04-2022 | Published: 29-04-2022

is followed by Decision Tree and Fuzzy logic. Later, a firefly and ACO-enhanced version of the SVM ensemble was implemented [6].

In this study, we conducted research to develop a Network Attack Detection and Alerting System by applying a machine learning approach to the system being built. In developing this system, we use various software, namely Snort IDS, Kafka, Elasticsearch Logstash Kibana (ELK Stack), ElastAlert, and Telegram. Each software has its function in developing the system proposed in this study. Snort is one of the IDS tools that serve to detect network attacks [7], by reading network packets, analyzing traffic with a predefined rule set, and generating alerts if there are matching rules [8]. This study uses Snort as an IDS tool because it is very easy to implement, and the ruleset can be customized as needed[9]. Apache Kafka is a platform messaging system that allows the consumption of large amounts of data or logs by applications [10] [11]. This study uses Apache Kafka as a messaging system to distribute logs generated by Snort, because of the scalability, high-volume, reliability, data transformations, and low latency [12]. ELK Stack is a combination of three applications i.e., Elasticsearch, Logstash, and Kibana that have their respective functions [13]. Elasticsearch is responsible for storing and collecting information centrally so that it can get the expected analysis results. Logstash is responsible for collecting all unstructured information and converting it into a structured format, and Kibana is a graphical interface (GUI) that ELK Stack users use to analyze logs[14]. This study uses ELK Stack as a Log Management System platform because ELK Stack is the most effective log monitoring tool [14] and it is an open system that can be the best and fast solution for developing a log analysis platform[15]. ElastAlert is a framework used to generate alerts obtained from Elasticsearch data. ElastAlert works with Elasticsearch with two-component types, namely rule types, and alerts. When the rule matches, the alert will be triggered [16]. Telegram is in charge of receiving alerts sent via the ElastAlert Framework. The basic considerations for using Telegram as a medium for sending alerts include security and convenience. In terms of security, based on research conducted by J. Botha et. al shows that Telegram is one of the Chat Applications that prioritizes Security and Privacy [17], and in terms of convenience, based on research conducted by I Made Ari Sulistya et. al, Telegram provides an API (Application Program Interface) that allows developers to create applications that are integrated with Telegram Messenger using Bots [8].

This study aims to develop a machine learning-based network attack detection and alerting system. In order to get a model that has the best performance to be applied to the system being built, we compare the performance of the eight selected machine learning algorithms. The dataset used in the learning process is the CSE-CICIDS2018 dataset, with the consideration that the dataset has comprehensive data and can represent network conditions in the real world [18], [19]. The application of the Model-Based Feature Selection method is also carried out to select important features according to the specified threshold parameter [20]. Feature selection is an important factor in improving machine learning model performance because it can remove unnecessary features, make the model clearer to understand, and reduce training and testing time [21]. The results of this study are expected to be a solution to overcome the problem of false positives and the number of false alarms on the IDS.

2. Research Methods

The methodology we use to conduct this research includes literature study related to previous research, dataset selection, feature selection, performance measure, machine learning model development, and architectural design for Network Attack Detection and Alerting System development.

2.1 Dataset Selection

Based on a survey conducted by [18], datasets are indispensable in conducting training and evaluating IDS performance. The dataset represents normal conditions and attacks in the real world. In the training and testing of IDS machine learning, a broad range of publically available datasets are used. To develop an effective machine learning model performance, the dataset must have adequate and useful information. The quality and quantity of datasets have a substantial impact on the accuracy of IDS, hence choosing an appropriate dataset is an important and unavoidable component of the process [22]. Maxime Labonne used the NSL-KDD, KDD Cup 99, CICIDS2017, and CSE-CICIDS2018 datasets for his research [19]. According to the findings of this study, CICIDS2017 has a more reliable dataset for measuring IDS performance than KDD Cup 99 and NSL-KDD, while CSE-CICIDS2018 has a more comprehensive dataset than CICIDS2017. The findings of this research were considered when CSE-CICIDS2018 was chosen as the dataset for our research.

2.2 Feature Selection

Datasets with a lot of features will increase the computational complexity, requiring more resources and time to analyze. The feature selection technique is one method used by researchers to address this issue [23]. Feature selection is an important factor in improving the performance of machine learning models because it can eliminate unimportant features, making the model clearer to comprehend and reducing training and testing time [21]. Feature selection approaches are commonly used to increase classifier algorithm

DOI: https://doi.org/10.29207/resti.v6i2.3989

accuracy, lessen the influence of the "curse of dimensionality," and improve machine learning prediction performance by choosing the best ones from a larger set of features [24].

