# Detecting Compromised IoT Devices Through XGBoost

Mauro A. A. da Cruz, Lucas R. Abbade, Pascal Lorenz, *Senior Member, IEEE*, Samuel B. Mafra, and Joel J. P. C. Rodrigues, *Fellow, IEEE*

***Abstract*— The evolution and rapid adoption of the Internet of Things (IoT) led to a rise in the number of attacks that target IoT environments. IoT environments are vulnerable to several attacks because many devices lack memory, processing power, and battery. Most of these vulnerabilities are relatively easy to mitigate when best practices are followed. However, even when best practices are followed, an attack to obtain a device credential and use it to generate false data is difficult to detect. Such an attack is called a replication attack and its impact can be catastrophic in crucial IoT scenarios such as smart transportation. In this sense, this paper proposes a solution to detect these attacks by analyzing abnormal network traffic through machine learning.**

***Index Terms*— Internet of Things, IoT, XGBoost, machine learning, security, replication attack, IoT-23.**

## I. Introduction

THE Internet of Things (IoT) is a diverse ecosystem composed by a plethora of Internet-connected "things" from the real or digital world that can be remotely monitored or managed [1]. IoT environments are composed by three main layers *i*) devices and infrastructure, which is self-explanatory, *ii*) an IoT platform that is also called to as IoT middleware, that hides complexity from end-users as well as applications [2], and *iii*) users and applications where end-users and applications, such as decision support tools are located [3]. Currently, many IoT devices are vulnerable because there is always a tradeoff between security and usability. Unfortunately, IoT manufacturers tend to maximize usability, fearing that average users will lose interest if a device cannot be used straight out of the box. This lack of optimal device security generally reflects on IoT networks and platforms, endangering the whole environment and jeopardizing the IoT concept's widespread adoption [4].

Security is a crucial element of any system and devices connected to the Internet demand additional precautions because threats can arrive anytime from any part of the globe. Most of the vulnerabilities are relatively easy to prevent, especially when the best practices are followed. However, even when most device vulnerabilities are mitigated, the replication attack is hard to detect and prevent. In the replication attack, intruders obtain device credentials (legitimately or not), potentially disrupting an IoT network or leaking data. In crucial IoT scenarios, like intelligent transportation systems, these attack can even cause a dangerous impact and major disruptions in daily life or even casualties. Then, detecting the occurrence of such an attack could vastly improve the security in IoT environments. Since IoT middleware is responsible for a significant part of the intelligence in IoT environments and it is located on a powerful server [3], they could use additional resources to detect such an attack. In this sense, the paper aims to detect such attacks through a machine learning technique known as the XGBoost.

Machine learning (ML) is an excellent tool when a problem requires discovering hidden patterns from a large dataset. A common issue in ML happens when a model becomes so familiar with the training data that it can no longer adapt to other data, then it is said the model is overfitted. When a model cannot identify the patterns, even in the training set, which translates into even worse generalization, then the model underfitted. The paper goals to detect the replication attack where an attacker obtains device credentials and uses them to disturb the IoT environment. In this sense, XGBoost was the chosen ML technique because it shows excellent results in various areas and is seen as an evolution of the also popular random forests and decision trees, which also produce accurate results with labeled data. Then, the main contributions of the paper are the following:

Manuscript received 19 October 2021; revised 6 March 2022; accepted 22 June 2022. Date of publication 15 July 2022; date of current version 29 November 2023. This work was supported in part by the Fundo de Apoio ao Desenvolvimento das Comunicações (FADCOM), Republic of Angola, presidential decree no 264/10, in November 2010; in part by the Fundação para a Ciência e a Tecnologia/Ministério da Ciência, Tecnologia e Ensino Superior (FCT/MCTES) through the National Funds and European Union (EU) Funds under Project UIDB/50008/2020; in part by Rede Nacional de Ensino e Pesquisa (RNP) through Ministério da Ciência, Tecnologia e Inovações (MCTIC) under the Centro de Referência em Radiocomunicações—CRR Project of the Instituto Nacional de Telecomunicações (Inatel), Brazil, under Grant 01250.075413/2018-04; and in part by the Brazilian National Council for Research and Development (CNPq) under Grant 313036/2020-9 and Grant 431726/2018-3. The Associate Editor for this article was Z. Lv. *(Corresponding author: Joel J. P. C. Rodrigues.)*

Mauro A. A. da Cruz is with the National Institute of Telecommunications (INATEL), Santa Rita do Sapucaí 37540-000, Brazil, and also with the Network and Telecommunication Research Group, University of Haute Alsace, 68008 Colmar, France (e-mail: maurocruzter@gmail.com).

Lucas R. Abbade and Samuel B. Mafra are with the National Institute of Telecommunications (INATEL), Santa Rita do Sapucaí 37540-000, Brazil (e-mail: lucasabbade@hotmail.com; samuelbmafra@inatel.br).

Pascal Lorenz is with the Network and Telecommunication Research Group, University of Haute Alsace, 68008 Colmar, France (e-mail: lorenz@ieee.org).

Joel J. P. C. Rodrigues is with the College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266555, China, and also with the Instituto de Telecomunicações, 6201-001 Covilhã, Portugal (e-mail: joeljr@ieee.org).

Digital Object Identifier 10.1109/TITS.2022.3187252

- A review of top security threats for IoT environments and their counters;
- Detecting replication attacks in IoT networks (when an attacker impersonates a legitimate IoT device) through XGBoost;
- The code created and used in this work is open-source and published on GitHub, to aid developers and researchers seeking to make similar work in the future;
- Demonstration that XGBoost presents better accuracy regarding the state-of-the-art (for the used public datasets).

