RTMCA: Real-Time Machine Learning model for large-scale cybersecurity attacks

Mario de Lucas Garcia
Illinois Institute of Technology — Universidad Politécnica de Madrid
Department of Computer Science — ETSI Informáticos
`mdelucasgarcia@hawk.iit.edu` — `m.delucas@alumnos.upm.es`

Ignacio Gomez Valverde
Illinois Institute of Technology — Universidad Politécnica de Madrid
Department of Computer Science — ETSI Informáticos
`igomezvalverde@hawk.iit.edu` — `ignacio.gvalverde@alumnos.upm.es`

Abstract

The surge in cybersecurity threats has needed a deeper understanding and innovative solutions to counteract malicious activities. Among these threats, this study [1] collected data from seven categories: DDoS, DoS, Mirai, spoofing, recon, web and brute force. In this paper, we present some approaches for identifying those attacks. These approaches include *Logistic Regression*, *Random Forest*, *XGBoost* and *Voting Classifier*. Our best estimator, our *XGBoost*, achieved an impressive accuracy of **99,9 %**.

Moreover, we introduce a real-time machine-learning model for detecting SYN flood attacks, demonstrating an accuracy ranging from **85%** to **96%** depending on the analyzed traffic network at a given moment.

In conclusion, this paper addresses open issues and challenges in cybersecurity while also delving into potential future research directions.

1 Introduction

The proliferation of small personal devices (e.g., smartphones) and substantial computing devices or services (e.g., cloud computing or online banking) has led to a highly digital interconnected world. It is estimated that there are more than 13.14 billion IoT devices worldwide and forecasts predict that there will be more than 29 billion devices by 2030 [1] and 5.3 billion users worldwide [2]. People use these devices for a diverse set of applications including gaming, video conference and remote working to name a few.

Consequently, millions of data bytes are generated, processed, exchanged, shared, and utilized every minute to yield specific outcomes in diverse applications, as outlined in this study [2].

Securing data, machines (devices), and user privacy in cyberspace has emerged as a paramount concern for individuals, business organizations, and national governments since this interconnectivity has also raised the number of cyber-attack[3] events each year [3]. As per cybersecurity experts, the year 2017 alone witnessed cyber-attacks potentially causing damage amounting to US$5 billion, and projections indicate a continued growth in such incidents in the future [4].

In recent years, Machine Learning (ML), Deep Learning (DL), and Data Mining (DM), have been employed to enhance cybersecurity measures. Examples include malware analysis, specially for

[1]Number of IoT connected devices worldwide 2019-2023, with forecasts to 2030 from Statista.

[2]Number of internet and social media users worldwide as of October 2023 from Statista.

[3]Based on the National Institute of Standards and Technology (NIST) a cyber attack is any kind of malicious activity that attempts to collect, disrupt, deny, degrade or destroy information system resources or the information itself.

zero-day malware detection, threat analysis, anomaly-based intrusion detection of prevalent attacks on critical infrastructures, and many others [5]. However, despite the notable progress in the use of those computational intelligence techniques leading to enhanced performances, robustness against cyber-attacks remains a significant challenge.

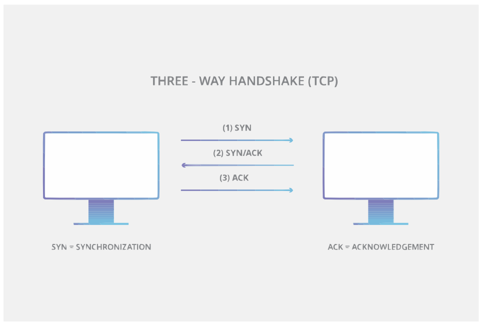
Therefore, this paper attempts to propose a novel approach for detecting SYN flood attacks in real-time, leveraging machine-learning techniques for classification task and using the extraordinary CICIoT 2023 dataset [1].

The rest of the paper is structured as follows. Section 2 gives an in-depth description of the problem we are facing and Section 3 shows what the CICIoT 2023 dataset contains and why it is so valuable for this project. Section 4 provides a comprehensive review of the machine learning approach we have used and the results are shown in section 5. Future research directions and conclusions are described in Section 6.

