



## **Research Report**

**Detection and Response: Enhancing Intrusion Detection Systems**

by:

Group 5

**Joemar Lugtu - 301355179**

**Mark Labiano - 301346652**

**Syed - 301318212**

**Tarun Kumar Ravikumar – 301364055**

for:

**CBER-710.....**

**Centennial College**

**School of Engineering Technology and Applied Science**

to:

**Dr. Sk Md Mizanur Rahman**

on:

**April 6, 2023.**

## Table of Contents

<b>1</b>	<b>Abstract .....</b>	<b>5</b>
<b>2</b>	<b>Introduction .....</b>	<b>7</b>
<b>3</b>	<b>Background .....</b>	<b>10</b>
<b>4</b>	<b>Related Work .....</b>	<b>12</b>
<b>5</b>	<b>Explanation of Methodology and Technology .....</b>	<b>15</b>
5.1	Experimental Setup .....	16
5.2	Data Collection .....	17
<b>6</b>	<b>Proposed Approach.....</b>	<b>20</b>
6.1	Performance Analysis .....	25
<b>7</b>	<b>Results .....</b>	<b>30</b>
<b>8</b>	<b>Conclusion .....</b>	<b>35</b>
<b>9</b>	<b>General Information .....</b>	<b>37</b>
9.1	Definition of Terms .....	38
9.2	Research Management Plan.....	38
<b>10</b>	<b>References .....</b>	<b>39</b>

## List of Tables

Table 1 - Overview of Previous research .....	13
---	----

## List of Figures

Figure 1 - Network Architecture .....	17
Figure 2 - SSH Brute Force.....	18
Figure 3 - FTP Brute Force .....	19
Figure 4 – CIC FlowMeter Capturing Packets .....	19
Figure 5 - Proposed Approach.....	21
Figure 6 - Dropping categorical Columns .....	22
Figure 7 - Dropping categorical column (Timestamp).....	22
Figure 8 - Dropping missing values .....	22
Figure 9 - Dropping zero columns.....	22
Figure 10 - Normalizing Datasets .....	23
Figure 11 – Label Encoding .....	23
Figure 12 - Feature Selection using RF .....	24
Figure 13 - Decision Tree.....	24
Figure 14 - Random Forest.....	24
Figure 15 - Multi-layer Perceptron .....	25
Figure 16 - Naive Bayes.....	25
Figure 17 - KNN .....	25
Figure 18 - Confusion Matrix (Decision tree Online Dataset) .....	26
Figure 19 - Confusion Matrix (Decision Tree).....	26
Figure 20 - Confusion Matrix (Random forest).....	27
Figure 21 - Confusion Matrix (MLP) .....	27
Figure 22 - Confusion Matrix (Naive Bayes) .....	28
Figure 23 - Confusion Matrix (KNN).....	28
Figure 24 - Accuracy.....	28
Figure 25 - Precision .....	29
Figure 26 - Recall.....	29
Figure 27 - F1-Score .....	29
Figure 28 - Training Time Comparison .....	31
Figure 29 - Prediction Time Comparison .....	32
Figure 30 - Performance Analysis.....	33
Figure 31 - Performance Analysis 2.....	34

# Chapter 1

## **1 Abstract**

In this study, we delved into the intricate domain of Intrusion Detection Systems (IDS) within cybersecurity, specifically focusing on the detection of brute force attacks using machine learning (ML) techniques. Leveraging the CSE-CIC-IDS-2018 dataset and our bespoke data collection setup, we embarked on an exploration of various ML algorithms and feature selection methodologies to enhance IDS performance. The study encompassed meticulous data preprocessing, rigorous model training, comprehensive performance evaluations, and detailed analyses of computational efficiency. Our key contributions include identifying critical feature dimensions that significantly impact classification performance, evaluating the effectiveness of diverse ML classifiers in detecting SSH and FTP brute force attacks, and presenting practical insights essential for implementing scalable and efficient IDS solutions in real-world cybersecurity scenarios.

## Chapter 2

## 2 Introduction

In today's modern age, the Internet holds immense importance as a fundamental medium for communication and global information exchange. Information and Communication Technology (ICT) are essential in various aspects of business and daily activities. Yet, the emergence of big data has led to more complex and prevalent cyber-attacks on ICT systems, underscoring the critical necessity for advanced network security solutions. As society depends more on digital tools like computers and the Internet, it's crucial to develop robust and secure programs, frameworks, and networks that can withstand these evolving threats. [1,2]

In the realm of safeguarding computer networks, Intrusion Detection Systems (IDS) are pivotal for swiftly identifying and responding to security threats, significantly enhancing network security. These systems continuously monitor network traffic, detecting anomalies that may signal potential attacks. However, IDS technology faces several challenges, including the need to improve detection accuracy, reduce false positives, and effectively identify both known and unknown threats.

IDS are typically categorized into three types: signature-based, anomaly-based, and hybrid systems. Anomaly-based systems can detect unknown threats by pinpointing deviations from normal behavior, whereas signature-based systems excel at identifying known attacks but often generate false alarms. Hybrid IDS, combining both approaches, have emerged as a robust solution. Current anomaly-based IDS confront accuracy issues due to limited network traffic diversity and recent attack patterns. [3,4,5,6]

Hybrid IDS, leveraging machine learning (ML) techniques for anomaly detection, have shown promise in overcoming these challenges. ML automates the learning process from examples to build models that distinguish between normal and abnormal activities, thereby enhancing IDS efficiency. Feature selection (FS) is another critical aspect influencing IDS effectiveness. It involves selecting relevant features for building accurate models, thereby improving preprocessing and classifier methods. [7,8,9]

## Detection And Response: Enhancing Intrusion Detection Systems

This study delves into the complexities of IDS using the CSE-CIC-IDS-2018 dataset, specifically focusing on brute force attacks targeting SSH and FTP. The research encompasses data cleaning, exploratory data analysis, normalization, and feature reduction using extraction techniques to streamline processing and reduce model complexity. Various ML algorithms like Decision Trees (DT), Naïve Bayes, KNN, MLP, and Random Forest were tested, with DT and RF showing promising outcomes.

The study's contributions include:

1. Investigating large datasets related to harmful network activities.
2. Identifying feature dimensions that impact classification performance, leading to improved detection accuracy.
3. Utilizing the CSE-CIC-IDS-2018 dataset for NIDS and evaluating ML classifiers for detecting SSH and FTP brute force attacks.
4. Employing our custom setup for data collection, ensuring precise and relevant datasets for analysis and evaluation.
5. Presenting comprehensive performance evaluations, including CPU processing time and model size considerations, essential for intrusion detection systems' practical implementation and scalability.



# Chapter 3

### **3 Background**

In today's digital environment, cybercriminals are becoming more adept at exploiting unknown vulnerabilities to bypass security measures. One of the prevalent types of network attacks is brute force attacks, which are increasingly difficult to detect due to the widespread adoption of high-speed networks and encryption in network traffic. Brute force attacks involve systematically trying all possible character combinations until the correct input is found, often using extensive password dictionaries with millions of entries. If successful, these attacks allow hackers to gain unauthorized access to sensitive data, applications, and resources, potentially leading to further malicious actions.

There are several signs that can indicate a brute force attack, such as numerous failed login attempts originating from the same IP address, login attempts using multiple usernames from the same IP, logins from multiple IP addresses for a single account, sequential login credentials, logins from referral URLs like mail or IRC clients, unusually high bandwidth consumption during a session, and a significant number of authentication failures.

Secure Shell (SSH) is a widely used communication protocol allowing remote access to systems through encrypted channels. Brute force attacks involve systematically attempting authentication with various passwords until the correct one is found. Attackers often use automated scripts and applications for these brute force attempts, targeting hosts directly exposed to the Internet or WAN with SSH and FTP services running.

Brute force attacks have been a prominent issue, with reports indicating their prevalence in attack types detected by Intrusion Prevention Systems (IPS). Various organizations and services, including Microsoft Office 365, Alibaba, GitHub, WordPress, and even German routers, have been targets of massive brute force attacks, resulting in compromised accounts and significant damages.

## Detection And Response: Enhancing Intrusion Detection Systems

To mitigate brute force attacks, network administrators can implement measures such as enhancing password complexity, limiting login attempts, using Captchas to deter automated scripts, and implementing two-factor authentication for added security.

While existing detection methods for brute force attacks have relied on traditional data models and tools like SSH tunneling and firewalls, there's a growing need for more dynamic and efficient detection mechanisms. Machine learning techniques have shown promise in network security for detecting complex attacks. However, their effectiveness in detecting brute force attacks specifically hasn't been extensively explored, making it a potential area for further research and development.

# Chapter 4

#### 4 Related Work

Network intrusion detection datasets are scarce compared to datasets for malicious code, with KDD CUP 99 being the most commonly used for evaluating IDS. However, in this study, we focus on developing an IDS model using the CSE-CIC-IDS-2018 dataset, which contains the latest common network attacks.[10] This dataset from the Canadian Institute for Cybersecurity introduces the concept of profiles, allowing for versatile generation of network events by both agents and individuals across various network protocols and topologies. It has been updated with the standards from CIC-IDS-2017, ensuring data integrity with minimal duplicates or unclear information, and is readily available in CSV format for immediate use without additional processing. [11]

As data complexity increases, feature selection has become crucial for IDS development. This process involves eliminating irrelevant and redundant features to identify the optimal subset that best represents patterns in different classes. Feature selection offers several benefits, including reduced dimensionality for improved algorithm performance, enhanced data efficiency by eliminating noisy or redundant data, increased model accuracy, and a better understanding of data generation processes. [12,13]

Numerous studies have utilized machine learning techniques with the CSE-CIC-IDS-2018 dataset for intrusion detection, as summarized in Table 1 after reviewing relevant literature and research articles.

