IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.Doi Number

Detection and Response: Enhancing Intrusion Detection Systems

Syed K. Ikramuddin[1], Sk. Md. Mizanur Rahman[1,]

[1]School of Engineering Technology and Applied Science, Centennial College, Scarborough, ON M1G 3T8, Canada.

ABSTRACT This article comprehensively evaluates Intrusion Detection Systems (IDS) using machine learning (ML) techniques, leveraging the CSE-CIC-IDS-2018 dataset alongside a custom-generated dataset. The study addresses the challenge of dataset relevance to real-world scenarios by replicating attack setups within a different network architecture. A thorough comparison of ML model performance in various contexts is made possible by this method. Models trained on the unique dataset showed higher precision, recall, F1-score, and accuracy, resulting in good experiment performance. On the other hand, Models trained on the available dataset did not perform well enough. These results highlight how crucial it is to match training data with real-world network environments to improve IDS adaptability. This study shows how particular network settings affect the significance of features, offering important insights for creating IDS models that are more adapted to real-world threats to cybersecurity.

INDEX TERMS Brute force, CSE-CIC-IDS-2018, denial of service, IDS, intrusion detection system, machine learning, web attacks.

I. INTRODUCTION

In order to detect and stop harmful activity in network environments, intrusion detection systems, or IDS, are essential. The accuracy and usability of the datasets used to train and assess machine learning (ML) models have a major impact on how well these machine learning models function. Historically, datasets such as KDD CUP 99, ISC2012, and ADFA13 have served as standard references for IDS research. These datasets, however, frequently lack the complexity and diversity required to depict modern network settings. [1][2]

Both valid and attack traffic are included in the extensive collection of network traffic scenarios provided by the CSE-CIC-IDS-2018 dataset. The Canadian Institute for Cybersecurity created this large dataset, which covers a range of attack methods such as web-based, brute force, and denial of service (DoS) attacks. Its wide range of attack scenarios and varied feature set make it an invaluable standard for assessing IDS performance. [3]

The possibility of a mismatch between training environments and real-world applications is a major problem with current datasets [4]. In this study, we use a domain-specific dataset to address this difficulty. Using the same tools and methods as those in the CSE-CIC IDS-2018 dataset, but in a different network setup, we were able to successfully duplicate attack scenarios. With this method, we may examine how the distinct features of particular networks may affect the effectiveness of IDS models.

We evaluated machine learning models against our custom dataset after training them on the CSE-CIC-IDS-2018 dataset in order to acquire a deeper understanding of these dynamics. This procedure makes it possible to compare how well the model performs in various network conditions. We also examine the best machine learning approaches for intrusion detection systems (IDS) applications, with a focus on lowering false positive rates, speeding up training and prediction, and enhancing feature selection processes and computing effectiveness.

A variety of machine learning algorithms, including as random forests, XGBoost, and deep learning models, have been studied in the past, with varying degrees of success [5][6]. These models' effectiveness frequently depends on how well the training datasets match the particular situations they are intended to cover.

II. OUR CONTRIBUTION

For Intrusion Detection Systems (IDS), this study offers a comprehensive evaluation of machine learning models in real-time scenarios. To ensure consistency and comparability throughout the experiment, we presented cyber-attacks using the same tools and techniques as the CSE-CIC-IDS-2018. The only distinction is that the network architecture used in the data gathering setup is different from the original dataset.

Even though there are many IDS datasets available, our research highlights a significant drawback of these models: when trained on randomized outcome datasets, they do not perform optimally. When it comes to applying these models to our particular network architecture, where the relationship between attribute and feature values varies and differs from

standard datasets like the CIC-IDS-2018, this inefficiency is especially visible.

Our network configuration and dataset, highlight these discrepancies and provide a unique perspective on how network-specific features impact model performance. ML models were trained using the CSE-CIC-IDS-2018 dataset, and their performance was assessed using a dataset we created ourselves that was gathered from our particular network environment. It also made it possible to assess the models' ability to identify network intrusions under different scenarios.

The study also contains a thorough investigation to find the best machine learning (ML)-based intrusion detection system (IDS) algorithms that can offer valuable insights into how feature importance and architectural differences affect model performance. Better cybersecurity solutions can be developed as a result of this, expanding the ways that context-specific datasets might support the efficacy and vulnerability of IDS models.

III. RELATED WORKS

Over time, intrusion detection systems (IDS) have undergone significant change, mostly due to developments in machine learning (ML) methods and the accessibility of a variety of datasets, including UNSW-NB15, Bot-IoT, CSE-CIC-IDS-2018, and KDD99. Numerous research has looked into using machine learning (ML) to identify various cyber-attacks across different network datasets. [2][5]

A study by Jaw and Wang [7] examined feature selection and ensemble-based solutions for IDS. They emphasized the efficiency of ensemble methods in detecting network intrusions, where it highlighted hybrid feature selection with ensemble classifier as a critical component for improving model performance. A similar study focused on network intrusion detection through comparative analysis based on feature selection and ensemble machine learning. These approaches significantly enhanced the accuracy of intrusion detection systems, especially in complex environments. [8]

In order to identify anomalies in network traffic, Vanin et al. [9] examined AI/ML-based IDS, demonstrating a taxonomy of machine learning methods and details of popular datasets. A thorough analysis identified the advantages and disadvantages of several machine learning models. Deep learning neural networks for DDoS attack detection were investigated by Sumathi et al. [10], who showed how well they worked to increase detection accuracy by using long short-term memory (LSTM) recurrent neural networks (RNN) strategy.

By 2023, IDS model optimization has become the main priority. In their study of AI-based intrusion detection techniques, Sowmya and Anita [11] provided an overview of different AI-based detection mechanisms and insights on the challenges of multi-classification of attacks. In parallel, Singh and Vigila [12] suggested using a proposed Principal Component analysis based Fuzzy Extreme Learning Machine (PCA-FELM) in their intrusion detection for mobile ad hoc networks (MANET), a network with dynamic topology. This creative method provided a strong way to improve by increasing detection rates. Songma et al. [13] enhanced accuracy and reduced computing load by employing a phased approach to IDS optimization, particularly with the CSE-CIC-IDS-2018 dataset. To address the particular issues of the IoT ecosystem, Bhavsar et al. [14] created an anomaly detection-based intrusion detection system for IoT systems. Similarly, Gaber et al. [15] used machine learning and optimization techniques to propose an Industrial Internet of Things (IIoT) IDS.

A cloud-based IDS that uses deep learning based on convolutional neural networks (CNNs) to improve cloud security by detecting and classifying cyberattacks was proposed by Aljuaid and Alshamrani [16]. Paidipati et al. [17] decreased computation complexity and false positives by using an ensemble of optimization and reinforcement models for DDoS attack detection and classification on cloud-based Software-Defined Networks (SDNs). Mohsenabad and Tut [18] highlighted approaches on developing highly accurate models using the fewest possible features by concentrating on bio-inspired optimization algorithms such as the Artificial Bee Colony (ABC), Flower Pollination Algorithm (FPA), and Ant Colony Optimization (ACO) feature-selection techniques. Göcs and Johanyák [19] supplemented this by offering a thorough feature analysis of the CSE-CIC-IDS-2018 dataset, showing that feature selection improves IDS model correctness, particularly in large, complicated datasets.

Talukder et al. [20] advanced IDS research by using oversampling and feature extraction to handle the difficulties of imbalanced data, feature embedding, and dimension reduction. Their research demonstrated that handling imbalanced datasets is crucial for improving detection accuracy and reducing computational complexity in large-scale environments. Yuan et al. [21] proposed CoSen-IDS, a novel cost-sensitive intrusion detection system tailored for imbalanced data in 5G networks, a strategy that guides Generative Adversarial Networks to amplify multiple minority traffic classes simultaneously in the training dataset. Another 2024 study [22] introduced a novel IDS for cloud computing, reflecting the need for feature selection and usage of a hybrid firefly algorithm with the hybrid classifier. In the realm of vehicular networks, Neto et al. [23] addressed DoS and spoofing attacks in the Internet of Vehicles (IoV) and the development of the CICIoV2024 dataset, highlighting how IDS can be adapted to the development of intra-vehicular communication security.

Optimizing hyperparameters in ML models is another key area of focus. Using the CSE-CIC-IDS-2018 dataset, Witcha and Siriporn [24] reported on XGBoost hyperparameter optimization which significantly improved model performance. Bakır and Ceviz [25] introduced a comprehensive approach to improving IDS performance through genetic algorithm-based hyperparameter tuning and hybrid feature selection, enhancing both detection accuracy and efficiency.

In general, IDS research shows a distinct trend toward combining optimization methods with advanced machine

learning approaches. These advancements highlight the importance of adapting IDS ML models to specific network environments, such as cloud computing, IoT, and IoV, to successfully mitigate modern cybersecurity challenges.

IV. EXPERIMENTAL SETUP

In order to preserve consistency and comparability with the CSE-CIC-IDS-2018 datasets, all attack setups in this investigation were duplicated using the tools and procedures outlined on the UNB website [3]. Two virtual machines (VMs) running VMware Workstation 17 Player function as the data collection environment. The first VM, the "Victim," operates on CentOS 7 with two gigabytes of RAM and two virtual processors (vCPUs). SSH and FTP servers are set up and operational in this virtual machine. It also has the CIC Flowmeter tool, which is used for collecting data packets related to brute force attacks. The second VM, known as the "Attacker," runs Kali Linux and possesses similar specifications, including two gigabytes of RAM and two virtual CPUs.

Under the setup shown in Figure 1, the Attacker VM uses the Patator tool to execute brute-force attacks on the SSH and FTP services configured and operational on the Victim VM. Using a list of usernames and passwords, the Attacker VM attempts repeated login. These brute-force attacks produce network traffic that the CIC Flowmeter on the Victim VM records and saves in CSV form. Under a controlled environment, this configuration lets the simulation and monitoring of brute force assaults on SSH and FTP services.

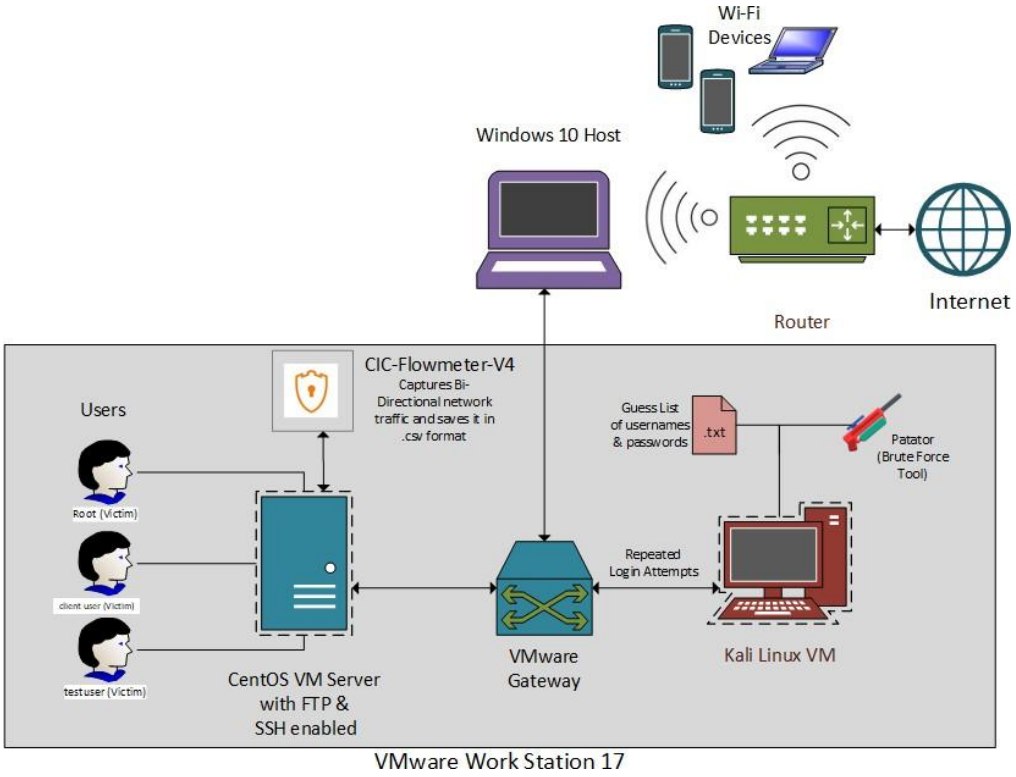


FIGURE 1. SSH and FTP Brute Force attacks setup.