2.3 Performance Measure

The confusion matrix is commonly used in evaluating the performance of the developed learning algorithms. According to [25], for binary classification, a 2x2 matrix with actual class and prediction class is described in Figure 1.

		Actual Class		
		p	**n**	**total**
Predictive class	**p'**	True Positive **(TP)**	False Positive **(FP)**	**P'**
	n'	False Negative **(FN)**	True Negative **(TN)**	**N'**
	total	P	N	

Figure 1. Confusion Matrix

From Figure 1, in determining the classification there are formulas used [4], [25], [26], including:

1) True Positive (TP): The positive value that is predicted to be correct, i.e., the Actual Class and Predictive Class values are both positive.
2) True Negative (TN): The predicted negative values, i.e., the Actual Class and Predictive Class values are both negative.
3) False Positive (FP): The positive result that is incorrectly predicted, i.e., the Actual Class value is negative while the Predictive Class value is positive.
4) False Negative (FN): The negative value that is incorrectly predicted, i.e., the Actual Class value is positive while the Predictive Class value is negative.

In the case of IDS, we can assume positive means an attack occurred, while negative means no attack. The metrics derived from the confusion-matrix table above [4], [25] are explained as follows:

1) Accuracy: The most intuitive performance metric is the ratio of predicted right observations to total observations, which can be calculated using the formula 1:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

2) Recall: The predicted negative values, i.e., the Actual Class and Predictive Class values are both negative, as the following formula 2:

$$Recall = \frac{TP}{P} = \frac{TP}{TP+FN} \quad (2)$$

3) Precision: The positive result that is incorrectly predicted, i.e., the Actual Class value is negative while the Predictive Class value is positive, as the following formula 3:

$$Precision = \frac{TP}{P'} = \frac{TP}{TP+FP} \quad (3)$$

4) F_1-Score: The negative value that is incorrectly predicted, i.e., the Actual Class value is positive while the Predictive Class value is negative, as the following formula 4:

$$F_1 = 2 * \frac{Precision*Recall}{Precision+Recall} = \frac{2TP}{2TP+FP+FN} \quad (4)$$

2.4. Model Development

Algorithm selection is required for implementing Machine Learning-based IDS. Algorithms such as Decision Tree, Random Forest, SVM, and others can be utilized [27]. Some of these algorithms have their respective advantages and disadvantages, so research is needed to get a model that has the best performance [28]. The machine learning model development process that we did is represented in Figure 2.

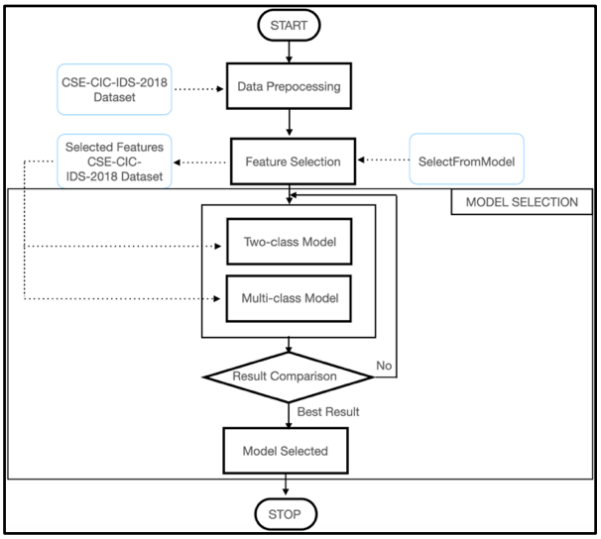


Figure 2. Machine Learning Model Development

The steps of model development in our research are as follows:

Step 1, Data Preprocessing: clean-up, split, and dataset analysis.

Step 2, Feature Selection Selection of feature subsets from the dataset to be implemented in machine learning modeling.

Step 3, Model Selection: The selection of the model that has the best algorithm performance, the stages include the following:

Stage 1, Two-class Model: two-class classification is used to train and test datasets;

DOI: https://doi.org/10.29207/resti.v6i2.3989

Stage 2, Multi-class Model: multi-class classification is used to train and test datasets;

Stage 3, Result Comparison: compare algorithm performance to choose the best machine learning model;

Stage 4, Model Selected: the best model from the Result Comparison is selected.