*Table 1 - Overview of Previous research*

Study	Methodology/Findings
S. Ullah et al. [14]	Compared various machine learning algorithms (Random Forest, Bayes, Logistic Regression, K-Nearest Neighbors, Decision Trees) and performed feature selection using Random Forest with 30 features on the CSE-CIC-IDS-2018 dataset, with Decision Trees yielding the best results.
M. A. Khan. [15]	Developed HCRNNIDS, a hybrid convolutional recurrent neural network-based NIDS, and compared it with machine learning algorithms (Decision Trees, Logistic Regression, XGBoost) using feature selection by Random Forest (30

	features) on the CSE-CIC-IDS-2018 dataset, showing superior results with HCRNNIDS.
J. Kim et al. [10]	Discussed IDS models using various machine learning algorithms (Artificial Neural Networks, Support Vector Machines, Convolutional Neural Networks, Recurrent Neural Networks) and found that Convolutional Neural Networks outperformed traditional techniques on the CSE-CIC-IDS-2018 dataset.
R. Qusyairi et al. [3]	Proposed an ensemble learning technique incorporating Logistic Regression, Decision Trees, and gradient boosting after comparisons with single classifiers on the CSE-CIC-IDS-2018 dataset, identifying 23 significant traits out of 80.
S. Chimphee et al. [4]	Focused on IDS using the CSE-CIC-IDS-2018 dataset, employing data preprocessing, feature selection, and seven classifier machine learning algorithms (including Multi-Layer Perceptron and XGBoost), with Multi-Layer Perceptron providing the most successful outcomes.
S. Malliga et al. [17]	Examined denial of service (DoS/DDoS) attacks, deep-learning-based detection algorithms, and the evolution of attack patterns, highlighting the need for improved techniques to handle dynamic attacker behavior.
A. Alzaqebah et al. [18]	Improved network intrusion detection systems using a modified grey wolf optimization algorithm, achieving high accuracy and performance in detecting regular and anomalous traffic.

# Chapter 5

## **5 Explanation of Methodology and Technology**

### **5.1 Experimental Setup**

The data collection setup comprises two virtual machines (VMs) deployed on VMware Workstation 17 Player. One VM runs CentOS 7 (referred to as the "Victim") with 2 GB of RAM and 2 vProcessors allocated to it. This CentOS VM is configured with an SSH server and an FTP server, both of which are operational. Additionally, the CentOS VM is equipped with the CIC Flowmeter tool to capture data packets related to brute force attacks. On the other hand, the second VM runs Kali Linux (referred to as the "Attacker") with similar specifications, including 2 GB of RAM and 2 vProcessors. The Kali Linux VM utilizes the Patator tool to execute brute force attacks against the CentOS VM.

This setup enables the simulation and monitoring of brute force attacks in a controlled environment. The Victim VM represents a target system with services susceptible to brute force attacks, while the Attacker VM emulates a malicious actor attempting to gain unauthorized access using brute force techniques.

The CIC Flowmeter on the Victim VM captures network traffic and data packets generated during these attacks, providing a dataset for later analysis. The collected data from this setup is essential for training and testing machine learning algorithms aimed at detecting and mitigating brute force attacks. By simulating real-world attack scenarios in a controlled environment, researchers can gather meaningful data to develop and refine intrusion detection systems.



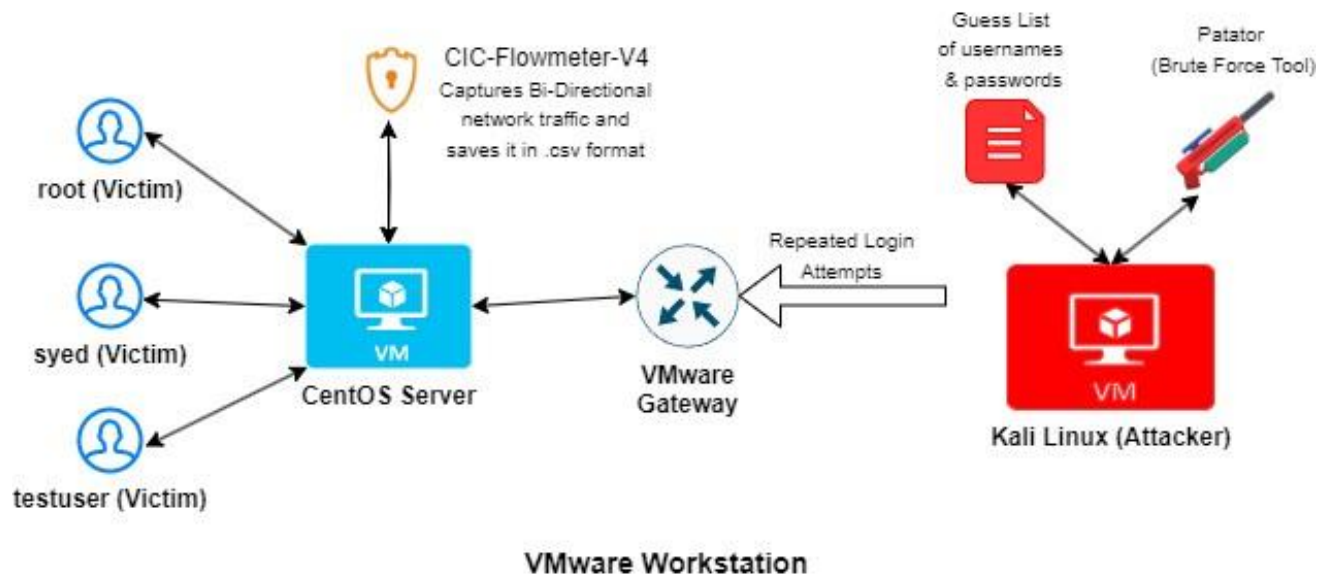


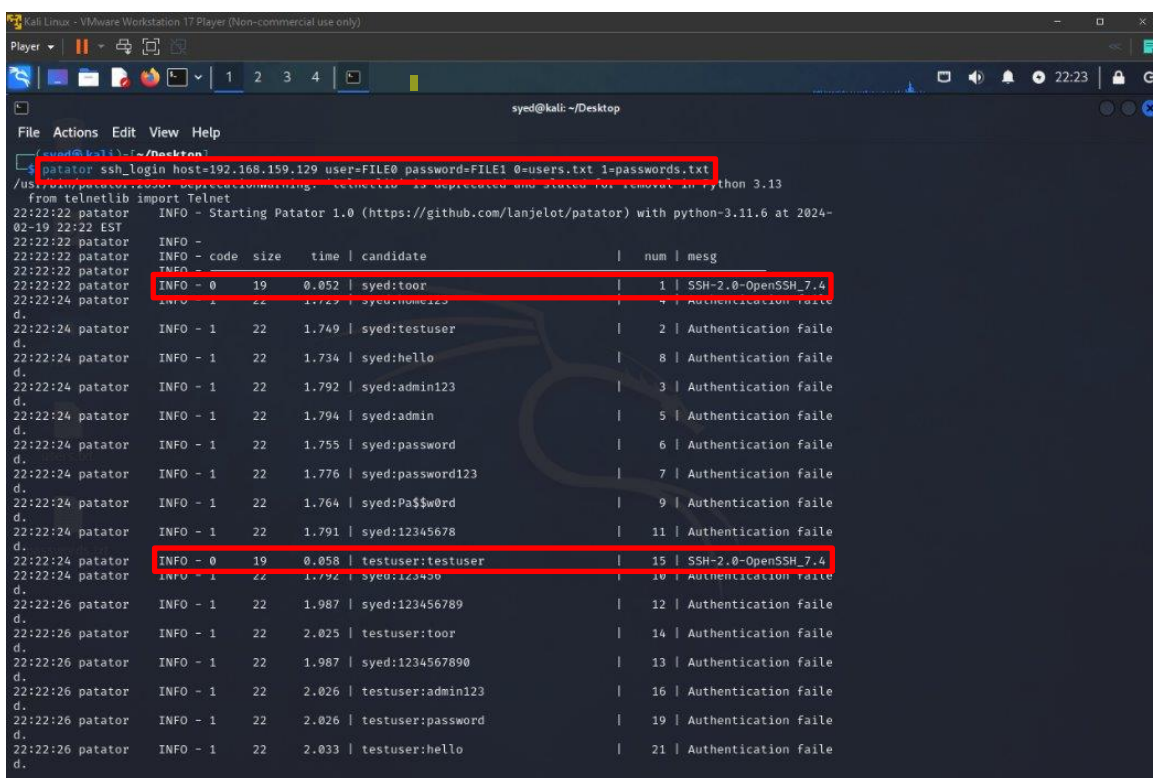
Figure 1 - Network Architecture

The research was conducted using Python 3.11 within Google Colab due to the significant volume of large data involved. Google Colab's environment includes an Intel Xeon CPU with 2 virtual CPUs (vCPUs) and 13GB of RAM, making it suitable for tasks such as data preparation, handling, preprocessing, analysis, training, and evaluating metrics. A recommended model was developed and evaluated using libraries like Numpy, Pandas, and Scikit Learn. Specifically, Pandas and Numpy were utilized for data handling and preprocessing, while Scikit Learn was employed for model training, evaluation, and assessing metrics. Furthermore, data visualization was performed using the Seaborn program and Matplotlib. Detailed insights into the research methodology are provided in subsequent sections of the study.

