



An automatic unsupervised complex event processing rules generation architecture for real-time IoT attacks detection

José Roldán-Gómez¹ · Jesús Martínez del Rincon² · Juan Boubeta-Puig³ · José Luis Martínez¹

Accepted: 13 December 2022
© The Author(s) 2023, corrected publication 2023

Abstract

In recent years, the Internet of Things (IoT) has grown rapidly, as has the number of attacks against it. Certain limitations of the paradigm, such as reduced processing capacity and limited main and secondary memory, make it necessary to develop new methods for detecting attacks in real time as it is difficult to adapt as has the techniques used in other paradigms. In this paper, we propose an architecture capable of generating complex event processing (CEP) rules for real-time attack detection in an automatic and completely unsupervised manner. To this end, CEP technology, which makes it possible to analyze and correlate a large amount of data in real time and can be deployed in IoT environments, is integrated with principal component analysis (PCA), Gaussian mixture models (GMM) and the Mahalanobis distance. This architecture has been tested in two different experiments that simulate real attack scenarios in an IoT network. The results show that the rules generated achieved an F1 score of .9890 in detecting six different IoT attacks in real time.

Keywords Attack detection · Cybersecurity · Internet of things · Complex event processing · Machine learning

1 Introduction

The Internet of Everything (IoE) has grown rapidly in the last decade and it seems highly unlikely that this growth will stop or slow down any time soon, due to the obvious potential offered by this new paradigm. Proof of this is the

increasing number of interactions with certain applications designed for this paradigm through devices such as smartphones or wearables. IoE can be considered an extension of the Internet of Things (IoT) [1]. While the two key elements of IoT are things and networks, in IoE there are five key concepts: things, networks, people, data and processes [2]. IoT and IoE have proved useful in a myriad of contexts and applications, such as healthcare applications, home automation and resource management, among many others more [3–6].

The fast growth of IoE and IoT is positive for the development of many applications, but, this growth also means facing a number of challenges in different domains [7], such as the heterogeneity of manufacturers and protocols, ubiquitous computing and a dependence on batteries in many cases. In this paper we focus on the cybersecurity of IoT systems, specifically on the detection of network attacks in IoT environments. As IoT is a subset of IoE, the ability to detect attacks in real time in IoT environments allows us to defend both IoT environments and improve the defenses of IoE environments.

It is essential to understand that solutions from other paradigms cannot always be directly applied in IoT environments, mainly due to the limitations of IoT devices.

✉ José Roldán-Gómez
jose.roldan@uclm.es

Jesús Martínez del Rincon
j.martinez-del-rincon@qub.ac.uk

Juan Boubeta-Puig
juan.boubeta@uca.es

José Luis Martínez
JoseLuis.Martinez@uclm.es

¹ Albacete Research Institute of Informatics (I3A), University of Castilla-La Mancha, Campus Universitario s/n, 02071 Albacete, Spain

² Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Belfast BT3 9DT, UK

³ UCASE Software Engineering Research Group, Department of Computer Science and Engineering, University of Cadiz, Avda. de la Universidad de Cadiz 10, 11519 Puerto Real, Cadiz, Spain

These limitations include: low computational capacity, limited bandwidth, low-cost sensors, and limited memory and battery capacity. If we add to this the increase in the use of this type of devices [8, 9], which has led to an increase in the number of cybercriminals who focus on this paradigm, it is understandable that researchers have had to adapt or design new solutions in different areas of security, such as cryptography [10] or reliability models [11]. Within the different areas of IoT cybersecurity, this work focuses on real-time IoT attack detection because the early detection of an attack can be vital in protecting the system. This is important to improve data protection in both IoT and IoE environments.

To detect network attacks in real time within IoT environments we need to meet two non-negotiable requirements. The first is that the system can be deployed in IoT environments with the limitations mentioned above, and the second is that the system must be able to process a large amount of data, thus allowing the system to be scalable and to work in networks of different sizes. Complex Event Processing (CEP) [12] is a technology that perfectly meets these requirements as it allows a large amount of data to be collected in the form of simple events. By means of rules defined by an expert, situations of interest can be extracted from these simple events, thus forming complex events. This functionality is ideal, for example, for detecting network attacks in real time. To this end, network packets are defined as simple events and the detected attacks are the resulting complex events. The successful deployment of CEP engines in IoT environments has been widely demonstrated [13–16]. Although CEP is very advantageous for real-time attack detection, it has one limitation, namely the need for a domain expert who is able to define the rules that must be followed to carry out such detection.