2.4. System Architecture Design

The next step is to implement the model into the Network Attack Detection and Alerting System architecture. In designing this architecture, we utilize some software, including Snort IDS, Apache Kafka, ELK Stack, ElastAlert, and Telegram. The software were combined to design the architecture of the Network Attack Detection and Alerting System as shown in Figure 3.

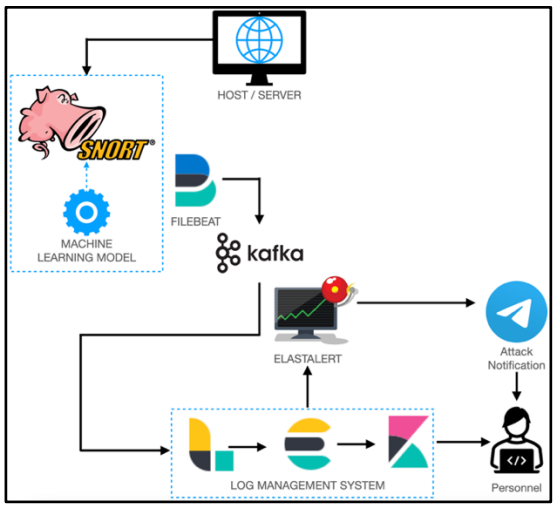


Figure 3. Architectural Design Development of Network Attack Detection and Alerting Systems

The following steps in the intrusion detection simulation are based on the Network Attack Detection and Alerting System Architecture:

Step 1, Capture/sniff network traffic from the host/server and capture the log events as pcap files.

Step 2, Extract network features from the pcap file in the CSE-CICIDS2018 dataset format using CICFlowMeter[29].

Step 3, Predict the intrusion detection findings using the Final Machine Learning Model obtained from Model Development.

Step 4, Send the prediction results to Apache Kafka using Filebeat. Filebeat is a beat shipper used to send log files [30].

Step 5, Apache Kafka distributes logs sent from Filebeat to the Log Management System.

Step 6, Log Management System (ELK Stack) performs network traffic classification on logs received as normal or attack and displays the results.

Step 7, ElastAlert sends logs with attack classifications to Telegram using the ElastAlert framework.

3. Results and Discussions

3.1 *Data Prepocessing*

The CSE-CICIDS2018 dataset must be cleaned before it can be utilized for machine learning algorithm analysis and training. The following steps were applied to perform a clean-up of the dataset [31]:

Step 1, Remove duplicate headers from the dataset that appear as rows.

Step 2, Infinity and inf values are substituted for each other.

Step 3, Remove whitespaces and non-word characters from the column's name.

We analyzed the CSE-CICIDS2018 dataset after cleaning it up by classifying traffic into two groups, benign/normal and attack, as shown in Table 1, as well as classifying traffic into the attack type based on the attack label, as shown in Table 2.

Table 1. Benign and Attack Numbers and Percentages of CSE-CICIDS2018 Dataset

Label	Number	Percentage
Benign	13,484,708	83.07%
Attack	2,748,235	16.93%
Total	16,232,943	

Table 2. Attack Type and Percentages of CSE-CICIDS2018 Dataset

Label	Number	Percentage
Benign	13484708	83.0700%
DDOS attack-HOIC	686012	4.2260%
DDoS attacks-LOIC-HTTP	576191	3.5495%
DoS attacks-Hulk	461912	2.8455%
Bot	286191	1.7630%
FTP-BruteForce	193360	1.1912%
SSH-Bruteforce	187589	1.1556%
Infiltration	161934	0.9976%
DoS attacks-SlowHTTPTest	139890	0.8618%
DoS attacks-GoldenEye	41508	0.2557%
DoS attacks-Slowloris	10990	0.0677%
DDOS attack-LOIC-UDP	1730	0.0107%
Brute Force –Web	611	0.0038%
Brute Force -XSS	230	0.0014%
SQL Injection	87	0.0005%

3.2 Model-Based Feature Selection

Model-Based Feature Selection evaluates all features at once, to identify relationships [32]. At this stage, the feature selection steps carried out include the following:

Step 1, Remove Constant Features: Constant Feature is a feature that displays the same value, one value for all dataset observations. This feature does not provide information that would allow machine learning models to distinguish or predict targets. Removing Constant

DOI: https://doi.org/10.29207/resti.v6i2.3989

Features is done by removing zero-variance features or removing features that have a standard deviation of 0 [33].

