An Efficient Real-Time NIDS Using Machine Learning Methods

Konda Srikar Goud(✉), M. Shivani, B. V. S. Selvi Reddy, Ch. Shravyasree, and J. Shreeya Reddy

Department of Information Technology, BVRIT Hyderabad College of Engineering for Women, Hyderabad, Telangana, India
kondasrikargoud@gmail.com

Abstract. Recent developments in network technology and related services have caused a significant rise in data traffic. However, there has also been a massive rise in the negative consequences of cyber-attacks. Many new types of network attacks are emerging. As a result, designing a robust Intrusion detection system (IDS) has become essential. This paper presents a framework for designing an efficient IDS to enhance detection accuracy and reduce false positives on real-time data. This research used the CIC-IDS 2017 dataset to train Machine Learning models such as Logistic Regression, K Nearest Neighbor, Gaussian Naive Bayes, and Random Forest. Machine learning models often perform well on benchmark datasets but may encounter challenges when applied to real-time traffic scenarios. So, we created a Real-time dataset and tested it on the trained models. In the evaluation, the Random Forest classifier outperformed all other models and achieved an accuracy of 99.99% .

Keywords: Intrusion Detection System · Cyber-attacks · DDoS attack · Botnet · Random Forest · Real-time dataset

1 Introduction

An IDS is a device which monitors network traffic for unusual activity and warns the user when it is discovered. Software that looks for harmful activity on a network or machine. Any unlawful behavior or violation is frequently noted, either centrally via a security information and event management (SIEM) system or by sending an administrator a notification. In order to discriminate between valid and false alerts, a SIEM system aggregates outputs from several sources and applies alarm filtering techniques. Intrusion detection systems are prone to raise erroneous warnings while monitoring networks for potentially dangerous behavior as a result, after first installation, businesses need to modify their IDS products. To discriminate between safe network traffic and malicious activity, intrusion detection systems must be properly configured. The two main categories of IDS are host-based and network-based ones. They are as follows: (1) Network Intrusion Detection (NIDS): A NIDS Is a security instrument that scans

P. Pareek et al. (Eds.): IC4S 2023, LNICST 536, pp. 172–185, 2024.
https://doi.org/10.1007/978-3-031-48888-7_15

network traffic for probable intrusions or malicious activity. It compares live packet analysis results to known attack patterns or anomalies. To record and examine traffic, NIDS sensors are positioned strategically throughout the network architecture. It produces notifications to inform administrators of any questionable activity that is found. By monitoring actual traffic and spotting assaults that may evade firewall regulations, NIDS support firewalls. (2) HIDS: A security technology known as a Host Intrusion Detection System keeps track of the actions and conduct of specific hosts or endpoints within a network. It concentrates on preserving the integrity and security of particular equipment, including servers, workstations, or IoT gadgets. In order to identify unauthorized access, malicious software, or unusual activity at the host level, HIDS employs a variety of approaches, including log analysis, file integrity verification, and system call monitoring.

In order to meet the needs of an efficient IDS, researchers have considered the possibility of using ML and DL techniques. ML and DL fall under the broad umbrella of Artificial Intelligence (AI) and are focused on learning useful information from large amounts of data. These techniques gained immense popularity in network security over the past decade due to the development of high-performance graphics processors (GPUs). Both ML and DL are effective tools for learning useful features from network traffic and for predicting normal or abnormal activities on the basis of the learned patterns. ML-based IDS relies heavily on feature engineering for learning useful information from network traffic. DL-based IDS don't rely on feature engineering and are capable of learning complex features automatically from the raw data because of its deep structure.

Section 2 discusses the literature review of various methods for detecting attacks. In Sect. 3, we discuss proposed model with algorithms. Section 4 of this article talks about simulation and results analysis. The Results are discussed in Sect. 5. Finally, we discuss conclusion in Sect. 6.

2 Literature Survey

Intrusion detection systems aim to identify and respond to unauthorized activities or malicious behavior within a network. Traditional IDS typically use rule-based methods that rely on pre-defined patterns or signatures of known attacks. However, these rule-based systems often struggle to detect novel or sophisticated attacks, which led to the exploration of ML-based IDS.

Authors in [1] demonstrates how ML and DL techniques were used on the CICIDS 2017 dataset. Four feature selection strategies are the subject of the work. The paper discusses the performance of these algorithms with accuracy. In [2], Researchers primarily study and assess the 1999 NSL- KDD datasets and the 1999 KDDCUP datasets using SVMs for intrusion detection. Additionally, in [3], researchers focuses on network intrusion detection system using machine learning classifiers on KDD99 dataset.