While set up for Web Attacks, the Attacker VM in Figure 2 executes a range of web attacks, including SQL injection, cross-site scripting (XSS), and brute force attacks, on the DVWA platform using automated and Python scripts coupled with the Selenium framework. The Damn Vulnerable Web Application (DVWA) is installed on the victim virtual machine. Scripts are used to automate the attacks, which continuously bombard the web application running on the victim virtual machine. The Victim VM captures the network traffic for these attacks in a CSV file after the CIC Flowmeter records it. This process allows the simulation and monitoring of several web attacks in a controlled environment.

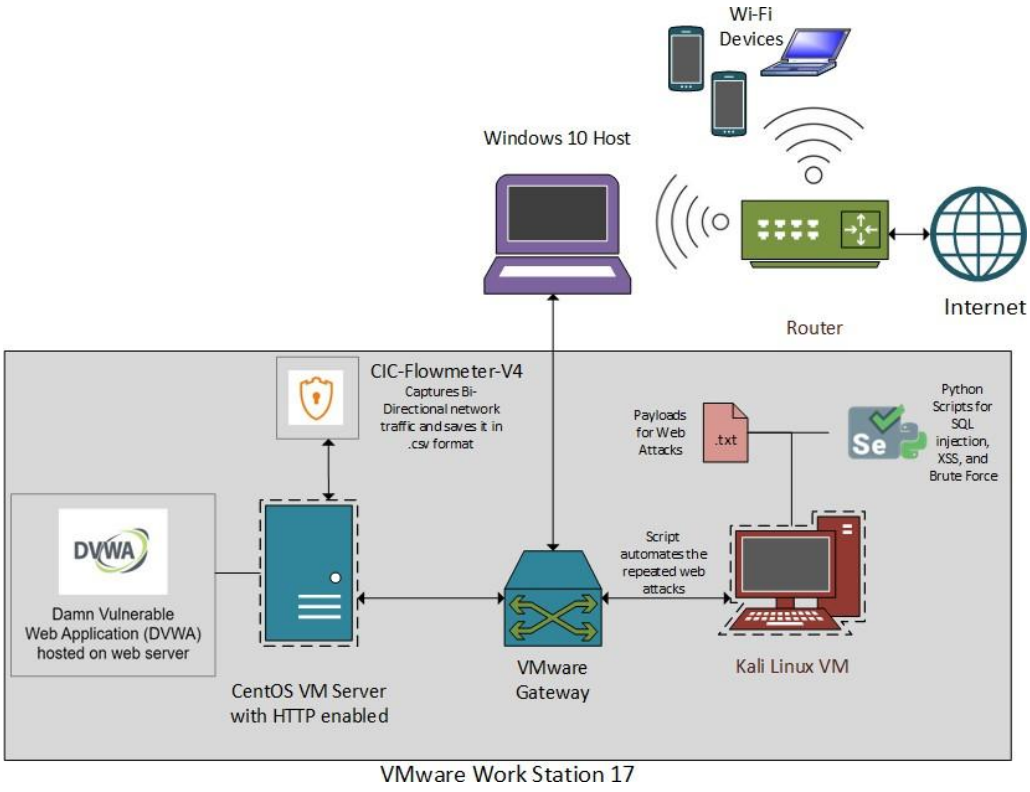


FIGURE 2. **Web attacks setup.**

In the Denial of Service (DoS) attack scenario shown in Figure 3, the Attacker VM utilizes tools like GoldenEye and Slowloris to execute an HTTP Flood Attack on the Victim VM. CIC Flowmeter collects the network traffic produced by these DoS attacks on the Victim VM. Worth mentioning is that the CIC Flowmeter saves the data in CSV format. This configuration provides an isolated environment for simulating as well as monitoring multiple DoS attacks.

Later in these cases, monitor mode is configured on the system's interface to collect normal network data. This configuration gives you all 802.11 (wireless) data and also tries to get wired data, which gives you a broad baseline of normal network activity. This baseline is important to differentiate normal from malicious traffic in the network. The data collected, when referenced for training machine learning models, improves the accuracy of intrusion detection systems. In addition, to train and test the machine learning algorithms, it is necessary to have adequate data to simulate the different cyber-attacks.

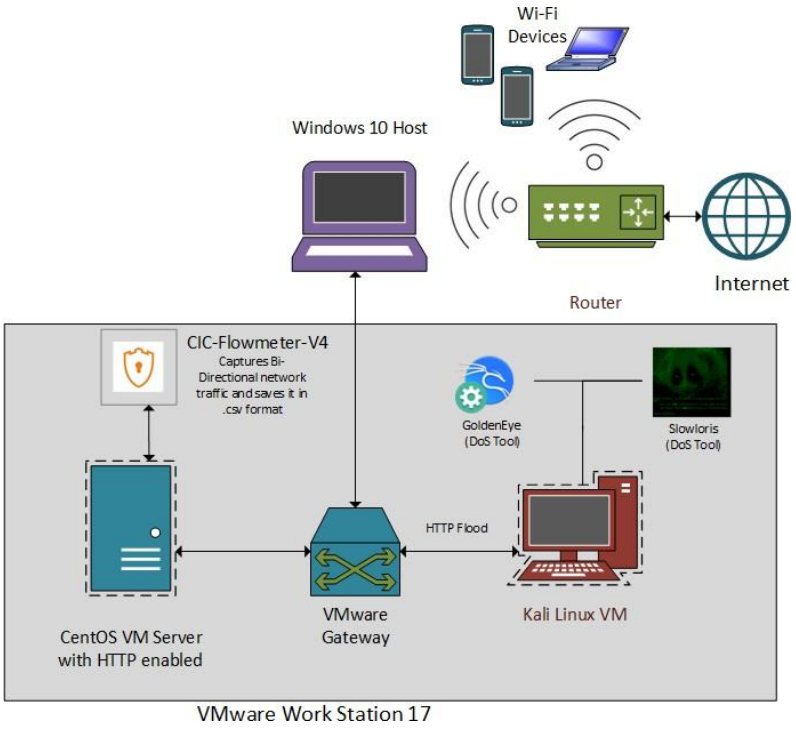


FIGURE 3. **Denial of Service (DoS) attacks setup.**

V. METHODOLOGY

The method, described in Figure 4, relies on two datasets: the CSE-CIC IDS 2018 and a custom-generated dataset. This begins with data preprocessing, including fixing missing values, removing duplicate rows, dropping categorical columns, and removing null values or columns with 0. Then, feature scaling is performed via normalization and label encoding techniques to ready the data for machine learning-based models.

Next, the Random Forest algorithm is used for feature selection to find the important features that improve model accuracy. The next part includes training and testing several ML models such as Decision Trees, Random Forest, MLP (Neural Networks), K-nearest Neighbors (KNN), Naïve Bayes, and XGBoost using these selected features.Last but not least, we compare these models' performances to determine which one works best for intrusion detection.

Python 3.11 was used for data preparation, handling, preprocessing, analysis, model training, and metrics evaluation on the Google Colab platform. Scikit-Learn, Pandas, and Numpy were the libraries used to develop and assess the model. Specifically, Scikit-Learn was utilized for model training, assessment, and metric computation, while Pandas and Numpy were useful for data preparation and manipulation. Data visualization was done using Microsoft Excel and the Seaborn library. The next sections of this study provide more details on the research methodology.

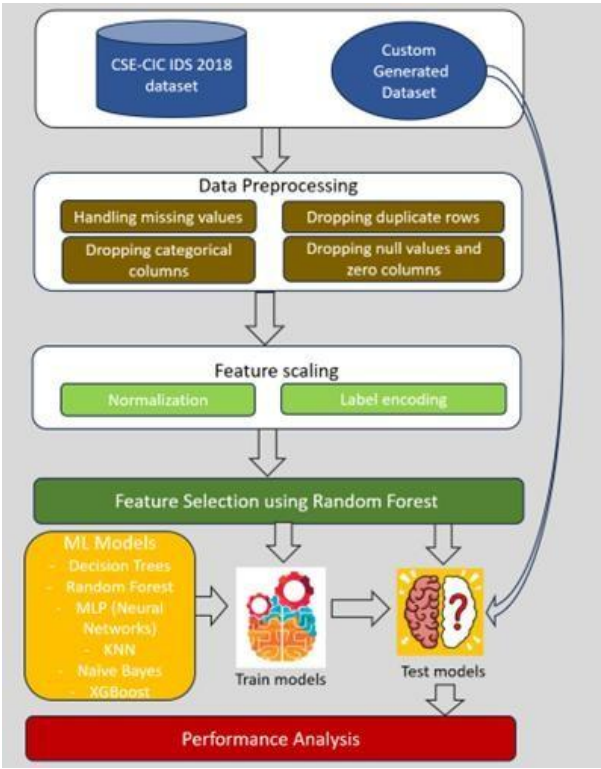


FIGURE 4. **Proposed Methodology.**

A. DATA COLLECTED

The main dataset for this research is a customized dataset generated through CICFlowMeter-V4 [26]. This dataset mimics the features of cyber-attacks, specifically for research improvement purposes. This allows for more in-depth analysis and experimentation toward enhancing attack detection mechanisms. After data collection, the dataset was labeled as per the CSE-CIC-IDS 2018 data for consistency and ease for comparison.

Alongside the custom dataset, this research uses CSE-CIC-IDS 2018, a benchmark dataset comprising various benign and malicious intents that depict real-time traffic flows across networks. This dataset is generated using CICFlowMeter-V3, and it includes 83 features related to network flow directions, number of packets, labels, and FlowID. Among these features, five of them are categorical (SourceIP, DestinationIP, SourcePort, DestinationPort, Protocol), and the other 78 (according to the initial data) are continuous data. The dataset is provided as CSV files, which makes it easy to integrate into machine learning pipelines [3].

In addition, we focused on refining the dataset to contain only specific subtypes of attacks on well-defined dates in February 2018 in the CIC-IDS dataset: 'FTP-BruteForce' and 'SSH-Bruteforce' attacks conducted on Wednesday, February 14, 2018, 'DoS-GoldenEye' and 'DoS-Slowloris' attacks on Thursday, February 15, 2018, and 'Brute Force-Web,' 'Brute Force-XSS,' and 'SQL Injection' attacks which occurred on Thursday, February 22, 2018, and Friday, February 23, 2018. As shown in Table 1 [3], the distribution of records in the CSE-CIC-IDS 2018 dataset.

TABLE I
DISTRIBUTION OF RECORDS IN THE CIC-IDS 2018 DATASET

Label	Number of Records
Benign	3,759,925
FTP-BruteForce	193,360
SSH-Bruteforce	187,589
DoS-GoldenEye	41,508
DoS-Slowloris	10,990
BruteForce-Web	611
BruteForce-SQLInjection	87
BruteForce-XSS	230

After collecting the data, we labeled them accordingly: 'Benign' for normal traffic, 'FTP-BruteForce' for FTP brute force attacks, 'SSH-Bruteforce' for SSH brute force attacks, 'DoS-GoldenEye' for Denial of Service attacks using the GoldenEye tool, 'DoS-Slowloris' for Denial of Service attacks using the Slowloris tool, 'BruteForce-Web' for brute force attacks on DVWA, 'BruteForce-SQLInjection' for repeated SQL injection attacks on DVWA, and 'BruteForce-XSS' for repeated XSS attacks on DVWA. The distribution of records in the custom-generated dataset is provided in Table 2.

TABLE 2
DISTRIBUTION OF RECORDS IN THE CIC-IDS 2018 DATASET

Label	Number of Records
Benign	24,760
FTP-BruteForce	1,874
SSH-Bruteforce	1,013
DoS-GoldenEye	4,662
DoS-Slowloris	4,727
BruteForce-Web	2,678
BruteForce-SQLInjection	551
BruteForce-XSS	1,260

B. DATA PREPROCESSING

In the data preprocessing phase, several steps are taken to clean and prepare the datasets for machine learning analysis. Initially, unnecessary columns are dropped to streamline the datasets; the 'Flow ID', 'Src IP', 'Src Port', 'Dst IP', and 'Timestamp' columns are discarded. Datasets are then checked for values equal to infinity, which are replaced with `NaN` to process any anomalies. The datasets are checked for `NaN` values to make sure all missing data has been detected, and any rows that contain them are removed to provide clean datasets. Additionally, to get rid of non-informative data, columns in the training dataset that have only zeros in every row are found and eliminated. These preprocessing procedures are crucial for transforming the datasets for accurate and quick machine learning model training.