## 5.2 Data Collection

The dataset utilized in this study is the CIC-IDS 2018 benchmark dataset, which captures contemporary benign and malicious network activities, reflecting real-time network traffic scenarios. The dataset, generated using CICFlowMeter-V3, comprises 83 features providing information on network flow directions, packets, labels, and FlowID. Among these features, five are categorical (SourceIP, DestinationIP, SourcePort, DestinationPort, Protocol), while the remaining 78 are continuous. The data is provided in CSV format, making it compatible with machine learning pipelines.

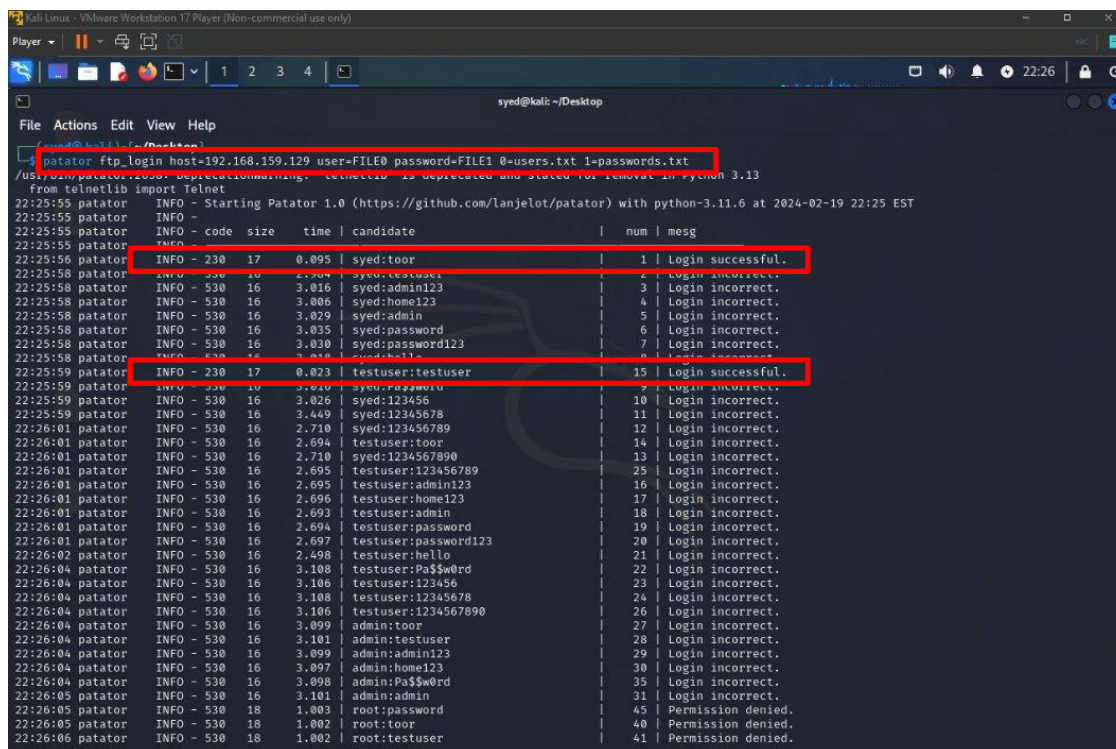
We generated a dataset similar to the CIC-IDS 2018 benchmark dataset specifically focusing on brute force attacks using our own experimental setup. This dataset mirrors the characteristics of real-world brute force attacks and was created to enhance our research. This customized dataset enables us to conduct detailed analyses and experiments focused on improving the detection. Before using the datasets, certain pre-processing steps are necessary to ensure its usability and compatibility with machine learning algorithms.



```
syed@kali: ~/Desktop
patator ssh_login host=192.168.159.129 user=FILE0 password=FILE1 0=users.txt 1=passwords.txt
INFO - Starting Patator 1.0 (https://github.com/lanjelot/patator) with python-3.11.6 at 2024-02-19 22:22 EST
INFO - code size time | candidate | num | msg
INFO - 0 19 0.052 | syed:toor | 1 | SSH-2.0-OpenSSH_7.4
INFO - 1 22 1.729 | syed:123456789 | 2 | Authentication failed
INFO - 1 22 1.749 | syed:testuser | 3 | Authentication failed
INFO - 1 22 1.734 | syed:hello | 4 | Authentication failed
INFO - 1 22 1.792 | syed:admin123 | 5 | Authentication failed
INFO - 1 22 1.794 | syed:admin | 6 | Authentication failed
INFO - 1 22 1.755 | syed:password | 7 | Authentication failed
INFO - 1 22 1.776 | syed:password123 | 8 | Authentication failed
INFO - 1 22 1.764 | syed:Pa$$w0rd | 9 | Authentication failed
INFO - 1 22 1.791 | syed:12345678 | 10 | Authentication failed
INFO - 0 19 0.058 | testuser:testuser | 11 | SSH-2.0-OpenSSH_7.4
INFO - 1 22 1.792 | syed:123456 | 12 | Authentication failed
INFO - 1 22 1.987 | syed:123456789 | 13 | Authentication failed
INFO - 1 22 2.025 | testuser:toor | 14 | Authentication failed
INFO - 1 22 1.987 | syed:1234567890 | 15 | Authentication failed
INFO - 1 22 2.026 | testuser:admin123 | 16 | Authentication failed
INFO - 1 22 2.026 | testuser:password | 17 | Authentication failed
INFO - 1 22 2.033 | testuser:hello | 18 | Authentication failed
```

Figure 2 - SSH Brute Force

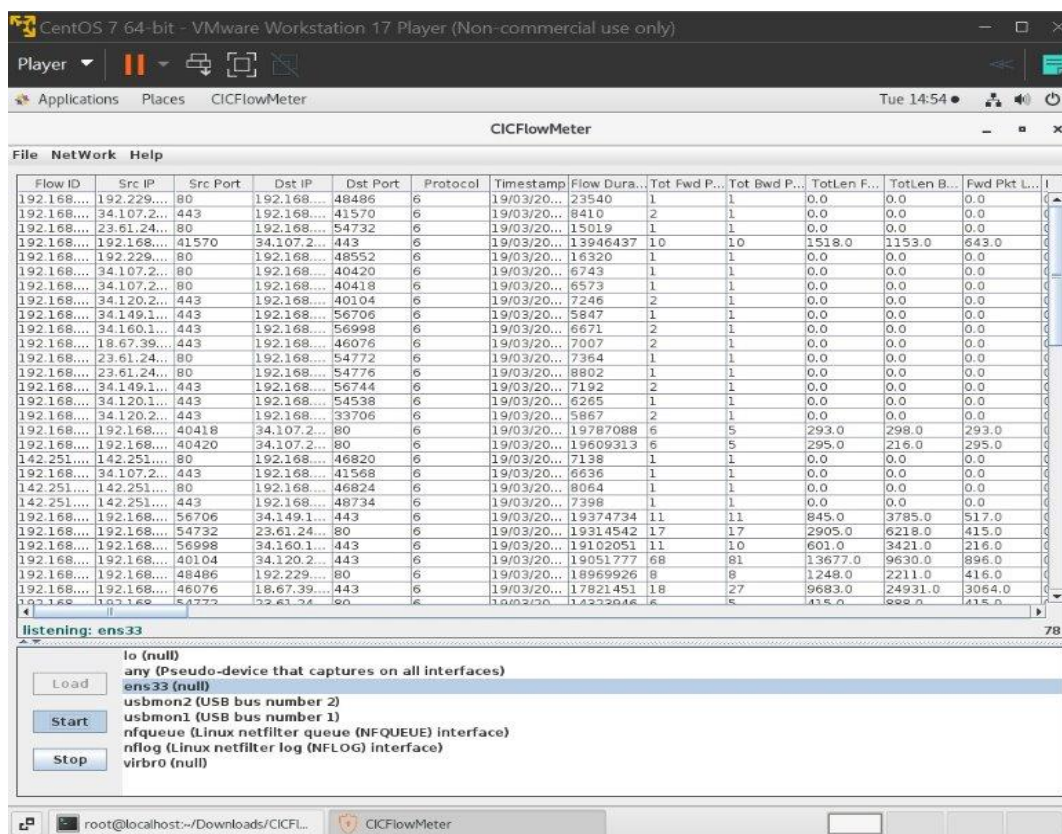
## Detection And Response: Enhancing Intrusion Detection Systems



The screenshot shows a terminal window with the command `patator ftp_login host=192.168.159.129 user=FILE0 password=FILE1 @-users.txt 1-passwords.txt` and its output. The output is a table of login attempts. Two rows are highlighted with red boxes: one for 'syed:toor' and another for 'testuser:testuser', both showing 'Login successful'.

