Development of an Intelligent Classifier Model for Denial of Service Attack Detection

Álvaro Michelena[1]*, Jose Aveleira-Mata[2], Esteban Jove[1], Héctor Alaiz-Moretón[3], Héctor Quintián[1], José Luis Calvo-Rolle[1]

[1] University of A Coruña, CTC, CITIC, Department of Industrial Engineering, Ferrol, A Coruña, (Spain)
[2] University of León, RIASC: Research Institute of Applied Sciences in Cybersecurity, León, (Spain)
[3] University of León, Department of Electrical and Systems Engineering, León, (Spain)

Received 23 February 2023 | Accepted 31 July 2023 | Published 21 August 2023

Abstract

The prevalence of Internet of Things (IoT) systems deployment is increasing across various domains, from residential to industrial settings. These systems are typically characterized by their modest computational requirements and use of lightweight communication protocols, such as MQTT. However, the rising adoption of IoT technology has also led to the emergence of novel attacks, increasing the susceptibility of these systems to compromise. Among the different attacks that can affect the main IoT protocols are Denial of Service attacks (DoS). In this scenario, this paper evaluates the performance of six supervised classification techniques (Decision Trees, Multi-layer Perceptron, Random Forest, Support Vector Machine, Fisher Linear Discriminant and Bernoulli and Gaussian Naive Bayes) combined with the Principal Component Analysis (PCA) feature extraction method for detecting DoS attacks in MQTT networks. For this purpose, a real dataset containing all the traffic generated in the network and many attacks executed has been used. The results obtained with several models have achieved performances above 99% AUC.

Keywords

Cybersecurity, DoS Attacks, Feature Extraction, MQTT, Soft Computing, Supervised Classifiers.

DOI: 10.9781/ijimai.2023.08.003

I. Introduction

IoT (Internet of Things) allows daily objects to acquire new functionalities, such as gathering information from the environment or performing actions in the environment through actuators. Thanks to internet connectivity, these devices can collect, analyze, and share data between objects, software applications, and cloud platforms. Concepts such as smart cities [1] and Industry 4.0 [2] have emerged thanks to healthcare devices, industrial sensors, and actuators connected to the Internet.

Recent market studies have predicted that the number of connected devices will be more than 70% of total internet connections, with the number growing by 180% in the next four years [3].

IoT systems present new cybersecurity challenges due to the heterogeneous growth in the number of devices and linked services. Operating in resource-constrained environments, such as networks with low transfer rates due to interference, low power consumption, and small embedded processors, requires the use of simple protocols and devices, which may limit security aspects [4], [5].

The different protocols can be represented like a layered structure, where each of them provides a different functionality [6], being the most widely used architecture the three-layer topology. Considering the studies on the protocols used in IoT environments [7], [8], they can be classified according to Table I.

* Corresponding author.
E-mail address: alvaro.michelena@udc.es

TABLE I. IoT Protocol Classification in Three Layers

Protocols	Layers
XMPP, MQTT, CoAP, Web-Socket, HTTP REST	Application
UDP, TCP, 6LoWPAN	Network
LoRa, IEEE 802.15 (BLE, Bluetooth, ZigBee), IEEE 802.11(Wi-Fi)	Physical

Malicious actors can exploit a diverse range of attack vectors, based on the special behaviours of this kind of environment. As a result, there is a growing interest in cybersecurity topics research around IoT. In the review addressed by Lu & Xu [9], a clear upward trend in research on "IoT cybersecurity" is shown.

Attackers usually exploit vulnerabilities of specific IoT protocols embedded in TCP/IP networks [10]. One of the most common attacks is a denial of service (DoS) which consists of the attacker saturating the network with a large volume of traffic until the system cannot provide [11] service. One of the most famous attacks that have been performed on the Internet was the "Mirai" botnet, developed on September 2016. It performed a DDoS attack, based on a distributed denial of service over "DynDNS" servers, being one of the largest DNS service providers systems. "Mirai" attack generated 1.2 terabits of malicious traffic, forcing to set of "DynDNS" servers, the out of service during several hours, which caused the fall of widespread of internet services such as Twitter, Netflix, Reddit, and GitHub [12]. A more recent botnet attack was "dark_nexus" which dated in 2020 compromised around 1370 devices. Bitdefender analysis report [13] shows how "dark_nexus" works, with a behaviour very similar to Mirai.

A. IoT Cybersecurity Solutions

This subsection addresses state of the art to show the most popular solutions for protecting IoT environments.

The research work conducted by Idriss et al. [14], delves into various options for implementing cybersecurity in IoT systems, being the most notable of them, the implementation of a hardware module that allows adding randomness to the encryption in a more lightweight way than other methods, calling PUF (Physical Unclonable Functions) based lightweight authentication. Amanlou et al. [15] proposes a lightweight authentication system for IoT systems using the MQTT protocol, a temporary key exchange algorithm FCDHE, and the shared key authentication (PSK) algorithm. This combination provides mutual authentication between IoT network devices thanks to an authentication scheme known as ECDHE-PSK. This implementation would also improve IoT cybersecurity using this protocol. However, the systems deployed previously must be modified.

In the last few years, new IoT cybersecurity approaches have been published. Zhu & Deng [16], include IoT security situation classification based on support vector machines and security situation awareness based on Markov game model. Choudhary & Pahuja [17], present a new technique called Steering Convention for Vitality Effective Systems (SC-VFS) that improves vitality proficiency and ensures the safety of sensitive information in remote sensor networks, with a focus on detecting doppelganger attacks in IoT-based intelligent health applications. Berjón et al. [18], introduce the SCIFI-II system, which simplifies the development of applications in IoT contexts by allowing the distribution of events between event brokers and designing components that are decoupled from the event brokers.

To address cybersecurity without modifying existing systems, implementing Intrusion Detection Systems (IDS) is the main solution since they can analyse the traffic generated by the environment, without intervening in its configuration. There are several types of intrusion detection systems, depending on the paradigm applied in their detection module, being theses rule-based IDS or anomaly-based IDS [19]. The anomaly-based IDS paradigm observes network traffic features to detect attacks by identifying altered behaviour within the network.

Anomaly detection systems (IDS) are an effective solution for implementing attack detection in IoT systems. They are highly versatile in detecting new types of attacks and can adapt to new protocols. Anomaly IDS systems utilize classification models created with soft computing techniques, such as machine and deep learning, supported by neural networks [20]. The implementation of these procedures requires training models using high quality datasets [21].

B. Objectives

Based on the state of the art addressed previously, this paper aims to develop a functional IDS with an intelligent model for detecting DoS attacks on the MQTT protocol. The model will be constructed thanks to applying soft computing techniques based on machine learning techniques.