The study in [4] provides an overview of ensemble classifiers which are suitable for classifying data IDS with machine learning. In [5], researchers proposes an optimized intrusion detection system that improves upon the existing IDS which detects malicious packets in the network. Researchers in [6] explored six different ML techniques to find out the best technique for detecting DDOS attacks in machine learning. The authors in

[7], proposed a research project has been suggested to utilize a hybrid system that makes use of misuse detection methods for well-known sorts of invasions. By recognizing the assaults that are known thanks to anomaly detection systems, it also educates and trains itself. The major goal of is to build a real-time IDS that can detect intrusions by examining incoming and egressing network data in real-time. The deep neural network used in the proposed system was trained the model using 28 features from the NSL-KDD dataset.

Authors in [9] focuses on using machine learning methods for wireless sensor network intrusion detection. It explores the use of ML algorithms, including decision trees, SVM, k-nearest neighbors (KNN), and random forests, on the CICIDS 2017 dataset. The paper discusses the performance of these algorithms in detecting different types of intrusions and compares their effectiveness.

In [10], Researchers suggested that datasets like CSE-CIC-IDS2018 were made to train prediction algorithms on anomaly-based intrusion detection for network traffic. The study describes a paradigm for real-time intrusion detection [11]. In [12], researchers suggest a deep learning-based intrusion detection solution for industrial control systems (ICS). It trains deep learning models including deep neural networks (DNN) and long short-term memory (LSTM) networks using the CICIDS 2017 dataset, to find intrusions in ICS networks. The paper highlights the performance of different models and assesses how well they execute real-time intrusion detection.

Additionally, in [13], scientists concentrate on the creation of a DL-based anomaly-based network intrusion detection system. For the purpose of extracting pertinent characteristics from the authors' CICIDS 2017 dataset suggested an improved feature selection method. For intrusion detection, the study applies deep auto-encoders and deep belief networks, and it assesses how well they perform is measured by the F1-score, recall, accuracy, and precision.

The findings were summed up by the authors in [14], who also highlighted the potential of DL for detecting intrusions in IOT networks. The performance and resilience of the proposed system were both increased by the authors' recommendations for future research initiatives. The article in [15] gives an overview of the various feature selection methods used in utilizing machine learning to detect intrusions systems. It investigates various methodologies, including wrapper, filter, and embedding approaches, and how they apply to the CICIDS 2017 dataset. The article examines how feature selection affects the effectiveness of intrusion detection models and offers suggestions for choosing the right attributes for efficient detection.

3 Proposed Methodology

Here we'll discuss the proposed framework. The primary goal of this framework is to increase the model's accuracy. By identifying the optimal features using the feature importance. It is an attribute of Random Forest Classifier. The initial step is to pre-process the CICIDS2017 dataset. The preprocessing process begins with data cleaning, addressing missing values, outliers, and inconsistent data. The techniques used in data cleaning are binning, replacing with mean and median values. Feature selection techniques are then applied and selects the most relevant features. Feature scaling is performed to ensure

that features are on a consistent scale. Categorical variables are encoded into numerical representations, and techniques are employed to handle imbalanced data, ensuring the dataset adequately captures both normal and at- tack samples. Finally, training and testing sets are created from the preprocessed dataset for model training and evaluation. These preprocessing steps ensure the quality, relevance, and compatibility of the CICIDS 2017 dataset with ML algorithms, ultimately enhancing the performance of the real-time IDS.

