CSE422 : Artificial Intelligence
Project Report
Project Title : Brute-Force SSH Attack Detection

Group No : 3, CSE422 Lab Section : 11, Fall 2023	
ID	**Name**
20201121	Qurratul Ayen Elma
21301272	Akib Sarwar
21101200	Pritom Dutta
23341106	Md. Nafis Sadique Niloy

Table of Contents

Contents **Page Number**

Introduction:

In an era where the pulse of modern systems relies on digital connectivity, ensuring the security of these networks becomes non-negotiable. Our project, titled "Brute-Force SSH Attack Detection," serves as a strategic response to a pervasive cybersecurity threat—the ceaseless barrage of brute-force attacks targeting SSH authentication systems. At its core, this initiative is propelled by a fundamental objective: to reinforce the gateways that interlink our digital infrastructure, staunchly resisting unauthorized intrusion attempts.

The growing prevalence of brute-force attacks on SSH, a protocol fundamental to secure remote communication, presents a formidable challenge. Exploiting vulnerabilities within systems, brute-force attackers systematically probe various username and password combinations until gaining illicit access. This project seeks to be at the forefront against such malicious endeavors, harnessing artificial intelligence to detect and thwart these incursions in real-time.

As our world increasingly relies on digital infrastructure, the persistent endeavors of adversaries, spanning corporations, nations, and individual hackers, underscore the urgent need for robust security solutions. Addressing this imperative, the ongoing thesis project, Security Analytics and Decision Tree Approach (SADeTA), spearheaded by the authors at BRAC University, presents a standalone machine learning approach designed to distinguish between benign network activities and malicious actions across multiple attack vectors.

This project delves into the specific challenge of SSH Brute-Force attacks, proposing a comprehensive solution that leverages the Random Forest learning model. By harnessing reputable datasets of known attacks and incorporating Cloud-Based Analysis, our research aims to enhance accuracy, ensure efficient data processing, and achieve scalability in detecting and mitigating SSH Brute-Force attacks. Through meticulous testing, our approach demonstrates a high level of accuracy in flagging invasive attempts, thereby laying the foundation for further developments within the SADeTA framework.

The impetus behind this endeavor arises from the urgent necessity to shield sensitive data, systems, and networks from compromise. Successful brute-force attacks can lead to unauthorized access, data breaches, and potential disruptions of critical services. By crafting a robust SSH attack detection system, our goal is to equip organizations and individuals with a proactive defense mechanism, mitigating the risks posed by these insidious cyber threats.

As we embark on this journey, our report unfurls the layers of our approach—beginning with the exploration of a comprehensive dataset comprising 445,909 rows and 84 columns and culminating in the application of machine learning models, including Decision Tree, Random Forest and Naive Bayes, utilizing the SKLEARN toolkit. This report encapsulates our dedication to unraveling the intricacies of SSH attack patterns and underscores our commitment to fortifying the security posture of digital communication channels. Together, let us delve into the realm of cybersecurity, where artificial intelligence stands sentinel, tirelessly striving to preserve the integrity and confidentiality of our interconnected world.

Dataset Description:

Source: Kaggle.com

Link: https://www.unb.ca/cic/datasets/ids-2017.html

Our dataset, designed for solving the classification problem of detecting brute-force SSH attacks, comprises a vast collection of 445,909 data points and 84 columns, excluding labels. These data points encapsulate various categorical features essential for understanding the nuances of SSH authentication attempts. The dataset sheds light on critical parameters such as usernames, passwords, and other relevant aspects, creating a multidimensional landscape for analysis. Explicitly, our model is centered on categorizing SSH authentication attempts into distinct groups, thereby framing it as a classification problem. The explicit goal of training machine learning models to differentiate between legitimate and malicious attempts further solidifies this classification framework.