To achieve a functional IDS, several tasks are addressed. These tasks are described throughout the paper as follows:

- Study the data sets available to develop the intelligent model that applied the soft computing techniques chosen (Section II).
- Collect a new MQTT dataset (How this dataset has been constructed will be addressed in Section III) because no MQTT datasets exist with normal and DOS traffic for applying machine learning methods.
- Chose and test a set of machine learning methods for application to the previously defined dataset, to achieve the best model for deployment in the IDS (Sections IV and V).

II. Related Works

In order to implement a set of machine learning techniques for getting a functional model that will be inserted in an IDS, it is necessary to work with a specific dataset. This dataset consists of labeled traffic frames, each one tagged as standard/normal network traffic or traffic with hostile purposes. Thanks to the models obtained after a training process, the IoT MQTT behaviour is modeled as well as the recognition of the most important features for understanding this behaviour.

Using general purposed datasets collected from TCP/IP networks can be a solution for modeling attacks such as botnets, without focusing on the special characteristics of IoT systems [22]. To obtain anomaly-based IDS capable of detecting DoS attacks, well-known datasets are used. Some of them were created like over general purposed networks (non IoT networks) such as the NSL-KDD dataset [23], which an enhanced version of the KDD99 dataset was developed in 1993. Some research works address the use of different artificial intelligence techniques for modeling traffic and detecting distributed denial of service DDoS attacks, caused by IoT system botnets [24], [25]. Liu et al. [26], also use the NSL-KDD dataset for getting the model that will be included in the IDS, in this case, Kontiki is the software utilized for simulating IoT environments where CoAP (Constrained Application Protocol) works.

MQTT is typically utilized to connect small devices with restricted bandwidth in IoT [27] and Industry 4.0 environments [28]. MQTT is a publish/subscribe protocol designed for lightweight machine-to-machine (M2M) communications, being ideal for connecting small devices to networks with low bandwidth. MQTT architecture follows a star topology with a central server node called a Broker. The communication is based on topics. Clients can create and publish topics, while others that want to receive information from that topic, can subscribe to it. The broker side handles all the load of the overall system. This operation can be seen in detail in Fig. 1.

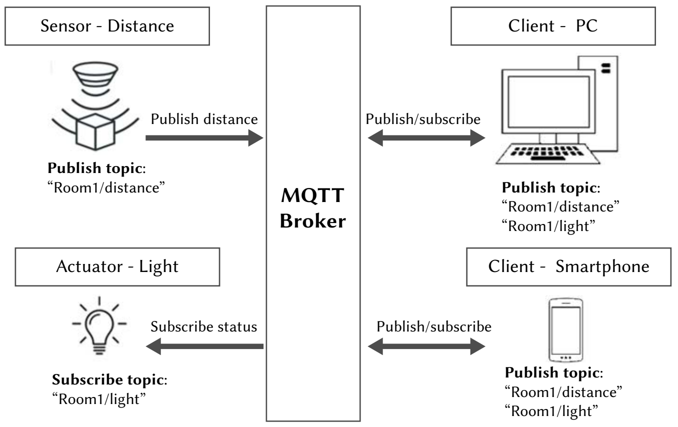
Sensor - Distance
Publish distance
Publish topic: "Room1/distance"
MQTT Broker
Client - PC
Publish/subscribe
Publish topic: "Room1/distance" "Room1/light"
Actuator - Light
Subscribe status
Subscribe topic: "Room1/light"
Client - Smartphone
Publish/subscribe
Publish topic: "Room1/distance" "Room1/light"

Fig. 1. MQTT environment.

MQTT does not specify any networking or routing techniques; it uses TCP as a transport protocol and TLS/SSL for security. IoT application protocols, such as MQTT, can be supported by transport layer security (TLS), but there are no mechanisms in place to protect IoT devices from denial of service (DoS), being this susceptible to this kind of attack. Several datasets have recently been created that focus on attacks on MQTT systems. For example, "MQTT-iot-ids2020" [29] was generated using a simulated MQTT architecture that consists of twelve sensors sending random messages, a server that manages the connections called "broker", a simulated camera, and an attacker. On top of this environment, the attacker performs network scanning and brute force attacks to decrypt access credentials.

The "TON_IoT" dataset [30] focuses on modern IoT systems using the MQTT protocol. It was generated in a simulated environment with the NSX-VMware platform [31] where network scanning process and DoS attacks are performed on the MQTT environment.

III. Case Study

A Denial of Service (DoS) attack involves flooding a network with a high traffic volume to the point where the system cannot provide its intended services. This attack particularly affects IoT systems, since they have limited computational capacity and most protocols they use are for processing information in real time [32].

The previously described datasets use simulated environments and traffic that is collected by only considering the frames of the MQTT protocol. This paper presents the development of a dataset that aggregates all traffic from a real-world environment, utilizing an IoT system with the MQTT protocol. Notably, any denial of service attacks against the broker within this system is labeled. Therefore, an MQTT environment is developed to simulate real traffic thanks to a broker programmed in "node.js" with the "Aedes" library [33]. It uses an actuator with a relay, a distance sensor, and two clients: a smartphone and a computer. All the traffic generated in this environment, including the interactive Internet traffic, is captured by a router with the "OpenWRT OS" installed.

Several DoS attacks are performed on the environment, taking into account the vulnerabilities of the protocol. An attacker scans the network with a search engine like "Shodan" through the well-known port 1883 [23]. Thanks to this, it is possible to find out which servers use this protocol as a broker, being this the vulnerable part of the MQTT system, due to this centralizes all control of the system.

The attacks are performed with a tool developed for performance testing called "Malaria MQTT" [34]. This tool sends many messages to the broker, simulating 1000 clients, sending 1000 messages per second with a size of 100 characters. Thanks to this, it is not possible for the broker to respond to all of these messages, generating a service failure in the IoT environment.

To generate the dataset, all the traffic in the test environment developed is captured, standard internet browsing traffic and traffic generated by the IoT environment. The router registers all the traffic for generating a PCAP file. The set of PCAP files contains a lot of information and many fields. In this way it is simplified by a dissecting procedure. With this purpose, a tool developed for the authors was designed [35]. The dissecting tool works as follows:

- The frames in a pcap file must be organized to analyze a DoS attack effectively. During an attack, a large number of frames may be generated in a short period of time, and the capture tool (such as tcpump with OpenWRT) may overlap several frames with the same timestamp. To obtain useful information about the attack, it is necessary to separate these overlapping frames based on their timestamps.
- The frames are dissected by taking some fields common to all the frames. These common fields are chosen, taking as an example the AWID dataset, which is from 802.11 protocol. All fields that make up the MQTT protocol are included, resulting in 65 fields for each frame.
- To properly label each frame as either part of an attack or normal traffic, it is necessary to consider the timestamp of when each attack begins and ends. Each frame should be tagged based on this information, allowing for a clear distinction between attack frames and normal traffic.