Step 2, Remove Quasi-Constant Features: Quasi-Constant Feature is a feature that displays the same value for most of the dataset observations. In general, this feature provides very little information that allows machine learning models to distinguish or predict targets. Removing Quasi-Constant Features is done by removing low-variance features or removing features that have a VarianceThreshold of 0.01 [33]

Step 3, Spearman's Rank: The Spearman Correlation can be used to determine the correlation between features. The Spearman Correlation scale ranges from -1 to +1, where 0 indicates no correlation, and the correlation gets stronger as it approaches the absolute value of 1 [34]. To identify correlations between features, in implementing Spearman's Rank Correlation, it is necessary to select the appropriate threshold. Table 3 shows the interpretation of the Correlation Coefficient of absolute quantities that can be used as a reference for determining the threshold [34].

Table 3. Absolute Magnitude of The Observed Correlation Coefficient and Its Interpretations

Absolute Magnitude	Interpretation
0.00-0.10	Negligible correlation
0.10-0.39	Low correlation
0.40-0.69	Medium correlation
0.70-0.89	High correlation
0.90-1.00	Very high correlation

At this stage, features that have a very high correlation with each other will be removed, because these features will not provide predictions but only cause noise [31]. Based on the interpretation obtained from Table 3, this study uses a threshold value of 1 to identify features that have a very high correlation. Figure 4 provides a heatmap that visualizes the correlation between distinct pairs of features after deleting the very highly correlated features. Figure 4 shows that feature pairs with a higher correlation are represented by a darker color spectrum, whereas feature pairs with a lower correlation are represented by a lighter color spectrum.

Step 4, Feature Selection using SelectFromModel: SelectFromModel is a feature selection method that removes features if the relevant coefficients or important features are less than the parameter threshold [20]. This strategy is applied to estimators with significant characteristics or coefficients [21]. Eight classifier algorithms are implemented as estimators in this method, including XGBoost, Random Forest, Extra Trees, Gradient Boosting, Decision Tree, AdaBoost, Logistic Regression, and Stochastic Gradient Descent (SGD). The eight classifier algorithms are implemented as models in feature selection using default parameters. Table 4 shows the selected features from the application of Model-Based Feature Selection.

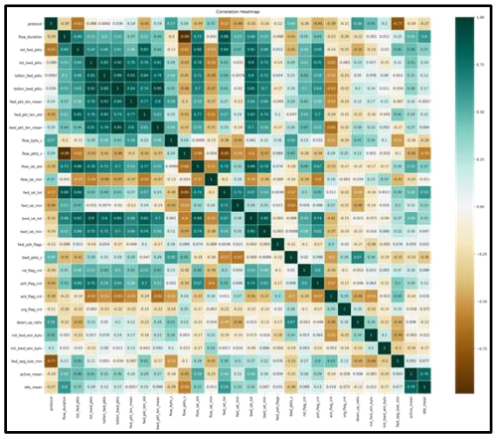


Figure 4. Correlation Heatmap after the removal of very highly correlated features

Table 4. Selected Features of Several Classifier Algorithms

Estimator/Classifier Algorithm	Remained/Selected Features
XGBoost	totlen_fwd_pkts fwd_pkt_len_mean flow_pkts_s fwd_iat_tot init_fwd_win_byts init_bwd_win_byts fwd_seg_size_min
Random Forest	flow_duration totlen_fwd_pkts fwd_pkt_len_mean flow_byts_s flow_pkts_s flow_iat_min fwd_iat_tot fwd_iat_min bwd_pkts_s init_fwd_win_byts init_bwd_win_byts fwd_seg_size_min
Extra Trees	flow_duration fwd_pkt_len_std flow_pkts_s fwd_iat_tot bwd_pkts_s ack_flag_cnt init_fwd_win_byts init_bwd_win_byts fwd_seg_size_min
Gradient Boosting	totlen_fwd_pkts flow_iat_min fwd_iat_tot init_fwd_win_byts init_bwd_win_byts fwd_seg_size_min
Decision Tree	totlen_fwd_pkts bwd_pkt_len_mean flow_pkts_s fwd_iat_tot init_fwd_win_byts init_bwd_win_byts fwd_seg_size_min
AdaBoost	totlen_fwd_pkts fwd_pkt_len_std flow_iat_min init_fwd_win_byts init_bwd_win_byts
Logistic Regression	totlen_bwd_pkts flow_byts_s

DOI: https://doi.org/10.29207/resti.v6i2.3989

Estimator/Classifier Algorithm	Remained/Selected Features
	flow_pkts_s bwd_pkts_s init_fwd_win_byts init_bwd_win_byts
SGD	flow_duration totlen_fwd_pkts totlen_bwd_pkts flow_iat_std flow_iat_min fwd_iat_tot fwd_iat_min bwd_iat_min

3.3 Performance Comparison and Model Selection

The subset of features from the feature selection stage is employed in the learning process with the same classifier algorithm at this level. We go through a learning process to classify two-class and multi-class cases. The prediction results of each algorithm with the Two-class classification are represented in Table 5 and the Multi-class classification in Table 6.