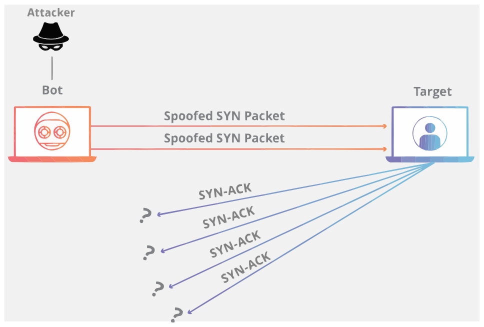
2 The Problem

Among all the different cyberattacks, we have put our focus on SYN Flood attacks. This kind of intrusion stands out as a particularly disruptive one, exploding vulnerabilities in the TCP handshake process. The conventional methods employed for SYN Flood detection face significant challenges in real-time network environments, paving the way for a critical examination of the problem at hand.

SYN Flood attacks have evolved over time, becoming increasingly sophisticated and evasive. The traditional three-way handshake (TWH) vulnerability, where an attacker inundates a target server with a barrage of SYN requests without completing the handshake, se Figure 1, poses a great challenge to conventional security measures.



(a) TCP SYN Handshake



(b) TCP SYN Flood Attack

Figure 1: TCP TWH vs SYN Flood Attack[6]

Attackers continuously refine their techniques, exploiting loopholes in network protocols and leveraging the scalability of modern infrastructures. The sheer volume and distributed nature of these attacks make them a potent weapon for cybercriminals seeking to disrupt services, compromise data integrity, and undermine the overall stability of networked systems.

The criticality of real-time SYN Flood detection cannot be overstated. As cyber threats evolve, the response time to emerging threats becomes a pivotal factor in safeguarding network integrity. Organizations must transition from reactive to proactive security measures, necessitating the development and implementation of robust real-time detection mechanisms.

The main goal of this paper is to present a study on how machine learning techniques can be applied to make a real time system capable of detecting networks packets corresponding to SYN FLood attacks. The subsequent sections of this paper delve into the intricacies of our proposed machine learning-based algorithm, which aims to overcome these challenges and provide an effective, real-time solution to SYN Flood attacks in contemporary network environments.

3 The Dataset

This section provides an introduction to the CICIot2023 dataset, offering an in-depth description of how the dataset was curated by the CIC IoT Lab [1]. The IoT topology employed in generating the CICIoT2023 consists of 105 IoT devices. Among these, 67 IoT devices actively participated in the attacks, while an additional 38 Zigbee and Z-Wave devices were interconnected with five hubs. This topology is designed to emulate a real-world deployment of IoT products and services within a smart home environment. The list of devices includes smart home, cameras, sensors, and micro-controllers, all interconnected and configured to facilitate the execution of various attacks and capture the corresponding attack traffic.

Each packet transmitted through the network is archived in a PCAP file[4]. Subsequently, employing Python functions, 47 features were identified from each network packet. Table 1 provides a comprehensive description of all the extracted features.

The benign data in the dataset encapsulates the legitimate use of the IoT network. Then, they uses tools such as udp-flood[5], hping3[6] or ettercap[7]. Figures 2 and 3 illustrate the distribution of the dataset among the 33 different attacks, along with benign traffic. Notably, the dataset exhibits unbalanced classes, a challenge addressed in Section 5.

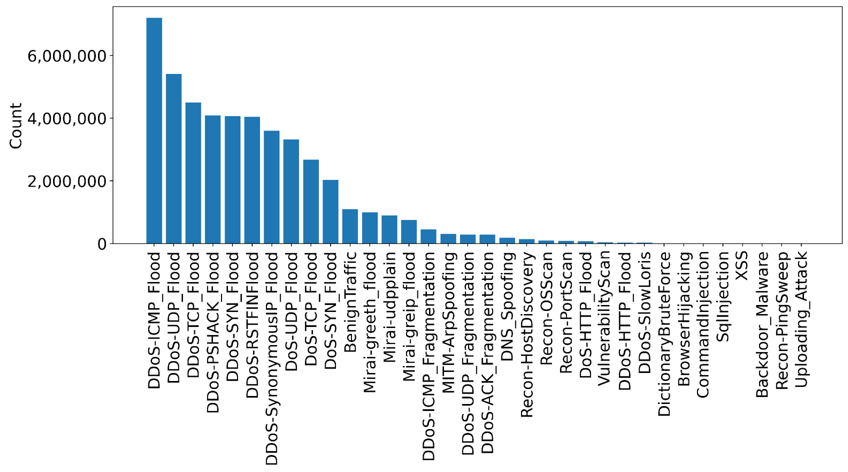


Figure 2: Number of rows for each scenario[1].