The resulting dataset contains all traffic generated by the described environment, capturing both the normal operation traffic and the traffic under a DOS attack on the MQTT protocol. The dataset comprises a CSV file in which 65 fields delineate each captured frame. It compiles a total of 94,625 frames, 45,513 of which are labeled as "under attack" in the "type" field, while 49,112 are labeled as "normal". This dataset is currently accessible online [36].

IV. Soft Computing Techniques Used

Two stages are implemented to detect DoS attacks in MQTT networks with a functional IDS based on an intelligent model. The first one reduces the dataset dimensionality, while in the second one, a set of classification methods are implemented, choosing the best one.

Therefore, this section is divided into two subsections. Section IV.A will describe the feature extraction method employed, while Section IV.B will define the six different classification techniques implemented.

A. Feature Extraction Method

As discussed in Section III, the working dataset contains a total of 65 variables. This large number of features can lead to a high computational cost in the model training process and a certain mathematical complexity in the classifiers. Therefore, in these scenarios, it is very common to use dimensionality reduction techniques, based on feature extraction, to minimize the number of variables in the dataset in order to reduce the computational cost and obtain simpler classifiers with good performance. Nowadays, there is a wide variety of techniques and algorithms for dimensionality reduction, being Principal Component Analysis (PCA) one of the most common.

1. Principal Component Analysis

Principal Component Analysis (PCA) is an unsupervised multivariate statistical approach developed by Pearson [37] and is generally used for dimensional reduction. The variation of a multivariate dataset is described by this technique as a set of uncorrelated variables corresponding to linear combinations of the original parameters. In general, the principal purpose of this strategy is to generate a new set of orthogonal axes that maximize data variance, avoiding the loss of information. This is accomplished by computing the eigenvalues of the correlation matrix. The initial set can then be linearly translated into lower dimension space using the eigenvectors [38]. Fig. 2 shows an example in $\mathbb{R}^2$ of obtaining the principal components.

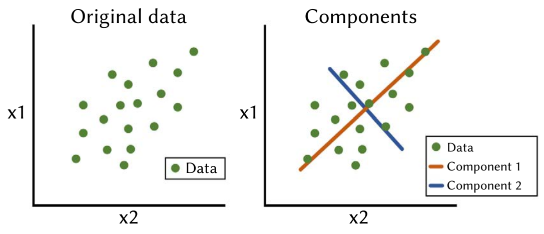

Fig. 2. PCA example.

B. Classification Methods

This subsection describes briefly the six supervised classification techniques implemented in this research.

1. Decision Trees

One of the simplest and most widely used supervised machine learning techniques are decision trees (DT). This method is based on generating a model with a hierarchical tree structure with a root node, branches, decision nodes, and response nodes, also known as leaves [39].

The model starts at the root node, where one of the dataset variables is evaluated. Then, according to the variable value, one of the output branches is selected to re-evaluate the data in a decision node. This process is repeated until the data reaches a response node where the sample is classified with the value associated with the leaf node. In general, the decision tree divides the data according to the value of its variables, so it is essential to find the optimal division boundaries. For this purpose, this method calculates each variable's entropy, or Gini index, to know its impurity degree. By estimating this value, the information gain value can be determined by comparing the impurity of the data set before and after the node splitting. Since the decision tree has a hierarchical structure, the tree is built from top to bottom using the variables with the highest information gain at the nodes of the first stages.

On the other hand, the decision trees algorithm generates models that are easy to understand and interpret; however, this technique cannot achieve good performance in complex problems since it generates large and complex trees that tend to cause overfitting.

2. Multi-Layer Perceptron

Artificial neural networks are one of the most widely used techniques in the field of soft computing. This method uses artificial neurons linked in layers to generate a structure of interconnected neurons that emulates the functioning of a human brain [40]. In this way, neural networks consist of an input layer, one or more hidden layers, and an output layer. Each of these layers is composed of one or more artificial neurons. These neurons sum the input values weighted by weights related to each input, and an independent value, also known as bias. Then, an activation function is applied to this value to obtain the neuron's output result.

Information flows through the network's hidden layers from the input to the output layer. In contrast, the training process is executed from the output layer to the input layer, applying a method known as backpropagation. The training process calculates the necessary gradients to optimize and adjust each network connection's weights.

Different network architectures can be developed depending on the configuration of the layers and the connections of layers and neurons. However, one of the simplest and most commonly used structures is the Multilayer Perceptron neural network (MLP), which is characterized by each neuron being connected to all the neurons of the next layer.

3. Random Forest

Random Forest (RF) is a well-known supervised machine learning technique commonly applied in classification and regression tasks based on implementing a certain number of decision trees [41].

Its performance is based on hiring a certain number of decision trees to generate a more accurate and robust model. Each random forest tree is different since it is trained with different random subsets selected from the training data. The Bootstrap Aggregation, or Bagging, is used to obtain the data subsets. This technique generates as many subsets as decision trees used in the model.

Finally, with each tree trained, Random Forest uses each decision tree to classify the input data. The classification of all the trees is then analyzed, and the most common prediction is taken as the model's output classification.

4. Support Vector Machine

Other well-known supervised techniques are the Support Vector Machines (SVM) developed by Cortes and Vapnik [42]. These methods are a group of machine learning algorithms often used for classification and regression tasks. The main objective of SVMs is to achieve a hyperplane that maximizes the minimum distance, known as the margin, between the hyperplane and the nearest samples of each class. This margin is used to determine a boundary for classifying new data samples.

The above SVMs definition assumes that a linear boundary can separate the classes. However, most real-world datasets are not linearly separable. To solve this problem, SVMs use data transformations, $\langle x_i,x_j\rangle \rightarrow \langle \phi(x_i),\phi(x_j)\rangle$, for mapping the data into a higher dimensional space, where a linear boundary can separate it. The specific transformation implemented, $\phi(x)$, depends on the kernel function selected.

5. Fisher Linear Discriminant

The Linear Discriminant Analysis (LDA), or Fisher Linear Discriminant Analysis, is a supervised classification machine learning technique developed by R.A. Fisher [43].

The main goal of the Fisher Linear Discriminant is to find the best linear combination of features that separates different training data classes as much as possible. Therefore, LDA searches out the hyperplane where the means of each class are as far apart as possible and the classes have the least variance in their data. The objective function, Equation (1) defined as $J(\theta)$ is maximized in the optimization process.

$$J(\theta) = \frac{(\mu_1 - \mu_2)^2}{\hat{s}_1^2 + \hat{s}_2^2} \tag{1}$$

where μ_1 and μ_2 are the mean value of class 1 and 2 respectively, and $\hat{s}_1$ and $\hat{s}_2$ correspond to the within-class variance 1 and 2.

6. Naive Bayes

Naive Bayes, also known as Naive Bayesian (NB), are straightforward machine learning methods, frequently used for classification issues, that are based on the Bayes statistical theorem. Additionally, these approaches presuppose that given the class, data properties are conditionally independent [44]. Although this assumption is generally excessively strong, Naive Bayes performance still produces outcomes that are very competitive and computationally efficient.