INFO	code	size	time	candidate	num	msg
22:25:55	INFO	-	-	syed:toor	1	Login successful.
22:25:58	INFO	-	-	syed:toor	2	Login incorrect.
22:25:58	INFO	-	-	syed:admin123	3	Login incorrect.
22:25:58	INFO	-	-	syed:home123	4	Login incorrect.
22:25:58	INFO	-	-	syed:admin	5	Login incorrect.
22:25:58	INFO	-	-	syed:password	6	Login incorrect.
22:25:58	INFO	-	-	syed:password123	7	Login incorrect.
22:25:58	INFO	-	-	syed:123456	8	Login incorrect.
22:25:59	INFO	-	-	testuser:testuser	15	Login successful.
22:25:59	INFO	-	-	testuser:testuser	9	Login incorrect.
22:25:59	INFO	-	-	syed:123456	10	Login incorrect.
22:25:59	INFO	-	-	syed:12345678	11	Login incorrect.
22:26:01	INFO	-	-	syed:123456789	12	Login incorrect.
22:26:01	INFO	-	-	testuser:toor	14	Login incorrect.
22:26:01	INFO	-	-	syed:1234567890	13	Login incorrect.
22:26:01	INFO	-	-	testuser:123456789	25	Login incorrect.
22:26:01	INFO	-	-	testuser:admin123	16	Login incorrect.
22:26:01	INFO	-	-	testuser:home123	17	Login incorrect.
22:26:01	INFO	-	-	testuser:admin	18	Login incorrect.
22:26:01	INFO	-	-	testuser:password	19	Login incorrect.
22:26:01	INFO	-	-	testuser:password123	20	Login incorrect.
22:26:02	INFO	-	-	testuser:hello	21	Login incorrect.
22:26:04	INFO	-	-	testuser:Pa\$\$w0rd	22	Login incorrect.
22:26:04	INFO	-	-	testuser:123456	23	Login incorrect.
22:26:04	INFO	-	-	testuser:12345678	24	Login incorrect.
22:26:04	INFO	-	-	testuser:1234567890	26	Login incorrect.
22:26:04	INFO	-	-	admin:toor	27	Login incorrect.
22:26:04	INFO	-	-	admin:testuser	28	Login incorrect.
22:26:04	INFO	-	-	admin:admin123	29	Login incorrect.
22:26:04	INFO	-	-	admin:home123	30	Login incorrect.
22:26:04	INFO	-	-	admin:Pa\$\$w0rd	35	Login incorrect.
22:26:05	INFO	-	-	admin:admin	31	Login incorrect.
22:26:05	INFO	-	-	root:password	45	Permission denied.
22:26:05	INFO	-	-	root:toor	40	Permission denied.
22:26:06	INFO	-	-	root:testuser	41	Permission denied.

Figure 3 - FTP Brute Force



The screenshot shows the CICFlowMeter application interface. The top section displays a table of network flows with columns for Flow ID, Src IP, Src Port, Dst IP, Dst Port, Protocol, Timestamp, Flow Dura..., Tot Fwd P..., Tot Bwd P..., TotLen F..., TotLen B..., and Fwd Pkt L... The bottom section shows the 'listening: ens33' status and a list of capture devices including 'lo (null)', 'any (Pseudo-device that captures on all interfaces)', 'ens33 (null)', 'usbmon2 (USB bus number 2)', 'usbmon1 (USB bus number 1)', 'nfqqueue (Linux netfilter queue (NFQUEUE) interface)', 'nflog (Linux netfilter log (NFLOG) interface)', and 'viro0 (null)'. The 'Start' button is highlighted.

Figure 4 – CIC FlowMeter Capturing Packets

# Chapter 6

## 6 Proposed Approach

Our proposed paradigm is depicted in the schematic block diagram shown in the Figure 5. The approach is organized into several phases as follows:

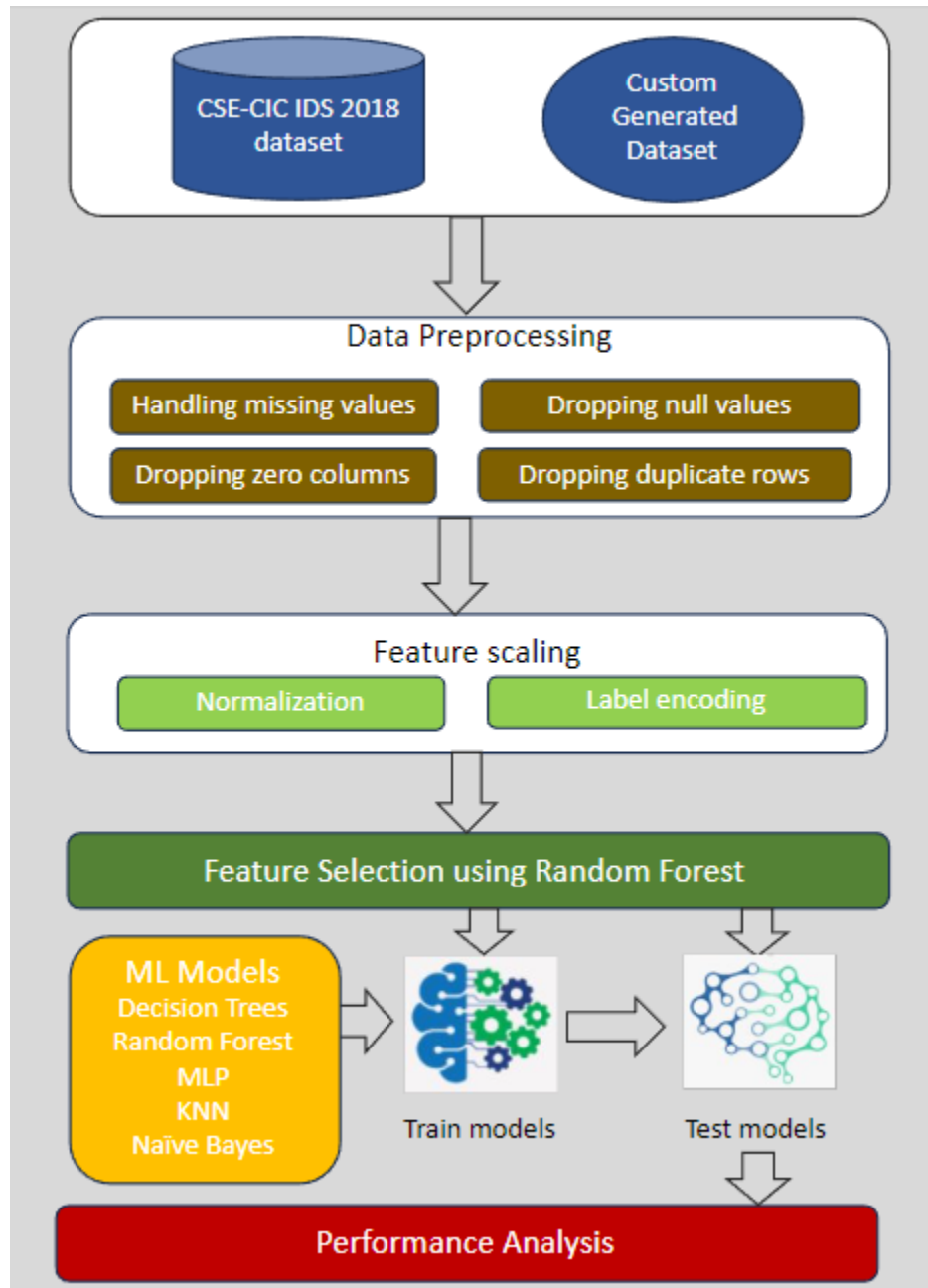


Figure 5 - Proposed Approach

1. Data Preprocessing (Step-1): Initially, data preprocessing tasks are performed, including handling missing values, dropping categorical columns, removing zero column, dropping duplicate and null rows. This step is crucial for enhancing data quality and involves tasks such as data type conversions and data cleaning to prepare the dataset for analysis.

```
df2.drop(columns=['Flow ID', 'Src IP', 'Src Port', 'Dst IP'], inplace=True)  
df3.drop(columns=['Flow ID', 'Src IP', 'Src Port', 'Dst IP'], inplace=True)
```

Figure 6 - Dropping categorical Columns

```
[ ] df = df.drop(columns=['Timestamp'],axis = 1)  
df2 = df2.drop(columns=['Timestamp'],axis = 1)
```

Figure 7 - Dropping categorical column (Timestamp)

```
[ ] df = df.dropna()  
  
[ ] df2 = df2.dropna()
```

Figure 8 - Dropping missing values

```
# Print out use the common zero columns list  
print(common_zero_columns_list)  
['Bwd URG Flags', 'Bwd Pkts/b Avg', 'Bwd Blk Rate Avg', 'CWE Flag Count', 'Fwd Blk Rate Avg', 'Bwd Byts/b Avg', 'Fwd URG Flags', 'Fwd Pkts/b Avg', 'Fwd Byts/b Avg']  
  
[ ] df.drop(common_zero_columns_list, axis=1, inplace=True)  
  
[ ] df2.drop(common_zero_columns_list, axis=1, inplace=True)
```

Figure 9 - Dropping zero columns

2. Feature Scaling (Step-2): Input features are normalized, and output features are label encoded to ensure uniform scaling of data, which is essential for machine learning (ML) algorithms.

```
[ ] from sklearn.preprocessing import StandardScaler, MinMaxScaler
    sc = MinMaxScaler()
    X_train_std = sc.fit_transform(X_train)
    X_test_std = sc.transform(X_test)
    X_my_std = sc.transform(X_my)
    X_my_test_std = sc.transform(X_my_test)
    print(X_my_std.shape)
```

Figure 10 - Normalizing Datasets

```
[ ] df["Label"].unique()

array(['Benign', 'FTP-BruteForce', 'SSH-Bruteforce'], dtype=object)

▶ from sklearn.preprocessing import LabelEncoder
  le = LabelEncoder()
  df["Label"] = le.fit_transform(df["Label"])
  df2["Label"] = le.transform(df2["Label"])
  df3["Label"] = le.transform(df3["Label"])

[ ] df["Label"].value_counts()

0    667626
1    193360
2    187589
Name: Label, dtype: int64
```

Figure 11 – Label Encoding

3. Feature Selection using Random Forest (Step-3): Random Forest, apart from being a robust prediction model, is also utilized for feature selection in ML. It evaluates the significance of each feature during training by assessing its contribution to reducing impurity or inaccuracy in the model. This feature ranking capability simplifies the selection process, enhancing model efficiency and interpretability.