Table 5. Performance Evaluation of Two-Class Classification Model

Model (selected features subset)	Accuracy	Precision	Recall	F1-Score	Time	Rank
XGBoost (7 features)	0.990	0.990	0.990	0.990	0:14:04.79	**1**
Random Forest (12 features)	0.989	0.989	0.989	0.989	2:13:27.37	2
Extra Trees (9 features)	0.988	0,988	0.988	0.988	0:52:41.07	3
Gradient Boosting (6 features)	0.988	0.988	0.988	0.987	0:32:18.17	4
Decision Tree (7 features)	0.987	0.987	0.987	0.987	0:03:23.38	5
AdaBoost (5 features)	0.986	0.986	0.986	0.986	0:18:57.66	6
Logistic Regression (6 features)	0.853	0.858	0.853	0.808	0:00:49.11	7
SGD (8 features)	0.831	0.854	0.831	0.754	0:00:39.17	8

Table 6. Performance Evaluation of Multi-class Classification Model

Model (selected features subset)	Accuracy	Precision	Recall	F1-Score	Time	Rank
XGBoost (7 features)	0.984	0.980	0.984	0.979	3:00:03.36	**1**
Random Forest (12 features)	0.983	0.977	0.983	0.979	2:13:00.73	2
Extra Trees (9 features)	0.982	0.977	0.982	0.979	0:51:48.05	3
Gradient Boosting (6 features)	0.964	0.960	0.964	0.961	10:00:04.13	5
Decision Tree (7 features)	0.981	0.977	0.981	0.978	0:03:27.54	4
AdaBoost (5 features)	0.839	0.715	0.839	0.711	0:26:15.11	6
Logistic Regression (6 features)	0.831	0.732	0.831	0.777	0:22:26.78	7
SGD (8 features)	0.831	0.690	0.831	0.754	0:09:27.71	8

Based on the performance evaluation of Model-Based Feature Selection as shown in Table 5 and Table 6, the XGBoost algorithm shows the best algorithm performance evaluation for the classification of Two-class and Multi-class, with an accuracy value of 99 percent and a processing time of 14 minutes 4 seconds 79 milliseconds for Two-class classification and an accuracy value of 98.4 percent, and a processing time of 3 hours 3 seconds 36 milliseconds for Multi-class classification. These results indicate a decrease in the performance of the XGBoost Algorithm for Multi-class classification. Further research is needed to analyze the decline in performance and methods that can be used to overcome these problems.

3.4 Development and Simulation

In the development of a Network Attack Detection and Alerting System, there is a requirement not only to detect network traffic in the form of normal or attack but also to detect the types of attacks that occur. This is the basis for selecting the XGBoost Model with Multi-class classification to be applied to the system being built.

In developing this system, the steps we took include the following:

Step 1, On IDS Server, we create a shell script to capture snort with tcpdump.log, extract dataset with CICFlowMeter, do dataset prepocessing, create dataset with selected subset features, predict the result with machine learning model, and create result in CSV format namely ml_snort.csv. Then, we configure filebeat.yml to send ml_snort.csv to Apache Kafka.

Step 2, On Apache Kafka, we run the Kafka server, then create topic snort on Kafka server, and verify the received message