The remainder of this paper is organized as follows. Section II reviews the most common security threats for IoT devices and platforms. The section also provides an overview regarding similar studies to support the new contribution and presents general directives to mitigate the majority of the presented threats. Section III provides an overview of decision trees, random forest, and XGBoost based techniques. Section IV introduces the experimental scenario, details the necessary steps to deploy XGBoost based solutions successfully, and presents and analyses the obtained results. Section V concludes the paper by presenting the final remarks and suggestions for future works.

## II. Background and Related Work

IoT security is a hot topic because several devices lack the proper mechanisms to fend off attacks and it is common for them to be targeted by attackers. They can also target IoT platforms and gain access to privileged information, which they can sell to third parties or demand ransom to give data back. In information systems, security involves protecting data from corruption or leakage, which means that only authorized parties should be able to send and access data. In this sense, below, the most common security threats for IoT platforms and IoT devices are overviewed.

### A. Security Threats for IoT Platforms and Networks

Networks are critical in IoT systems because they provide connections for devices. Combined with IoT platforms, networks allow devices to interact with applications and deliver storage and advanced computation capabilities. It is a consensus that IoT networks should only disclose data to authorized entities [5] because this type of data leakage could have unforeseen consequences. In this sense, the most common security threats for IoT networks and platforms identified in [6] are the following: *a*) sniffing attacks, *b*) Denial of Service (DoS) attacks, *c*) spoofing attacks, *d*) routing attack, *e*) IoT cloud service manipulation, and *f*) privilege escalation. Next, each security threat is briefly described as follows.

*a*) **Sniffing attack**: The attacker monitors and captures packets that are sent or received by a given network or host to obtain credentials or other sensitive data. The tool that is used to capture network data is called a packet Sniffer. Packet sniffers are not always malicious tools because they can enable network administrators to diagnose network issues. Two popular packet Sniffers are *Observer* and *Wireshark* [7].

*b*) **Denial of Service (DoS) and Distributed DoS attacks**: An attacker floods a network or a host with huge amounts of data, causing entire systems to become unavailable. From the perspective of a server, a DoS is relatively easy to counteract since it involves a single machine and blocking the IP address of the attacker can nullify the threat. However, mitigating a Distributed DoS (DDoS) in which multiple devices perform DoS attacks on the same target [8] is more complex.

c) **Spoofing attack**: The attacker impersonates a network entity to receive compromising data such as credentials from one of the legitimate entities. Attackers can spoof an IP address in scenarios where the IP is used for access control and for gaining various network privileges. In RFID solutions, attackers can clone data from a legitimate RFID tag and impersonate it. The Man-in-the-middle (MitM) attack, where the attacker impersonates both the sender and receiver, is the more advanced Spoofing attack.

*d*) **Routing attack**: The attacker changes how packet routing is performed on an IoT network, generating routing loops, fraudulent error messages, or even obtaining sensitive information. A simple routing attack can consist of a node advertising itself as the shortest path to a common network destination [9], then *i*) dropping all packets, or *ii*) forward data to a malicious server. In *i*), the attacker effectively stops the network. In *ii*), the attacker can obtain sensitive data such as device credentials.

*e*) **IoT cloud service manipulation**: The attacker gains control or access to one or multiple cloud services. When this attack is successful, the intruder can obtain data directly from the database or even disrupt an IoT environment by generating false alerts. An example of such is an exploit in Docker containers that could only be executed with administrator privileges for a reasonable amount of time, which meant that a malicious process inside the container could inherit its privileges. However, recent Docker versions allow containers to be deployed without administrative privileges to reduce the impact of such an attack.

*f*) **Privilege escalation**: The attacker gains access to a low-level account (legitimately or not) to access other protected services. The premise of the attack can be understood by the following example: a company that uses keycards as an access control mechanism only allows visitors to access the common rooms. However, a visitor notices that any subsequent room is accessible with the visitor keycard once inside a restricted area. Then, he/she proceeds to exploit this vulnerability and access other restricted areas.

The threats to IoT platforms are generally mitigated through *i*) authentication and authorization, *ii*) traffic filtering and firewalls, and *iii*) encryption protocols.

*i*) **Authentication and authorization**: Most Web applications require users to provide their credentials, generally through a username and password combination. This action is called authentication and allows the server to certify that the user is registered. After the authentication process, the server verifies which actions can be performed and it is called authorization. Authentication can also be simplified as an answer to "who are you?" and authorization as "what can you do?". A well-configured authorization can mitigate

privilege escalation as well as IoT cloud service manipulation attacks.

*ii*) **Traffic filtering and firewalls**: A firewall is a network entity (software) that analyzes the network traffic and decides to block or allow traffic based on previously configured security rules [10]. A well-configured firewall can mitigate various attacks, especially from known threat sources. However, firewall rules need continuous increments, especially in an IoT context.

*iii*) **Encryption protocols**: These protocols ensure data confidentiality, so users that "eavesdrop" unauthorized communications cannot decode data. Most encryption protocols make use of symmetric and asymmetric key management. One of the most popular protocols on the Internet is HTTPS (HyperText Transfer Protocol Secure) because of the added security. HTTPS's security relies on TLS (Transport Layer Security), a cryptography protocol that uses a private and public key pair to encrypt communications. In IoT communications, cryptography protocols could be used to encrypt MQTT (Message Queuing Telemetry Transport) and CoAP (Constrained Application Protocol) communications. Without encryption, device data is transmitted in plain text, which means that any attacker that intercepts it can read the message.

### *B. Security Threats for IoT Devices*

A device is likely the most fragile component of IoT domains. They are mainly limited regarding battery life and computational capabilities because security contrasts with performance. A compromised node can compromise the entire network by generating erroneous measurements or even carrying out malicious network attacks. In this sense, the most common attacks related to the device layer identified in [6] are the following: *a*) Hardware Trojan attacks, *b*) Replication attacks, *c*) tampering attacks and malicious code injection, and *d*) battery-draining attacks.