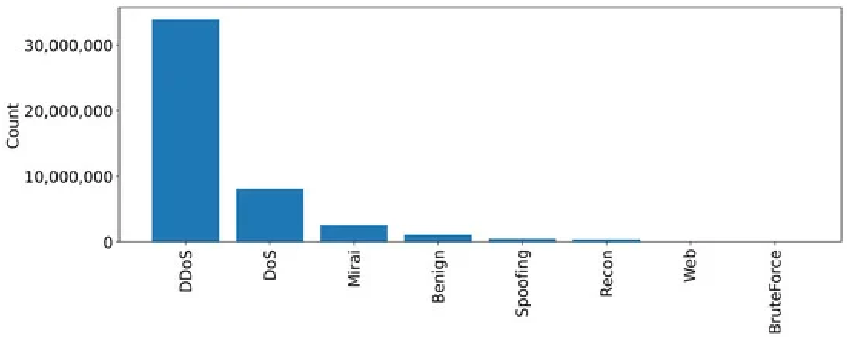


Figure 3: Number of rows for each category[1].

[4]PCAP file description

[5]udp-flood documentation.

[6]hping3 for DoS/DDoS attacks.

[7]ettercap for spoofinf attacks.

#	Feature	Description
1	ts	Timestamp
2	flow duration	Duration of the packet's flow
3	Header Length	Header Length
4	Protocol Type	IP, UDP, TCP, IGMP, ICMP, Unknown (Integers)
5	Duration	Time-to-Live (ttl)
6	Rate	Rate of packet transmission in a flow
7	Srate	Rate of outbound packets transmission in a flow
8	Drate	Rate of inbound packets transmission in a flow
9	fin flag number	Fin flag value
10	syn flag number	Syn flag value
11	rst flag number	Rst flag value
12	psh flag number	Psh flag value
13	ack flag number	Ack flag value
14	ece flag number	Ece flag value
15	cwr flag number	Cwr flag value
16	ack count	Number of packets with ack flag set in the same flow
17	syn count	Number of packets with syn flag set in the same flow
18	fin count	Number of packets with fin flag set in the same flow
19	urg count	Number of packets with urg flag set in the same flow
20	rst count	Number of packets with rst flag set in the same flow
21	HTTP	Indicates if the application layer protocol is HTTP
22	HTTPS	Indicates if the application layer protocol is HTTPS
23	DNS	Indicates if the application layer protocol is DNS
24	Telnet	Indicates if the application layer protocol is Telnet
25	SMTP	Indicates if the application layer protocol is SMTP
26	SSH	Indicates if the application layer protocol is SSH
27	IRC	Indicates if the application layer protocol is IRC
28	TCP	Indicates if the transport layer protocol is TCP
29	UDP	Indicates if the transport layer protocol is UDP
30	DHCP	Indicates if the application layer protocol is DHCP
31	ARP	Indicates if the link layer protocol is ARP
32	ICMP	Indicates if the network layer protocol is ICMP
33	IPv	Indicates if the network layer protocol is IP
34	LLC	Indicates if the link layer protocol is LLC
35	Tot sum	Summation of packets lengths in flow
36	Min	Minimum packet length in the flow
37	Max	Maximum packet length in the flow
38	AVG	Average packet length in the flow
39	Std	Standard deviation of packet length in the flow
40	Tot size	Packet's length
41	IAT	The time difference with the previous packet
42	Number	The number of packets in the flow
43	Magnitude	Average of the lengths of in/out packets
44	Radius	Variance of the lengths of in/out packets
45	Covariance	Covariance of the lengths of incoming and outgoing packets
46	Variance	Variance of the lengths of in/out packets
47	Weight	Number of incoming packets × Number of outgoing packets

Table 1: Features extracted from PCAP files.