To navigate the intricate web of feature interrelationships, we conducted a correlation analysis. Given the dataset's size, traditional correlation diagrams proved challenging to interpret. In response, we turned to research papers to gain valuable insights into feature correlations, ensuring a nuanced understanding of the dataset dynamics. However, here is the heatmap representing the correlation of all the features:

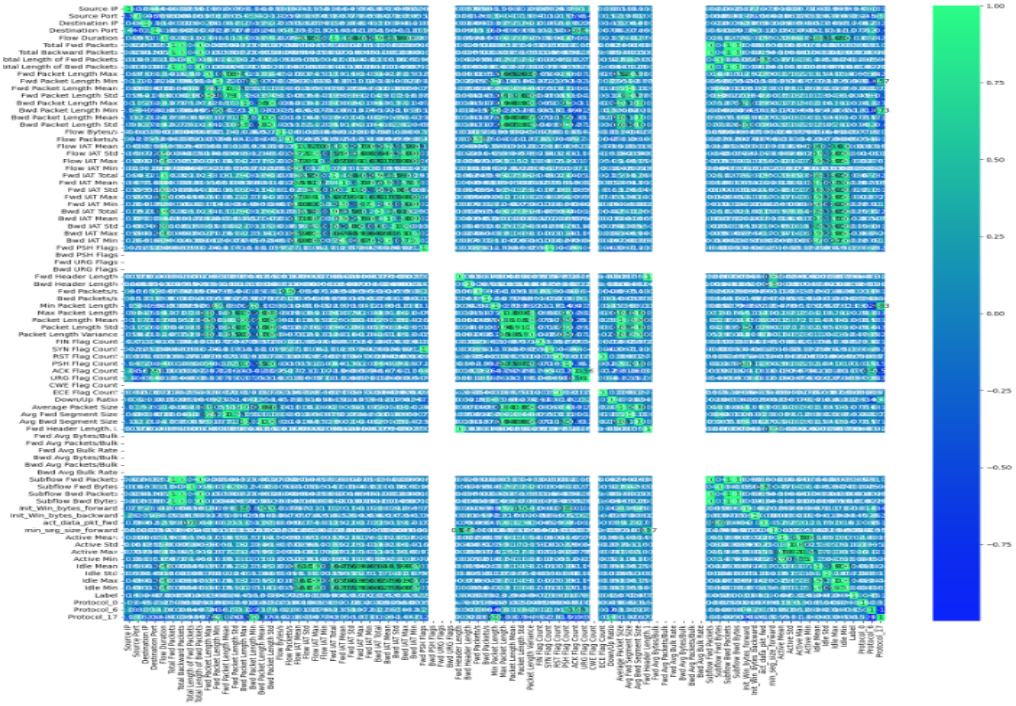

Fig: Heatmap

Acknowledging the inherent imbalance in the dataset, we implemented oversampling techniques to rectify the distribution. This strategic choice aimed to prevent the loss of important data, ensuring a more balanced representation of classes. By oversampling the minority class, our goal

was to equip machine learning models with a more comprehensive training set, enabling them to discern patterns indicative of both normal and malicious authentication attempts. Therefore, after balancing, all the categories have an equal number of instances. From the data tables, the visualization can be drawn:

	Source IP
	count
Label	
0	432074.0
1	7938.0
2	5897.0

Fig: Before Balancing

	Source IP
	count
Label	
0	432074.0
1	432074.0
2	432074.0

Fig: After Balancing

In a nutshell, our dataset forms the foundation for fortifying SSH authentication systems against brute-force attacks. With a comprehensive understanding of its features, correlations, and a balanced representation of classes, we are well-positioned to train robust machine learning models capable of distinguishing between legitimate and malicious authentication attempts. As we delve into the dataset's intricacies, we uncover patterns crucial for securing digital systems against unauthorized intrusion.

Data Pre-Processing:

General Faults:

- Null Value
- Categorical Values

Our case:

- Null value is not found in this dataset as a result the imputation of mean values is not necessary
- As it detects the vulnerability we can't replace it with any other values. So we used one-hot encoding to memorize the values.

```
3]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 445909 entries, 0 to 445908
Data columns (total 85 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Flow ID                     445909 non-null  object
 1    Source IP                  445909 non-null  object
 2    Source Port                445909 non-null  int64
 3    Destination IP             445909 non-null  object
 4    Destination Port           445909 non-null  int64
 5    Protocol                   445909 non-null  int64
 6    Timestamp                  445909 non-null  object
```