C. FEATURE SCALING

Label encoding and normalization techniques were employed in the feature scaling procedure to prepare the datasets ready for feature selection technique. Label Encoding involves converting the categorical labels into a set of integer values [20]. To facilitate the target variable's evaluation by the models, the categorical 'Label' column in both datasets was first converted into numerical format using label encoding. A LabelEncoder was fitted to the 'Label' column to achieve this. It is possible to over-polarize any one feature, which could affect the learning process, due to our scaling normalization technique, which guarantees that every feature contributes equally to the model [13]. The MinMaxScaler was then used to carry out normalization, which scaled the feature values within a specified range, often between 0 and 1. To maintain feature scaling consistency across the two datasets, the MinMaxScaler was first fitted and transformed on the training set, and then the same transformation was applied to the test set. These actions are essential for improving machine learning algorithms' performance and convergence of datasets, particularly for those that are sensitive to input feature scaling.

D. FEATURE SELECTION

The Random Forest technique, which uses entropy gain to determine feature importance, was used for selecting features in the proposed approach. Entropy gain, also referred to as information gain, quantifies the reduction in entropy or uncertainty in the data when the dataset is split based on a specific feature. Features that yield higher entropy

gain are deemed more critical for the prediction task. The Random Forest classifier inherently provides a measure of feature importance, allowing for the ranking and selection of the most impactful features for constructing machine learning models. In this study, the top 15 features were chosen based on their importance scores.

The following figures 5, 6, and 7 illustrate the feature importance for different types of attacks on two different datasets as determined by the Random Forest classifier:

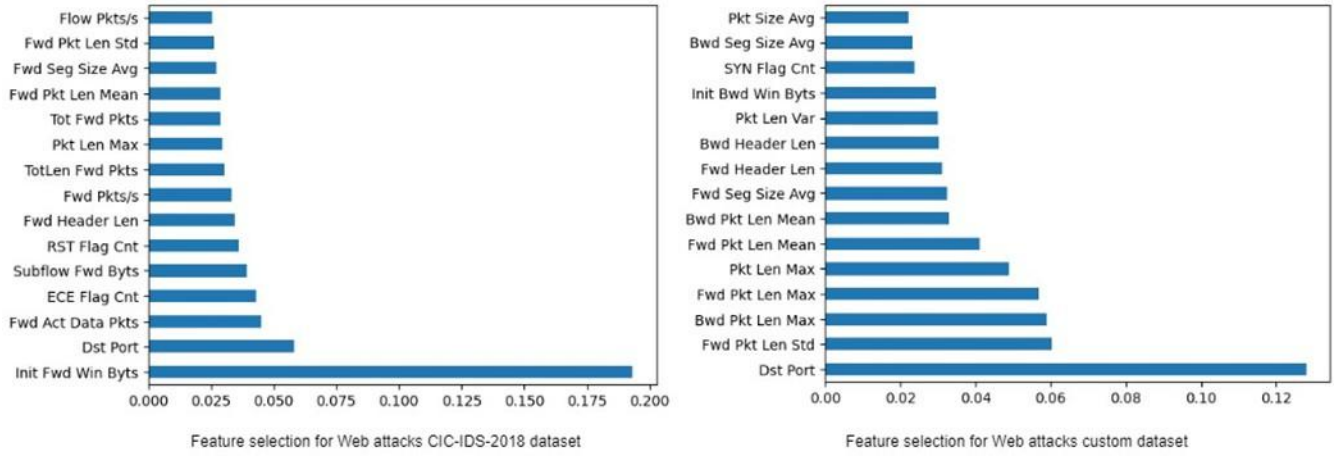


FIGURE 5. Feature Selection for Web Attacks: CIC-IDS-2018 Dataset vs. Custom Dataset.

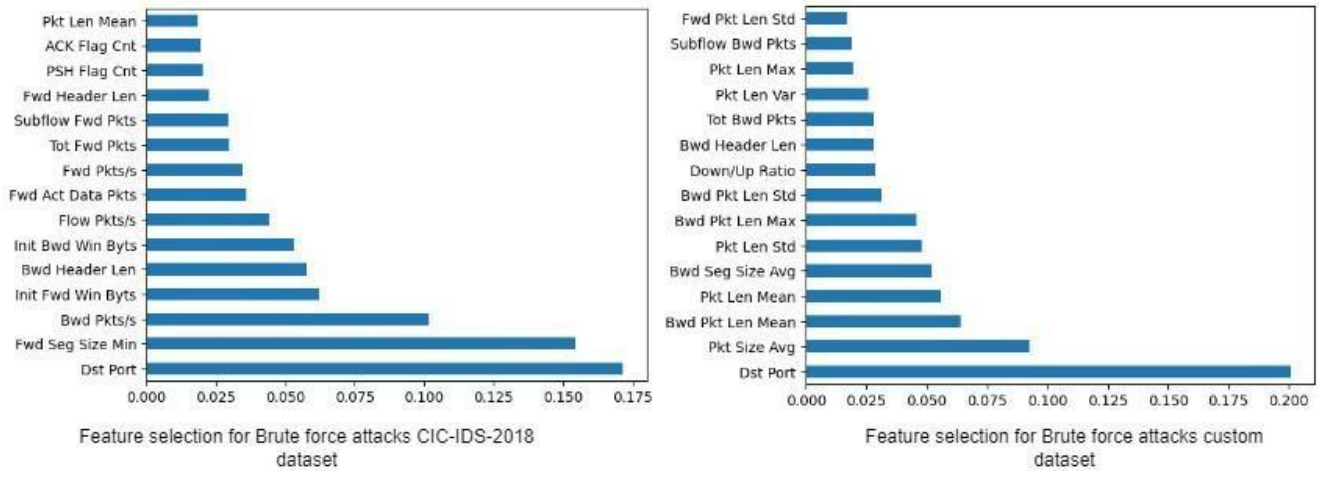


FIGURE 6. Feature Selection for Brute Force Attacks: CIC-IDS-2018 Dataset vs. Custom Dataset.

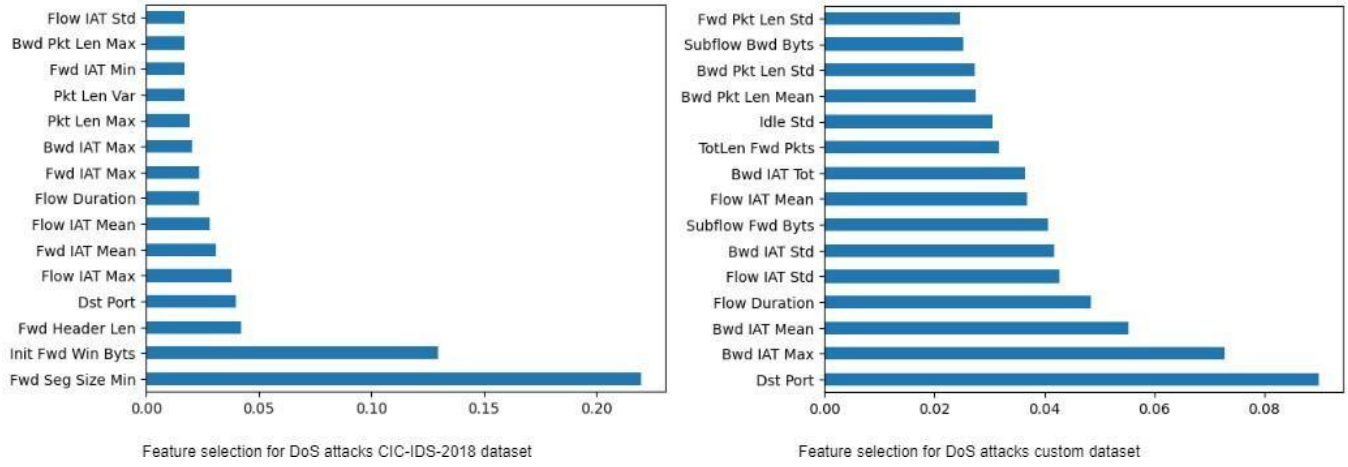


FIGURE 7. Feature Selection for DoS Attacks: CIC-IDS-2018 Dataset vs. Custom Dataset.

The analysis of feature importance reveals that different types of attacks exhibit distinct sets of important features. Each type of attack, such as Web, Brute Force, and DoS attacks, demonstrates unique patterns in the data. Because of this, the characteristics that are most crucial for identifying one kind of attack might not be as relevant for another. The feature importance rankings for the different attack types make this clear.

The CSE-CIC-IDS-2018 dataset and the custom-generated dataset have different attributes that are considered relevant, even for the same type of attack. This discrepancy is likely due to differences in network architecture. Understanding the network environment and data characteristics is crucial for developing robust and effective models capable of accurately detecting a wide range of network attacks.

VI. RESULTS

Two datasets—the CSE-CIC-IDS-2018 dataset and a custom-generated dataset—were used to assess machine learning models. The findings showed that the models' prediction abilities varied significantly based on the dataset they were trained on. The performance differences between models trained on the CSE-CIC-IDS-2018 dataset and those trained on the custom-generated dataset are graphically depicted by the confusion matrices given below. The classification results for a particular attack type across various machine learning models are displayed in each matrix.

During testing, models trained on the CSE-CIC-IDS-2018 dataset had trouble correctly predicting different kinds of attacks. The confusion matrices, which display a high rate of misclassification across several attack situations, make this clear. The models had difficulty generalizing from the CSE-CIC-IDS-2018 dataset to the test data generated in our setups, as reflected in the confusion matrices presented in Figures 8, 9, and 10.

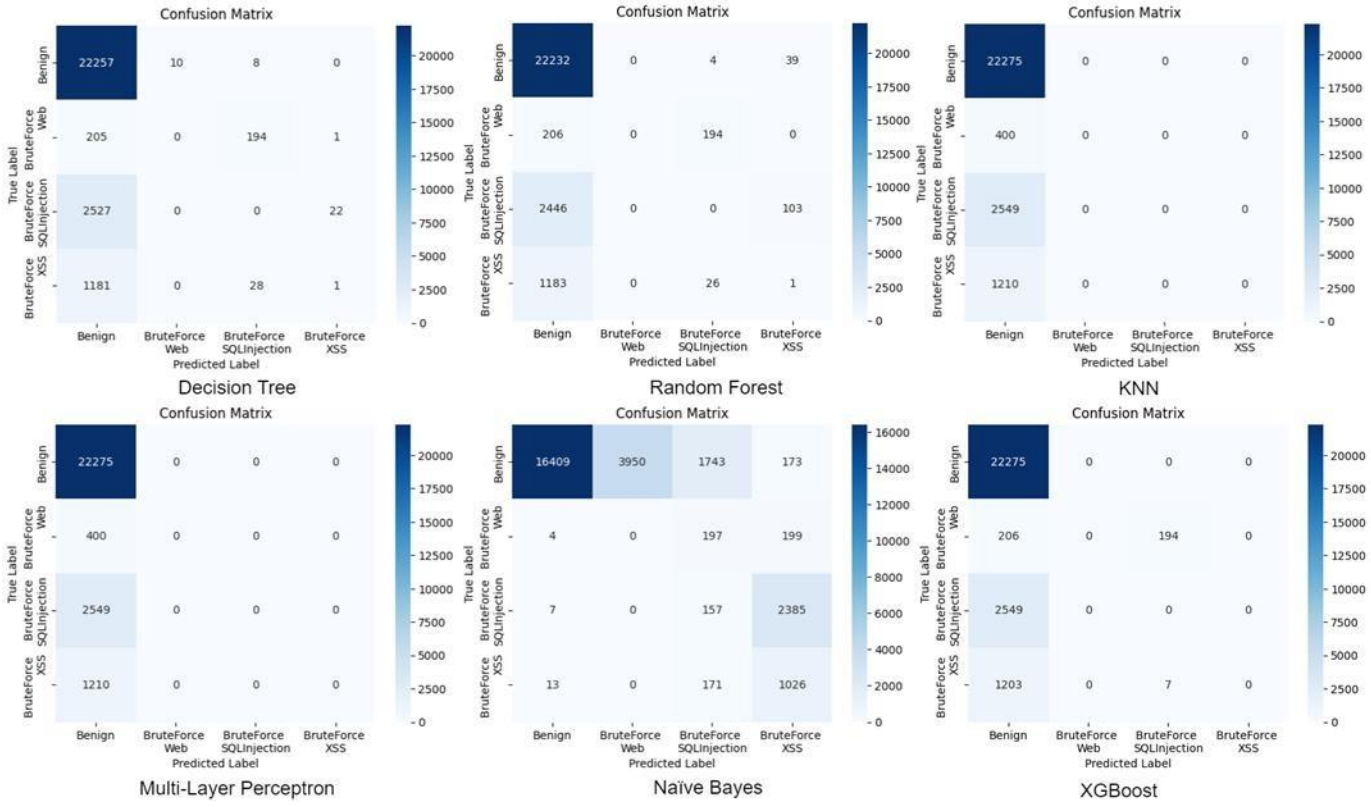


FIGURE 8. Confusion Matrices for Different Machine Learning Algorithms trained on Web Attacks CIC-IDS-2018 Dataset.