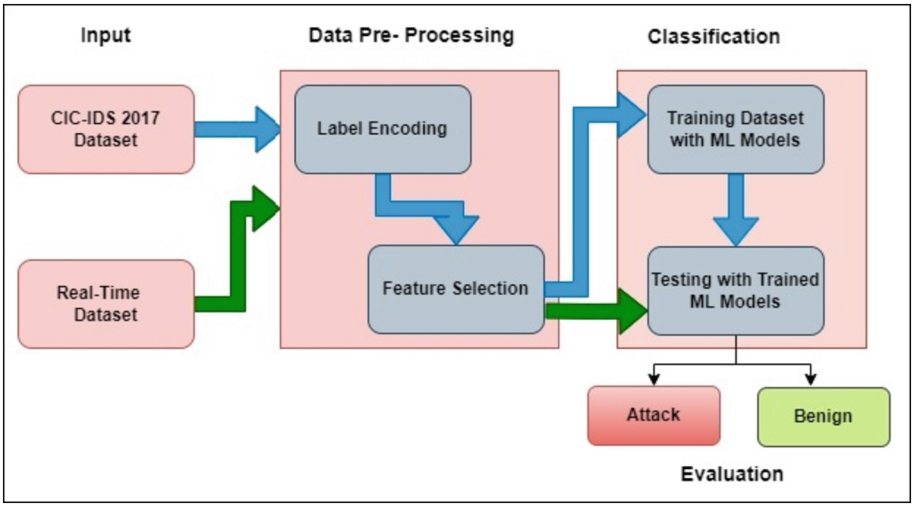

Fig. 1. Architecture of the proposed framework

The creation of a real-time intrusion detection system (IDS) is the aim of this effort. The preprocessing and feature selection of the CICIDS 2017 dataset are done. To build a predictive model, training is done using the Logistic Regression, KNN, Gaussian Naive Bayes, and Random Forest methods. The model is then tested on real-time data to detect attacks. By utilizing machine learning algorithms, the IDS aims to accurately and promptly identify malicious activities in network traffic, enabling proactive measures to be taken for maintaining network security. The Fig. 1 depicts the Architecture of the proposed framework.

3.1 Random Forest Classifier

Random Forest Classifier is initialized with 100 estimators, then trained on the training data. The confusion matrix is created to assess the model's performance after predictions are made based on the test data. The confusion matrix provides a comprehensive overview of the class labels, both projected and actual, supporting the evaluation of false positives, true positives, true negatives, and false negatives. This evaluation aids in assessing the model's Accuracy, Precision, Recall, and F1 Score. By utilizing the scikit-learn library and following these steps, the Random Forest classifier can effectively detect intrusions

in real-time scenarios, providing valuable insights for security purposes.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(fi - yi)^2 \tag{1}$$

3.2 Gaussian Naive Bayes Classifier

The initialization and training phases of the classifier include estimating the statistical parameters of the Gaussian distribution for each class using the training data. Subsequently, predictions are made on the test data by calculating the probability of each sample belonging to each class & selecting the group that has the highest probability. The model's performance is then assessed using the confusion matrix, which offers details on the predicted and actual class labels. This enables the evaluation of accuracy, precision, recall, and other performance parameters by allowing the examination of false positives, true positives, true negatives, and false negatives. By utilizing the Gaussian Naive Bayes algorithm and following these steps, the classifier can effectively detect intrusions in real-time scenarios, providing valuable insights for IDS applications.

$$\mathrm{P(x)} = \frac{1}{\sqrt{2\pi\sigma^2}}\mathrm{e}^{-}\frac{(x-\mu)^2}{2\sigma^2} \tag{2}$$

3.3 Logistic Regression Classifier

The data is first scaled using the StandardScaler to standardize the features and ensure consistent scaling across different variables. The Logistic Regression model is then initialized with increased max iter to ensure convergence. The scaled training data are used to train the model and teach it the logistic regression equation's coefficients. The learned coefficients are then used to make predictions on the scaled test data. After that, the confusion matrix is computed to assess the model's effectiveness and reveal differences between the predicted and real class labels. By utilizing the confusion matrix to examine false positives, true positives, true negatives, and false negatives, it is possible to assess accuracy, precision, recall, and other performance parameters. By utilizing the Logistic Regression algorithm and following these steps, the classifier can effectively detect intrusions in real-time scenarios, providing valuable insights for IDS applications.

$$Z = \left(\sum_{i=1}^{n} w_i x_i\right) + b \tag{3}$$

3.4 K Nearest Neighbor

Data preprocessing and feature engineering steps, such as removing irrelevant columns, handling missing values, and encoding categorical variables are done. It splits using the training data, a K-nearest neighbors (KNN) classifier is created, trained, and applied to the dataset. The code then makes predictions on the test set and evaluates the classifier's accuracy using the accuracy score metric. This code provides a basic framework for

implementing a KNN classifier for intrusion detection on the CICIDS 2017 dataset, but further modifications and enhancements may be needed for optimal performance and customization to specific requirements.

$$\text{dist}\,(x, z) = \left(\sum_{r=1}^{d} |x_r - z_r|^p\right)^{1/p} \tag{4}$$

3.5 Pearson Correlation Coefficient

The linear connection between two continuous variables is measured by the Pearson correlation coefficient, sometimes known as Pearson's r. It is calculated by taking the sum of the product of the differences between each variable's value and its mean, and then dividing it by the product of the standard deviations of the variables. The resulting coefficient, which ranges from −1 to 1, shows the strength and direction of the link. The perfect correlation is represented by a value of 1, the perfect correlation is represented by a value of −1, and the perfect correlation is not represented by a value of The Pearson correlation coefficient can be used for feature selection by calculating the correlations between each feature and the target variable, and then selecting features based on their absolute correlation values or by setting a correlation threshold. However, it assumes linearity and may not capture non-linear relationships or be suitable for all types of data.

$$r = \frac{n\left(\sum xy\right) - \left(\sum x\right)\left(\sum y\right)}{\left[\, n\sum x^2 - \left(\sum x\right)^2\right]\left[n\sum y^2 - \left(\sum y\right)^2\right]} \tag{5}$$

4 Simulation and Results Discussion

In this section, we evaluate the proposed methodology's results with those obtained using the LR, KNN, Naive Bayes and RF classifiers. For the experiment, we considered the following datasets and simulation environment.