*a*) **Hardware Trojan attack**: This is a process in which a malicious action is purposely inserted in the circuitry during the assembly or construction phase [11]. This action stays hidden and is unnoticed until it is triggered by the manufacturer or other third party aware of the action. In other words, it is a backdoor that is added during the manufacturing process. When a device is compromised during the manufacturing process, the issue might not be solved if the vulnerability is placed directly on the hardware itself. There is no real way to prevent or mitigate such an attack.

*b*) **Replication attack**: This is a process in which attackers insert a device with the same credentials as a target device, generating false data or obtaining other security grants like cryptographic shared keys. This type of attack requires that attackers possess the credentials (generally username and password) of a legitimate device and can be hard to detect. This attack can also threaten IoT platforms because once the device credentials are obtained, the attacker can gain a degree of access to IoT platforms. This attack cannot be prevented, but it can be mitigated with a proper detection mechanism.

*c*) **Tampering attack and malicious code injection**: This attack generally occurs when an attacker obtains physical access to a device, inserts malicious code, or retrieves the device logs [12]. This type of attack can also be remotely performed by exploiting the default or commonly used username/password combinations and uploading a changed version of Firmware. *Mirai* is an example of malware frequently used for this type of attack that can hijack IoT devices remotely and create botnets to perform distributed denial of service (DDoS) attacks [13].

*d*) **Battery draining attacks**: It consists in reducing a device's battery life by continually sending data to it and reducing its "sleep time" or forcing constant reply messages [14].

Most of the threats to IoT devices can be mitigated by limiting physical access to the IoT devices. Other recommendations that can mitigate IoT device threats are the following: *i*) individual device credentials, *ii*) mechanisms to modify device credentials, and *iii*) source code protection.

*i*) Individual device credentials: IoT devices should have an individual username and password combination. Otherwise, an attacker can disrupt an entire IoT network in an unimaginable manner by obtaining a single devices' credentials. *ii*) Mechanisms to modify device credentials: The ability to modify device credentials refers mainly to the password. This is important because there is always a possibility that an attacker can obtain a password. *iii*) Source code protection: A device source code determines its operation rules and various constraints. Thus, it should be protected from external access through source-code protection mechanisms that guarantee that source-code cannot be retrieved and alert users (administrators) when new source-code versions are uploaded to the device.

### *C. Related Studies*

The only threats without a clear counter are the hardware trojan attack (which cannot be countered) and the replication attack, which can only be mitigated with a proper intrusion detection mechanism. In this sense, the two main approaches to detect intrusion consist of signature detection and anomaly detection. Signature detection techniques store the unique characteristics of known attacks in databases to compare with the incoming traffic. Anomaly detection usually uses machine learning to detect hidden patterns on the incoming data and differentiate legitimate traffic from malware traffic.

Signature-based detection is excellent for identifying known threats but can be ineffective against new attacks whose signatures are not in the database. Signature-based detection requires constant maintenance because the new signatures must be continuously inserted into the database. Regarding anomaly detection techniques, since they mostly use machine learning, a model must be trained based on a dataset to detect the anomalies. A dataset used in recent studies is IoT-23 [15], because it is available to the public and contains labeled benign and malicious IoT network traffic captured from 2018 to 2019 [16]. The dataset labels several threats like Okiru, Torii, Mirai, port scanning DDos, C&C, and heartbeat. The issue with IoT-23 is that it is a skewed dataset because most of the recorded data is malicious.

In a recent study, [17], a security framework called ADEPT was created to detect suspicious activities in the network. ADEPT uses a combination of K-NN (K-Nearest Neighbor), Random forests (RF), and SVM (Support Vector Machine) algorithms. Furthermore, the study also used three datasets: UNSW-NB15 [18], NSS Mirai Dataset [19], and IoT-23. After training the model, the experiments containing data from the three datasets showed great promise, with ADEPT successfully identifying attacks with diverse characteristics.

In [20], the IoT-23 dataset was used to train two distinct models, the first model was based on Multi-Class Decision Forest and the second was based on a Multi-Class Neural Network. In both models, the percentage of true negatives was 100%. Regarding the percentage of true positives, the multi-class decision forest was 99.8% and the multi-class neural network was 99.7%. In [21], an ensemble method using a DNN (Deep Neural Network), LSTM (Short-Term Memory) and logistic regression as a meta-classifier was used for intrusion detection. The work demonstrated the efficiency of such an approach by using data from the IoT-23, LITNET-2020 [22], and NetML-2020 datasets [23].

In [24], a lightweight bot detection mechanism called BotFP was created, and three variants were evaluated: BotFP-Clus, BotFP-MLP, and BotFP-SVM. BotFP-Clus was trained with the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm. BotFP-MLP used Multi-Layer Perceptrons, and BotFP-SVM used Support Vector Machines. The study uses the CTU-13 dataset [25] combined with the IoT-23 dataset to evaluate each model. The results were promising, considering BotFP successfully detected all bot occurrences on the CTU-13 dataset and few false positives.

Although a labeled dataset will be used to carry out the experiments in this paper, another option could be based on using unsupervised ML techniques that do not need labeled outputs. One of the most significant advantages of using unsupervised techniques comes from the fact that, since they do not need a labeled dataset, it can be useful in anomaly detection [26]. Most studies use supervised techniques (including this one) because they are easier to deploy, debugging, and less complex.

The paper focuses on detecting abnormal network traffic to mitigate the replication attack by finding hidden data patterns. In this sense, the paper will use ML because it is an excellent tool to identify hidden patterns in data. An XGBoost model will be trained using data from the publicly available dataset IoT-23. XGBoost was chosen as the ML technique because it is obtaining great results on Kaggle competitions [26], which includes tasks from Kaggle and some big companies like Google, demonstrating its flexibility across various scenarios. Kaggle competitions are great benchmark for ML techniques and techniques because their competitions are very prestigious, giving good payouts and therefore attract a lot of competitors. This means that when an algorithm or technique starts dominating several categories, it deserves special attention.