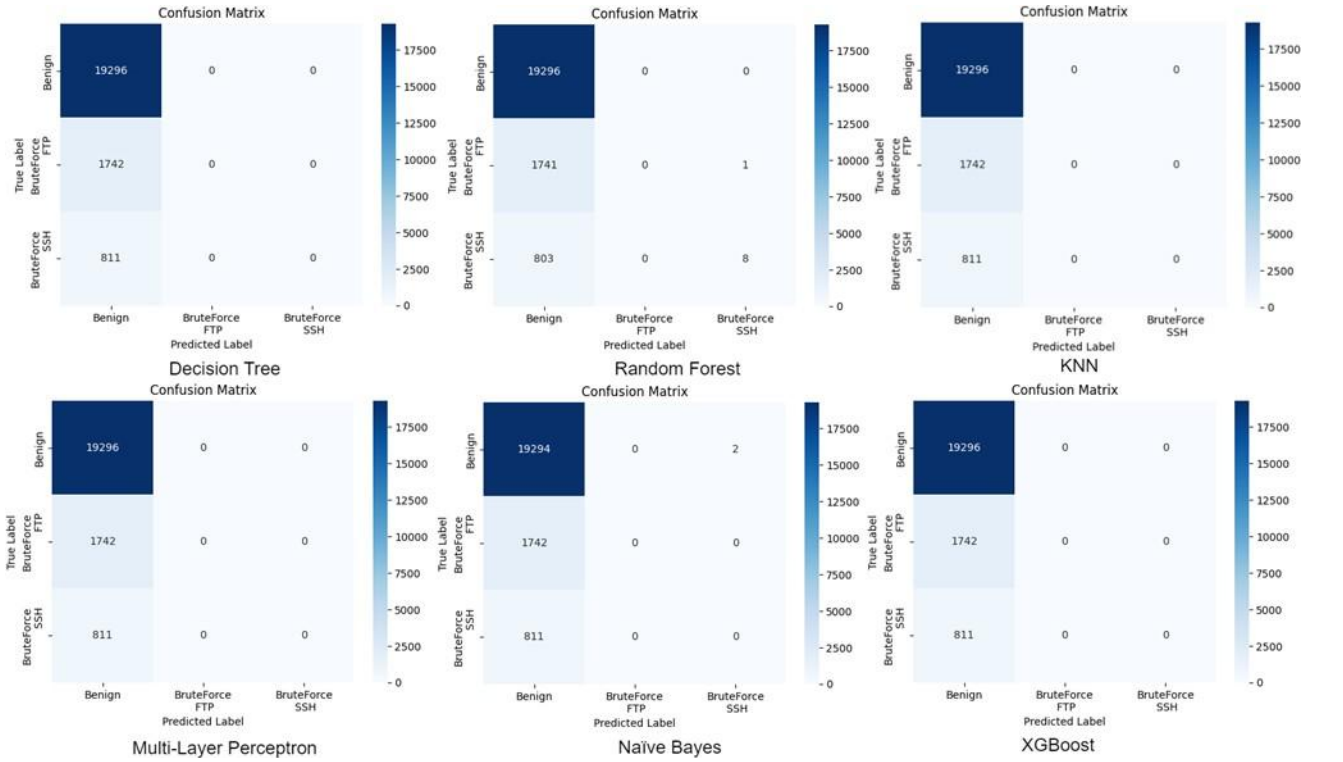


FIGURE 9. Confusion Matrices for Different Machine Learning Algorithms trained on Brute Force Attacks CIC-IDS-2018 Dataset.

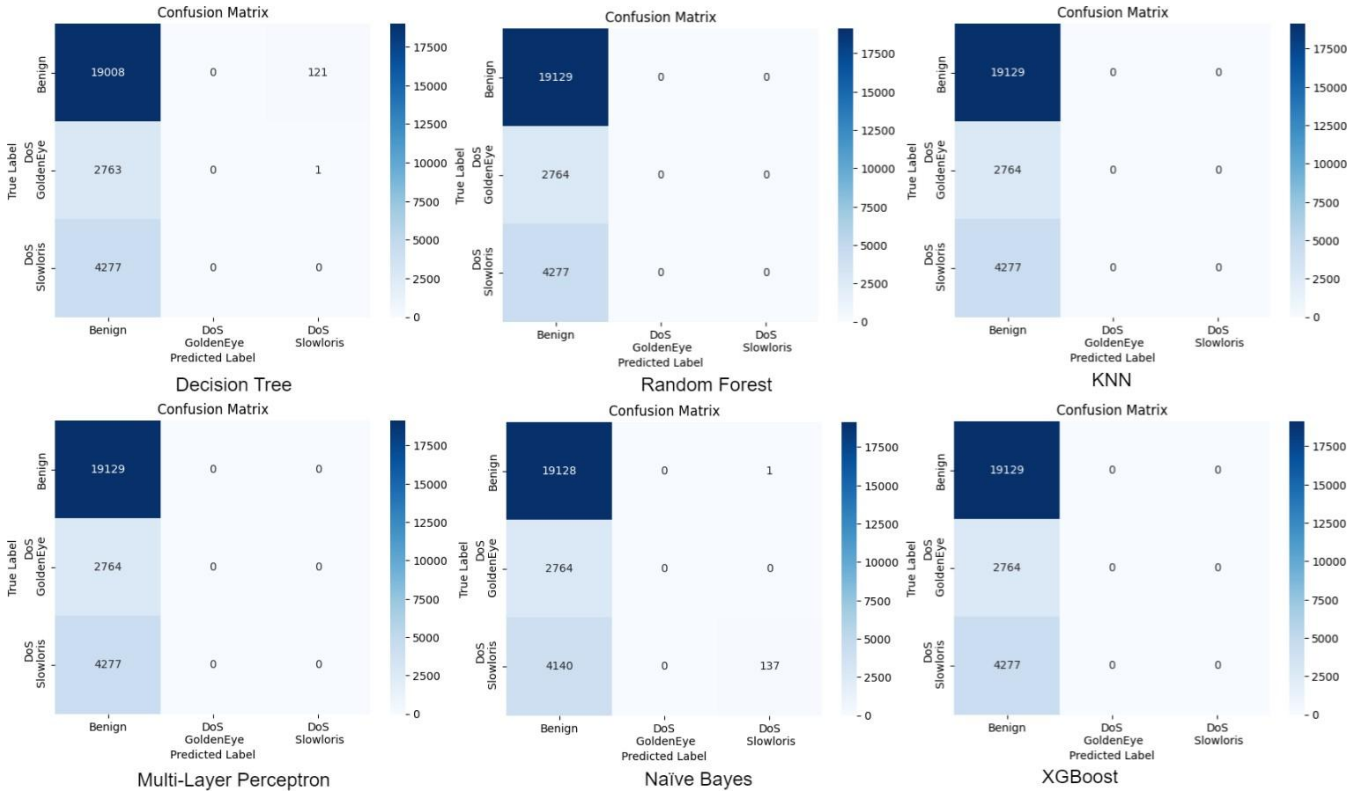


FIGURE 10. Confusion Matrices for Different Machine Learning Algorithms trained on DoS Attacks CIC-IDS-2018 Dataset.

In contrast, when the models were trained on the custom-generated dataset, they exhibited a high level of accuracy in predicting the attacks. The confusion matrices for different attack types and machine learning models clearly indicate that the models were able to correctly classify the majority of the attack instances. This superior performance can be

attributed to the custom dataset's closer alignment with the test data, which was also generated using our specific network setups, as it is displayed in the confusion matrices from Figures 11, 12, and 13.

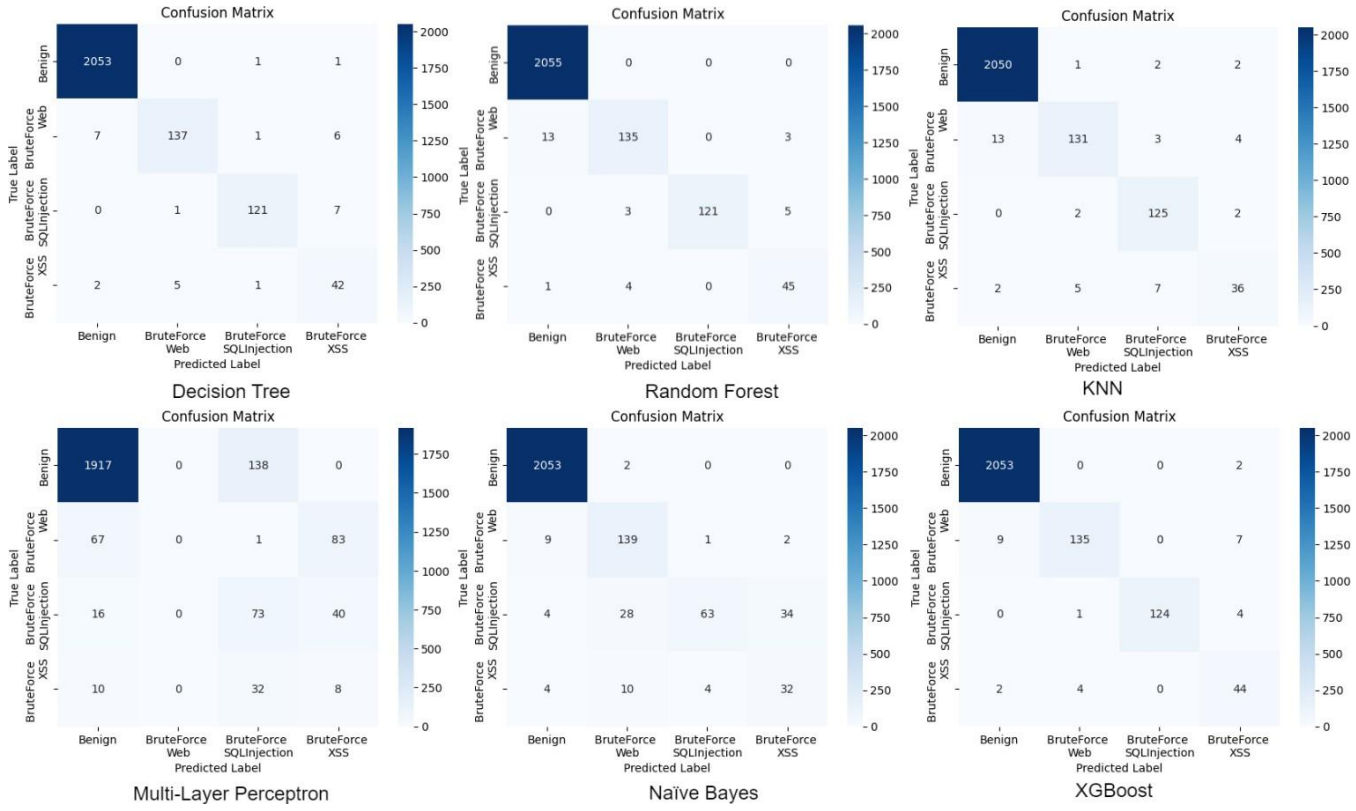


FIGURE 11. **Confusion Matrices for Different Machine Learning Algorithms trained on Web Attacks Custom Dataset.**