4.1 Dataset Description

The CICIDS2017 dataset captures the network behavior of various types of network traffic, including both normal traffic and malicious activities. Intrusion detection offers a thorough understanding of network behavior. With over 2.8 million instances and 79 features per instance, the dataset provides a comprehensive representation of network behavior. The features include packet headers, payload information, and flow statistics. Source and destination IP addresses, ports, protocol types, and flags are just a few of the details found in packet headers. Payloads include packet data, and flow statistics record the characteristics of the overall network flow. The development and evaluation of intrusion detection systems use the CICIDS 2017 dataset as a benchmark.

The dataset provides labeled data, with each network flow labeled as either normal or belonging to a specific attack category. Web assaults, denial-of-service attacks, reconnaissance attacks and botnet attacks are among the several types of attacks. Among

Table 1. Description of files containing CICIDS-2017 dataset

Name of Files	Day Activity	Attacks Found
Monday-WorkingHours.pcap_ISCX.csv	Monday	Benign(Normal human activities)
Tuesday-WorkingHours.pcap_ISCX.csv	Tuesday	Benign, FTP-Patator, SSH-Patator
Wednesday-WorkingHours.pcap_ISCX.csv	Wednesday	Benign, DoS GoldenEye, Dos Hulk Dos Slowhttptest, Dos slowloris, Heartbleed
Thursday-WorkingHours.Morning-WebAttacks.pcap_ISCX.csv	Thursday	Benign, Web Attack-Brute Force, Web Attack-Sql Injection, Web Attack-XSS
Thursday-WorkingHours.Afternoon - Infiltration.pcap_ISCX.csv	Thursday	Benign, Infiltration
Friday-WorkingHours -Morning.pcap_ISCX.csv	Friday	Benign, Bot
Friday-WorkingHours –Afternoon-PortScan.pcap_ISCX.csv	Friday	Benign, Portscan
Friday-WorkingHours –Afternoon-DDos.pcap_ISCX.csv	Friday	Benign, DDos

others. This labeling allows researchers and developers to train and evaluate their intrusion detection and prevention systems on real-world attack scenarios. With regard to each network flow, the CICIDS 2017 dataset provides a wealth of details and properties, such as source IP and destination IP addresses, port numbers, protocol types, packet sizes, and time duration. The Table 1 and Table 2 presents the description and features of the CIC-IDS dataset.

4.2 Simulation Environment

To generate a real-time dataset, you will need specific tools and virtual machines. The necessary tools include VirtualBox, Wireshark, and CICFlowmeter. Virtual machines such as Windows XP and Kali Linux are required for this process, and they should be installed within a virtual box environment. The goal is to use these virtual machines to carry out various attacks, including DoS, DDoS, Slowloris, and Synflood attacks. While performing these attacks, it is essential to have Wireshark running to capture the live traffic, which will generate a Pcap file. The next step involves using CICFlowmeter to convert the Pcap file into a CSV file. Finally, the two CSV files are merged for testing purposes.

Performing Slowloris Attack

Step 1: Start Kali Linux, and then Start up your terminal. Use the command below to create a new directory on your desktop with the name Slowloris. Navigate to the Slowloris directory you need to create. The Slowloris tool must now be copied from Github so that it can be installed on your Kali Linux computer. You just need to type the following URL in your terminal inside the Slowloris directory you generated to do that: https://github.com/GHubgenius/slowloris.pl.git. Figure 2 depicts the slowloris attack in kali.

Step 2: Now, in order to execute that kind of command, you must first check your machine's IP address. It is now time to launch the Apache server. Enter the following command to do so: sudo service apache2 start. Figure 3 depicts the wireshark file capture.