## III. XGBoost

Traditional software is becoming obsolete in areas like e-commerce and social media because clients expect to view information relevant to their preferences, which is generally achieved through machine learning (ML) [28]. A machine learning approach considers machines (typically computers) making predictions and improving their accuracy based on the analyzed data [29]. Machine learning techniques are good for finding hidden patterns in data but require large amounts of training data to generate models that produce accurate results [30]. Luckily, datasets on various topics are available on the Internet and can be used for research, free of charge.

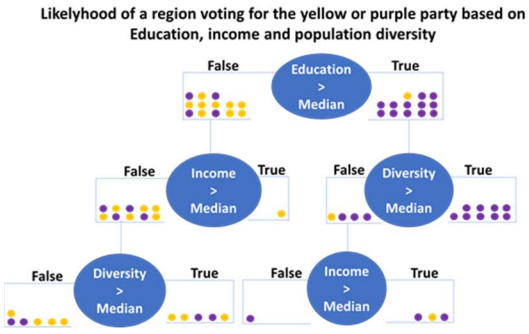


Fig. 1. Illustration of a decision tree that tries to determine the probability of a state voting for a yellow (represented as False and located on the left side) or a purple party (represented as True and located on the right side).

Recently, a ML technique, called XGBoost, has become increasingly popular because of its immense benefits. XGBoost uses gradient boosted decision trees, which means that understanding XGBoost, implies understanding decision trees. Decision trees are supervised models that best predict the result with only two possible outcomes (e.g., spam or not spam). Decision trees can also be applied to non-binary outcomes, but such an approach increases the complexity. Decision trees are visually represented as binary trees where a dataset is broken into smaller subsets, called nodes, as the tree grows, which represent true or false statements [31] based on variables from the dataset. The main issue with decision trees is that the order in which the variables are analyzed (questions are formulated) can change the outcome. Thus, the first variable to be analyzed should provide more information gain than the others in the sequence [32].

A typical application for decision trees is predicting the outcome of an election based on known variables, such as education, income, and diversity. Depending on the data in each feature variable, the tree's root should be different because the variable located in the root should contain more information. Figure 1 shows a decision tree that determines the probability of a state voting for a yellow or a purple party, and education was chosen as the tree's root. When a variable cannot be represented as a Boolean (false or true), the numerical data is split into thresholds that determine if the result is more likely to go left or right (false or true in Figure 1 example).

In mathematical terms, the tree decides which outcome is more likely by calculating the information gain, determined by the Gini index or entropy. Gini determines the information gain through expression (1), and entropy determines the information gain through expression (2). In both expressions,

$p_i$ represents the probability of class $i$ occurring, and with binary outcomes, there are only two classes. Their behavior is also similar because when they reach the maximum value, it means that the probability of each outcome is the same, and there is little information gain. When a node reaches its minimum value, the node is pure and there is no need for further nodes.

$$Gini = \sum_{i=0}^{n} p_i^2 \tag{1}$$

$$Entropy = -\sum_{i=0}^{n} p_i log_2 p_i \tag{2}$$

The Gini index and entropy generally produce similar results on the same dataset and can be used to determine the root of the decision tree. Since choosing the tree's root can impact the prediction of the outcome, it is more common to use random forests, which are multiple decision trees trained based on the available data. Since various trees are created, several permutations of the variables in the dataset are created for each tree. After complete the training, each tree votes for an outcome (true or false), and the algorithm chooses the outcome with the majority of votes. Due to their nature, random forests are beneficial in problems with two possible outcomes such as spam classification and other security-related issues.

XGBoost is seen as an evolution to Random forests and decision trees and utilizes the gradient descent algorithm to minimize prediction errors. XGBoost (XGB) is similar to Gradient Boosting at its core. It builds decision trees sequentially, approximating its predictions to the observed values by reducing the residuals at each step. The most important differences are regularization parameters added to stimulate pruning the trees and prevent overfitting and software optimizations to maximize computing efficiency and mathematical simplifications to reduce calculation complexity while retaining accuracy. The resulting model is easy to implement, quick to train and achieves very good accuracy in a wide range of problems [33].

An XGBoost classifier starts with a base prediction of 0.5. It then calculates each observation's residuals (observed value - predicted probability) and starts to build the first tree. When building the tree, XGBoost will begin by adding every observation's residual to a single leaf and calculating the similarity score of the leaf using expression (3), where $r_i$ is the residual of observation $i$, $p_i$ is the last predicted probability of observation $i$ (for the first tree, $p_i$ will always be 0.5), and $\lambda$ is a regularization parameter.

$$SimilarityScore = \frac{\left(\sum_{i=0}^{n} r_i\right)^2}{\sum_{i=0}^{n} (p_i \times (1 - p_i)) + \lambda} \tag{3}$$

The classifier will then try to split the leaf using the data features. For continuous features, XGB will test splitting based on quantiles, and calculate the gain of splitting at each quantile. The feature with the highest gain is chosen and XGB continues trying to split with the other features. Gain is calculated using expression (4), where root similarity is the original leaf similarity containing all the residuals, $leaf_1$ and $leaf_2$ are the resulting leaves of the split.