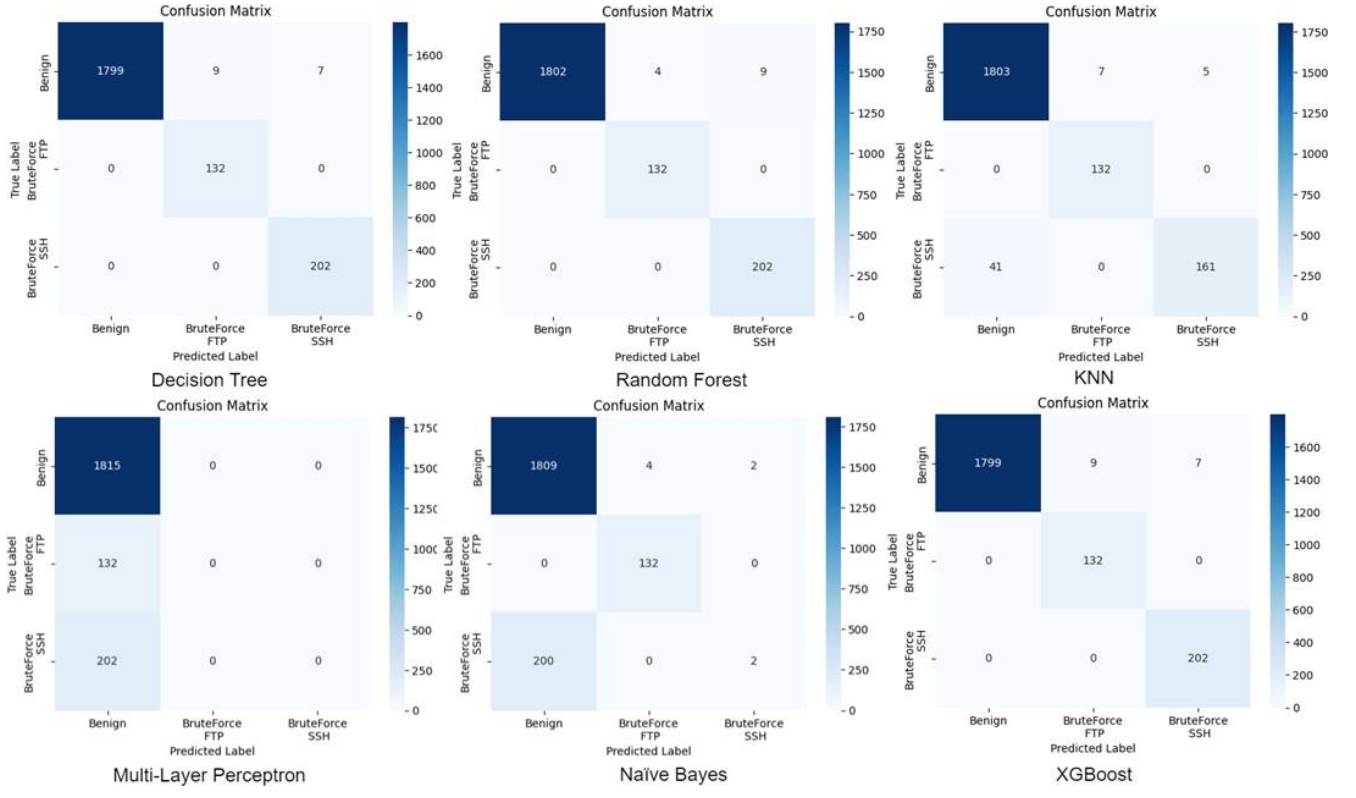


FIGURE 12. **Confusion Matrices for Different Machine Learning Algorithms trained on Brute Force Attacks Custom Dataset.**

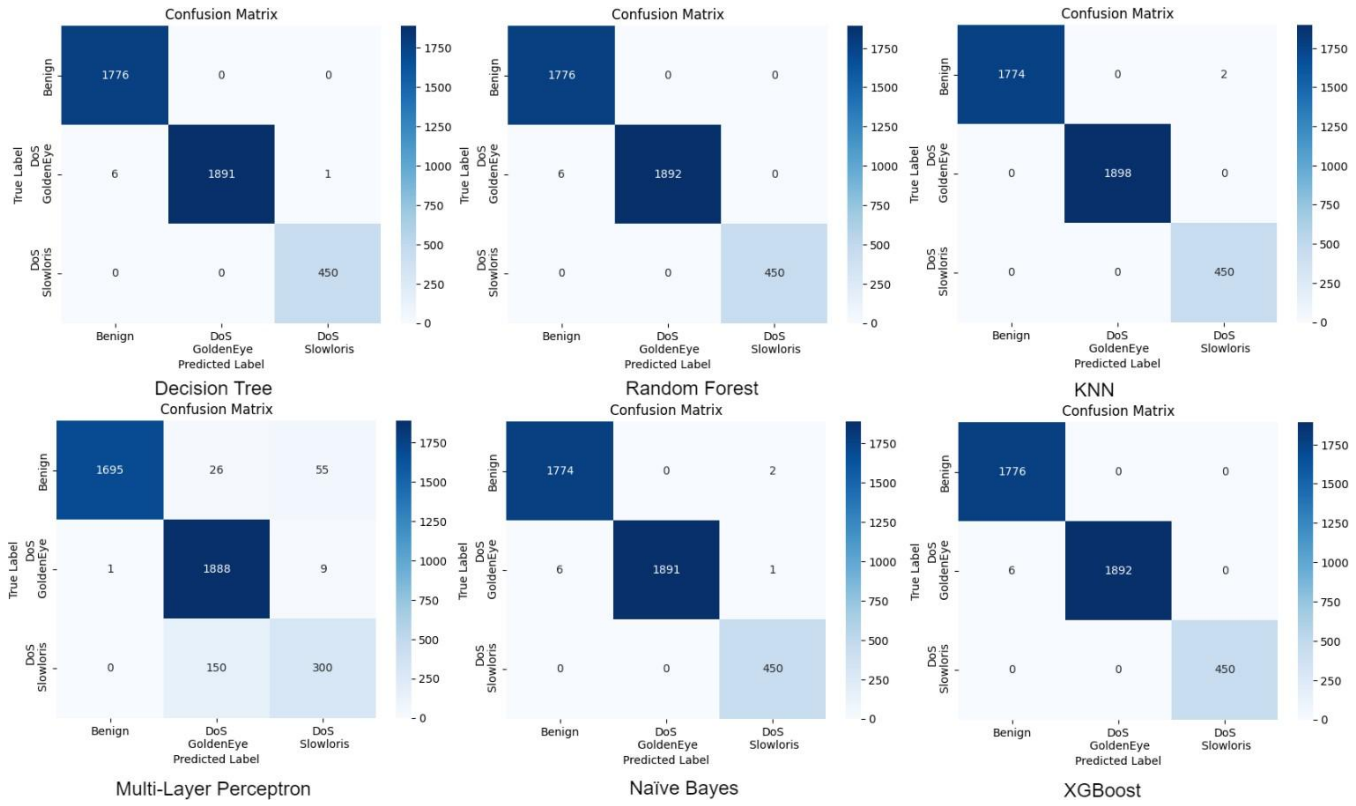


FIGURE 13. **Confusion Matrices for Different Machine Learning Algorithms trained on DoS Attacks Custom Dataset.**

For testing the models, we utilized data generated by our setups for each type of attack. The custom dataset likely captured the nuances and specific patterns present in the test data, allowing the models to learn and generalize more effectively. This finding highlights the need for using representative and context-specific data for training machine detection systems using machine learning.

One of the main reasons for the loss in model performance resides in the differences between the network architectures provided as the source model for the CSE-CIC-IDS-2018 dataset and the powerful custom dataset that was generated. Although the same network collection methodology was used, the CSE-CIC-IDS-2018 dataset is collected from a comprehensive source, whereas the custom dataset was formed through a significantly less comprehensive imaging network setup. This disparity resulted in some of the network parameters becoming more significant in the custom dataset. Moreover, correlations between features are accumulated, and signatures were found to differ significantly from the network architecture of the CICFlowmeter. Consequently, the sensitive attributes used for training, based on the same type of attack machine learning model, have changed, affecting the predictive performance of the model.

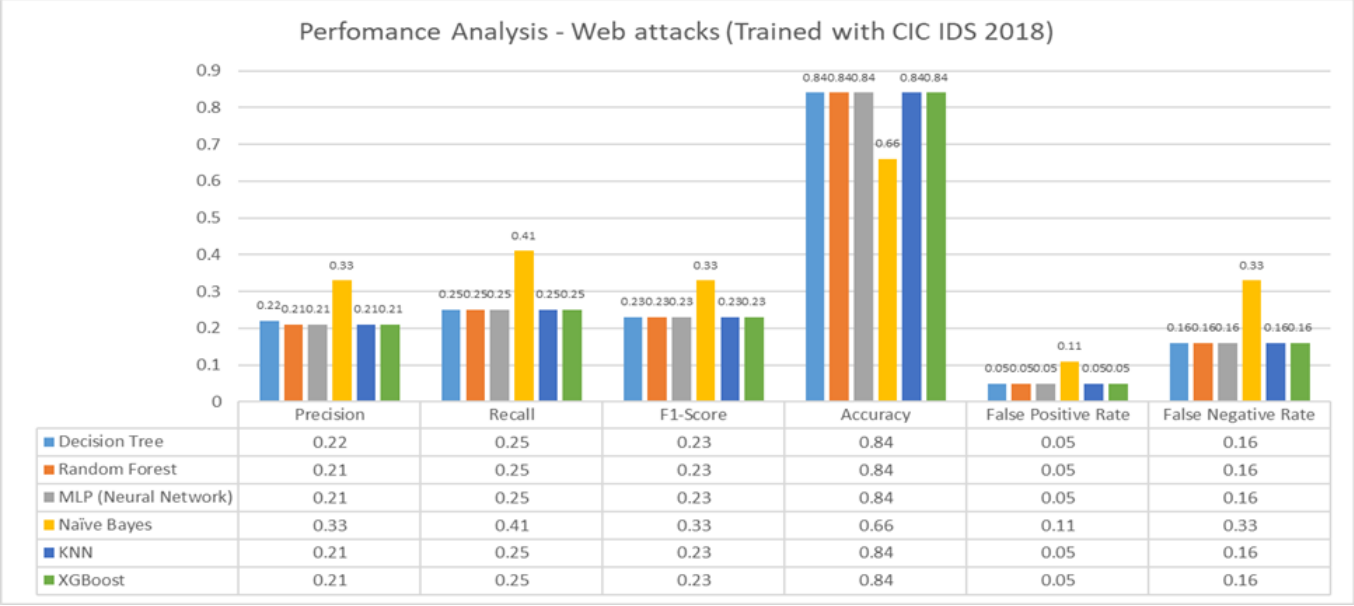
VII. PERFORMANCE ANALYSIS

Next, we take a look at the performance of multiple machine learning models—XGBoost, Decision Tree, Random Forest, MLP (Neural Networks), KNN, and Naïve Bayes—trained on the CIC-IDS 2018 dataset and a custom-generated dataset for the detection of different types of cyber attack types, such as Web attacks, DoS, and Brute Force attacks. A variety of metrics are used to evaluate the models, including precision, recall, F1 score, accuracy, and false positive and false negative rates. Beyond evaluating them, the models are compared with each other not only on the accuracy of the models but also on their training and prediction times to get a pretty good understanding of their efficiency. This analysis gives some ideas about the efficacy and computational efficiency of each classification of different types of attacks by a model under controlled conditions.

Web Attacks trained on the custom dataset (Figure 14) as well as the CIC-IDS-2018 dataset (Figure 15) shows its accuracy and effectiveness on the custom dataset. In Figure 14, models such as Decision Tree, Random Forest, and XGBoost provide high precision, recall, F1 score, and predictive power, achieving high levels of accuracy of 0.99 on XGBoost and Random Forest, along with low false positive and negative rates. Conversely, MLP (Neural Network) shows lower performance, with a precision of 0.33 and an accuracy of 0.84, suggesting it is less suitable for detecting web attacks on this dataset. Figure 15, on the other hand, demonstrates that models trained on the CIC-IDS-2018 dataset perform significantly worse, with precision, recall, and F1-scores ranging from 0.21 to 0.41, and accuracy at 0.84. Additionally, false positive and negative rate is notably higher, particularly for Naïve Bayes, indicating the CIC-IDS-2018 dataset's limitations in accurately capturing web attack patterns.