Table 2. Features of CICIDS-2017 Dataset

Feature no.	Features	Feature no.	Features
1.	Destination Port	41.	Packet Length Mean
2.	Flow Duration	42.	Packet Length Std
3.	Total Fwd Packets	43.	Packet Length Variance
4.	Total Backward Packets	44.	FIN Flag Count
5.	Total Length of Fwd Packets	45.	SYN Flag Count
6.	Total Length of Bwd Packets	46.	RST Flag Count
7.	Fwd Packet Length Max	47.	PSH Flag Count
8.	Fwd Packet Length Min	48.	ACK Flag Count
9.	Fwd Packet Length Mean	49.	URG Flag Count
10.	Fwd Packet Length Std	50.	CWE Flag Count
11.	Bwd Packet Length Max	51.	ECE Flag Count
12.	Bwd Packet Length Min	52.	Down/Up Ratio
13.	Bwd Packet Length Mean	53.	Average Packet Size
14.	Bwd Packet Length Std	54.	AvgFwd Segment Size
15.	Flow Bytes/s	55.	AvgBwd Segment Size
16.	Flow Packets/s	56.	Fwd Header Length
17.	Flow IAT Mean	57.	FwdAvg Bytes/Bulk
18.	Flow IAT Std	58.	FwdAvgPackets/Bulk
19.	Flow IAT Max	59.	FwdAvg Bulk Rate
20.	Flow IAT Min	60.	BwdAvg Bytes/Bulk
21.	Flow IAT Total	61.	BwdAvg Packets/Bulk
22.	Fwd IAT Mean	62.	BwdAvg Bulk Rate
23.	Fwd IAT Std	63.	SubflowFwd Packets
24.	Fwd IAT Max	64.	SubflowFwd Bytes
25.	Fwd IAT Min	65.	SubflowBwd Packets
26.	Bwd IAT Total	66.	SubflowBwd Bytes
27.	Bwd IAT Mean	67.	Init_Win_bytes_forward
28.	Bwd IAT Std	68.	Init_Win_bytes_backward
29.	Bwd IAT Max	69.	act_data_pkt_fwd
30.	Bwd IAT Min	70.	min_seg_size_forward
31.	Fwd PSH Flags	71.	Active Mean
32.	Bwd PSH Flags	72.	Active Std
33.	Fwd URG Flags	73.	Active Max
34.	Bwd URG Flags	74.	Active Min
35.	Fwd Header Length	75.	Idle Mean
36.	Bwd Header Length	76.	Idle Std
37.	Fwd Packets/s	77.	Idle Max
38.	Bwd Packets/s	78.	Idle Min
39.	Min Packet Length	79.	Label
40.	Max Packet Length		

Now that we need to determine whether your server is active or not, use the following command to find out its status: service apache2 status

Step 3: The tool should now be run using the following command: perl slowloris.pl -dns 10.0.2.15 –options

Step 4: Capture the live traffic data using wireshark.

Step 5: A Pcap file will be generated from wireshark tool. Convert the Pcap file to CSV file using CICFlowmeter tool.

Performing Synflood Attack

Step 1: Firstly, we should get IP address of 2 virtual Machines- Windows XP and Kali-linux. Communication should be done for both VMs using ping command.