$$Gain = simLeaf_1 + simLeaf_2 - simRoot \tag{4}$$

Once the tree is built, XGB will attempt to prune it bottom-up using the $\gamma$ parameter, eliminating splits where the resulting gain is smaller than $\gamma$. Pruning will stop when the gain is larger than $\gamma$ even if there are splits further up with smaller gains. The output value for each of the tree's leaves is then calculated using Expression (5), where $r_i$ is the residual of observation $i$ for each observation in the leaf, $p_i$ is the last predicted probability of observation $i$ (for the first tree it is 0.5), and $\lambda$ is the regularization parameter. Note that this expression is very similar to expression (3), and both of them will output large values for leaves containing very similar values, while penalizing leaves where residuals are too different, which makes sense as similar residuals indicate good confidence in the tree output.

$$Outputvalue = \frac{\sum_{i=0}^{n} r_i}{\sum_{i=0}^{n} (p_i \times (1 - p_i)) + \lambda} \tag{5}$$

XGB will keep building trees this way until new trees fail to improve prediction. The final prediction is a log(odds) of the sum of each tree output value times the learning rate ($\eta$ parameter) for every tree m, as seen in expression (6). The initial prediction of 0.5 does not get added because the log(odds) of a 0.5 probability is 0. The final log(odds) value is then converted into a probability using expression (7). A deeper analysis of XGBoost mathematical formulation can be found in the original paper [34].

$$\log(odds) = \sum_{m=0}^{M} \eta \times outputvalue_m \tag{6}$$

$$Probability = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}} \tag{7}$$

## IV. Experimentation Scenario and Result Analysis

This section presents the stages taken to identify replication attacks through XGBoost and analyzes the testing data results. XGBoost was chosen to solve such an issue because they performed well in Kaggle competitions, are easy to train, produce accurate results in labeled datasets, and are consistent across various scenarios. The following steps were taken to train the model: *i*) Obtain or produce a dataset, *ii*) clean dataset, *iii*) grid search to find the best training parameters.

- **Clean dataset**: At this step, gathered data is analyzed to decide which data will be used and generally also involves data normalization (also known as a feature normalization in machine learning). Data normalization is widespread in machine learning, especially when numerical data is used and it consists on rescaling attributes to a given range (generally from 0 to 1). Without rescaling numerical data, the model might not correctly estimate the importance of the numerical values. One of the advantages of XGBoost (since it is a tree-based classifier) is that normalization is not necessary for numerical data [35]. Categorical features were one-hot

TABLE I
GRID SEARCH PARAMETERS TO FIND THE OPTIMAL TRAINING PARAMETERS CONSIDERING $\eta$,$\gamma$ ,$\lambda$, DEPTH, SCALE, AND ROUNDING

| η | γ | λ | depth | scale | rounding | accuracy | F1 | Rank |
|---|---|---|---|---|---|---|---|---|
| 0.1 | 0 | 1 | 6 | 0.3 | 0.3 | 0.883 | 0.937 | 1 |
| 0.1 | 0 | 0 | 6 | 0.3 | 0.5 | 0.882 | 0.937 | 2 |
| 0.1 | 0 | 0 | 6 | 0.3 | 0.3 | 0.882 | 0.937 | 2 |
| 0.5 | 0 | 1 | 6 | 0.3 | 0.3 | 0.882 | 0.937 | 2 |
| 0.1 | 1 | 1 | 6 | 1.0 | 0.3 | 0.880 | 0.936 | 3 |
| 0.1 | 1 | 1 | 6 | 1.0 | 0.3 | 0.880 | 0.936 | 3 |
| 0.1 | 0 | 1 | 6 | 1.0 | 0.3 | 0.880 | 0.936 | 3 |
| 0.1 | 1 | 1 | 15 | 1.0 | 0.3 | 0.880 | 0.936 | 3 |
| 0.1 | 1 | 1 | 15 | 1.0 | 0.3 | 0.880 | 0.936 | 3 |
| 0.1 | 1 | 0 | 15 | 1.0 | 0.3 | 0.880 | 0.936 | 3 |

encoded, which is a technique similar to normalization but for labeled data, and it is based on transforming every category into a boolean variable, where 1 (one) represents True, and 0 (zero) False. The following features were one-hot encoded: network protocol (tcp, icmp, udp), application-layer protocol (DNS, SSL, SSH, DHCP, HTTP, IRC), and connection state, which has 13 (thirteen) categories

- **Grid search to find the best training parameters**: When training machine learning models, a crucial aspect is choosing the training hyperparameters set before training. Grid search is a technique that searches for the best parameter values of a chosen model. For XG Boost, the hyperparameters on the grid search were $\eta$, $\gamma$, $\lambda$, depth, scale, and rounding. Depth refers to the number of questions made in each decision tree. $\eta$ is the learning rate to scale each tree's output value. $\gamma$ and $\lambda$ are regularization parameters to stimulate tree pruning and avoid overfitting. Scale gives more weight to the less common observation value to compensate for skewed data. Rounding is the threshold used when rounding the percentage output by the model to a final label. The grid search results are presented in Table I. Results below rank 3 are omitted and their difference was small. Also, the difference between the ranked 1 and ranked 3 parametrization was minimal. Overall, variations of the depth = 6 occupy most of the top positions.

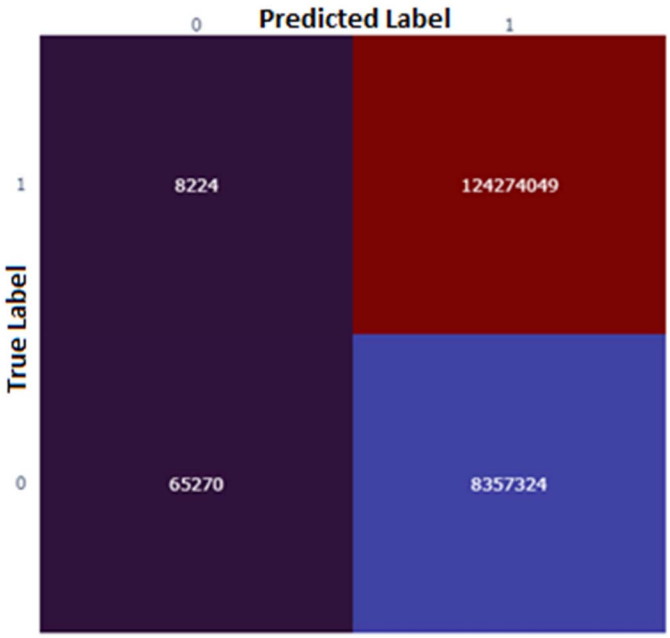


Fig. 2. Confusion matrix results of the test data – 1 means attack.