Under this technique, different algorithms can be applied. In the current research, Bernoulli and Gaussian methods have been tested.

Bernoulli Naive Bayes: Each feature is thought to correlate to a binary value. In this model, the probability is obtained using Equation (2).

$$P(x_i \mid c) = P(x_i \mid c)b_i + (1 - b_i)(1 - p(x_i \mid c)) \tag{2}$$

To use this approach, all data features must be binary; if a feature contains any other type of data, a binarization process is carried out.

Gaussian Naive Bayes: The numerical attribute values in Gaussian NB have a normal distribution and are shown concerning the mean and standard deviation. Equation (3) is used in this approach to determine the probability of the features.

$$P(x_i \mid c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} exp\left(-\frac{(x_i - \mu_c)^2}{2\sigma_c^2}\right) \tag{3}$$

where σ is the standard deviation and μ the mean value.

V. Experiments and Results

The present section describes the setup of the experiments and the results obtained.

A. Experiments Setup

This section provides the experiment configurations, including the tools and metrics used to measure and compare the performance of each classifier. The experiments were implemented using Python and several libraries such as Scikit-learn, Pandas, Numpy, TensorFlow and Keras.

To configure the experiments, the fundamental stages of machine learning problems summarized in Fig. 3, were followed. Each of these stages is described in detail below.

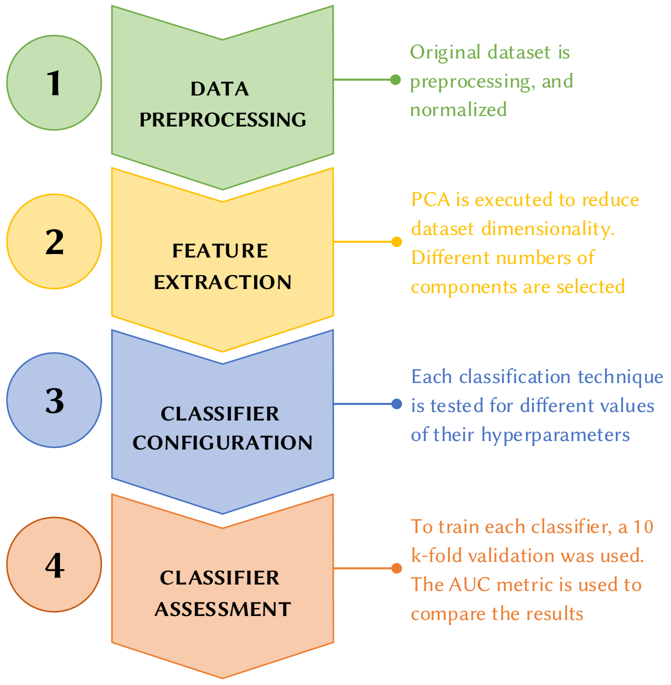
1
DATA PREPROCESSING
Original dataset is preprocessing, and normalized
2
FEATURE EXTRACTION
PCA is executed to reduce dataset dimensionality. Different numbers of components are selected
3
CLASSIFIER CONFIGURATION
Each classification technique is tested for different values of their hyperparameters
4
CLASSIFIER ASSESSMENT
To train each classifier, a 10 k-fold validation was used. The AUC metric is used to compare the results

Fig. 3. Experiment setup scheme.

1. Data Preprocessing

The first step was the preprocessing of the study case dataset. Once the data was analyzed, samples with missing data and constant variables for all the samples were removed from the dataset. On the other hand, the non-numerical variables were codified to numerical features, and finally, the data were normalized using the z-score method, with a mean value of 0 and a standard deviation of 1.

2. Feature Extraction

After preparing the dataset, the PCA technique was employed to reduce the number of features and choose the most important dataset variables. The number of principal components to retain was determined by analyzing the variance explained by each component, focusing on those that explain a significant amount of variance. In order to obtain the best classifier, both in terms of performance and computational cost efficiency, in this research, the experiments were carried out taking into account the different number of components.

3. Classifier Configuration

Each of the supervised classification techniques presented in Section IV has been tested for different configurations. Table II shows the hyperparameters that have been configured for each algorithm as well as the values that have been implemented. Each of the model's hyperparameters is briefly described below.

- Decision Trees (DT): for this technique, decision trees have been evaluated for different maximum depth parameters, from 5 to 50 layers of the tree with intervals of 5. In addition, it has also been tested, not indicating a maximum depth value ("None"). This way, the nodes are expanded until all leaves contain less than two samples.
- Multi-layer Perceptron (MLP): MLP neural networks have been analyzed for different network structures, considering the number of hidden layers, the number of neurons in the hidden layers, and the dropout percentage. The dropout corresponds to the middle layers used to control the regularization of the neural network and avoid overfitting problems. The following values have been taken into account for each parameter:
  - Number of hidden layers: 1, 2, and 3 hidden layers.
  - Number of neurons in hidden layers: 5, 10, 15, and 20 neurons per layer.
  - Dropout: 0 and 20%.

  It is important to note that the *ReLu* function was used as activation function in the neurons of the hidden layers and *Softmax* in the output layer. This configuration is commonly used in classification tasks with neural networks.
- Random Forest (RF): the parameter to be determined in this technique is the number of decision trees that conform the model. In this case, the algorithm performance was evaluated for models of 10 to 100 trees with increments of 10 trees.
- Support Vector Machines (SVM): in this case, different configurations of the Support Vector Machine have been tested by modifying the algorithm kernel, which indicates the transformation function, and the data regularisation factor, *C*. The strength of the regularisation is inversely proportional to *C*. The values used in these hyperparameters are:
  - Kernel: linear, polynomial, rbf (Radial Basis Function) and sigmoid.
  - Data regularization *C*: 0.001, 0.01, 0.1 and 1.
- Fisher Linear Discriminant (LDA): the performance of this technique has been tested for three different algorithms solvers (least squares, *lsqr*, singular value decomposition, svd and eigenvalue decomposition, *eigen*).
- Naive Bayes (NB): as already mentioned in IV.B.6 the performance of Bernoulli and Gaussian Naive Bayes have been evaluated.

4. Classifier Assessment

For the training process of each model, k-fold cross-validation with a k value of 10 has been used. In addition, the Area Under the receiving operating Curve (AUC) has been considered as the evaluation metric, which is widely used in classification tasks. The relationship between true positive and false positive rates is established by this parameter,

TABLE II.Configurations Tested

Evaluated technique	Evaluated configuration	Tested values
Decision Trees	Maximum depth	5:5:50 and None
Multi-layer Perceptron	Number of hidden layes Neurons in hidden layers Dropout (%)	1:1:3 5:5:20 0, 10
Random Forest	Number of trees	10:10:100
Support Vector Machine	Data regularization Kernel	0.001, 0.01, 0.1, 1 linear, poly, rbf, sigmoid
Fisher linear discriminant	Solver	svd, lsqr, eigen
Naive bayes	Algorithm	Bernoulli, Gaussian

which boasts two key benefits. Firstly, it offers a unified evaluation of classifier performance, and secondly, it remains unaffected by variations in class distribution.