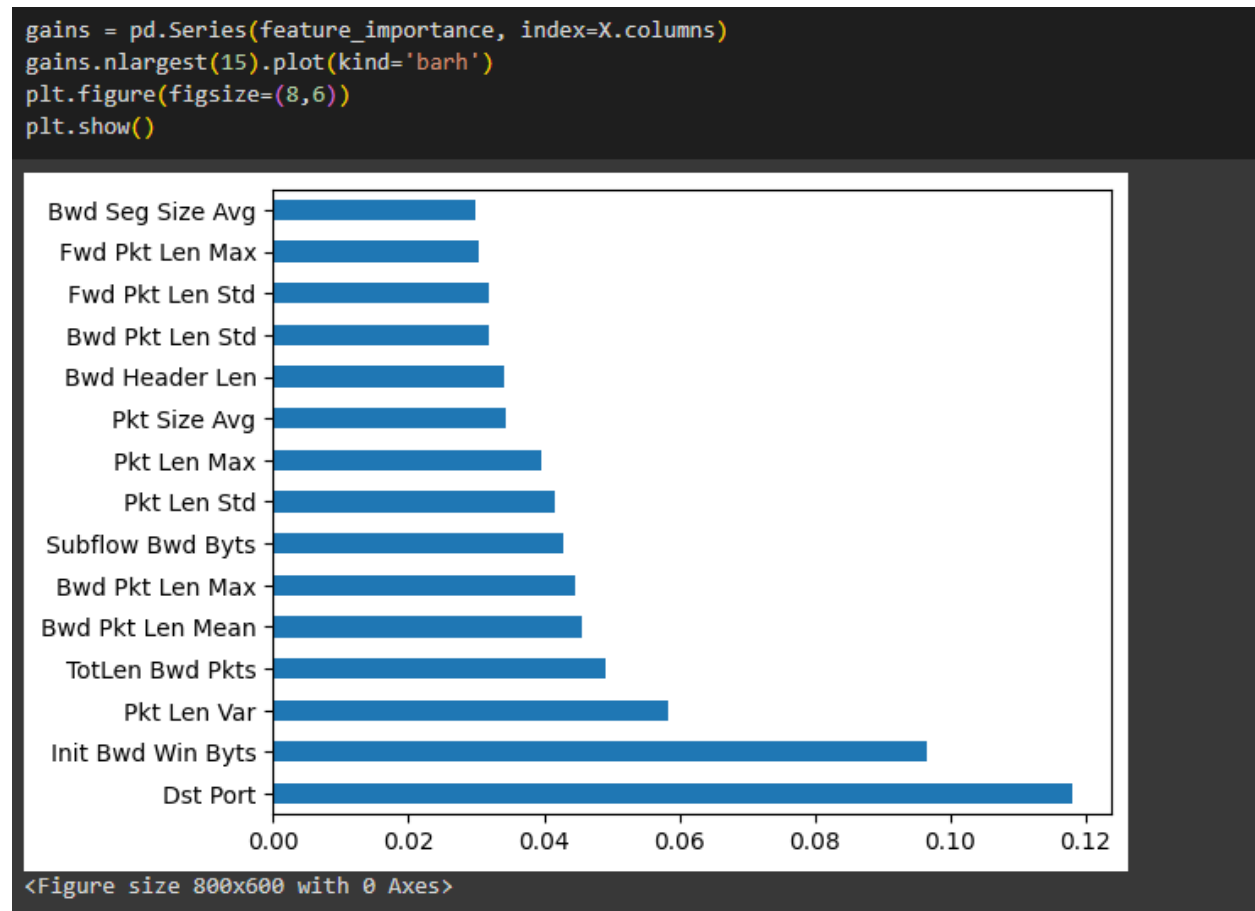


Figure 12 - Feature Selection using RF

4. Model Training (Step-4): ML algorithms such as Decision Trees (DT), Random Forest (RF), Multi-Layer Perceptron (MLP), Naive Bayes, and K-Nearest Neighbors (KNN) Classification are utilized for model training.

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion="entropy")
clf = clf.fit(X_train_std, y_train)
```

Figure 13 - Decision Tree

```
random = RandomForestClassifier(n_estimators = 5, criterion = 'entropy')
random_cross = fit_and_evaluate(random, X_my_std , X_my_test_std , y_my, y_my_test)
```

Figure 14 - Random Forest



```
NeuralNet_model = MLPClassifier(hidden_layer_sizes=(100,100,100),  
                                max_iter=50, alpha=0.0001, solver='sgd',  
                                verbose=10, random_state=21,tol=0.00000001)  
NeuralNet_model.fit(X_my_std, y_my)
```

Figure 15 - Multi-layer Perceptron

```
NB_model = GaussianNB()  
NB_model.fit(X_my_std, y_my)
```

Figure 16 - Naive Bayes