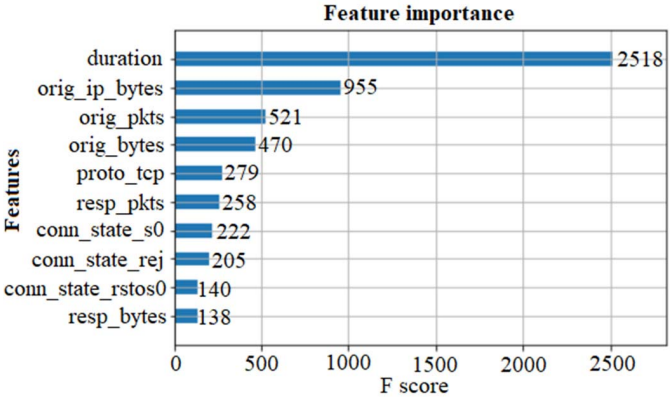


Fig. 3. Feature importance in function of the F score for the most relevant features.

### *A. Result Analysis*

The number one ranked parametrization was used to evaluate the model ($\eta = 0.1$, $\gamma = 0$, $\lambda = 1$, depth = 6, scale = 0.3, and rounding = 0.3). The trained model presents 93.6% accuracy, the precision of 93.7%, a recall score of 99.9%, and an F1 score of 96.7% in the data used in the experiments. The results are summarized in the confusion matrix presented in Figure 2 and shows the model detects more false positives than false negatives, which is preferable in security applications. With a higher number of false positives, compromised devices are more likely to be detected. If the model accused more false negatives than positives, several infected devices would remain undetected. The hidden cost of a high number of false positives is that they require further investigation from a network administrator.

When identifying if the request was from an attacker, the trained model attributed more importance to the following features: *Duration of the request* (2875 splits), *orig_ip_bytes* (898 splits), *orig_ip_pkts* (600 splits), *Orig_Bytes* (566), *Resp_bytes* (258), *resp_pkts* (243), *conn_state_rej* (205), *proto_tcp* (200), *conn_state_S0* (155), and *resp_IP_bytes* (145). The feature importance is displayed in Figure 3. Features with a higher number of splits are more relevant to the model.

The duration of the request determines how long the request took to be fulfilled, so it makes sense that the model determines that requests with a higher duration originate from an attacker in a distant region, thus giving higher importance for the duration. It is hard to determine the reasons for the model attributing importance to the other features because, to humans, little information can be extracted from them. Orig_ip_bytes is the number of bytes from the originator (external host) IP. Orig_pkts packets is the number of packets that came from the external host. Orig_Bytes refers to the payload bytes of the external host. Resp_Bytes refers to the payload bytes sent by the server. Resp_pkts is the number of

packets that came from the server, conn_state_rej means that connection was attempted by the originator and rejected by the server. Proto_tcp means that the used protocol was TCP. Conn_state_s0 means that the connection attempt was seen by the server and received no reply. Rest_ip_bytes is the number of bytes from the server (external host) IP.

## V. Conclusion and Future Works

Security is a crucial and delicate topic often neglected in IoT environments because IoT devices generally present numerous constraints such as limited memory, processing power, and battery. This lack of security attracts various attackers that easily exploit device vulnerabilities. The paper presented an overview of the most common threats to IoT platforms and devices as well as the best practices to mitigate these threats. The study focused on detecting replication attacks where attackers obtain device credentials to disrupt the IoT environment. The replication attack is hard to detect and can have severe consequences, especially in crucial IoT scenarios like smart transportation systems. In intelligent transportation systems, invalid data could cause delays or even the loss of human life and should be addressed, as presented in this paper.

The paper presented a detailed framework to create a lightweight and low overhead model capable to achieve about 93.6% accuracy on the entirety of the IoT-23 dataset. IoT-23 contains devices data from infected and healthy devices recorded from 2018 to 2019. The model presented in this paper should be more useful for developers in real scenarios seeking a powerful, simple, and robust solution. The other works exploring IoT-23 utilize only a part of the entire subset (which may be highly susceptible to overfitting) or use much larger and complicated models like DNN.

The obtained results were extremely promising because, after the training phase, the system presented an accuracy of 93.6%, a recall score of 0.99, a precision of 93.6%, and an F1 score of 96.7%. The trained model also detected more false positives than negatives, which is a good result in security solutions because otherwise, infected devices might not be detected. Also, the trained model could be integrated into an IoT middleware solution and further and block access devices that were flagged as suspicious, further increasing the security of IoT environments.

Future works may focus on evaluating the system's efficiency with other datasets to verify the flexibility of the trained model. Moreover, the efficiency of XGBoost in this particular problem could be compared with Random forests or a GAN (Generative Adversarial Network) where the model trains itself based on a dataset. GANs considers two neural networks and their training is different in the sense that, during training, one neural network is a forger that tries to deceive a fraud detector (which is the other neural network). In the context of this work, the forger would attempt to replicate requests that are similar to a legitimate request and the fraud detector would attempt to detect the fraudulent request.