On the other hand, the computational cost of each classifier implemented has also been measured. For this purpose, the average training time of each configuration of the models has been considered. In this sense, it must be taken into account that the experiments have been executed on a computer with an Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz and a RAM memory of 8GB.

B. Results

The results derived from the experimental setup outlined above are shown in this section. First, to determine the number of components to reduce the initial dataset, an initial Principal Component Analysis was executed to identify the components and their respective percentage of variance explained. Fig. 4 shows the results obtained in bar graph format.

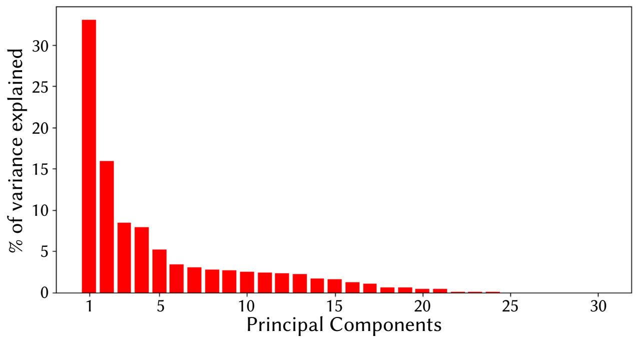

Fig. 4. PCA initial analysis.

Based on the achieved results, three different component selections will be considered for the experiments for evaluating the performance of combining dimensional reduction with the above-described classification techniques in terms of classification accuracy and computational cost. The three component selection criteria are as follows:

- Components with a percentage of explained variation greater than 10%: in this case the first 2 components are selected.
- Components with a percentage of variance explained greater than 5%: in this case the first 5 components are selected.
- Components with a percentage of explained variation greater than 0.01%: in this case the first 24 components are selected.

After selecting the different numbers of components to be used in each experiment, we trained and evaluated each technique's performance using the proposed configurations.

Before presenting the results, since the models were tuned and evaluated using k-fold cross-validation, it is essential to highlight that all the tables in this section depict the average AUC and training time values.

Table III presents the results obtained using decision trees. As can be seen, this technique achieved excellent results, exceeding 99% in terms of AUC, with the different configurations tested. Furthermore, it is noticeable that using fewer components significantly reduces the training time, with a slight loss in classifier performance, lower than 0.3% in terms of AUC. This technique has very low training times, less than 1 second in some of its configurations.

On the other hand, Table IV shows the results obtained with the Multi-Layer Perceptron neural networks (MLP). This technique exhibits high performance, reaching more than 99.8% of AUC in some configurations. In this case, reducing the number of components also reduces the classifier's performance. Comparing the results obtained, a reduction of more than 1% in terms of AUC can be produced using 24 or 2 components. On the other hand, the training time is not affected by the number of components, i.e., the number of neurons in the network's input layer. The network dimension, determined by the number of hidden layers and neurons per layer, is the main factor affecting computational cost. Additionally, it is observed that using dropout in the network does not improve classifier performance and significantly increases the training time, as it involves adding a new layer (the regularisation layer). Generally speaking, this technique presents a higher computational cost than decision trees.

Table V presents the performance of the Random Forest method for its different configurations. This technique achieves very good classifiers, with an AUC of over 99% in all configurations tested. Regarding the results, it can be observed that using a greater number of trees does not significantly improve the model's performance. For example, comparing the 100-tree model with the 10-tree model showed a difference of less than 0.1%. Similarly, the classifier's performance does not deteriorate significantly when using fewer components, achieving a reduction of 0.2% AUC when comparing models trained with 24 components to models adjusted with 2. However, models with fewer trees combined with a reduced number of selected components minimize the computational cost measured in training time, reducing it by more than 70% in some cases.

On the other hand, Table VI shows the performance of support vector machines. With this technique, very different results were obtained among the evaluated configurations. In general, it can be observed that using a reduced number of components greatly affects the classifier's performance, with a loss of more than 20% of AUC in many cases. Moreover, the best results are obtained with the highest value of the hyperparameter C, which implies low data regularisation. For this technique, the best model obtained was the one that uses the polynomial kernel with $C = 1$, which reaches a 98.32% AUC considering 24 components. Finally, highlight that reducing the number of components used does not reduce the training time. Compared to the other techniques, except for MLP, the computational cost of this technique is much higher.

The performance results of Fisher's Linear Discriminant Analysis are presented in Table VII. It can be observed that changing the algorithm's solver does not affect the classifier's performance, and this hyperparameter only influences the training time. In this regard, the svd (Singular value decomposition) method is the most computationally expensive compared to the other solvers tested. Additionally, when analyzing the impact of the number of components on the classifier's performance, it is evident that reducing the number of components significantly compromises the classifier's performance, lowering the AUC value and the training time. For this technique, the optimal classifier is obtained using the lsqr (Least squares solution) algorithm and components, which achieves over 89% AUC with an average training time of 0.159 seconds.

Finally, the Gaussian and Bernoulli naive Bayes were tested and the results are presented in Table VIII. With this technique, the Gaussian model fits better to the problem posed and performs better than the Bernoulli algorithm. On the other hand, it can be observed how considering a greater number of components improves the AUC result of the classifier and increases the training time.

Fig. 5 summarizes the best AUC results obtained for each of the techniques and the different number of components. This graph shows how using a greater number of components improves the results measured by the AUC metric and how the best classifiers are obtained with the Decision Trees, Multi-layer Perceptron, and Random Forest.

TABLE III. Decision Trees Results

PCA / Model setup	2 components		5 components		24 components	
Nº of trees	AUC (%)	T. time (s)	AUC (%)	T. time (s)	AUC (%)	T. time (s)
5	98.27	0.064	98.43	0.135	98.83	0.705
10	99.00	0.098	99.02	0.224	99.24	1.263
15	99.09	0.112	99.19	0.262	99.30	1.679
20	99.09	0.116	99.24	0.273	99.33	2.013
25	**99.10**	0.116	99.25	0.278	99.35	2.042
30	99.09	0.115	99.28	0.276	99.33	2.038
35	99.10	0.118	99.27	0.276	99.34	2.038
40	99.10	0.117	**99.29**	0.275	**99.35**	2.037
45	99.09	0.117	99.27	0.275	99.34	2.038
50	99.09	0.117	99.27	0.275	99.35	2.038
None	99.09	0.117	99.28	0.280	99.33	2.039