4 The Models

The aim is to detect whether the network packets are malicious (SYN Flood) or benign (other packets), i.e. a binary classification.

As for the methods chosen, we first used logistic regression, the same method used by the authors of the original dataset paper[1]. Additionally, a Random Forest classifier and an eXtreme Gradient Boosting classifier were defined. Finally, and being a combination of the three previous ones, a Voting classifier is defined. Each of these models has its own advantages, parameters and mathematical foundations, the objective is to find the best possible model and its parameters combination.

4.1 Logistic Regression

Logistic Regression is a linear model used for binary classification. Given input features X and parameters W, the logistic regression model calculates sigmoid function[7], a S-shaped curve that maps any real-valued number to a value between 0 and 1. It is defined as:

$$Sigmoid(X, W) = P(Y = 1|XW) = \frac{1}{1 + e^{-(X \cdot W)}}$$

where e is the base of the natural logarithm. The right-hand side of the equation gives us the probability that $Y = 1$ given X. This is our hypothesis function and we want to choose W such that it best predicts our data.

To find the best parameters W, we use a method called *maximum likelihood estimation (MLE)*. The goal of MLE is to find the parameters that maximize the likelihood of making the observations given the parameters. We can define our likelihood function $L(W)$[7] as:

$$L(W) = \prod_{i=1}^{n} P(Y_i = 1|X_i)^{y_i}(1 - P(Y_i = 1|X_i))^{(1-y_i)}$$

After taking logs:

$$L(W) = \sum_{i=1}^{n}[y_i log(P(Y_i = 1|X_i)) + (1 - y_i) log(1 - P(Y_i = 1|X_i))]$$

Where:

- n is the number of observations in our data.
- y_i is the actual output for the i-th observation.
- $P(Y_i = 1|X_i)$ is the predicted probability for the i-th observation.

The maximun value is reached using gradient descent algorithm to iterate using MLE.

4.2 Random Forest Classifier

Random Forest Classifier is a non-linear model used for binary or multi-class classification. It is based on the idea of ensemble learning, which combines multiple weak learners (decision trees) to create a strong learner (random forest).

A decision tree splits data based on features to make predictions. In a random forest, each tree is trained on a subset of the data and features. The final prediction is obtained by averaging or voting. The mathematical foundation involves aggregating individual tree predictions, leading to a robust and versatile model. To make a prediction for a new input, each tree in the random forest votes for a class label, and the final prediction is the majority vote among all the trees. This is called bagging or bootstrap aggregating [8], see Figure 4.

The reason for testing this kind of algorithm is because it usually fits classification problems, where the output variable is usually a single answer, very accurately. It also handles missing values whilst maintaining this accuracy high, even when large amounts of data are missing thanks to bagging and replacement sampling[9].

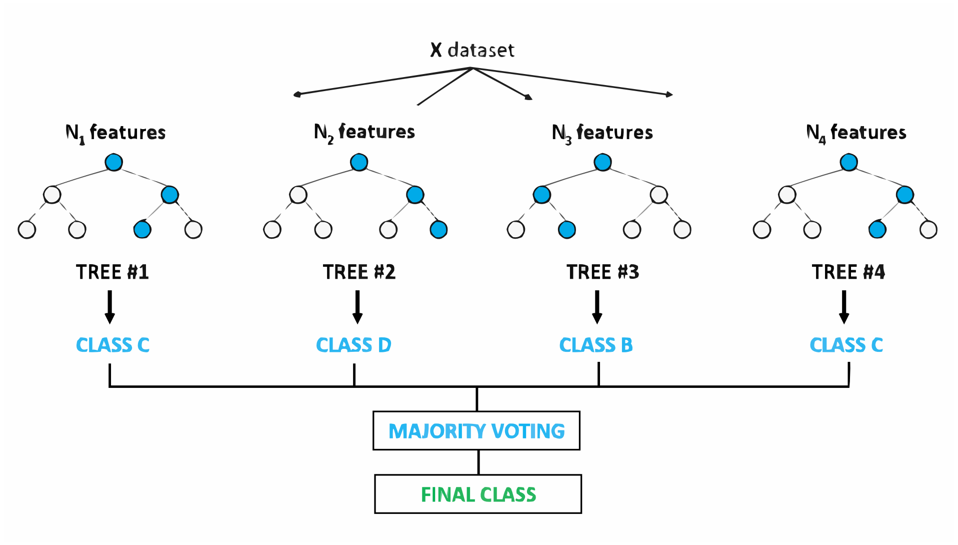


Figure 4: Random Forest Diagram[10].