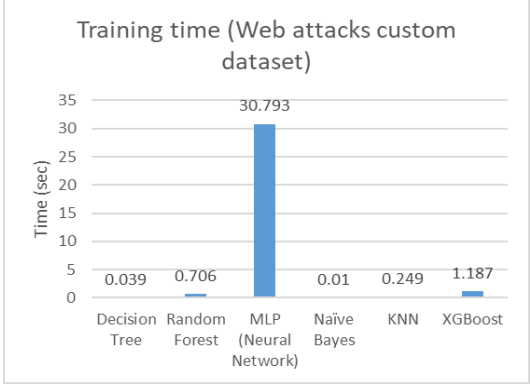
	Precision	Recall	F1-Score	Accuracy	False Positive Rate	False Negative Rate
Decision Tree	0.92	0.92	0.92	0.99	0.01	0.02
Random Forest	0.95	0.93	0.94	0.99	0.01	0.02
MLP (Neural Network)	0.33	0.36	0.36	0.84	0.05	0.16
Naïve Bayes	0.79	0.76	0.76	0.96	0.01	0.04
KNN	0.91	0.89	0.9	0.98	0.01	0.02
XGBoost	0.93	0.93	0.93	0.99	0.01	0.01

FIGURE 14. **Performance Analysis of Web Attacks Detection Models (Trained with Custom Dataset).**

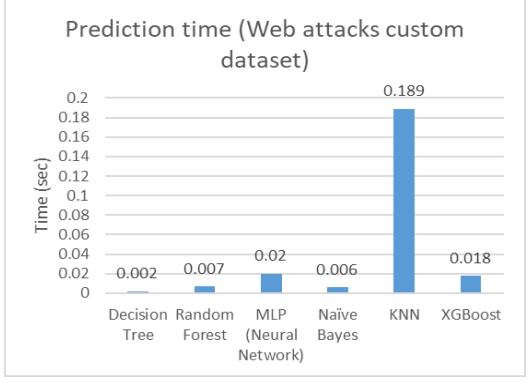


	Precision	Recall	F1-Score	Accuracy	False Positive Rate	False Negative Rate
Decision Tree	0.22	0.25	0.23	0.84	0.05	0.16
Random Forest	0.21	0.25	0.23	0.84	0.05	0.16
MLP (Neural Network)	0.21	0.25	0.23	0.84	0.05	0.16
Naïve Bayes	0.33	0.41	0.33	0.66	0.11	0.33
KNN	0.21	0.25	0.23	0.84	0.05	0.16
XGBoost	0.21	0.25	0.23	0.84	0.05	0.16

FIGURE 15. **Performance Analysis of Web Attacks Detection Models (Trained with CIC-IDS 2018).**



(a)



(b)

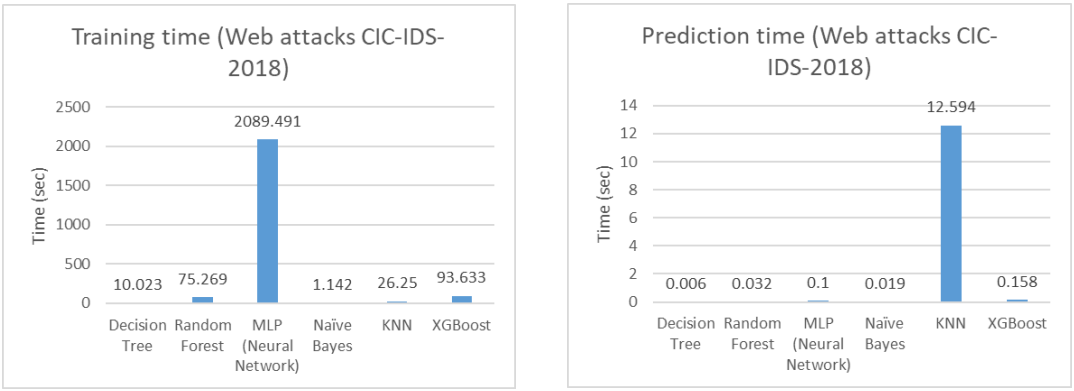
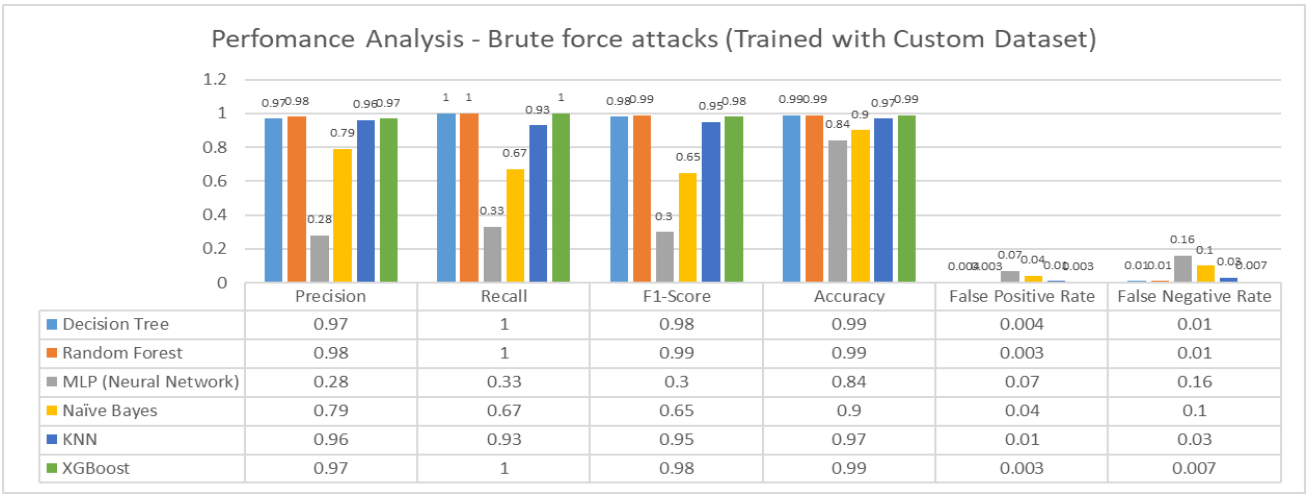


FIGURE 16. **Computational Time comparison for Web Attacks: (a) Training Time of Web Attacks Detection Models (Custom Dataset); (b) Prediction Time of Web Attacks Detection Models (Custom Dataset); (c) Training Time of Web Attacks Detection Models (CIC-IDS 2018 Dataset); (d) Prediction Time of Web Attacks Detection Models (CIC-IDS 2018 Dataset).**

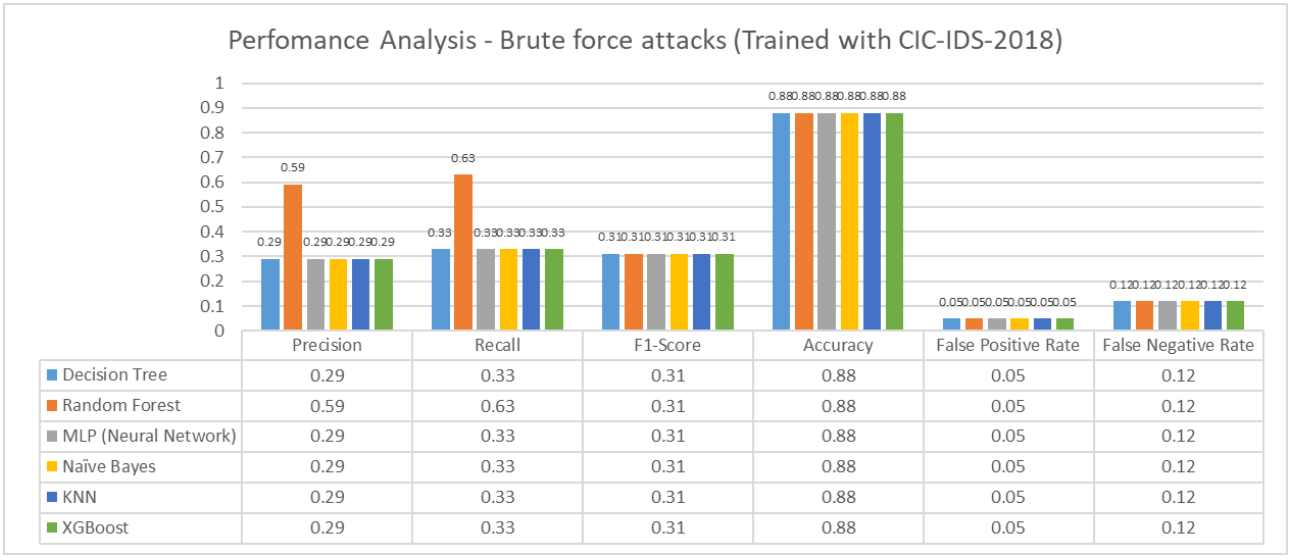
In addition, the computational efficiency of these models (Figures 16a, 16b, 16c, and 16d) shows significant differences in training and prediction times. MLP (Neural Network) consistently takes the longest to train, requiring 30.793 seconds on the custom dataset (Figure 16a) and an extensive 2089.491 seconds on the CIC-IDS-2018 dataset (Figure 16c), making it unsuitable for real-time applications. KNN, while relatively quick to train, exhibits the longest prediction times, especially on the CIC-IDS-2018 dataset, where it takes 12.594 seconds (Figure 16d). In contrast, Decision Tree, Random Forest, and XGBoost offer a better balance, with short training and prediction times on both datasets. Naïve Bayes stands out with the fastest training and prediction times, particularly on the custom dataset, where both phases take almost 0.01 seconds, though it is less effective in detection accuracy.

The performance comparison between brute force attacks trained on the custom dataset (Figure 17) and the CIC-IDS-2018 dataset (Figure 18) demonstrates a significant improvement in model accuracy and effectiveness with the custom dataset. In Figure 17, models like Decision Tree, Random Forest, and XGBoost achieve near-perfect precision, F1-score, recall, and accuracy (0.99), along with an almost 0% false positive and negative rate. In contrast, Figure 18 shows that models trained on the CIC-IDS-2018 dataset perform poorly, with a uniform drop in precision, recall, and F1-score (around 0.29 to 0.33), and accuracy at 0.88. The CIC-IDS-2018 dataset also exhibits higher false positive and false negative rates (0.05 and 0.12), highlighting its limitations in accurately capturing brute force attack characteristics.