```
knn = KNeighborsClassifier(n_neighbors=9)  
knn.fit(X_my_std, y_my)
```

Figure 17 - KNN

5. Performance Analysis (Step-5): The model's performance is evaluated using various established metrics including Precision, Recall, Confusion Matrix, Accuracy, and F1-score. These metrics provide insights into the model's effectiveness and help in selecting the best-performing model for intrusion detection system (IDS) tasks

## 6.1 Performance Analysis

We use several performance evaluation metrics to gauge the effectiveness of our proposed model, including accuracy, precision, recall, F1-score, and the confusion matrix. These metrics collectively offer insights into the model's ability to accurately classify instances, its overall accuracy, and its capacity to minimize false positives and false negatives. Here's an explanation of these performance metrics:

1. Confusion Matrix: The confusion matrix, a critical evaluation tool for machine learning models, illustrates four combinations of predicted and actual values: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). These elements are instrumental in assessing key metrics such as Recall, F1-score, Accuracy, and Precision.

## Detection And Response: Enhancing Intrusion Detection Systems

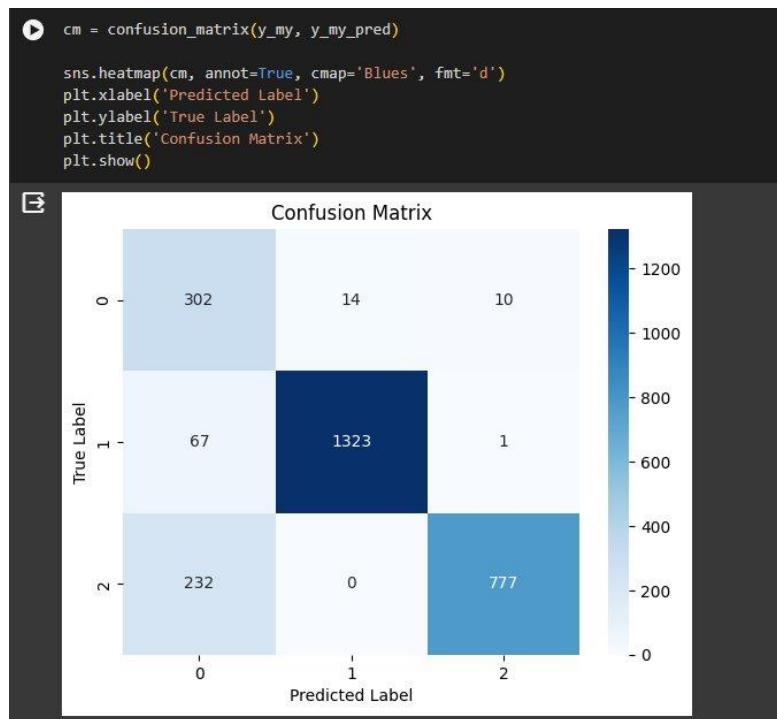


Figure 18 - Confusion Matrix (Decision tree Online Dataset)

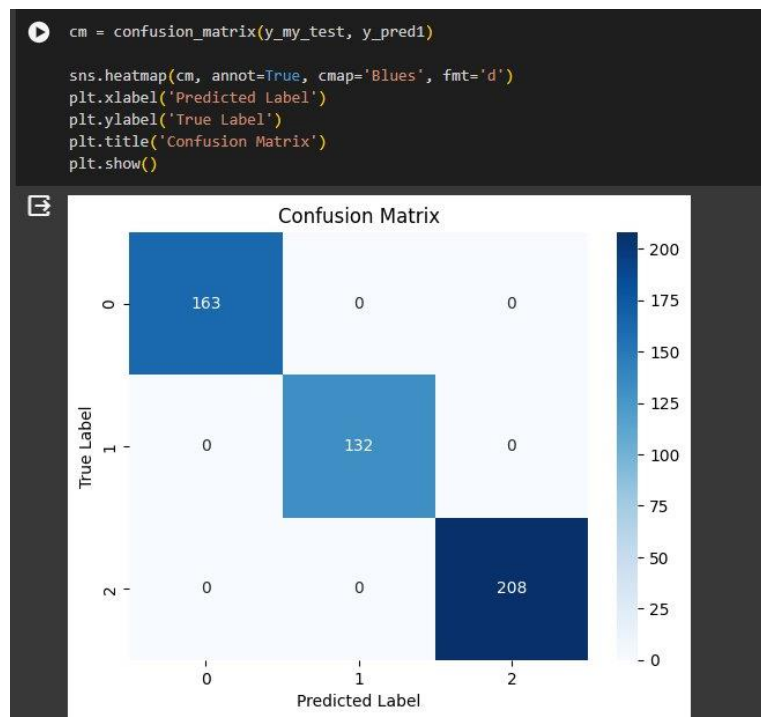


Figure 19 - Confusion Matrix (Decision Tree)

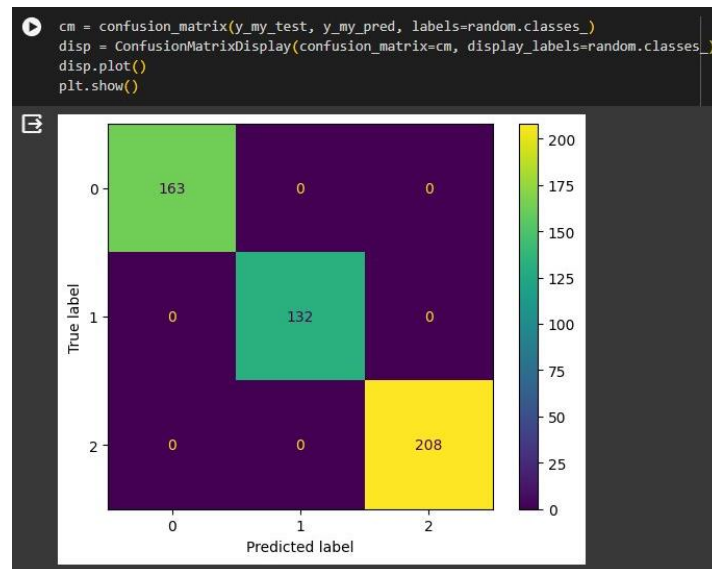


Figure 20 - Confusion Matrix (Random forest)

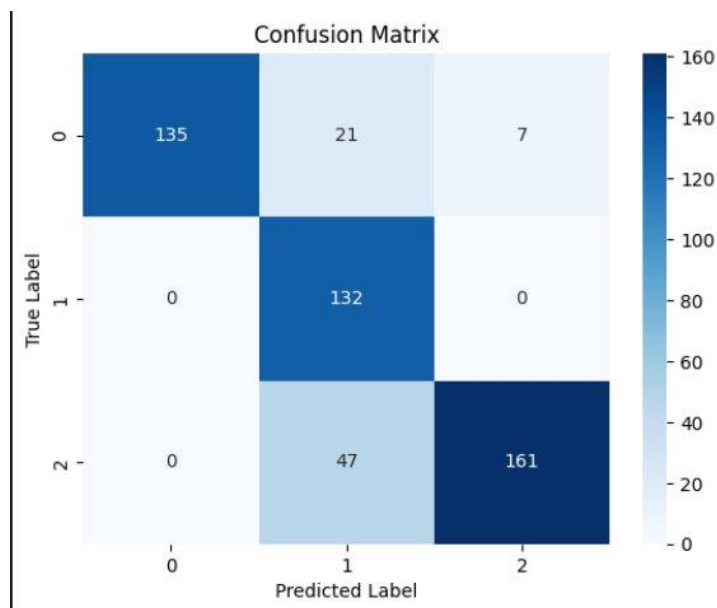


Figure 21 - Confusion Matrix (MLP)

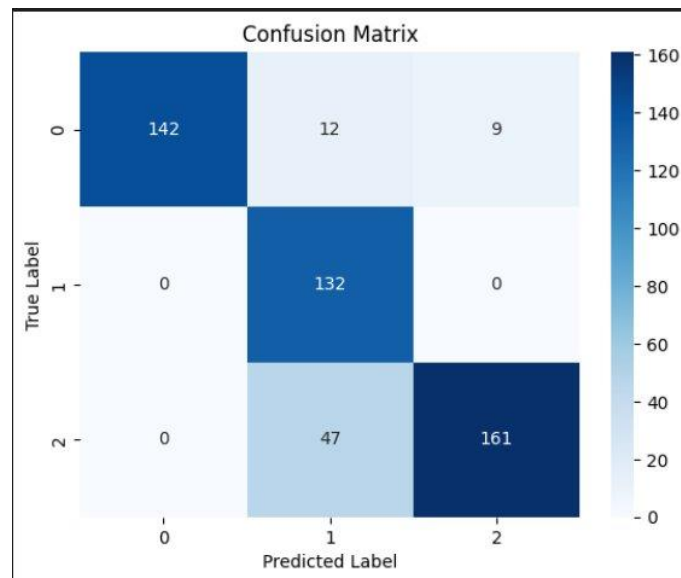


Figure 22 - Confusion Matrix (Naive Bayes)

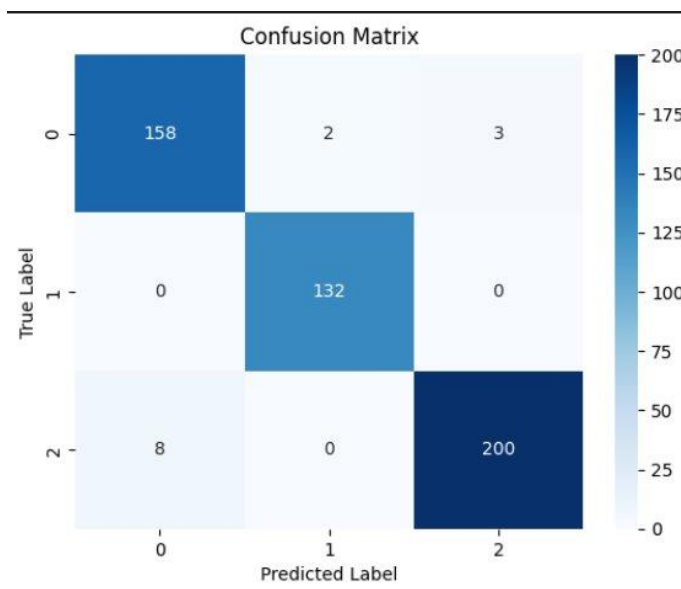


Figure 23 - Confusion Matrix (KNN)

2. Accuracy: This metric represents the proportion of correctly predicted observations out of the total observations.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Figure 24 - Accuracy

3. Precision: Precision measures the ratio of correctly predicted positive values to the total number of predicted positive values.

$$Precision = \frac{TP}{TP + FP}$$

*Figure 25 - Precision*

4. Recall: Recall calculates the ratio of correctly predicted positive values to all actual positive values.

$$Recall = \frac{TP}{TP + FN}$$

*Figure 26 - Recall*

5. F1-score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance in classification tasks.

$$F1 - Score = 2 \cdot \frac{(Precision \cdot Recall)}{(Precision + Recall)}$$

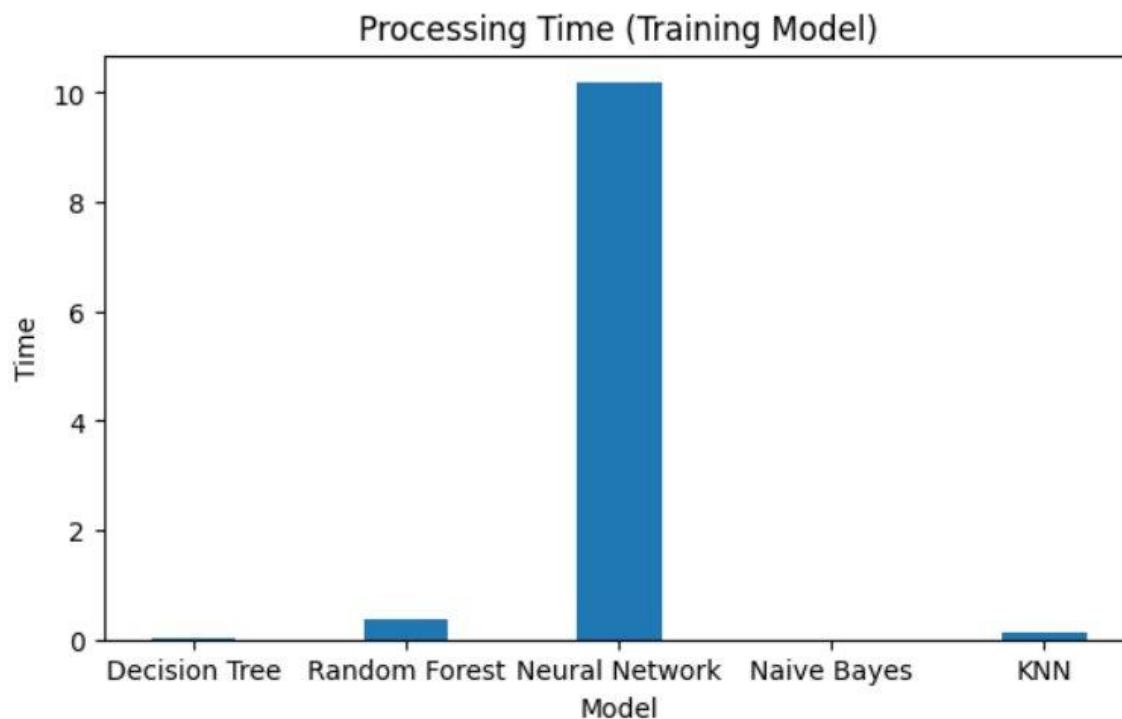
*Figure 27 - F1-Score*

# Chapter 7

## 7 Results

A synthetic conclusion was derived, and how the compilation process was designed and applied.

The study delves into a comprehensive analysis of various machine learning models, focusing on their performance and computational efficiency. By examining training and prediction times, as well as metrics such as Precision, Recall, F1-Score, and Accuracy, the study sheds light on the strengths and weaknesses of each model. This exploration provides valuable insights for selecting the most suitable models for IDS in machine learning applications.

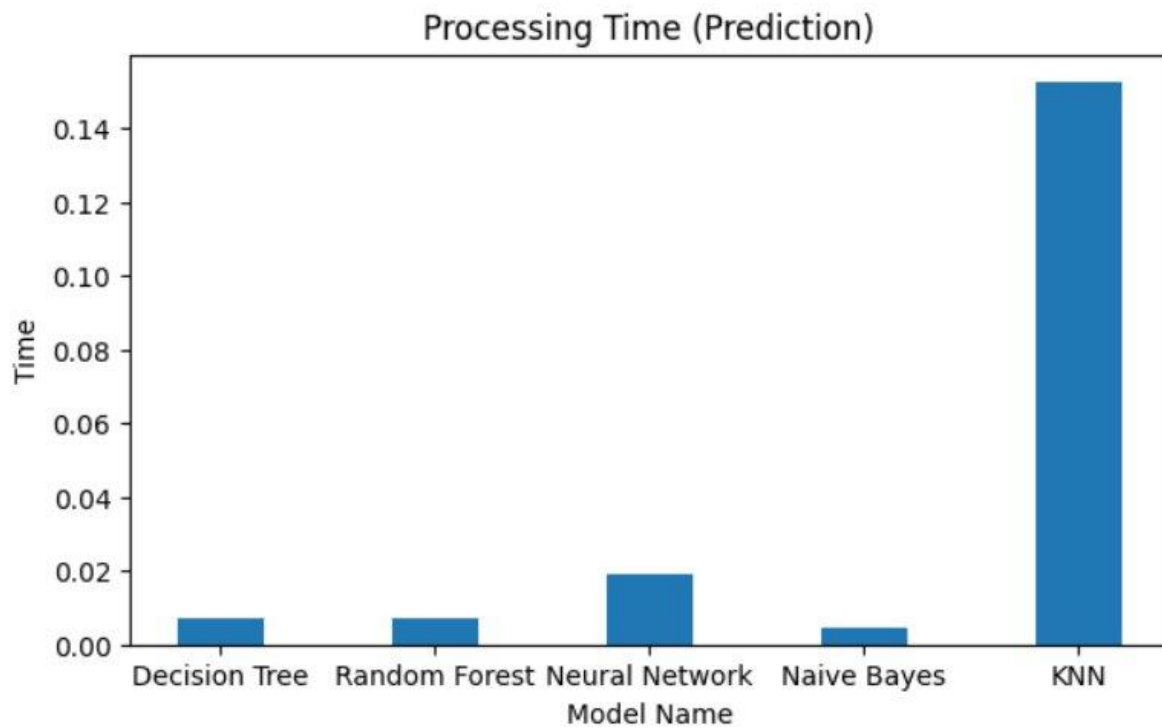


*Figure 28 - Training Time Comparison*

The bar chart titled "Processing Time (Training Model)" visually compares the time taken to train different machine learning models. The x-axis displays the models, while the y-axis shows the training time in seconds.

From the chart analysis, it's evident that the Neural Network model exhibits significantly longer training times compared to other models, with a processing time value of approximately 10 seconds. Conversely, models such as Decision Tree, Random Forest, Naive Bayes, and KNN display very short bars on the chart, indicating minimal processing time close to zero.

This information is crucial for discussing the computational efficiency of each model during the training phase, highlighting the relative speed at which these models can be trained and utilized for intrusion detection tasks.



*Figure 29 - Prediction Time Comparison*

The bar chart depicted in Figure 23 compares the prediction processing time among five distinct machine learning models. The analysis reveals that the K-Nearest Neighbors (KNN) algorithm necessitated the most time for processing predictions, approximately 0.14 seconds. In contrast, both the Decision Tree and Random Forest models exhibited swift prediction times, both clocking in at less than 0.02 seconds. Similarly, the Neural Network and Naive Bayes models showcased relatively rapid processing times, indicating their computational efficiency compared to the KNN model for prediction tasks within this specific context. These findings imply that in scenarios where prediction speed holds significance, opting for the Decision Tree and Random Forest models might be advantageous.





*Figure 30 - Performance Analysis*

The clustered bar chart titled "Performance Analysis" displays a comparison of different machine learning models based on Precision, Recall, F1-Score, and Accuracy metrics. The chart includes six models represented by different colored bars:

- Decision Tree (blue)
- Random Forest (orange)
- MLP (Multi-Layer Perceptron) (grey)
- Naïve Bayes (yellow)
- KNN (K-Nearest Neighbors) (light green)
- Decision Tree (Online Dataset) (dark green)

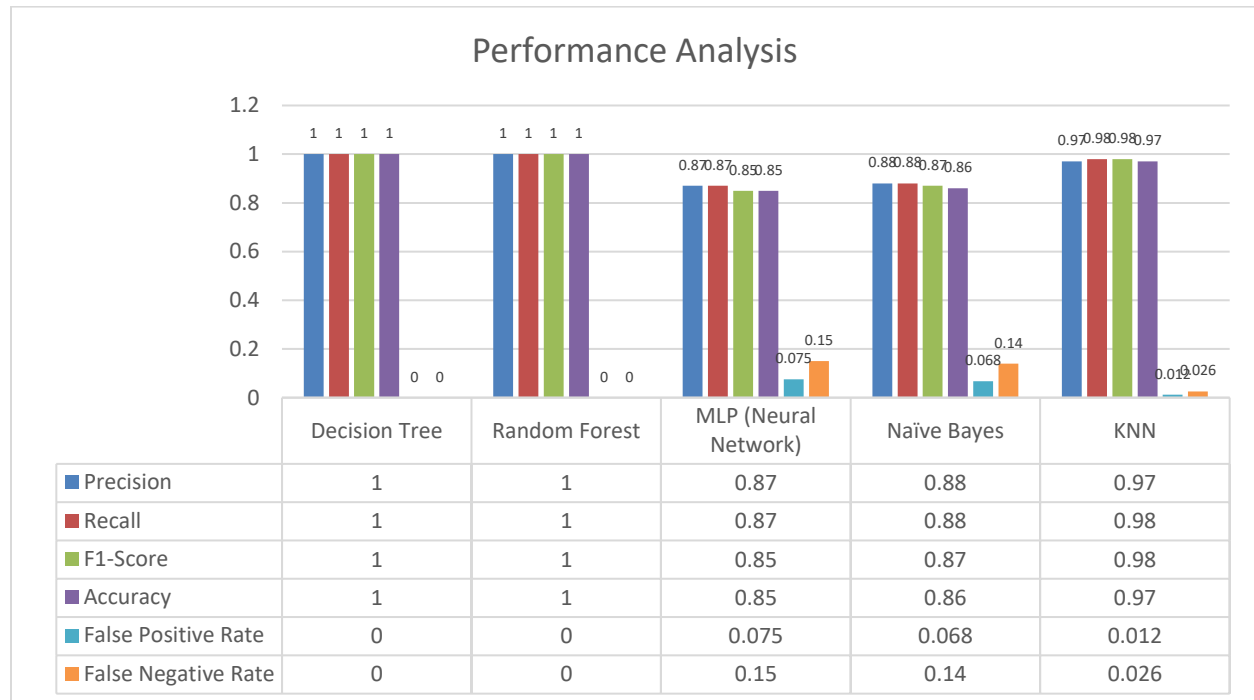
Upon examining the chart, several observations can be made:

- Both the Decision Tree and Random Forest models exhibit perfect scores of 1 across all metrics.