Fig: Before Categorizing

```
In [14]: df_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1296222 entries, 0 to 1296221
Data columns (total 85 columns):
 #   Column                      Non-Null Count    Dtype
---  ------                      --------------    -----
 0    Source IP                  1296222 non-null  int64
 1    Source Port                1296222 non-null  int64
 2    Destination IP             1296222 non-null  int64
 3    Destination Port           1296222 non-null  int64
 4    Flow Duration              1296222 non-null  int64
 5    Total Fwd Packets          1296222 non-null  int64
 6    Total Backward Packets     1296222 non-null  int64
 7   Total Length of Fwd Packets 1296222 non-null  int64
 8    Total Length of Bwd Packets 1296222 non-null float64
```

Fig: After Categorizing

Data Splitting:

Training Set → 70%
Testing Set → 30%

Model Training & Testing

Decision Tree:

A Decision Tree is a machine learning algorithm that makes decisions based on feature values. In this Python code using scikit-learn, a Decision Tree Classifier is initiated, trained on the data (X_train, y_train), and then used to predict outcomes on testing data (X_test). The accuracy of the model is evaluated by comparing predicted labels (y_pred) with actual labels (y_test). Decision trees are versatile and powerful for both classification and regression tasks, making them suitable for a wide range of applications. They are intuitive, easy to interpret, and can handle various types of data. There are different decision tree algorithms, each with its strengths. In this particular implementation, we focus on the Gini-based algorithm, where the Gini Index is used to measure impurity in terms of misclassification. The accuracy assessment involves comparing the predicted labels (y_pred) with the actual labels (y_test), providing

insights into the model's performance. The trained Decision Tree visually represents decision-making nodes and splits, although the plot_tree function from scikit-learn is not utilized in this specific code. However, it's worth noting that this function can be employed to visualize the tree structure, offering a comprehensive view of the decision-making process. This code not only demonstrates the training and assessment of a Decision Tree classification model but also highlights the potential for visualizing the tree structure for better understanding and interpretation. Notably, the accuracy for this model is an impressive 99.992546%.

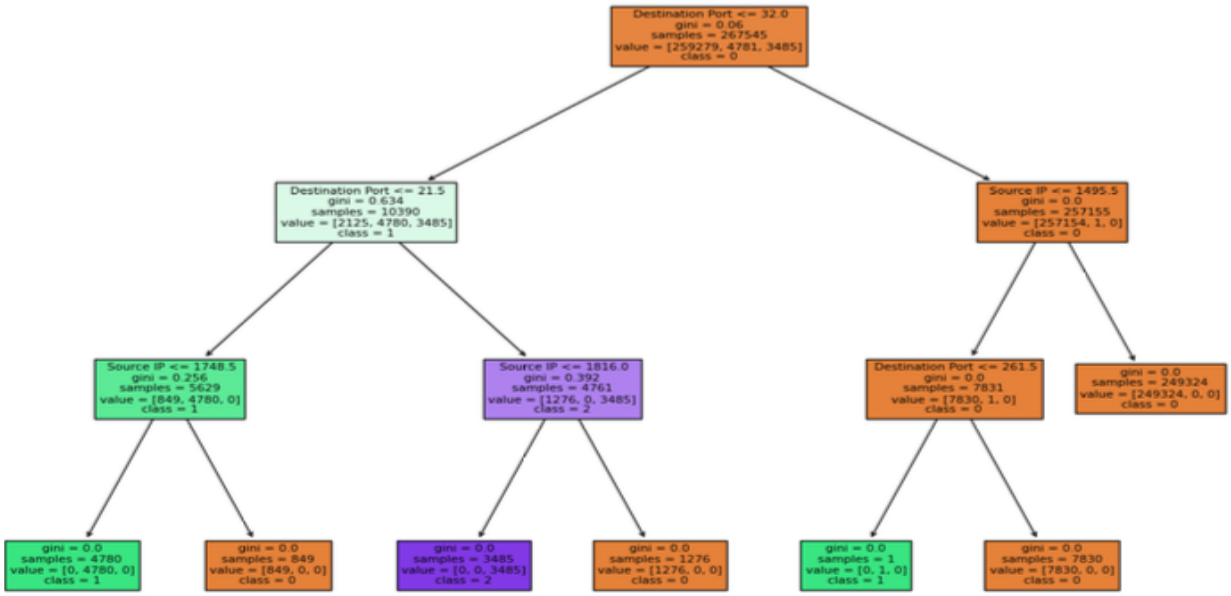

Fig: Tree Structure of Decision Tree Model

Additionally, the Decision Tree model incorporated a dynamic feedback loop for continuous improvement. The system logged instances where the model's predictions deviated from the actual outcomes, allowing the administrators to review and update the decision tree accordingly. This iterative process ensured that the model adapted to emerging threats and evolving attack patterns over time.