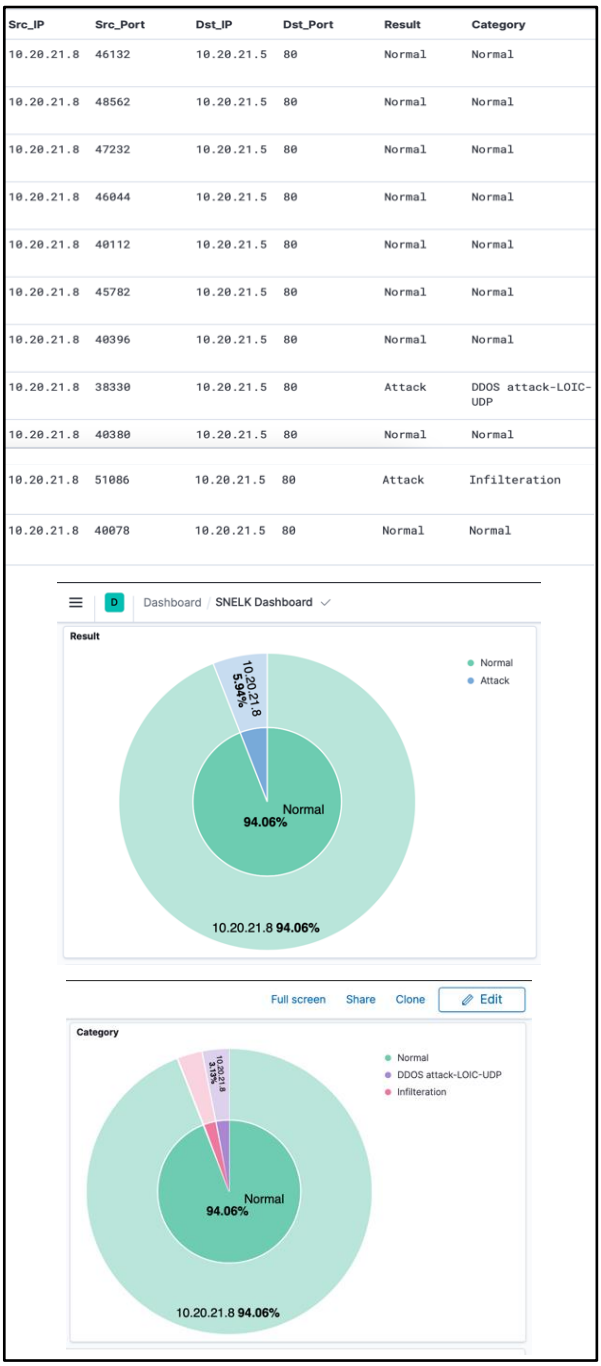
Step 3, On ELK Stack, we configure Logstash to set input from Kafka based on a predefined topic.

Step 4, Finally, we configure ElastAlert to send attack notifications to Telegram using the frequency rule type alerts with a filter query "Result: Attack".

To test the success of system development, attack simulations are carried out in the form of Network Scanning, Website Scanning, and Denial-of-Service Attack. The prediction results from the attack simulation are visualized on the ELK Stack as shown in Figure 5.

In Figure 5, the prediction results are obtained with normal and attack classifications, with the types of attacks in the form of DDOS attack-LOIC-UDP and Infiltration. The prediction results with the attack classification are sent to Telegram as an alert notification of attacks on the network. Figure 6 shows the network attack alert notification received by Telegram.

DOI: https://doi.org/10.29207/resti.v6i2.3989



Src_IP	Src_Port	Dst_IP	Dst_Port	Result	Category
10.20.21.8	46132	10.20.21.5	80	Normal	Normal
10.20.21.8	48562	10.20.21.5	80	Normal	Normal
10.20.21.8	47232	10.20.21.5	80	Normal	Normal
10.20.21.8	46044	10.20.21.5	80	Normal	Normal
10.20.21.8	40112	10.20.21.5	80	Normal	Normal
10.20.21.8	45782	10.20.21.5	80	Normal	Normal
10.20.21.8	40396	10.20.21.5	80	Normal	Normal
10.20.21.8	38330	10.20.21.5	80	Attack	DDOS attack-LOIC-UDP
10.20.21.8	40380	10.20.21.5	80	Normal	Normal
10.20.21.8	51086	10.20.21.5	80	Attack	Infilteration
10.20.21.8	40078	10.20.21.5	80	Normal	Normal

Figure 5. Visualization of Attack Simulation Results on ELK Stack

The alert notification received by Telegram is the result of an attack prediction, while the normal prediction result is not sent to Telegram. Thus, the goal of developing this system is achieved, namely being able to overcome the problem of false positives and the quantity of false alarms generated by IDS.