- The MLP model achieves scores of 0.87 for Precision, Recall, and Accuracy, with a slightly lower score of 0.85 for the F1-Score.
- The Naïve Bayes model obtains scores of 0.88 for Precision, Recall, and F1-Score, and a slightly lower score of 0.86 for Accuracy.

## Detection And Response: Enhancing Intrusion Detection Systems

- The KNN model demonstrates a high Precision score of 0.97, the highest Recall score of 0.98 among non-perfect models, an F1-Score of 0.98, and an Accuracy score of 0.97.
- The Decision Tree (Online Dataset) consistently scores 0.89 across all metrics.

Based on these findings, it is evident that the Decision Tree and Random Forest models perform exceptionally well on the analyzed datasets, outperforming the other models in terms of Precision, Recall, F1-Score, and Accuracy.



*Figure 31 - Performance Analysis 2*

The comparison and analysis of various machine learning models reveal insightful findings regarding their performance and computational efficiency. In terms of training time, the Neural Network model stands out with significantly higher processing requirements, while Decision Tree and Random Forest models demonstrate remarkable efficiency. Performance-wise, Decision Tree and Random Forest models excel across multiple metrics, showcasing their robustness in classification tasks. Furthermore, when considering prediction time as a crucial factor, the Decision Tree and Random Forest models emerge as top choices due to their swift processing capabilities. These results emphasize the importance of selecting appropriate models based on IDS requirements for optimal performance in machine learning applications.

# Chapter 8

## 8 Conclusion

The culmination of our research into Intrusion Detection Systems (IDS) and their effectiveness in detecting brute force attacks reveals a nuanced understanding of the intricate dynamics within network security. Our comprehensive exploration, utilizing the CSE-CIC-IDS-2018 dataset and a tailored data collection setup, has provided valuable insights into the capabilities and limitations of machine learning (ML) models for cybersecurity applications.

Through meticulous data preprocessing, robust model training, and rigorous performance evaluations, we have unearthed key findings that significantly contribute to the advancement of IDS technology. One of the pivotal outcomes of our study is the identification of critical feature dimensions that profoundly influence classification performance. This understanding is instrumental in enhancing detection accuracy and minimizing false positives and false negatives, thereby improving overall IDS efficacy.

Our evaluation of diverse ML classifiers for detecting SSH and FTP brute force attacks underscores the importance of algorithm selection in IDS design. Notably, the Decision Tree and Random Forest models emerged as top performers, showcasing remarkable efficiency in both training and prediction phases. These models exhibited high precision, recall, F1-score, and accuracy metrics, highlighting their robustness and suitability for real-time intrusion detection tasks.

Furthermore, our study emphasizes the significance of feature selection methodologies, such as Random Forest-based feature reduction, in streamlining model complexity and enhancing processing speed. This approach optimizes resource utilization and facilitates the deployment of scalable and efficient IDS solutions in dynamic cybersecurity environments. Looking ahead, future research endeavors may explore advanced ML techniques, ensemble learning strategies, and integration of real-time threat intelligence to further fortify IDS capabilities. Continual refinement and innovation in ML-based IDS are paramount in staying ahead of evolving cyber threats and safeguarding digital ecosystems effectively. Our study serves as a stepping stone towards developing resilient and adaptive IDS frameworks that can mitigate emerging security challenges comprehensively.

# Chapter 9

## **9 General Information**

### **9.1 Definition of Terms**

- Intrusion Detection Systems (IDS): Software or hardware systems designed to detect and respond to unauthorized access or malicious activities within computer networks.
- Brute Force Attack: A cyber attack method involving systematic attempts to gain access to a system by trying multiple username-password combinations.
- Machine Learning (ML): A subset of artificial intelligence (AI) that enables systems to learn from data and improve performance over time without explicit programming.
- CSE-CIC-IDS-2018 Dataset: A benchmark dataset containing network traffic data used for evaluating intrusion detection systems.

### **9.2 Research Management Plan**

- Objective: To investigate the effectiveness of ML techniques for detecting brute force attacks in IDS systems.
- Timeline: The research is conducted over our final semester, including data collection, preprocessing, model training, evaluation, and reporting phases.
- Resources: Utilizing Python, Google Colab, Numpy, Pandas, Scikit Learn, and visualization tools like Seaborn and Matplotlib for data analysis and presentation.
- Team Collaboration: Collaborative efforts among our team members to ensure comprehensive analysis and interpretation of results.
- Outputs: The research outcomes include performance evaluations, computational efficiency analyses, and recommendations for implementing ML-based IDS solutions.