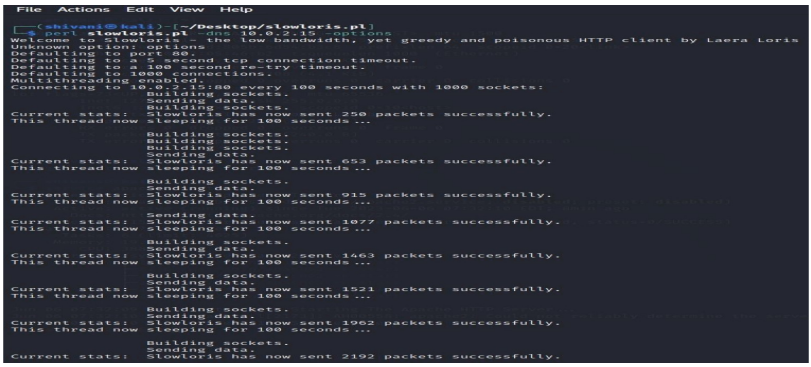

Fig. 2. Performing Slowloris Attack

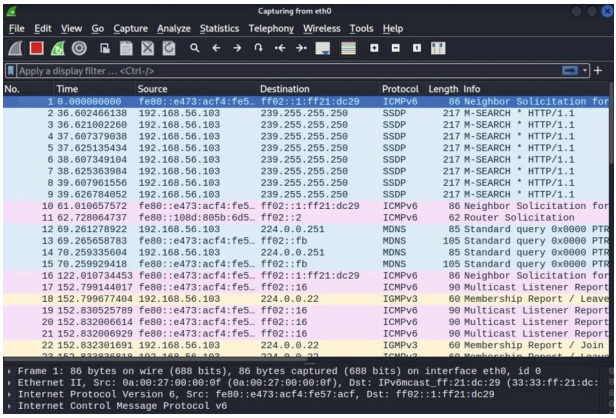

Fig. 3. Generating live traffic data

Step 2: We should enter into Metasploit using the following command: msfconsole

Metasploit is a widely used penetration testing framework which provides a comprehensive group of tools and exploits for assessing the security of computer systems. It allows security professionals to identify vulnerabilities, launch attacks, and test the effectiveness of security measures. Metasploit simplifies the process of scanning, exploiting, and gaining access to target systems, helping to identify potential weak- nesses before they can be exploited by malicious actors. With its extensive collection of exploits, payloads, and auxiliary modules, Metasploit is a powerful tool for ethical hacking and security assessment.

Step 3: Next command is search synflood. Enter show options. It shows some options about rhosts,num, rport.

Step 4: Set RHOST to IP address of other virtual machine and enter exploit.

In Metasploit, RHOSTS, RPORT, and NUM are parameters used to specify the target host, target port, and the number of concurrent targets, respectively.

- RHOSTS: RHOSTS refers to the remote hosts, which are the target IP addresses or hostnames that you want to scan or attack. It can be a single host or a range of hosts specified using CIDR notation or wildcard characters.
- RPORT: RPORT stands for remote port and represents the target port number on the remote host. It is the port to which you want to establish a connection or attempt an exploit.
- NUM: NUM is an abbreviation for the number of con- current targets. This parameter is typically used when launching attacks that require targeting multiple hosts simultaneously. By setting NUM to a specific value, you can control the number of concurrent targets that Metas- ploit will attack simultaneously, increasing the efficiency of the operation.

  - Set RHOSTS ipaddress
  - Set RPORT 134
  - Set NUM 50,000

Step 5: Capture the live traffic data using wireshark tool. A Pcap file will be generated from wireshark tool. Convert the Pcap file to CSV file using CICFlowmeter tool. Figure 4 depicts the Synflood attack.

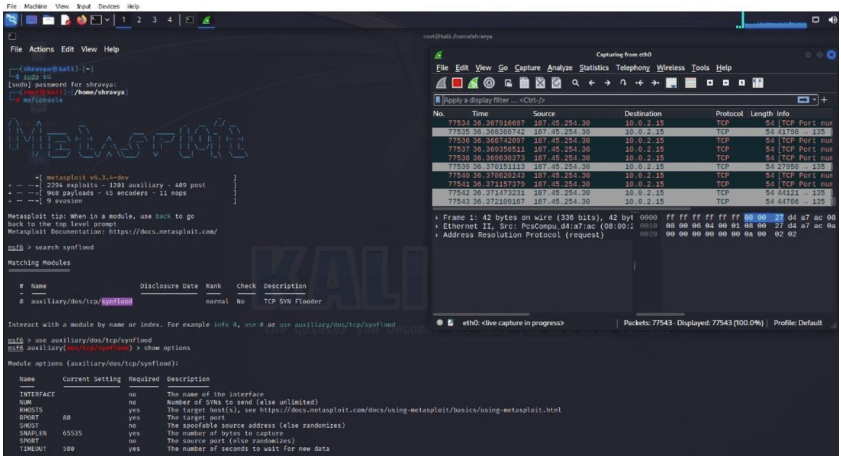

Fig. 4. Performing Synflood Attack

4.3 Performance Metrics

The performance evaluation phase, which tests and assesses the approach for generalization using several evaluation met- rics, is essential to the development of a strong ML model. To evaluate the model performance in this work, many evaluation metrics have been used.

Accuracy: Accuracy is a metric that counts the number of examples in a dataset and counts the fraction of accurately predicted instances out of those instances. It is calculated by dividing the overall forecasted number by the number of accurate estimates. The accuracy score goes from 0 to 1, with 1 denoting perfect accuracy. However, it is important to consider other metrics alongside accuracy, particularly when dealing with imbalanced datasets or when different types of errors have varying impacts. The formula for accuracy is:

$$\text{Accuracy} = (\text{Number of Correct Predictions}) \big/ (\text{Total Number of Predictions}) \tag{6}$$