TABLE IV. Multilayer Perceptron Results

PCA / Model setup			2 components		5 components		24 components	
Nº of hidden layers	Nº of neurons	Dropout (%)	AUC (%)	T. time (s)	AUC (%)	T. time (s)	AUC (%)	T. time (s)
1	5	0	96.22	20.755	99.11	20.714	99.57	20.786
1	5	10	96.04	21.794	97.33	21.933	99.45	22.043
1	10	0	98.20	21.259	99.27	21.116	99.68	21.405
1	10	10	98.18	22.446	99.28	22.560	99.68	22.884
1	15	0	98.25	21.496	99.30	21.328	99.75	21.579
1	15	10	98.22	22.680	99.29	22.810	99.70	23.044
1	20	0	98.41	22.077	99.35	21.936	99.76	22.032
1	20	10	98.33	24.207	99.39	23.131	99.74	23.330
2	5	0	93.35	22.652	92.48	22.663	99.58	22.250
2	5	10	96.39	24.766	99.07	27.514	99.46	24.297
2	10	0	98.37	22.977	99.35	23.039	99.73	22.941
2	10	10	96.50	25.435	99.37	25.471	99.68	25.243
2	15	0	98.47	23.459	99.42	23.508	99.73	23.286
2	15	10	98.50	25.942	99.41	26.003	99.76	25.722
2	20	0	98.62	24.099	99.47	23.861	99.79	23.591
2	20	10	98.51	26.379	99.42	26.391	99.78	26.056
3	5	0	98.12	24.411	93.58	24.170	99.59	23.799
3	5	10	95.40	27.229	99.13	26.911	99.49	26.760
3	10	0	98.43	25.194	99.43	24.801	99.73	24.457
3	10	10	98.22	28.644	99.34	27.994	99.73	27.849
3	15	0	98.65	25.787	99.53	25.205	99.79	25.111
3	15	10	98.52	29.346	99.45	28.693	99.76	28.716
3	20	0	98.79	26.494	99.48	25.795	99.82	25.287
3	20	10	98.52	30.360	99.47	29.480	99.80	29.156

TABLE V. Random Forest Results

PCA / Model setup	2 components		5 components		24 components	
Nº of trees	AUC (%)	T. time (s)	AUC (%)	T. time (s)	AUC (%)	T. time (s)
10	99.19	0.594	99.29	0.718	99.36	2.052
20	99.20	1.211	**99.33**	1.539	99.39	4.221
30	99.19	1.839	99.31	2.013	99.38	6.209
40	99.19	2.435	99.32	2.665	99.42	6.655
50	99.18	3.248	99.32	3.278	99.40	8.994
60	99.20	3.385	99.32	3.933	99.42	11.621
70	99.19	4.021	99.30	4.610	99.41	12.580
80	99.19	4.160	99.32	5.281	99.41	12.646
90	**99.21**	3.922	99.31	6.216	**99.44**	14.138
100	99.21	4.393	99.32	6.599	99.41	15.604

TABLE VI. Support Vector Machine Results

Model setup \ PCA		2 components		5 components		24 components	
Kernel	Reg.	AUC (%)	T. time (s)	AUC (%)	T. time (s)	AUC (%)	T. time (s)
linear	1	73.50	37.442	84.47	52.600	90.00	34.126
linear	0.1	73.50	25.434	79.90	28.180	89.97	23.385
linear	0.01	73.50	20.351	74.66	21.244	90.07	23.605
linear	0.001	73.50	19.763	74.64	21.188	89.20	28.774
poly	1	73.69	29.088	90.72	17.263	**98.32**	16.225
poly	0.1	73.69	49.309	75.52	19.876	91.04	22.407
poly	0.01	73.70	26.095	74.87	25.077	75.56	29.913
poly	0.001	73.71	21.021	74.86	25.688	75.55	37.652
rbf	1	**88.33**	26.963	**91.04**	18.143	98.26	16.359
rbf	0.1	73.73	30.471	90.27	25.654	98.27	21.541
rbf	0.01	73.50	27.578	74.65	30.738	90.46	43.467
rbf	0.001	73.50	30.304	74.65	39.303	73.29	62.536
sigmoid	1	71.15	27.657	71.16	26.914	85.77	46.046
sigmoid	0.1	73.03	27.566	73.48	29.337	88.95	49.293
sigmoid	0.01	78.27	40.877	79.53	38.644	82.70	55.067
sigmoid	0.001	73.23	40.016	74.61	49.282	73.49	58.673

TABLE VII. Fisher Linear Discriminant Results

Model setup \ PCA	2 components		5 components		24 components	
Solver	AUC (%)	T. time (s)	AUC (%)	T. time (s)	AUC (%)	T. time (s)
svd	73.51	0.041	74.52	0.049	89.27	0.236
lsqr	73.51	0.038	**74.52**	0.040	**89.27**	0.159
eigen	**73.51**	0.037	74.52	0.042	89.27	0.178

TABLE VIII. Naive Bayes Results

Model setup \ PCA	2 components		5 components		24 components	
Algorithm	AUC (%)	T. time (s)	AUC (%)	T. time (s)	AUC (%)	T. time (s)
Bernoulli	73.51	0.026	75.11	0.022	75.19	0.047
Gaussian	**80.37**	0.023	**78.80**	0.021	**83.55**	0.054

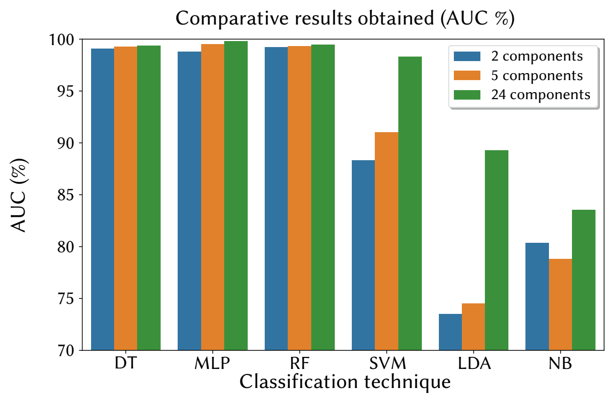

Fig. 5. Comparison of results.

VI. Conclusions and Future Work

This research analyses the performance of six supervised classification techniques in combination with the PCA dimensional reduction method to detect DoS attacks in data networks working with the MQTT protocol. The obtained results have been highly promising, reaching AUC values higher than 95% except for the LDA and Naive Bayes methods that have achieved, for their best configuration, a maximum of 89.27% and 83.55% AUC, respectively.

Considering only the classifier performance, MLP neural networks have been shown to detect better DoS attacks reaching 99.82% AUC for the network with 3 hidden layers, 20 neurons per layer, and without dropout layers. However, the computational cost of this technique, with a mean average training time of 25 seconds, is significantly higher than other methods that have also demonstrated excellent performance, such as, for instance, Decision Trees, with a maximum of 99.35% AUC and training times between 0.1 and 2 seconds, or Random Forest with more than 99% AUC and training times between 0.6 and 15 seconds depending on the selected configuration. SVMs also performed well in many configurations with values above 98% AUC but with training times above 30 seconds. Therefore, considering a computational performance-cost relationship, it can be concluded that decision trees are the best technique.