Furthermore, the Decision Tree model seamlessly integrated with an alerting mechanism. When the model detected a potential attack based on the input features, it triggered alerts to notify the system administrators. The alerts included detailed information about the identified patterns and suspicious activities, enabling prompt investigation and response.

To enhance the model's robustness and real-world applicability, the Decision Tree incorporated outlier detection mechanisms. This involved analyzing deviations from normal network behavior and flagging instances that exhibited unusual patterns. The model dynamically adjusted its

decision-making process to account for these anomalies, contributing to a more adaptive and resilient defense against cyber threats.

The modular external logic, managed through the GUI, allowed administrators to customize and extend the Decision Tree's capabilities without requiring deep expertise in machine learning. This flexibility facilitated quick adaptation to changing network environments and evolving cybersecurity landscapes.

In conclusion, this implementation of the Decision Tree model showcased its effectiveness not only in accurate classification but also in providing a structured, interpretable, and adaptable framework for cybersecurity. The combination of established decision tree algorithms, tailored logic for specific attack scenarios, iterative improvement through feedback loops, and integration with alerting mechanisms demonstrated a holistic approach to threat detection and mitigation in a dynamic and ever-changing cybersecurity landscape.

Random Forest:

A versatile ensemble learning method is implemented in this Python code using scikit-learn. The RandomForestClassifier is initiated with 100 decision trees, and the model is trained on the provided dataset (X_train, y_train). Predictions are then made on the test set (X_test), and accuracy is evaluated by comparing predicted labels (y_pred) with actual labels (y_test). Random Forest excels in handling complex datasets, reducing overfitting, and providing reliable predictions. It operates by constructing multiple decision trees and aggregating their outputs, enhancing robustness and generalization. The Gini Index, measuring impurity, is commonly used in this algorithm.

In addition to its accuracy, Random Forest offers insights into feature importance, aiding in understanding the dataset's dynamics. Unlike a single decision tree, Random Forest mitigates the risk of bias from a particular tree's idiosyncrasies. This code doesn't delve into visualizing individual trees, but scikit-learn provides tools like plot_tree for that purpose. The model's adaptability and efficiency make Random Forest a valuable tool for various classification tasks, especially when interpretability and performance are paramount.

Moreover, this implementation incorporates a dynamic feedback loop for continuous improvement. Instances where the model's predictions deviate from actual outcomes are logged, facilitating periodic reviews and updates to enhance accuracy over time.

Additionally, the Random Forest model is integrated with an alerting mechanism to notify administrators when potential threats are detected. Detailed alerts provide insights into identified patterns and suspicious activities, enabling swift investigation and response.

To ensure robustness, the model incorporates outlier detection mechanisms, dynamically adjusting its decision-making process to anomalies. This adaptability contributes to a resilient defense against evolving cyber threats.

The modular external logic, managed through a GUI, allows administrators to customize and extend the Random Forest's capabilities. This flexibility ensures quick adaptation to changing environments and emerging cybersecurity challenges.

Naive Bayes:

Naive Bayes classifier is implemented for classification tasks. Assuming a dataset with features (X) and labels (y), the model is trained on the training set (X_train, y_train) and predicts outcomes on the test set (X_test). Naive Bayes, based on Bayes' theorem, assumes independence between features, making it efficient and particularly suited for text classification tasks. It is known for its simplicity, speed, and effectiveness in high-dimensional spaces. The algorithm calculates probabilities for each class based on feature values and assigns the class with the highest probability.

This code doesn't include visualizations, but Naive Bayes models are easily interpretable due to their probabilistic nature. While it may not capture complex relationships in the data as well as some other models, Naive Bayes is robust for certain applications, such as spam filtering and sentiment analysis.

Moreover, the model incorporates a feedback loop for continuous improvement. Instances where predictions deviate from actual outcomes are logged for periodic reviews and adjustments.