4.3 eXtreme Gradient Boosting Classifier

Developed as a dedicated machine learning library, XGBoost has emerged as a go-to solution for tasks involving numbers, patterns, and predictions. Its prowess in handling regression, classification, and ranking challenges has solidified its position as a top-tier tool in the machine learning landscape. One of the key features that sets XGBoost apart is its parallel tree boosting capability, significantly enhancing both speed and effectiveness in various applications[11].

The heart of XGBoost's stellar performance lies in its innovative use of the regularization factor within the gradient descent formula—a method familiar to users of logistic regression. The objective function, denoted as $Obj^{(t)}$, combines the loss function (L) and the sum of regularization terms of all leaf nodes (Ω) in a tree. Mathematically[12], it is expressed as:

$$Obj^{(t)} = L(y, \hat{y}^{(t-1)}) + \sum_{i=1}^{t} \Omega(f_i)$$

With the objective function in place, the next steps involve calculating the gradient (g_i) and hessian (h_i) of the real values with respect to the predicted values for each leaf of the tree. This process leads to the assignment of a score to each leaf:

$$g_i = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}, \quad h_i = \frac{\partial^2 L(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}$$

$$score = -\frac{g_j}{h_j + \lambda}, \text{ with } \lambda \text{ being the regularization term}$$

After assigning scores to each leaf, the algorithm iteratively builds decision trees, favoring splits in nodes with higher scores. The process involves evaluating candidate splits, choosing the one that maximizes the improvement in the objective function, and updating leaf values accordingly. The regularization term guides the construction of simpler trees, preventing overfitting[12]. This iterative tree-building process continues for a specified number of iterations, contributing to the ensemble model's accuracy. The cumulative effect of multiple trees enhances predictive capabilities in the kind of system we are aiming to build.

4.4 Voting Classifier

A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output. It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting.

In our case, the voting iterates through the three methods explained in the previous subsections and decides which one is the best as shown in Figure 5.

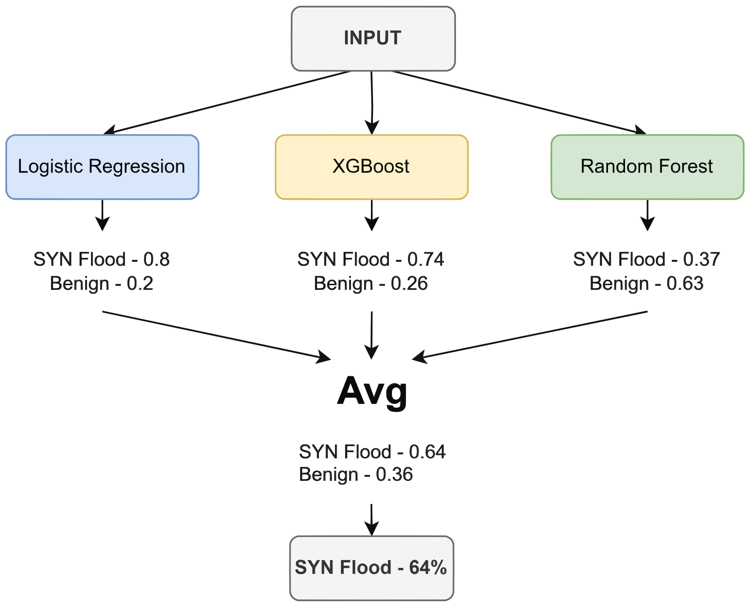


Figure 5: Example of a Voting Classifier iteration.