## References


[1] J. H. Nord, A. Koohang, and J. Paliszkiewicz, "The Internet of Things: Review and theoretical framework," *Expert Syst. Appl.*, vol. 133, pp. 97–108, Nov. 2019, doi: 10.1016/j.eswa.2019.05.014.

[2] J. Zhang, M. Ma, P. Wang, and X.-D. Sun, "Middleware for the Internet of Things: A survey on requirements, enabling technologies, and solutions," *J. Syst. Archit.*, vol. 117, Aug. 2021, Art. no. 102098, doi: 10.1016/j.sysarc.2021.102098.

[3] M. A. A. da Cruz, J. J. P. C. Rodrigues, J. Al-Muhtadi, V. V. Korotaev, and V. H. C. de Albuquerque, "A reference model for Internet of Things middleware," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 871–883, Apr. 2018, doi: 10.1109/JIOT.2018.2796561.

[4] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on IoT security: Application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82721–82743, 2019, doi: 10.1109/ACCESS.2019.2924045.

[5] S. Zeadally, A. K. Das, and N. Sklavos, "Cryptographic technologies and protocol standards for Internet of Things," *Internet Things*, vol. 14, Jun. 2021, Art. no. 100075, doi: 10.1016/j.iot.2019.100075.

[6] I. Farris, T. Taleb, Y. Khettab, and J. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 812–837, 1st Quart., 2019, doi: 10.1109/COMST.2018.2862350.

[7] H. Kim, H. Lee, and H. Lim, "Performance of packet analysis between observer and WireShark," in *Proc. 22nd Int. Conf. Adv. Commun. Technol. (ICACT)*, Phoenix Park, South Korea, Feb. 2020, pp. 268–271, doi: 10.23919/ICACT48636.2020.9061452.

[8] M. A. Aladaileh, M. Anbar, I. H. Hasbullah, Y.-W. Chong, and Y. K. Sanjalawe, "Detection techniques of distributed denial of service attacks on software-defined networking controller—A review," *IEEE Access*, vol. 8, pp. 143985–143995, 2020, doi: 10.1109/ACCESS.2020.3013998.

[9] M. Vigenesh and R. Santhosh, "An efficient stream region sink position analysis model for routing attack detection in mobile ad hoc networks," *Comput. Electr. Eng.*, vol. 74, pp. 273–280, Mar. 2019, doi: 10.1016/j.compeleceng.2019.02.005.

[10] S. Kim, S. Yoon, J. Narantuya, and H. Lim, "Secure collecting, optimizing, and deploying of firewall rules in software-defined networks," *IEEE Access*, vol. 8, pp. 15166–15177, 2020, doi: 10.1109/ACCESS.2020.2967503.

[11] Z. Huang, Q. Wang, Y. Chen, and X. Jiang, "A survey on machine learning against hardware trojan attacks: Recent advances and challenges," *IEEE Access*, vol. 8, pp. 10796–10826, 2020, doi: 10.1109/ACCESS.2020.2965016.

[12] T. Alladi, V. Chamola, B. Sikdar, and K. R. Choo, "Consumer IoT: Security vulnerability case studies and solutions," *IEEE Consum. Electron. Mag.*, vol. 9, no. 2, pp. 17–25, Mar. 2020, doi: 10.1109/MCE.2019.2953740.

[13] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017, doi: 10.1109/MC.2017.201.

[14] I.-G. Lee, K. Go, and J. H. Lee, "Battery draining attack and defense against power saving wireless LAN devices," *Sensors*, vol. 20, no. 7, p. 2043, Apr. 2020, doi: 10.3390/s20072043.

[15] S. Garcia, A. Parmisano, and M. J. Erquiaga, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," Zenodo, 2020, doi: 10.5281/zenodo.4743746.

[16] P. M. S. Sanchez, J. M. J. Valero, A. H. Celdran, G. Bovet, M. G. Perez, and G. M. Perez, "A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1048–1077, 2nd Quart., 2021, doi: 10.1109/COMST.2021.3064259.

[17] K. L. K. Sudheera, D. M. Divakaran, R. P. Singh, and M. Gurusamy, "ADEPT: Detection and identification of correlated attack stages in IoT networks," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6591–6607, Apr. 2021, doi: 10.1109/JIOT.2021.3055937.

[18] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Canberra, ACT, Australia, Nov. 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[19] *GitHub—Kaysudheera/NSS_Mirai_Dataset: This Dataset is Captured From a Mirai Type Botnet Attack on an Emulated IoT Network in OpenStack*. Accessed: Sep. 20, 2021. [Online]. Available: https://github.com/kaysudheera/NSS_Mirai_Dataset

[20] M. Hegde, G. Kepnang, M. Al Mazroei, J. S. Chavis, and L. Watkins, "Identification of botnet activity in IoT network traffic using machine learning," in *Proc. Int. Conf. Intell. Data Sci. Technol. Appl. (IDSTA)*, Valencia, Spain, Oct. 2020, pp. 21–27, doi: 10.1109/IDSTA50958.2020.9264143.