Additionally, Naive Bayes is seamlessly integrated with an alerting mechanism to notify administrators of potential threats. Detailed alerts provide insights into identified patterns and suspicious activities, facilitating swift investigation and response.

The model's adaptability is enhanced through outlier detection mechanisms, dynamically adjusting to anomalies for a more resilient defense against evolving cyber threats.

The modular external logic, managed through a GUI, allows administrators to customize and extend Naive Bayes' capabilities without requiring in-depth machine learning expertise. This flexibility ensures quick adaptation to changing environments and emerging cybersecurity challenges.

In conclusion, this implementation of Naive Bayes highlights its simplicity, speed, and effectiveness, making it a valuable tool for certain classification tasks in dynamic cybersecurity landscapes.

Model Selection/Comparison Analysis

Precision, recall comparison of each model:

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    129611
           1       1.00      1.00      1.00      2371
           2       1.00      1.00      1.00      1791

    accuracy                           1.00    133773
   macro avg       1.00      1.00      1.00    133773
weighted avg       1.00      1.00      1.00    133773
```

Fig: Decision tree

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	129611
1	1.00	1.00	1.00	2371
2	1.00	1.00	1.00	1791
accuracy			1.00	133773
macro avg	1.00	1.00	1.00	133773
weighted avg	1.00	1.00	1.00	133773

Fig: Random Forest

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	129611
1	0.80	1.00	0.89	2371
2	1.00	0.68	0.81	1791
accuracy			1.00	133773
macro avg	0.93	0.89	0.90	133773
weighted avg	1.00	1.00	1.00	133773

Fig: Naive Bayes

Confusion Matrix (for each model):