5 Results

First, it is crucial to highlight that the models implemented in *scikit-learn* feature a parameter known as *class_weight*, which proves instrumental in addressing the challenges posed by our unbalanced dataset. To tackle this, we computed the weights corresponding to each class within the dataset and subsequently supplied a dictionary encompassing each class along with its respective weight. Furthermore, in the context of our XGBoost implementation, we leveraged the *scale_pos_weight* parameter to achieve a balance specifically tailored for the binary classification task involving Benign Traffic and DoS SYN flood traffic.

Regarding the experiments, we conducted training sessions for all the models targeting the classification of the 34 classes within our dataset. This involved employing GridSearchCV to systematically identify the optimal estimator for the respective families under comparison. Subsequently, upon determining the best estimator for each family model (refer to Table 2 for details on the optimized hyperparameters), we proceeded to execute a series of experiments [8]. We test against our test set and the **Voting Classifier** outperforms all the models with an f1-score of **0.76**. XGB ends in the second position with an f1-score equals **0.72**. Table 3 presents all the metrics for the best estimators.

Concurrently, we undertook training sessions for all models, focusing on a binary classification task distinguishing between DoS SYN attacks and Benign traffic. Similar to the previous approach, we initiated a GridSearchCV search to pinpoint the optimal estimator for each family model (refer to Table 4 for comprehensive results). Subsequently, we conducted tests on our designated test set, revealing that the **XGB Classifier** emerged as the victor, albeit with a slight difference compared to the Voting Classifier. Specifically, XGB achieved an impressive f1-score of **0.99999**, while the

[8]This experiment can be found in main.ipynb file.

Model	Best Parameters
Logistic	{'C': 0.5, 'class_weight': 'balanced', 'penalty': 'l2'}
XGB	{'colsample_bytree': 0.5, 'gamma': 0.25, 'learning_rate': 0.1, 'max_depth': 5, 'subsample': 0.8}
Random Forest	{'max_depth': 7, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'n_estimators': 300}
Voting Classifier	{'voting': 'hard', 'weights': [1, 2, 1]}

Table 2: Best hyperparameters for classification task involving 34 classes.

Model	Accuracy Score	Recall Score	Precision Score	F1 Score
log_reg	0.78	0.48	0.54	0.47
xgb	0.99	0.77	0.7	0.72
random_forest	0.94	0.61	0.72	0.60
voting_classifier	**0.99**	**0.87**	**0.73**	**0.76**

Table 3: Model Performance with 34 classes classification task.

Voting Classifier closely followed with a score of **0.99998**. For a detailed breakdown of all metrics associated with the best estimators, please consult Table 5.

Model	Best Parameters
Logistic	{'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l2'}
XGB	{'colsample_bytree': 0.5, 'gamma': 0, 'learning_rate': 0.1, 'max_depth': 3, 'subsample': 0.8}
Random Forest	{'max_depth': 7, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'n_estimators': 200}
Voting Classifier	{'voting': 'hard', 'weights': [1, 2, 1]}

Table 4: Best hyperparameters for classification task involving 2 classes.

Model	Accuracy Score	Recall Score	Precision Score	F1 Score
log_reg	0.999	0.997	0.999	0.998
xgb	0.999	0.999	0.999	0.99999
random_forest	0.999	0.998	0.999	0.998
voting_classifier	0.999	0.999	0.999	0.99998

Table 5: Model Performance with binary classification task

In the context of our RTMCA, we executed numerous real-time tests wherein a single computer assumed the responsibility of initiating a SYN flood attack. We use a Windows laptop as a Victim and an Ubuntu laptop as the attacker. The command utilized for this purpose is as follows:

```
$ sudo hping3 -S --flood <IP_VICTIM>
```

Subsequently, employing the Scapy Python library, the VICTIM system analyzed each packet within the traffic. This involved the implementation of a callback function, wherein we systematically extracted all relevant features from the PCAP file, forwarding them to our optimal estimator. By comparing the source IP, we discerned whether the prediction manifested as a true/false positive or negative. For a comprehensive overview of our test results, please refer to Table 6.

Metric	**Demo**	Test 1	Test 2	Test 3
Packets read	1000	1000	1000	1000
Packets from Attacker (Ground Truth)	698	886	807	650
Recall	**1.0**	1.0	1.0	1.0
Precision	**0.96**	0.93	0.84	0.96
Accuracy	**0.96**	0.93	0.84	0.96

Table 6: Real-time results.