[21] V. Dutta, M. Choraś, M. Pawlicki, and R. Kozik, "A deep learning ensemble for network anomaly and cyber-attack detection," *Sensors*, vol. 20, no. 16, p. 4583, Aug. 2020, doi: 10.3390/s20164583.
[22] R. Damasevicius *et al.*, "LITNET-2020: An annotated real-world network flow dataset for network intrusion detection," *Electronics*, vol. 9, no. 5, p. 800, May 2020, doi: 10.3390/electronics9050800.
[23] O. Barut, Y. Luo, T. Zhang, W. Li, and P. Li, "NetML: A challenge for network traffic analytics," Apr. 2020, *arXiv:2004.13006*.
[24] A. Blaise, M. Bouet, V. Conan, and S. Secci, "Botnet fingerprinting: A frequency distributions scheme for lightweight bot detection," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 3, pp. 1701–1714, Sep. 2020, doi: 10.1109/TNSM.2020.2996502.
[25] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014, doi: 10.1016/j.cose.2014.05.011.
[26] M. Usama *et al.*, "Unsupervised machine learning for networking: Techniques, applications and research challenges," *IEEE Access*, vol. 7, pp. 65579–65615, 2019, doi: 10.1109/ACCESS.2019.2916648.
[27] C. Wang, C. Deng, and S. Wang, "Imbalance-XGBoost: Leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost," *Pattern Recognit. Lett.*, vol. 136, pp. 190–197, Aug. 2020, doi: 10.1016/j.patrec.2020.05.035.
[28] L. Micol Policarpo *et al.*, "Machine learning through the lens of e-commerce initiatives: An up-to-date systematic literature review," *Comput. Sci. Rev.*, vol. 41, Aug. 2021, Art. no. 100414, doi: 10.1016/j.cosrev.2021.100414.
[29] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Trans. Softw. Eng.*, vol. 48, no. 1, pp. 1–36, Jan. 2022, doi: 10.1109/TSE.2019.2962027.
[30] H. Taniguchi, H. Sato, and T. Shirakawa, "A machine learning model with human cognitive biases capable of learning from small and biased datasets," *Sci. Rep.*, vol. 8, no. 1, p. 7397, May 2018, doi: 10.1038/s41598-018-25679-z.
[31] S. E. Saad and J. Yang, "Twitter sentiment analysis based on ordinal regression," *IEEE Access*, vol. 7, pp. 163677–163685, 2019, doi: 10.1109/ACCESS.2019.2952127.
[32] F. Li, X. Zhang, X. Zhang, C. Du, Y. Xu, and Y.-C. Tian, "Cost-sensitive and hybrid-attribute measure multi-decision tree over imbalanced data sets," *Inf. Sci.*, vol. 422, pp. 242–256, Jan. 2018, doi: 10.1016/j.ins.2017.09.013.
[33] H.-S. Choi *et al.*, "XGBoost-based instantaneous drowsiness detection framework using multitaper spectral information of electroencephalography," in *Proc. ACM Int. Conf. Bioinf., Comput. Biol., Health Informat.*, Washington, DC, USA, Aug./Sep. 2018, pp. 111–121, doi: 10.1145/3233547.3233567.
[34] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
[35] GitHub. *Is Normalization Necessary? Issue #357 DMLC/XGBoost*. Accessed: Jan. 29, 2022. [Online]. Available: https://github.com/dmlc/xgboost/issues/357

**Mauro A. A. da Cruz** received the Ph.D. degree from the Université de Haute-Alsace (UHA), France, and the National Institute of Telecommunications (Inatel), Brazil, as an international joint cooperation, the M.Sc. degree from Inatel, and the B.Sc. degree (Licentiate, five-year) in informatics engineering from the Universidade Católica de Angola (UCAN), Angola. His research interests include the IoT, middleware for IoT, and mobile computing.

**Lucas R. Abbade** received the B.Sc. degree in computer engineering from the Instituto Nacional de Telecomunicações (INATEL), Brazil, in 2019. He is currently pursuing the master's degree with the Universidade de São Paulo (USP), São Paulo, Brazil. He is also an Equity Strats Analyst at Morgan Stanley, São Paulo. His research interests include machine learning, data science, quantitative finance, blockchain, and information security.

**Pascal Lorenz** (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees from the University of Nancy, France, in 1990 and 1994, respectively. From 1990 to 1995, he was a Research Engineer at WorldFIP Europe and Alcatel-Alsthom. He has been a Professor at the University of Haute-Alsace, France, since 1995. He is the author/coauthor of three books, three patents, and 200 international publications in refereed journals and conferences. He was the Chair of IEEE ComSoc France, the Chair of Vertical Issues in Communication Systems Technical Committee Cluster, the Chair of the Communications Systems Integration and Modeling TC, the Chair of the Communications Software TC, and the Chair of the TC on Information Infrastructure and Networking. He was an IEEE ComSoc Distinguished Lecturer during 2013–2014.

**Samuel B. Mafra** received the B.Sc. degree from Santa Catarina State University (UDESC), Joinville, Brazil, in 2010, the M.Sc. degree from the Federal University of Paraná (UFPR), Curitiba, Brazil, in 2012, and the D.Sc. degree in electrical engineering from the Federal Technological University of Paraná (UTFPR) in 2015. He held a post-doctoral position at UFPR in the period from September 2015 to May 2018. He is currently a Professor at the National Institute of Telecommunications (Inatel), Brazil, where he is currently an Assistant Professor. His research interests lie in the broad area of wireless communication theory, with applications to cognitive radio networks, cooperative communications, and the Internet of Things. He is a member of the IEEE Communications Society and Brazilian Telecommunications Society (SBrT).

**Joel J. P. C. Rodrigues** (Fellow, IEEE) is currently with the College of Computer Science and Technology, China University of Petroleum, Qingdao, China; the Senac Faculty of Ceará, Brazil; and a Senior Researcher at the Instituto de Telecomunicações, Portugal. He is the Leader of the Next Generation Networks and Applications (NetGNA) Research Group (CNPq). He has authored or coauthored over 1000 papers in refereed international journals and conferences, three books, two patents, and one ITU-T Recommendation. He was the Director for Conference Development—IEEE ComSoc Board of Governors and the Past-Chair of the IEEE ComSoc TCs on eHealth and on Communications Software. He is a fellow of AAIA. He is an IEEE Distinguished Lecturer and a Member Representative of the IEEE ComSoc on the IEEE Biometrics Council. He is the Editor-in-Chief of the *International Journal of E-Health and Medical Communications* and an editorial board member of several journals.