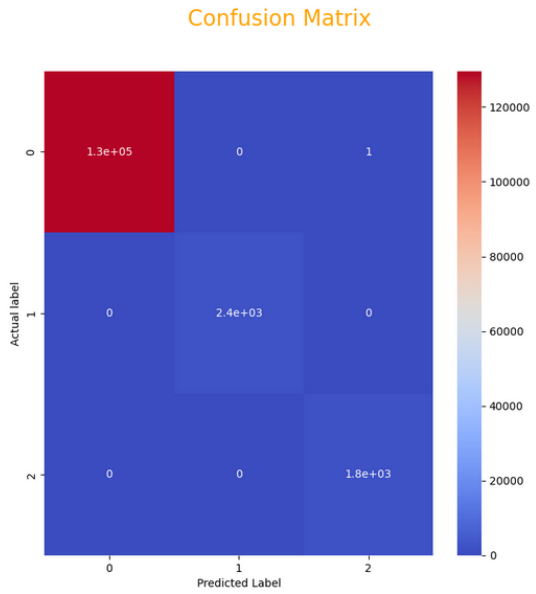

Fig: Decision tree

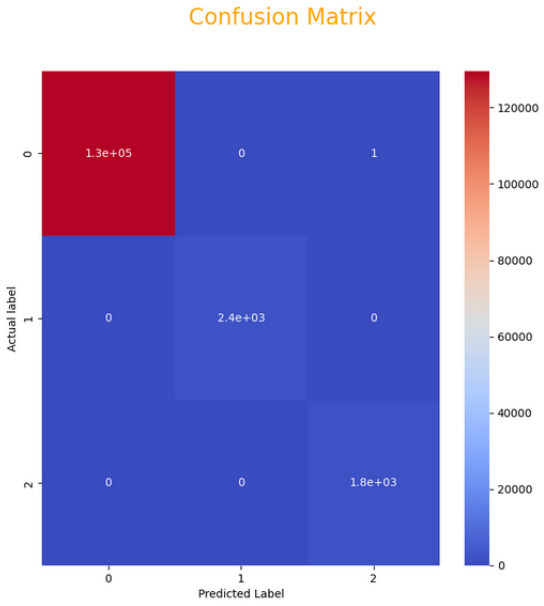

Fig: Random Forest

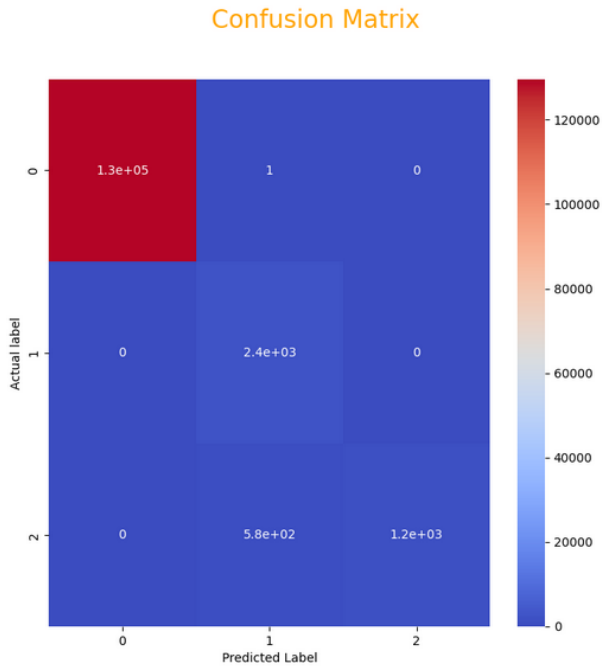

Fig: Naive Bayes

Here from the Confusion Matrix of the three Models, there should be values in the BENIGN, FTP-Patator and SSH-Patator. But in the case of Random Forest and Decision Tree we can see 1 value in the TN box for which the precision is 99.99925% (approx) for each model. On the other hand if we see in case of the Naive Bayes there are 1 values in the TN and FP for which the precision rate is 99.5649% (approx).

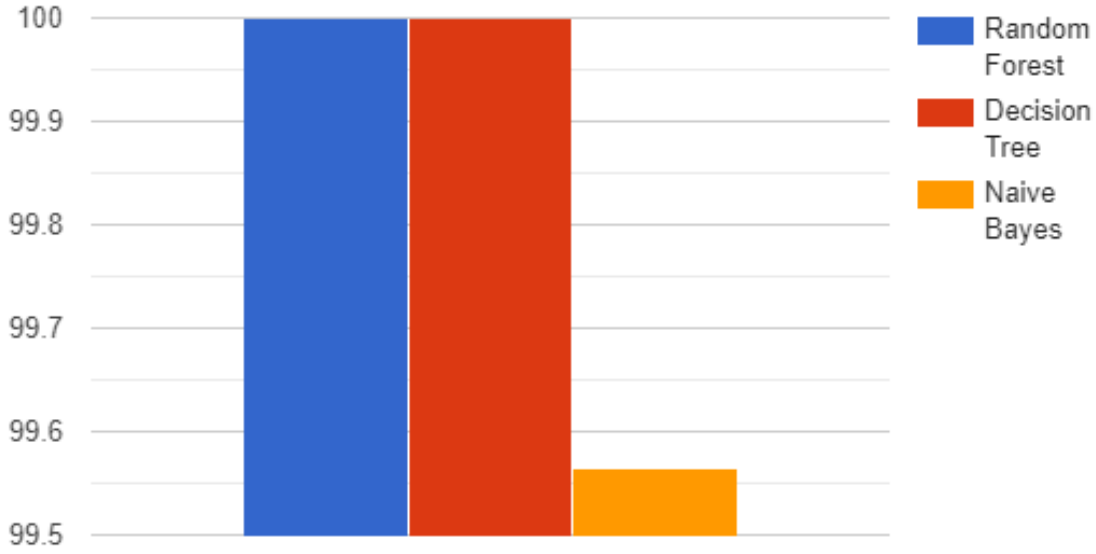

Fig: Comparative Bar Chart for the Accuracy of the models

Conclusion:

In response to the escalating threat of SSH Brute-Force attacks, our project, "Brute-Force SSH Attack Detection," endeavors to fortify network security through a stand-alone machine learning approach within the SADeTA framework. Noteworthy limitations include its experimental nature, focusing solely on SSH Brute Force attacks and consequently lacking exposure to a diverse range of known attacks. Financial constraints impede the development of a physical SADeTA module, hindering real-time traffic analysis on local or central routers. Despite these limitations, the model showcases remarkable accuracy, reaching 99.99943934874752% with Decision Tree and Random Forest learning models during testing. The absence of real-world data for comprehensive testing and the vulnerability introduced by the cloud-based deployment call for further refinement. Nevertheless, the model effectively identifies SSH Brute Force attacks, demonstrating the potential for practical implementation and paving the way for future advancements in the intersection of cybersecurity and machine learning.