On the other hand, comparing the results obtained by using a different number of components, it was observed that a significant reduction in the number of components can worsen the classifier's performance and reduce the model's training times. For Decision Trees, a maximum loss of 0.3% AUC is quite optimal when compared to the substantial reduction in training time, often exceeding 90%in some cases. This aspect is also similarly reflected in the Random

Forest models, in which component reduction greatly reduces the computational cost with minimal loss of classifier performance. However, in SVMs and MLP neural networks, using a smaller number of components does not reduce the training time and worsens the performance of these techniques.

Thanks to the high performance of the models achieved, these can be deployed in an IDS for detecting anomalous network behaviours, preventing attacks.

In future works, we will study the performance of other supervised and unsupervised classification techniques and other feature extraction methods to compare their performance against the proposal shown in this paper. Additionally, it will also be considered to test the performance of our proposal for detecting Denial of Service attacks in other types of IoT protocols, such as CoAP and LoRa, among others. On the other hand, the possibility of detecting other types of attacks in this protocol will also be studied. Finally, the development of an intelligent hybrid system capable of detecting different attacks in different IoT network protocols will be analyzed, making it possible to standardize and offer a handy tool for the field of cybersecurity.

Acknowledgment

Álvaro Michelena's research was supported by the Spanish Ministry of Universities (https://www.universidades.gob.es/), under the "Formación de Profesorado Universitario" grant with reference FPU21/00932. Spanish National Cybersecurity Institute (INCIBE) and developed Research Institute of Applied Sciences in Cybersecurity (RIASC). CITIC, as a Research Center of the University System of Galicia, is funded by Consellería de Educación, Universidade e Formación Profesional of the Xunta de Galicia through the European Regional Development Fund (ERDF) and the Secretaría Xeral de Universidades (Ref. ED431G 2019/01).

References

[1] T. M. Ghazal, M. K. Hasan, M. T. Alshurideh, H. M. Alzoubi, M. Ahmad, S. S. Akbar, B. Al Kurdi, I. A. Akour, "Iot for smart cities: Machine learning approaches in smart healthcare—a review," *Future Internet*, vol. 13, no. 8, 2021, doi: 10.3390/fi13080218.

[2] P. K. Malik, R. Sharma, R. Singh, A. Gehlot, S. C. Satapathy, W. S. Alnumay, D. Pelusi, U. Ghosh, J. Nayak, "Industrial internet of things and its applications in industry 4.0: State of the art," *Computer Communications*, vol. 166, pp. 125–139, 1 2021, doi: 10.1016/j.comcom.2020.11.016.

[3] M. Rothmuller, S. Barker, "Iot the internet of transformation 2020," *Juniper Research, Basingstoke, UK, Whitepaper*, 2020.

[4] M. Ahmad, T. Younis, M. A. Habib, R. Ashraf, S. H. Ahmed, "A review of current security issues in internet of things," *Recent Trends and Advances in Wireless and IoT-enabled Networks*, pp. 11–23, 2019, doi: 10.1007/978-3-319-99966-2.

[5] M. H. Khalid, M. Murtaza, M. Habbal, "Study of security and privacy issues in internet of things," *CITISIA 2020 - IEEE Conference on Innovative Technologies in Intelligent Systems and Industrial Applications, Proceedings*, 11 2020, doi: 10.1109/CITISIA50690.2020.9371828.

[6] B. Kepçeoğlu, A. Murzaeva, S. Demirci, "Performing energy consuming attacks on iot devices," in *2019 27th Telecommunications Forum (TELFOR)*, 2019, pp. 1–4.

[7] J. Granjal, E. Monteiro, J. S. Silva, "Security for the internet of things: A survey of existing protocols and open research issues," *IEEE Communications Surveys and Tutorials*, vol. 17, pp. 1294–1312, 2015, doi: 10.1109/COMST.2015.2388550.

[8] R. Yugha, S. Chithra, "A survey on technologies and security protocols: Reference for future generation iot," *Journal of Network and Computer Applications*, vol. 169, p. 102763, 11 2020, doi: 10.1016/j.jnca.2020.102763.

[9] Y. Lu, L. D. Xu, "Internet of things (iot) cybersecurity research: A review of current research topics," *IEEE Internet of Things Journal*, vol. 6, pp. 2103–2115, 4 2019, doi: 10.1109/JIOT.2018.2869847.

[10] J. Tournier, F. Lesueur, F. L. Mouël, L. Guyon, H. Ben-Hassine, "A survey of iot protocols and their security issues through the lens of a generic iot stack," *Internet of Things*, vol. 16, p. 100264, 12 2021, doi: 10.1016/J.IOT.2020.100264.

[11] E. Džaferović, A. Sokol, A. A. Almisreb, S. M. Norzeli, "Dos and ddos vulnerability of iot: a review," *Sustainable Engineering and Innovation*, vol. 1, no. 1, pp. 43–48, 2019.

[12] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, 2017, pp. 1093–1110.

[13] M. H. Khalid, M. Murtaza, M. Habbal, "Study of security and privacy issues in internet of things," in *2020 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA)*, 2020, pp. 1–5, IEEE.

[14] T. A. Idriss, H. A. Idriss, M. A. Bayoumi, "A lightweight puf-based authentication protocol using secret pattern recognition for constrained iot devices," *IEEE Access*, vol. 9, pp. 80546–80558, 2021, doi: 10.1109/ACCESS.2021.3084903.

[15] S. Amanlou, M. K. Hasan, K. A. A. Bakar, "Lightweight and secure authentication scheme for iot network based on publish–subscribe fog computing model," *Computer Networks*, vol. 199, p. 108465, 11 2021, doi: 10.1016/J.COMNET.2021.108465.

[16] X. Zhu, H. Deng, "A security situation awareness approach for iot software chain based on markov game model," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 7, pp. 59–65, 2022, doi: 10.9781/ijimai.2022.08.002.

[17] D. Choudhary, R. Pahuja, "Improvement in quality of service against doppelganger attacks for connected network," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 7, pp. 51–58, 2022, doi: 10.9781/ijimai.2022.08.003.

[18] R. Berjón, M. Mateos, M. E. Beato, A. F. García, "An event mesh for event driven iot applications," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 7, pp. 54–59, 2022, doi: 10.9781/ijimai.2022.09.003.

[19] H. J. Liao, C. H. R. Lin, Y. C. Lin, K. Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, pp. 16–24, 1 2013, doi: 10.1016/J.JNCA.2012.09.004.

[20] L. Aversano, M. L. Bernardi, M. Cimitile, R. Pecori, "A systematic review on deep learning approaches for iot security," *Computer Science Review*, vol. 40, p. 100389, 2021.