# Chapter 10

## 10 References

- [1] (PDF) Performance analysis of intrusion detection for deep learning model based on CSE-CIC-IDS2018 dataset. (2020, November 1). ResearchGate. [https://www.researchgate.net/publication/360104704\\_Performance\\_analysis\\_of\\_intrusion\\_detection\\_for\\_deep\\_learning\\_model\\_based\\_on\\_CSE-CIC-IDS2018\\_dataset](https://www.researchgate.net/publication/360104704_Performance_analysis_of_intrusion_detection_for_deep_learning_model_based_on_CSE-CIC-IDS2018_dataset)
- [2] (PDF) SSH-Brute Force Attack Detection Model based on Deep Learning. (2023, February 1). ResearchGate. [https://www.researchgate.net/publication/348541012\\_SSH-Brute\\_Force\\_Attack\\_Detection\\_Model\\_based\\_on\\_Deep\\_Learning](https://www.researchgate.net/publication/348541012_SSH-Brute_Force_Attack_Detection_Model_based_on_Deep_Learning)
- [3] Aljanabi, M.; Ismail, M.A.; Ali, A.H. Intrusion detection systems, issues, challenges, and needs. *Int. J. Comput. Intell. Syst.* **2021**, *14*, 560–571. [[Google Scholar](#)] [[CrossRef](#)]
- [4] Alzaqebah, A.; Aljarah, I.; Al-Kadi, O.; Damaševičius, R. A Modified Grey Wolf Optimization Algorithm for an Intrusion Detection System. *Mathematics* **2022**, *10*, 999. [[Google Scholar](#)] [[CrossRef](#)]
- [5] Ambusaidi, M.A.; He, X.; Nanda, P.; Tan, Z. Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Trans. Comput.* **2016**, *65*, 2986–2998. [[Google Scholar](#)] [[CrossRef](#)]
- [6] CanadianInstituteForCybersecurity. (n.d.). GitHub - CanadianInstituteForCybersecurity/CICFlowMeter: CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a network traffic Bi-flow generator and analyzer for anomaly detection that has been used in many Cybersecurity datasets such as Android Adware-General Malware dataset (CICAAGM2017), IPS/IDS dataset (CICIDS2017), Android Malware dataset (CICAndMal2017) and Distributed Denial of Service (CICDDoS2019). GitHub. <https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter>
- [7] Chimphee, S.; Chimphee, W. Machine learning to improve the performance of anomaly-based network intrusion detection in big data. *Indones. J. Electr. Eng. Comput. Sci.* **2023**, *30*, 1106–1119. [[Google Scholar](#)] [[CrossRef](#)]
- [8] Gautam, R.K.S.; Doegar, E.A. An Ensemble Approach for Intrusion Detection System Using Machine Learning Algorithms. In Proceedings of the 2018 8th International Conference on Cloud Computing, Data Science & Engineering, Noida, India, 11–12 January 2018. [[Google Scholar](#)]
- [9] IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. (n.d.). <https://www.unb.ca/cic/datasets/ids-2018.html>
- [10] Jaradat, A.S.; Barhoush, M.M.; Easa, R.B. Network intrusion detection system: Machine learning approach. *Indones. J. Electr. Eng. Comput. Sci.* **2022**, *25*, 1151–1158. [[Google Scholar](#)] [[CrossRef](#)]
- [11] Kaja, N.; Shaout, A.; Ma, D. An intelligent intrusion detection system. *Appl. Intell.* **2019**, *49*, 3235–3247. [[Google Scholar](#)] [[CrossRef](#)]
- [12] Karatas, G.; Demir, O.; Sahingoz, O.K. Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset. *IEEE Access* **2020**, *8*, 32150–32162. [[Google Scholar](#)] [[CrossRef](#)]
- [13] Khan, M.A. HCRNNIDS: Hybrid convolutional recurrent neural network-based network intrusion detection system. *Processes* **2021**, *9*, 834. [[Google Scholar](#)] [[CrossRef](#)]
- [14] Kim, J.; Shin, Y.; Choi, E. An Intrusion Detection Model based on a Convolutional Neural Network. *J. Multimed. Inf. Syst.* **2019**, *6*, 165–172. [[Google Scholar](#)] [[CrossRef](#)]
- [15] Malliga, S.; Nandhini, P.S.; Kogilavani, S.V. A Comprehensive Review of Deep Learning Techniques for the Detection of (Distributed) Denial of Service Attacks. *Inf. Technol. Control* **2022**, *51*, 180–215. [[Google Scholar](#)] [[CrossRef](#)]
- [16] Momand, A.; Jan, S.U.; Ramzan, N. A Systematic and Comprehensive Survey of Recent Advances in Intrusion Detection Systems Using Machine Learning: Deep Learning, Datasets, and Attack Taxonomy. *J. Sens.* **2023**, *2023*, 6048087. [[Google Scholar](#)] [[CrossRef](#)]
- [17] Muhsen, A.R.; Jumaa, G.G.; Bakri, N.F.A.; Sadiq, A.T. Feature Selection Strategy for Network Intrusion Detection System (NIDS) Using Meerkat Clan Algorithm. *Int. J. Interact. Mob. Technol.* **2021**, *15*, 158–171. [[Google Scholar](#)] [[CrossRef](#)]
- [18] Nassif, A.B.; Talib, M.A.; Nasir, Q.; Dakalbab, F.M. Machine Learning for Anomaly Detection: A Systematic Review. *IEEE Access* **2021**, *9*, 78658–78700. [[Google Scholar](#)] [[CrossRef](#)]
- [19] patator | Kali Linux Tools. (n.d.). Kali Linux. <https://www.kali.org/tools/patator/>



- [20] Qusyairi, R.; Saeful, F.; Kalamullah, R. Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems. In Proceedings of the International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), Bali, Indonesia, 7–8 July 2020. [\[Google Scholar\]](#)
- [21] Songma, S., Sathuphan, T., & Pamutha, T. (2023). Optimizing intrusion detection systems in three phases on the CSE-CIC-IDS-2018 dataset. *Computers*, 12(12), 245. <https://doi.org/10.3390/computers12120245>
- [22] Talukder, M. A., Islam, M. M., Uddin, M. A., Hasan, K. F., Sharmin, S., Alyami, S. A., & Moni, M. A. (2024). Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *Journal of Big Data*, 11(1). <https://doi.org/10.1186/s40537-024-00886-w>
- [23] Ullah, S.; Mahmood, Z.; Ali, N.; Ahmad, T.; Buriro, A. Machine Learning-Based Dynamic Attribute Selection Technique for DDoS Attack Classification in IoT Networks. *Computers* **2023**, 12, 115. [\[Google Scholar\]](#) [\[CrossRef\]](#)
- [24] Umar, M.A.; Chen, Z. Effects of Feature Selection and Normalization on Network Intrusion Detection. *TechRxiv* **2020**. preprint. [\[Google Scholar\]](#) [\[CrossRef\]](#)