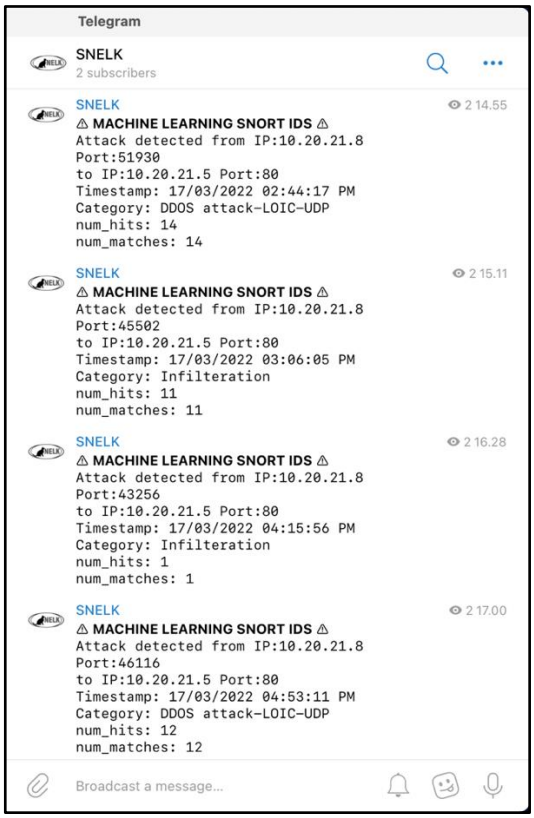


Figure 6. Network Attack Alert Notifications via Telegram

4. Conclusion

We successfully applied machine learning models in the development of an Attack Detection and Alerting System on the Network and performed attack detection simulations. Based on the performance evaluation of Model-Based Feature Selection, the XGBoost model was chosen to be used in the development of the proposed system. The simulation results show that the system has succeeded in visualizing the results of attack detection and sending notification alerts of the attack via Telegram. However, there are still shortcomings that must be followed up in further research related to machine learning models that can be used with the proposed approach. Some further research that can be done includes the utilization of other machine learning algorithms, the use of a more comprehensive dataset, reducing misclassification for Multi-class, and optimizing parameters.

Reference

[1] A. Tasneem, A. Kumar, and S. Sharma, "Intrusion Detection Prevention System using SNORT," *International Journal of Computer Applications*, vol. 181, pp. 21–24, Mar. 2018, doi: 10.5120/ijca2018918280.

[2] Z. Yu and J. J. P. Tsai, *Intrusion Detection A Machine Learning Approach*, vol. 3. London: Imperial College Press, 2011.

[3] Milan, H. Sardana, and K. Singh, "Reducing False Alarms in Intrusion Detection Systems-A Survey," *International Research Journal of Engineering and Technology (IRJET)*, vol. 05, no. 02, pp. 9–12, 2018.

DOI: https://doi.org/10.29207/resti.v6i2.3989

[4] B. Wahyudi, K. Ramli, and H. Murfi, "Implementation and Analysis of Combined Machine Learning Method for Intrusion Detection System," *International Journal of Communication Networks and Information Security*, vol. 10, pp. 295–304, Mar. 2018.

[5] Q. R. S. Fitni and K. Ramli, "Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems," in *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, Jul. 2020, pp. 118–124. doi: 10.1109/IAICT50021.2020.9172014.

[6] S. A. R. Shah, B. Issac, and S. M. Jacob, "Intelligent Intrusion Detection System Through Combined and Optimized Machine Learning," *International Journal of Computational Intelligence and Applications*, vol. 17, no. 02, p. 1850007, Jun. 2018, doi: 10.1142/S1469026818500074.

[7] A. Erlansari, F. F. Coastera, and A. Husamudin, "Early Intrusion Detection System (IDS) using Snort and Telegram approach," *SISFORMA*, vol. 7, no. 1, pp. 21–27, Jun. 2020, doi: 10.24167/sisforma.v7i1.2629.

[8] I. Made Ari Sulistya and G. Made Arya Sasmita, "Network Security Monitoring System on Snort with Bot Telegram as a Notification," *International Journal of Computer Applications Technology and Research*, vol. 9, no. 2, pp. 59–64, 2020.

[9] R. AM and R. Manicka chezian, "Intrusion Detection System Techniques and Tools: A Survey," *Scholars Journal of Engineering and Technology (SJET)*, vol. 5, no. 3, pp. 122–130, 2017.

[10] J. A. Shaheen, "Apache Kafka: Real Time Implementation with Kafka Architecture Review," *International Journal of Advanced Science and Technology*, vol. 109, pp. 35–42, Dec. 2017, doi: 10.14257/ijast.2017.109.04.

[11] T. V* and Dr. K. V, "Development of Kafka Messaging System and its Performance Test Framework using Prometheus," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 9, no. 1, pp. 1622–1626, May 2020, doi: 10.35940/ijrte.A2516.059120.

[12] B. R. Hiraman, C. Viresh M., and K. Abhijeet C., "A Study of Apache Kafka in Big Data Stream Processing," in *2018 International Conference on Information , Communication, Engineering and Technology (ICICET)*, Aug. 2018, pp. 1–3. doi: 10.1109/ICICET.2018.8533771.