6 Discussion

In this paper, we have presented a novel approach for detecting SYN flood attacks in real-time, leveraging machine learning techniques for classification tasks and using the CICIoT2023 dataset. We have compared four different models: Logistic Regression, Random Forest, XGBoost, and Voting Classifier, and evaluated their performance on both multi-class and binary classification problems. Our results show that XGBoost Classifier achieves the highest accuracy and f1-score, demonstrating their effectiveness and robustness in handling unbalanced and large-scale data. Moreover, we have implemented a real-time machine learning model for analyzing network traffic and identifying SYN flood packets, achieving an accuracy ranging from 85% to 96% depending on the analyzed traffic network at a given moment.

As future work, it would be of our concern to extend our approach to other types of cyberattacks, such as spoofing, brute force, or web attacks, and explore the possibility of using deep learning models, such as convolutional neural networks or recurrent neural networks, to capture more complex patterns and features from the network traffic. We also think it would be of general interest to test an improved real-time model on different network environments and devices, such as IoT hubs, routers, or firewalls, and evaluate its scalability and efficiency. Furthermore, the development a user-friendly and interactive interface for our model that allows users to monitor and visualize the network traffic and the detected attacks in real-time could also be helpful in the search of the other discussed ideas.

Finally, we hope to contribute to the open-source AI community by sharing our code, and encourage further research and collaboration on this important and challenging topic.

Acknowledgments and Disclosure of Funding

We would like to express our sincere gratitude to the open-source AI community for their invaluable contributions. The wealth of content shared by this community has greatly facilitated our research on the topic.

It is crucial to underscore that both authors have made **equal contributions** to the development of this research line.

Besides, we would like to point out the usage of Generative AI for proofreading and ensuring grammatical accuracy during the composition of this report. While every piece of information presented is drawn from our knowledge or references, the utilization of Generative AI adds an additional layer of assurance for grammatical correctness and coherence.

All the code is available on GitHub using this link[9].. In the models' folder of the Github repo you can access all the models implemented in one Jupyter Notebook with all the cell outputs.

[9]CS-584-MachineLearning-FinalProject

References

[1] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," 2023.

[2] D. Dasgupta, Z. Akhtar, and S. Sen, "Machine learning in cybersecurity: a comprehensive survey," *The Journal of Defense Modeling and Simulation*, vol. 19, no. 1, pp. 57–106, 2022.

[3] M. H. U. Sharif and M. A. Mohammed, "A literature review of financial losses statistics for cyber security and future trend," *World Journal of Advanced Research and Reviews*, vol. 15, no. 1, pp. 138–156, 2022.

[4] F. Alharbi, M. Alsulami, A. Al-Solami, Y. Al-Otaibi, M. Al-Osimi, F. Al-Qanor, and K. Al-Otaibi, "The impact of cybersecurity practices on cyberattack damage: The perspective of small enterprises in saudi arabia," *Sensors*, vol. 21, no. 20, p. 6901, 2021.

[5] A. Handa, A. Sharma, and S. K. Shukla, "Machine learning in cybersecurity: A review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 4, p. e1306, 2019.

[6] "Syn flood ddos attack." `https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/`. Accessed: 2023-12-02.

[7] "Logistic regression." `https://online.stat.psu.edu/stat462/node/207/`. Accessed: 2023-12-02.

[8] R. Rastogi, "Random forest classification and it's mathematical implementation." `https://medium.com/analytics-vidhya/random-forest-classification-and-its-mathematical-implementation-1895a7bb743e`, 2020. Accessed: 2023-12-03.

[9] NVIDIA, "Random forest." `https://www.nvidia.com/en-us/glossary/data-science/random-forest/`. Accessed: 2023-12-03.

[10] "Task 2 - random forest." `https://adibnihal10.blogspot.com/2020/04/random-forest-random-forest-algorithm.html`, 2020. Accessed: 2023-12-03.

[11] NVIDIA, "Xgboost." `https://www.nvidia.com/en-us/glossary/data-science/xgboost/`. Accessed: 2023-12-03.

[12] C. G. Tianqi Chen, "Xgboost: A scalable tree boosting system," 2016.