	Precision	Recall	F1-Score	Accuracy	False Positive Rate	False Negative Rate
Decision Tree	0.97	1	0.98	0.99	0.004	0.01
Random Forest	0.98	1	0.99	0.99	0.003	0.01
MLP (Neural Network)	0.28	0.33	0.3	0.84	0.07	0.16
Naïve Bayes	0.79	0.67	0.65	0.9	0.04	0.1
KNN	0.96	0.93	0.95	0.97	0.01	0.03
XGBoost	0.97	1	0.98	0.99	0.003	0.007

FIGURE 17. **Performance Analysis of Brute Force Attacks Detection Models (Trained with Custom Dataset).**



	Precision	Recall	F1-Score	Accuracy	False Positive Rate	False Negative Rate
Decision Tree	0.29	0.33	0.31	0.88	0.05	0.12
Random Forest	0.59	0.63	0.31	0.88	0.05	0.12
MLP (Neural Network)	0.29	0.33	0.31	0.88	0.05	0.12
Naïve Bayes	0.29	0.33	0.31	0.88	0.05	0.12
KNN	0.29	0.33	0.31	0.88	0.05	0.12
XGBoost	0.29	0.33	0.31	0.88	0.05	0.12

FIGURE 18. **Performance Analysis of Brute Force Attacks Detection Models (Trained with CIC-IDS 2018)**

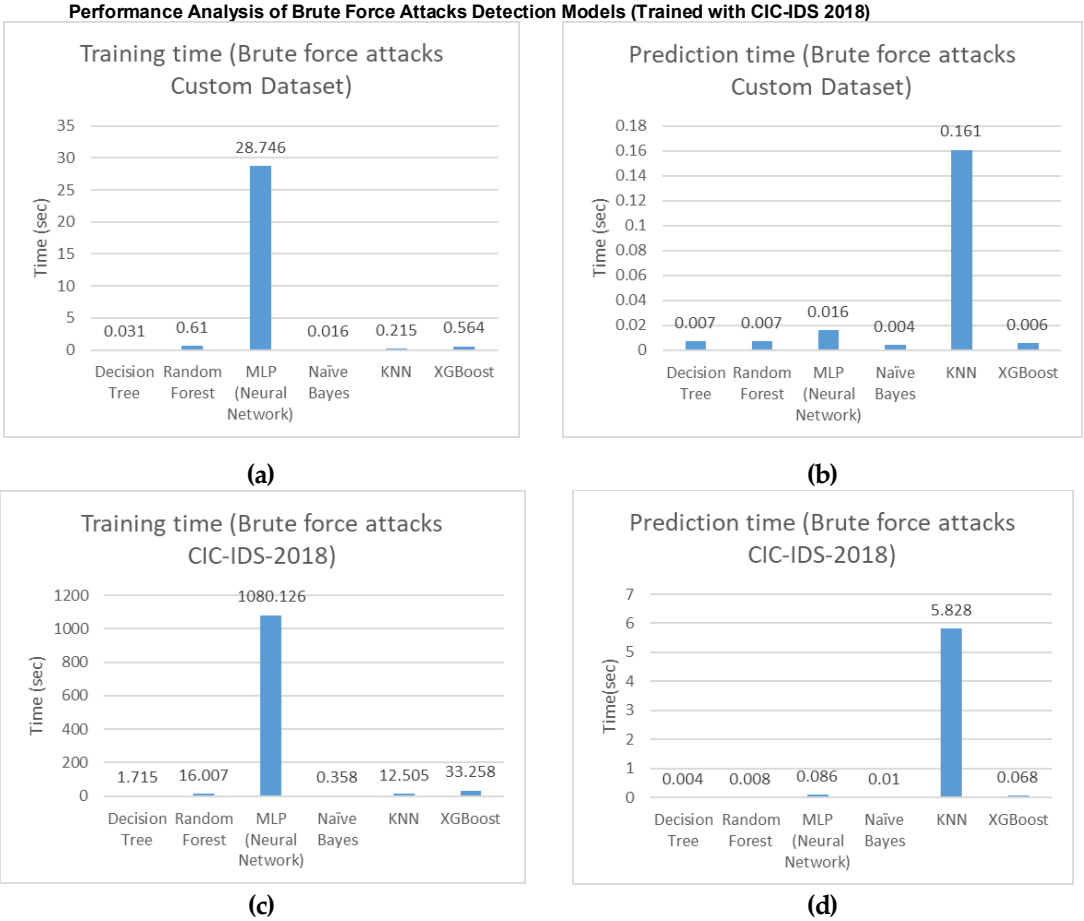


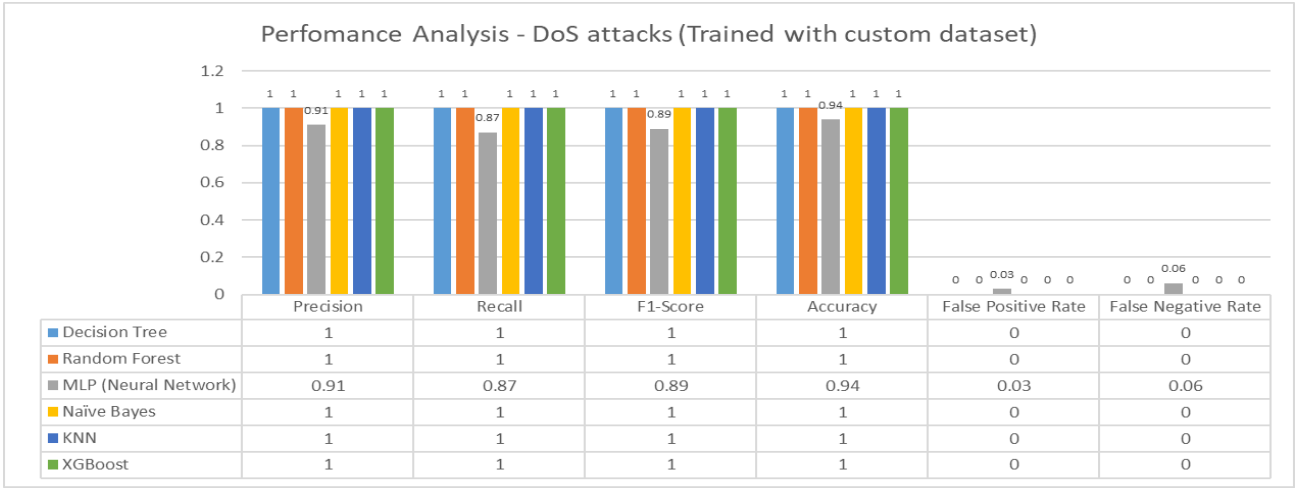
FIGURE 19. **Computational Time comparison for Brute Force Attacks: (a) Training Time of Brute Force Attacks Detection Models (Custom Dataset); (b) Prediction Time of Brute Force Attacks Detection Models (Custom Dataset); (c) Training Time of Brute Force Attacks Detection Models (CIC-IDS 2018 Dataset); (d) Prediction Time of Brute Force Attacks Detection Models (CIC-IDS 2018 Dataset).**

Further insights into the computational efficiency of these models (Figures 19a, 19b, 19c, and 19d) reveal distinct variations in training and prediction times. MLP (Neural Network) consistently demonstrates the longest training and prediction times, making it less practical for real-time intrusion detection, with over 1080 seconds of training time on the CIC-IDS-2018 dataset. KNN, although fast in training, shows the highest prediction time, on the CIC-IDS-2018 dataset (5.828 seconds), limiting its real-time applicability. In contrast, Decision Tree, Random

Forest, and XGBoost provide an optimal balance, with both minimal training times and low prediction times across both datasets. Notably, Naïve Bayes exhibits the fastest training and prediction times, making it computationally efficient, though less effective in terms of detection performance.

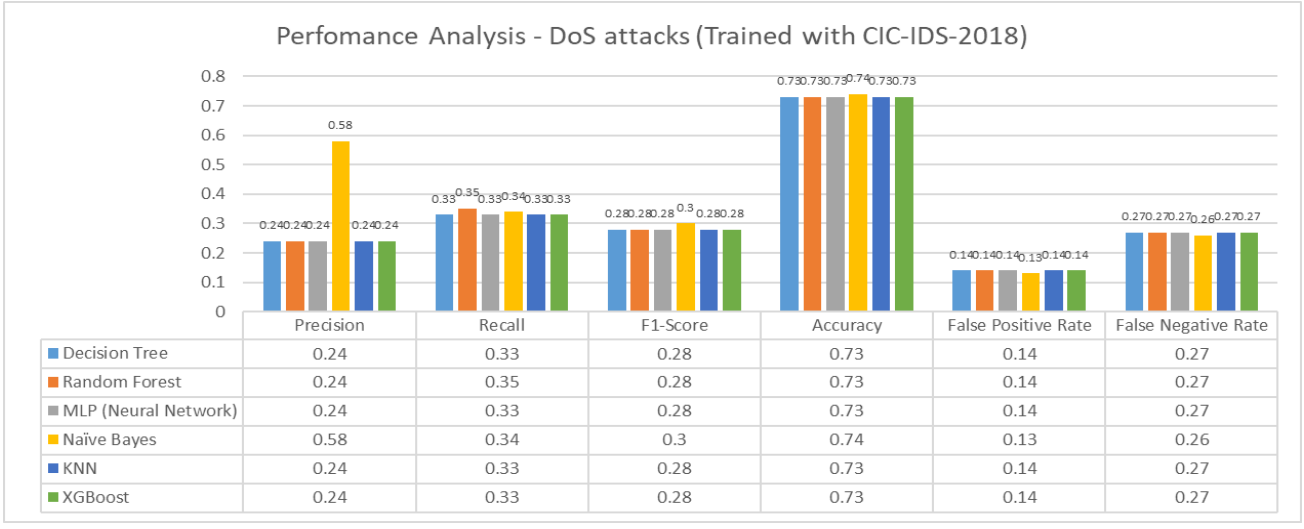
The comparison between models trained to detect DoS attacks on the custom dataset (Figure 20) and the CIC-IDS-2018 dataset (Figure 21) shows a notable improvement in accuracy and effectiveness with the custom dataset. In Figure 20, models like Decision Tree, Random Forest, Naïve Bayes, KNN, and XGBoost achieve perfect precision, recall, F1-scores, and accuracy, with no false positives or negatives. In contrast, MLP (Neural Network) performs slightly worse, with an F1-score of 0.89 and a false negative rate of 0.06, suggesting it's less suited for detecting DoS attacks on this dataset. On the other hand, Figure 21 shows that models trained on the CIC-IDS-2018 dataset perform significantly less effectively, with precision, recall, and F1-scores dropping to around 0.24 to 0.35, and the highest accuracy, achieved by Naïve Bayes, at 0.74. Additionally, false positive and negative rates are higher across all models, especially for Naïve Bayes and Random Forest, indicating the CIC-IDS-2018 dataset's limitations in accurately capturing DoS attack patterns.

Moreover, the computational efficiency of these models (Figures 22a, 22b, 22c, and 22d) shows clear differences in training and prediction times. MLP (Neural Network) consistently has the longest training time, taking 30.107 seconds on the custom dataset (Figure 22a) and a lengthy 982.04 seconds on the CIC-IDS-2018 dataset (Figure 22c), making it impractical for real-time use. KNN, although quick to train, has the longest prediction time, on the CIC-IDS-2018 dataset, where it takes 7.127 seconds (Figure 22d). In contrast, Decision Tree, Random Forest, and XGBoost offer a more balanced approach, with shorter training and prediction times on both datasets. Naïve Bayes stands out with the fastest training and prediction times, on the custom dataset, where both phases take almost 0.01 seconds, although its detection accuracy is lower.



	Precision	Recall	F1-Score	Accuracy	False Positive Rate	False Negative Rate
Decision Tree	1	1	1	1	0	0
Random Forest	1	1	1	1	0	0
MLP (Neural Network)	0.91	0.87	0.89	0.94	0.03	0.06
Naïve Bayes	1	1	1	1	0	0
KNN	1	1	1	1	0	0
XGBoost	1	1	1	1	0	0

FIGURE 20. **Performance Analysis of DoS Attack Detection Models (Trained with Custom Dataset).**



	Precision	Recall	F1-Score	Accuracy	False Positive Rate	False Negative Rate
Decision Tree	0.24	0.33	0.28	0.73	0.14	0.27
Random Forest	0.24	0.35	0.28	0.73	0.14	0.27
MLP (Neural Network)	0.24	0.33	0.28	0.73	0.14	0.27
Naïve Bayes	0.58	0.34	0.3	0.74	0.13	0.26
KNN	0.24	0.33	0.28	0.73	0.14	0.27
XGBoost	0.24	0.33	0.28	0.73	0.14	0.27

FIGURE 21. **Performance Analysis of DoS Attacks Detection Models (Trained with CIC-IDS 2018).**