Precision: A classification model's accuracy of correctly predicting future events is measured by a parameter called precision. The formula for calculating it is to divide the total number of true positive predictions by the total number of true positives and false positives. It focuses on minimizing false positives and provides a demonstration of the model's accuracy in identifying positive events. Precision values range from 0 to 1, with 1 representing perfect precision. A higher precision score indicates a lower rate of false positives, making it particularly important in applications where false positives have significant consequences, such as medical diagnoses or spam detection. The formula for recall is:

$$\text{Precision} = (\text{TP}) \big/ (\text{TP} + \text{FP}) \tag{7}$$

Recall: Recall, also known as true positive rate, is a statistic used to assess the effectiveness of a classification model, particularly in situations where the goal is to reduce false negatives. It calculates whether percentage of all real positive instance true positives plus false negatives were accurately anticipated positive instances. The formula for recall is:

$$\text{Recall} = (\text{TP}) \big/ (\text{TP} + \text{FN}) \tag{8}$$

F1 Score: By integrating precision and recall into a single value, the F1 score is a statistic that offers a fair evaluation of the effectiveness of a classification model. It is especially beneficial when there is an uneven distribution of students among classes or when both false positives and false negatives matter. The F1 score is calculated as the HM of recall and precision. The F1 score formula is as follows:

$$\text{F1 Score is calculated as} : 2 * (\text{Precision} * \text{Recall}) \big/ (\text{Precision} + \text{Recall}) \tag{9}$$

5 Results and Discussion

The below result provides a comparison of the performance metrics as Accuracy, Precision, F1 score and Recall for Four classification algorithms: LR, RF, and Naive Bayes, KNN. It states that Random Forest algorithm achieved the highest scores in all metrics, indicating its superior performance. The testing of a real-time dataset was conducted using the Random Forest classification, which resulted in an impressive accuracy of 99%. The model successfully predicted attacks with a label of 1. These results highlight the effectiveness of the Random Forest algorithm in detecting and classifying attacks in real- time scenarios, demonstrating its high accuracy and reliable performance. The results are listed below in the Table 3.

Table 3. Comparison of various algorithms

Models	Accuracy (%)	Precision (%)	Recall (%)	F1Score (%)
Random Forest	99.993	99.886	99.902	99.894
Logistic Regression	98.074	97.079	96.372	96.724
KNN	97.305	94.667	96.318	95.486
Gaussian Naïve Bayes	87.434	83.729	71.255	76.990

The ROC graph in the Fig. 5 below, which uses the four algorithms RF, KNN, LR, and Naive Bayes, shows the true positive rate on the y-axis and the false positive rate on the x-axis, respectively. The percentage of real positives that were accurately detected is known as the true positive rate. The percentage of genuine negatives that are mistakenly classified as positives is known as the false positive rate.

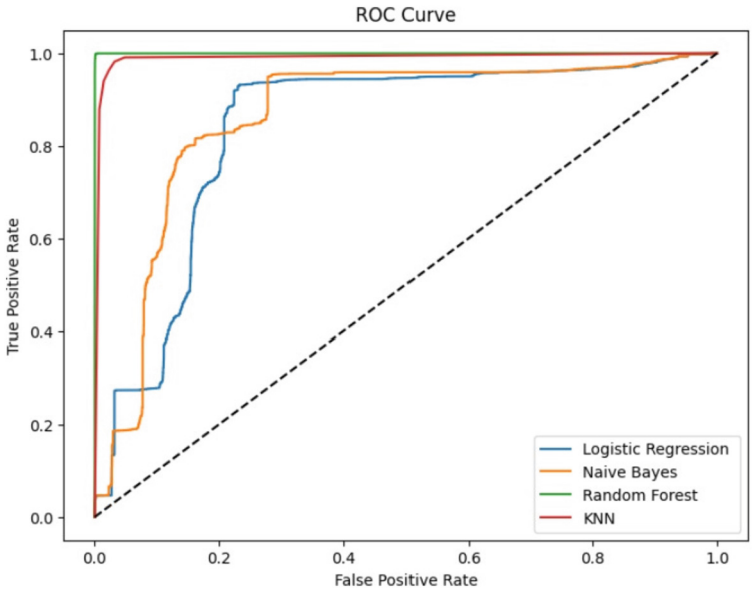

Fig. 5. ROC Curve

The outcomes demonstrate that ML models are useful for identifying attack traffic. The Table 4 presents the comparison of results of our proposed model with related papers. Based on the suggested framework, we note that our method outperforms the baseline classifiers in accuracy. Therefore, we draw the conclusion that the PC+RF architecture successfully addresses the feature selection and attack categorization issues.

Table 4. Comparison of results of our proposed model with related papers.