[13] P. Bavaskar, O. Kemker, and H. H. Syed, "A SURVEY ON: 'LOG ANALYSIS WITH ELK STACK TOOL,'" *SSRN Electronic Journal*, vol. 6, pp. 965–969, Mar. 2019.

[14] F. Ahmed, U. Jahangir, H. Rahim, K. Ali, and D.-S. Agha, "Centralized Log Management Using Elasticsearch, Logstash and Kibana," in *2020 International Conference on Information Science and Communication Technology (ICISCT)*, Feb. 2020, pp. 1–7. doi: 10.1109/ICISCT49550.2020.9080053.

[15] D. v Uday and G. S. Mamatha, "An Analysis of Health System Log Files using ELK Stack," in *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, May 2019, pp. 891–894. doi: 10.1109/RTEICT46194.2019.9016706.

[16] Q. Long, *ElastAlert Documentation*, Release 0.0.1. 2019.

[17] J. Botha, C. van 't Wout, and L. Leenen, *A Comparison of Chat Applications in Terms of Security and Privacy*. 2019.

[18] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, Sep. 2019, doi: 10.1016/j.cose.2019.06.005.

[19] M. Labonne, *Anomaly-based network intrusion detection using machine learning*. 2020.

[20] T. Akhtar *et al.*, "Effective Voting Ensemble of Homogenous Ensembling with Multiple Attribute-Selection Approaches for Improved Identification of Thyroid Disorder," *Electronics (Basel)*, vol. 10, no. 23, p. 3026, Dec. 2021, doi: 10.3390/electronics10233026.

[21] M. Huljanah, Z. Rustam, S. Utama, and T. Siswantining, "Feature Selection using Random Forest Classifier for Predicting Prostate Cancer," *IOP Conference Series: Materials Science and Engineering*, vol. 546, no. 5, p. 052031, Jun. 2019, doi: 10.1088/1757-899X/546/5/052031.

[22] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems," 2021, pp. 117–135. doi: 10.1007/978-3-030-72802-1_9.

[23] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. bin Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020, doi: 10.1109/ACCESS.2020.3009843.

[24] M. Iqbal, M. M. Abid, M. N. Khalid, and A. Manzoor, "Review of feature selection methods for text classification," *International Journal of Advanced Computer Research*, vol. 10, no. 49, pp. 138–152, Jul. 2020, doi: 10.19101/IJACR.2020.1048037.

[25] M. Gong, "A Novel Performance Measure for Machine Learning Classification," *International Journal of Managing Information Technology*, vol. 13, no. 1, pp. 11–19, Feb. 2021, doi: 10.5121/ijmit.2021.13101.

[26] M. F. Fibrianda and A. Bhawiyuga, " Analisis Perbandingan Akurasi Deteksi Serangan Pada Jaringan Komputer Dengan Metode Naïve Bayes Dan Support Vector Machine (SVM)," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 2, no. 9, pp. 3112–3123, 2018, [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/2559

[27] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An Adaptive Ensemble Machine Learning Model for Intrusion Detection," *IEEE Access*, vol. 7, pp. 82512–82521, 2019, doi: 10.1109/ACCESS.2019.2923640.

[28] M. Rabbani *et al.*, "A Review on Machine Learning Approaches for Network Malicious Behavior Detection in Emerging Technologies," *Entropy*, vol. 23, no. 5, p. 529, Apr. 2021, doi: 10.3390/e23050529.

[29] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018, pp. 108–116. doi: 10.5220/0006639801080116.

[30] F. Inigo Solomon, "Securing Websites & Webapplications Using Data Analytics," in *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, Feb. 2019, pp. 1–4. doi: 10.1109/ICCIDS.2019.8862089.

[31] C. Stumpf, "A machine learning based approach towards building an Intrusion Detection System," Dec. 11, 2019. https://github.com/cstub/ml-ids (accessed Apr. 18, 2022).

[32] R. Liang, "Feature selection using Python for classification problems," *https://towardsdatascience.com/feature-selection-using-python-for-classification-problem-b5f00a1c7028*, Aug. 07, 2019.

[33] S. Galli, "Feature Selection for Machine Learning - Code Repository," *https://github.com/solegalli/feature-selection-for-machine-learning*, Feb. 2018.

[34] P. Schober, C. Boer, and L. A. Schwarte, "Correlation Coefficients," *Anesthesia & Analgesia*, vol. 126, no. 5, pp. 1763–1768, May 2018, doi: 10.1213/ANE.0000000000002864.

DOI: https://doi.org/10.29207/resti.v6i2.3989