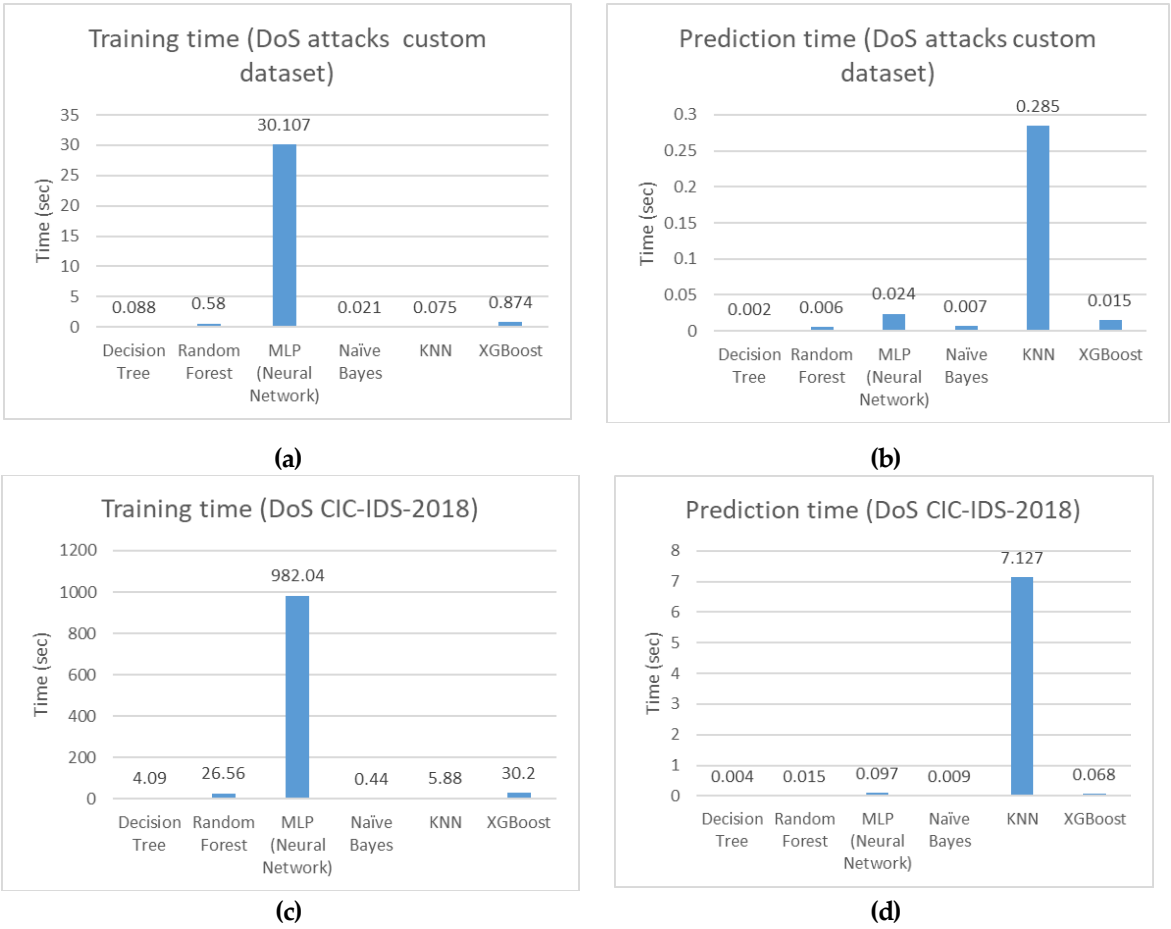


FIGURE 22. Computational Time comparison for DoS Attacks: (a) Training Time of DoS Attacks detection Models (Custom Dataset); (b) Prediction Time of DoS Attacks Detection Models (Custom Dataset); (c) Training Time of DoS Attacks Detection Models (CIC-IDS 2018 Dataset); (d) Prediction Time of DoS Attacks Detection Models (CIC-IDS 2018 Dataset).

VIII. CONCLUSION

In this study, we investigated the efficacy of various machine learning algorithms in the context of Intrusion Detection Systems (IDS) using both the CSE-CIC-IDS-2018 dataset and a custom-generated dataset. By replicating real-world attack scenarios and capturing network traffic data, we were able to train and evaluate ML models under different network architectures. The primary finding of this research is the significant impact of dataset characteristics and network environments on the performance of IDS models. Specifically, models trained on the custom dataset, which was tailored to the specific network architecture, demonstrated superior accuracy and reliability in detecting various types of attacks, including DoS, Brute Force, and Web attacks.

Our analysis revealed that context-specific data is crucial for enhancing the generalization capabilities of IDS models, as traditional datasets like CIC-IDS-2018 often fail to perform effectively in real-world network environments. This is because such datasets do not fully capture the complexities and dynamics of modern networks. Consequently, models trained on these traditional datasets showed suboptimal performance in production environments.

Our comprehensive evaluation identified Decision Tree, Random Forest, and XGBoost as the most effective algorithms, achieving nearly 99% accuracy in detecting cyber threats when trained on the custom dataset. These algorithms also exhibited the least training and prediction times, making them suitable for real-time detection scenarios. Although the K-Nearest Neighbors (KNN) algorithm demonstrated similar accuracy, its longer prediction time made it less efficient for practical deployment.

The results underscore the importance of collecting and using network-specific data for training IDS models, as this approach significantly enhances their effectiveness in production environments. Future research should focus on adapting IDS models to new and emerging environments, such as decentralized networks, cloud computing, and the Internet of Things (IoT). This includes refining feature selection techniques and exploring ensemble and deep learning methods to improve detection capabilities and efficiency in handling the unique challenges posed by these evolving environments.

ACKNOWLEDGMENT

This article is an enhanced research extension of the authors' capstone project, with contributions from Joemar Lugtu and Mark Edison Labiano.

REFERENCES

[1] S. Choudhary and N. Kesswani, "Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT," *Procedia Computer Science*, vol. 167, pp. 1561–1573, Jan. 2020, doi: 10.1016/j.procs.2020.03.367.

[2] 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*, Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 108–116. doi: 10.5220/0006639801080116.

[3] "IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB." Accessed: Aug. 27, 2024. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html

[4] A. Saadallah, F. Finkeldey, J. Buß, K. Morik, P. Wiederkehr, and W. Rhode, "Simulation and sensor data fusion for machine learning application," *Advanced Engineering Informatics*, vol. 52, p. 101600, Apr. 2022, doi: 10.1016/j.aei.2022.101600.

[5] S. Dwibedi, M. Pujari, and W. Sun, "A Comparative Study on Contemporary Intrusion Detection Datasets for Machine Learning Research," in *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, Nov. 2020, pp. 1–6. doi: 10.1109/ISI49825.2020.9280519.

[6] M. Ashfaq Khan, "HCRNNIDS: Hybrid Convolutional Recurrent Neural Network-Based Network Intrusion Detection System," *Processes*, vol. 9, p. 834, May 2021, doi: 10.3390/pr9050834.

[7] E. Jaw and X. Wang, "Feature Selection and Ensemble-Based Intrusion Detection System: An Efficient and Comprehensive Approach," *Symmetry*, vol. 13, no. 10, Art. no. 10, Oct. 2021, doi: 10.3390/sym13101764.

[8] S. Das *et al.*, "Network Intrusion Detection and Comparative Analysis Using Ensemble Machine Learning and Feature Selection," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4821–4833, Dec. 2022, doi: 10.1109/TNSM.2021.3138457.

[9] P. Vanin *et al.*, "A Study of Network Intrusion Detection Systems Using Artificial Intelligence/Machine Learning," *Applied Sciences*, vol. 12, no. 22, Art. no. 22, Jan. 2022, doi: 10.3390/app122211752.

[10] S. Sumathi, R. Rajesh, and S. Lim, "Recurrent and Deep Learning Neural Network Models for DDoS Attack Detection," *Journal of Sensors*, vol. 2022, no. 1, p. 8530312, 2022, doi: 10.1155/2022/8530312.

[11] T. Sowmya and E. A. Mary Anita, "A comprehensive review of AI based intrusion detection system," *Measurement: Sensors*, vol. 28, p. 100827, Aug. 2023, doi: 10.1016/j.measen.2023.100827.

[12] C. Edwin Singh and S. M. Celestin Vigila, "Fuzzy based intrusion detection system in MANET," *Measurement: Sensors*, vol. 26, p. 100578, Apr. 2023, doi: 10.1016/j.measen.2022.100578.

[13] S. Songma, T. Sathuphan, and T. Pamutha, "Optimizing Intrusion Detection Systems in Three Phases on the CSE-CIC-IDS-2018 Dataset," *Computers*, vol. 12, no. 12, Art. no. 12, Dec. 2023, doi: 10.3390/computers12120245.

[14] M. Bhavsar, K. Roy, J. Kelly, and O. Olusola, "Anomaly-based intrusion detection system for IoT application," *Discov Internet Things*, vol. 3, no. 1, p. 5, May 2023, doi: 10.1007/s43926-023-00034-5.

[15] T. Gaber, J. B. Awotunde, S. O. Folorunso, S. A. Ajagbe, and E. Eldesouky, "Industrial Internet of Things Intrusion Detection Method Using Machine Learning and Optimization Techniques," *Wireless Communications and Mobile Computing*, vol. 2023, no. 1, p. 3939895, 2023, doi: 10.1155/2023/3939895.

[16] W. H. Aljuaid and S. S. Alshamrani, "A Deep Learning Approach for Intrusion Detection Systems in Cloud Computing Environments," *Applied Sciences*, vol. 14, no. 13, Art. no. 13, Jan. 2024, doi: 10.3390/app14135381.

[17] K. K. Paidipati, C. Kurangi, J. Uthayakumar, S. Padmanayaki, D. Pradeepa, and S. Nithinsha, "Ensemble of deep reinforcement learning with optimization model for DDoS attack detection and classification in cloud based software defined networks," *Multimed Tools Appl*, vol. 83, no. 11, pp. 32367–32385, Mar. 2024, doi: 10.1007/s11042-023-16894-6.

[18] H. Najafi Mohsenabad and M. A. Tut, "Optimizing Cybersecurity Attack Detection in Computer Networks: A Comparative Analysis of Bio-Inspired Optimization Algorithms Using the CSE-CIC-IDS 2018 Dataset," *Applied Sciences*, vol. 14, no. 3, Art. no. 3, Jan. 2024, doi: 10.3390/app14031044.

[19] L. Göcs and Z. C. Johanyák, "Identifying relevant features of CSE-CIC-IDS2018 dataset for the development of an intrusion detection system," *Intelligent Data Analysis*, vol. Preprint, no. Preprint, pp. 1–27, Jan. 2024, doi: 10.3233/IDA-230264.

[20] Md. A. Talukder *et al.*, "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," *Journal of Big Data*, vol. 11, no. 1, p. 33, Feb. 2024, doi: 10.1186/s40537-024-00886-w.

[21] L. Yuan, J. Sun, S. Zhuang, Y. Liu, L. Geng, and W. Ma, "CoSen-IDS: A Novel Cost-Sensitive Intrusion Detection System on Imbalanced Data in 5G Networks," in *Advanced Intelligent Computing Technology and Applications*, D.-S. Huang, W. Chen, and Y. Pan, Eds., Singapore: Springer Nature, 2024, pp. 470–481. doi: 10.1007/978-981-97-5603-2_39.

[22] P. Rana, I. Batra, A. Malik, I.-H. Ra, O.-S. Lee, and A. S. M. Sanwar Hosen, "Efficacious Novel Intrusion Detection System for Cloud Computing Environment," *IEEE Access*, vol. 12, pp. 99223–99239, 2024, doi: 10.1109/ACCESS.2024.3424528.

[23] E. C. P. Neto *et al.*, "CICIoV2024: Advancing realistic IDS approaches against DoS and spoofing attack in IoV CAN bus," *Internet of Things*, vol. 26, p. 101209, Jul. 2024, doi: 10.1016/j.iot.2024.101209.

[24] W. Chimphlee and S. Chimphlee, "Hyperparameters optimization XGBoost for network intrusion detection using CSE-CIC-IDS 2018 dataset," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 13, p. 817, Mar. 2024, doi: 10.11591/ijai.v13.i1.pp817-826.

[25] H. Bakır and Ö. Ceviz, "Empirical Enhancement of Intrusion Detection Systems: A Comprehensive Approach with Genetic Algorithm-based Hyperparameter Tuning and Hybrid Feature Selection," *Arab J Sci Eng*, Apr. 2024, doi: 10.1007/s13369-024-08949-z.

[26] "Applications | Research | Canadian Institute for Cybersecurity | UNB." Accessed: Nov. 20, 2024. [Online]. Available: https://www.unb.ca/cic/research/applications.html

SYED K. IKRAMUDDIN was born in Hyderabad, India, in 2000. He received a B.E. degree in computer science from Osmania University, Hyderabad, India, in 2022, and recently completed a postgraduate program in Cybersecurity from Centennial College, Toronto, Canada, in 2024. His research interests include cybersecurity within AI, cloud, blockchain, and application security.