Method	Dataset	Accuracy (%)
Author in [1] designed a model ML and DL Algorithms	CICIDS 2017	97.24
Author in [2] proposed a model using SVM Algorithm	KDDCUP 99 and NSL-KDD	94.31
Author in [9] proposed a model using KNN,SVM,DT, RF algorithm	CICIDS 2017	92.19
Author in [12] designed a IDS model using DNN,LSTM	CICIDS 2018	96.47
Our Proposed Model	CICIDS 2017 and Real-time dataset	99.93

6 Conclusion

This study focuses on developing a real-time IDS using ML techniques. We leverage the CICIDS 2017 dataset for training our IDS model. The dataset encompasses attacks, including DoS, DDoS, FTP, SSH, and benign traffic. Through extensive data preprocessing, which involves removing duplicates and handling infinity values, we ensure the dataset's quality. We employ RF, KNN, LR, and Naive Bayes algorithms to train the IDS model. SMOTE is employed to address the class imbalance between benign and attack instances. Through feature selection, we identify and select the ten most relevant features. In the testing phase, we use our created real-time dataset to test the model. The accuracy achieved during the testing phase is an impressive 99.99% indicating the robustness of our model.

References

1. Hagar, A.A., Gawali, B.W.: Implementation of machine and deep learning algorithms for intrusion detection system. In: Intelligent Communication Technologies and Virtual Mobile Networks: Proceedings of ICICV 2022, pp. 1–20. Springer, Singapore (2022). https://doi.org/10.1007/978-981-19-1844-5_1
2. Ngueajio, M.K., Washington, G., Rawat, D.B., Ngueabou, Y.: Intrusion detection systems using support vector machines on the kddcup'99 and nsl-kdd datasets: A comprehensive survey. In: Proceedings of SAI Intelligent Systems Conference, pp. 609–629. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-16078-3_42
3. Prajapati, P.K., Singh, I., Subhashini, N.: Network intrusion detection using machine learning. In: Futuristic Communication and Network Technologies: Select Proceedings of VICFCNT 2021, vol. 1, pp. 55–66. Springer, Singapore (2023)
4. Yazdizadeh, T., Shabnam, H., Paula, B.: Intrusion detection using ensemble models. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 143–158. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-23633-4_11
5. Nayak, S., Anushka, AP., Reethika, R., Lakshmisudha, K.: Optimizing network intrusion detection using machine learning. In: Advances in Data Science and Information Engineering: Proceedings from ICDATA 2020 and IKE 2020, pp. 585–590. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-030-71704-9_40

6. Bindra, N., Sood, M.: Detecting DDoS attacks using machine learning techniques and contemporary intrusion detection dataset. Autom. Control. Comput. Sci. **53**, 419–428 (2019)
7. Dutt, I., Borah, S., Maitra, I.K., Bhowmik, K., Maity, A., Das, S.: Real-time hybrid intrusion detection system using machine learning techniques. In: Bera, R., Sarkar, S.K., Chakraborty, S. (eds.) Advances in Communication, Devices and Networking. LNEE, vol. 462, pp. 885–894. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-7901-6_95
8. Thirimanne, S.P., Jayawardana, L., Yasakethu, L., Liyanaarachchi, P., Hewage, C.: Deep neural network based real-time intrusion detection system. SN Comput. Sci. **3**(2), 145 (2022)
9. Dwivedi, R.K., Rai, A.K., Kumar, R.: Outlier detection in wireless sensor networks using machine learning techniques: a survey. In: 2020 International Conference on Electrical and Electronics Engineering (ICE3), pp. 316–321. IEEE. (2020)
10. Elhanashi, A., Gasmi, K., Begni, A., Dini, P., Zheng, Q., Saponara, S.: Machine learning techniques for anomaly-based detection system on CSE-CIC-IDS2018 dataset. In: International Conference on Applications in Electronics Pervading Industry, Environment and Society, pp. 131–140. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-30333-3_17
11. Singh, N.T., Chadha, R.: A review paper on network intrusion detection system. In: International Conference on Intelligent Cyber Physical Systems and Internet of Things, pp. 453–463. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-18497-0_34
12. Hijazi, A., El Safadi, A., Flaus, J.M.: A deep learning approach for intrusion detection system in industry network. In BDCSIntell, pp. 55–62 (2018)
13. Saba, T., Rehman, A., Sadad, T., Kolivand, H., Bahaj, S.A.: Anomaly-based intrusion detection system for IoT networks through deep learning model. Comput. Electr. Eng. **99**, 107810 (2022)
14. Acharya, N., Singh, S.: An IWD-based feature selection method for intrusion detection system. Soft. Comput. **22**, 4407–4416 (2018)
15. Goud, K., Gidituri, S.: Security challenges and related solutions in software defined networks: a survey. Int. J. Comput. Netw. Appl. **9**, 22–37 (2022)