[21] A. Khraisat, A. Alazab, "A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," *Cybersecurity*, vol. 4, pp. 1–27, dec 2021, doi: 10.1186/s42400-021-00077-7.

[22] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.

[23] S. Andy, B. Rahardjo, B. Hanindhito, "Attack scenarios and security analysis of mqtt communication protocol in iot system," in *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2017, pp. 1–6.

[24] D. H. Deshmukh, T. Ghorpade, P. Padiya, "Intrusion detection system by improved preprocessing methods and naïve bayes classifier using nsl-kdd 99 dataset," in *2014 International Conference on Electronics and Communication Systems (ICECS)*, 2014, pp. 1–7.

[25] M. Esmaeili, S. H. Goki, B. H. K. Masjidi, M. Sameh, H. Gharagozlou, A. S. Mohammed, "Ml-ddosnet: Iot intrusion detection based on denial-of-service attacks using machine learning methods and nsl- kdd," *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–16, 8 2022, doi: 10.1155/2022/8481452.

[26] J. Liu, B. Kantarci, C. Adams, "Machine Learning- Driven Intrusion Detection for Contiki-NG-Based IoT Networks Exposed to NSL-KDD Dataset," in *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*, New York, NY, USA, 2020, ACM.

[27] P. Sethi, S. R. Sarangi, "Internet of things: architectures, protocols, and applications," *Journal of Electrical and Computer Engineering*, vol. 2017, 2017, doi: 10.1155/2017/9324035.

[28] K. Ramamoorthy, S. Karthikeyan, T. Chelladurai, "An investigation on

industrial internet of things for mission critical things in industry 4 . 0 2 . literature review," *Seybold Report*, vol. 15, pp. 3294–3300, 2020.

[29] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, X. Bellekens, "Machine learning based iot intrusion detection system: An mqtt case study (mqtt-iot-ids2020 dataset)," in *International Networking Conference*, 2020, pp. 73–84, Springer.

[30] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, A. N. Anwar, "Ton-iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165130–165150, 2020, doi: 10.1109/ACCESS.2020.3022862.

[31] VMware, "Vmware nsx data center datasheet." [Online]. Available: https://kb.vmware.

[32] J. Deogirikar, A. Vidhate, "Security attacks in iot: A survey," *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017*, pp. 32–37, 2017, doi: 10.1109/I-SMAC.2017.8058363.

[33] "GitHub - moscajs/aedes: Barebone MQTT broker that can run on any stream server, the node way." [Online]. Available: https://github.com/moscajs/aedes.

[34] K. Palsson, "mqtt-malaria @ github.com," 2018. [Online]. Available: https://github.com/remakeelectric/mqtt-malaria.

[35] J. Aveleira-Mata, H. Alaiz-Moreton, "Functional prototype for intrusion detection system oriented to intelligent iot models," in *International Symposium on Ambient Intelligence*, 2019, pp. 179–186, Springer.

[36] "MQTT Dataset LE-229-18," 2019. [Online]. Available: https://joseaveleira.es/dataset.

[37] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

[38] H. Abdi, L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[39] L. Rokach, O. Maimon, "Decision trees," in *Data mining and knowledge discovery handbook*, Springer, 2005, pp. 165–192.

[40] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[41] A. Cutler, D. R. Cutler, J. R. Stevens, "Random forests," in *Ensemble machine learning*, Springer, 2012, pp. 157– 175.

[42] C. Cortes, V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[43] J. Yang, Z. Jin, J.-y. Yang, D. Zhang, A. F. Frangi, "Essence of kernel fisher discriminant: Kpca plus lda," *Pattern Recognition*, vol. 37, no. 10, pp. 2097–2100, 2004.

[44] I. Rish, *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, 2001, pp. 41–46.

Álvaro Michelena

Álvaro Michelena is a Ph.D. student in Computational Science at the University of A Coruña, Spain, su. He received a M.S. in Industrial Computing and Robotics from the University of A Coruña in 2021. He has worked for a year and a half as a Research Assistant at the Centre for Information and Communications Technology Research (CITIC) of the University of A Coruña where he has collaborated on different research projects. His main research areas are related to applying intelligent techniques for anomaly detection and system modeling.

Jose Aveleira-Mata

Jose Aveleira-Mata is a Ph.D. student on Production and Computing Engineering at the University of Leon, Spain. His research interests include Internet of Things, Cloud Computing, Wireless Sensor Networks and Network Security. He has several papers published in international conferences, as well as scientific publications in JCR journals, on topics related to cybersecurity.

Esteban Jove

Esteban Jove received a M.S. degree in Industrial Engineering from the University of Leon in 2014. After two years working in the automotive industry, he joined the University of A Coruña, Spain, where he has been Assistant Professor of Power Electronics in the Faculty of Engineering since 2016. He received his Ph.D at the University of La Laguna in 2020, and his research has been focused on the use of intelligent techniques for nonlinear systems modelling and anomaly detection using one-class techniques.

Héctor Alaiz-Moretón

Héctor Alaiz-Moretón received his degree in Computer Science, performing the final Project at Dublin Institute of Technology, in 2003. He received his PhD in Information Technologies in 2008 (University of Leon). He has worked as a lecturer since 2005 at the School of Engineering at the University of Leon. His research interests include knowledge engineering, machine and deep learning, networks communication, and security. He has several works published in international conferences, as well as books, more than 90 scientific publications between JCR papers, Lecture Notes and Scientific Workshops. He has been a member of scientific committees in conferences. He has headed several PhD Thesis and research competitive projects. Actually, he is the vice main of RIASC (Institute of Applied Sciences to Cybersecurity).

Héctor Quintián

Héctor Quintián is currently Assistant Professor at University of A Coruña (UDC). Along his academic career, he has published many papers in national and international scientific journals and several book chapters. As far as research activity is concerned, it is worth highlighting his publication activity; over the last 10 years, 62 research papers in journals indexed with relative quality index, all of them in the JCR. Around 70% of them have been published in journals located in the first two quartiles of their categories. He has published a total of 65 contributions in conferences of which 80% correspond to international conferences, most of them indexed at the CORE ranking and at the GII-GRIN-SCIE (GGS) Conference Rating. In addition, it has organized a large number of scientific conferences in various editions (40), all of them of recognized international prestige, His main research lines are focused on artificial intelligence, and not supervised learning developing several algorithms with application to industrial modelling systems.

José Luis Calvo-Rolle

José Luis Calvo-Rolle received M.S. and Ph.D. degrees in Industrial Engineering from the University of Leon in 2004 and 2007, respectively. He is Full Professor in the Systems Engineering and Automation Area, of the Industrial Engineering Department, University of A Coruña. Currently, he is the director of that department and the head of the Environmental Radioactivity Laboratory. In addition, he coordinates the Cybernetic Science and Technology Research Group. His main research areas are focused on the application of intelligent techniques and systems for optimization, diagnosis, modeling and control.