This work focuses on designing and implementing an architecture capable of generating CEP rules automatically, and without supervision, from historical data in order to detect and classify network attacks without the need for a domain expert. We apply unsupervised dimensionality reduction and clustering techniques to model normality using rules, and then apply anomalous data detection concepts to detect attacks as deviations from normality. In this way, effective and efficient rules can be generated without the need for labeled data in training.

To evaluate this proposal, a baseline scenario is deployed on MQTT and subjected to six different attacks. By using these attacks, we perform two different experiments based on evolving scenarios in which attacks are added iteratively. In the first experiment a new attack is carried out for the first time in each iteration, and if it is detected, it is then used to retrain and improve the model to be tested in the next iteration. In this way we can observe

how the system can detect anomalies and generate rules to detect them by adding new families. In the second experiment the attacks are distributed uniformly over the iterations, so that in each iteration both new attacks and new packets of already known attacks appear. This experiment is useful to check how existing rules evolve while generating new attack families in the same iteration.

The main contributions of this paper are the following:

- The integration of PCA with GMM and the Mahalanobis distance for the first time in a CEP engine, which allows us to generate CEP rules in an unsupervised manner.
- The generated CEP rules allow the CEP engine to detect attacks in IoT environments in real time.
- The use of dynamic tables makes it possible to generate new rules very easily without the need to modify the CEP application in real time.
- Our proposed framework is able, from an initial state in which it has only been exposed to normal traffic, to detect unseen attacks as anomalies and progressively and incrementally incorporate them in the rule set.
- The architecture has been successfully evaluated using an MQTT network use case using two different experimental scenarios, achieving an average F1 score >.9890%.

The remainder of this article is structured as follows. Section 2 describes the concepts necessary to understand the whole article, and Sect. 3 discusses related works, that is to say those that focus on generating CEP rules automatically. Then, Sect. 4 describes the design and implementation of the proposal, and the results are described and discussed in Sect. 5. Lastly, the conclusions and future work are presented in Sect. 6.

2 Background

This section introduces the key concepts of this article, namely MQTT (Message Queue Telemetry Transport) and CEP.

2.1 MQTT protocol

MQTT is a protocol that operates at the application layer and is supported by TCP/IP. It is oriented to network communication through a publisher/subscriber scheme using topics. In this way, devices (clients) that require information subscribe to the corresponding topic. The clients that generate this information publish in that topic. There is a central node called the broker that is in charge of orchestrating the behavior of the network, receiving the packets and forwarding them to the corresponding nodes.

This protocol is especially useful in IoT networks because it is particularly lightweight, which has made it very popular within the IoT paradigm [17].

Figure 1 shows the diagram of an MQTT network with 3 clients and 3 topics. Each shape represents a different topic, and in this way we can observe how the messages of these topics move. We can also see how the clients subscribe to the topics they need and receive the information associated with them. This information is produced or collected by other MQTT clients which are in charge of publishing it in that topic.

2.2 Complex event processing

CEP is a technology whose aim is to detect situations of interest by collecting and correlating events [18]. To achieve this, as a general rule, a domain expert defines CEP rules that allow the checking of specific situations in the event streams. Thus, when a rule is fulfilled, a complex event is generated that identifies a situation of interest.

More specifically, a CEP engine is a specific piece of software used to perform this type of data processing in real time. In our case, Siddhi CEP [19] is used.

The language used to define the rules in a CEP engine is called the Event Processing Language (EPL). There are many EPLs, and in our case SiddhiQL is the one provided by Siddhi CEP.

Simple events are the raw data received by the CEP engine. In the case of real-time network attack detection, these simple events will be the network packets. However, this may change depending on the context and the problem statement.

CEP rules are the patterns described and implemented by a domain expert. These CEP rules describe the situations of interest to be identified, and are written in a particular EPL, which this may vary depending on the CEP engine used. In this work Siddhi is used, and in our case each CEP rule can identify a family of attacks.

Complex events identify a situation of interest and are generated by CEP rules. Every time one of these rules is fulfilled, a complex event is generated. In our case, a complex event identifies that an attack of a particular family has been detected.

Figure 2 shows the three main stages in CEP technology. These stages are as follows:

- **Event capture:** This is the earliest stage and consists of the reception of the simple events to be analyzed and correlated by the CEP engine. These simple events vary according to the context. In this article, as discussed above, the simple events contain the network packet information.
- **Analysis:** The second stage is responsible for correlating simple events, so that it is possible to detect when a situation of interest arises. These situations of interest are identified when the simple events meet the

Fig. 1 MQTT network operation diagram

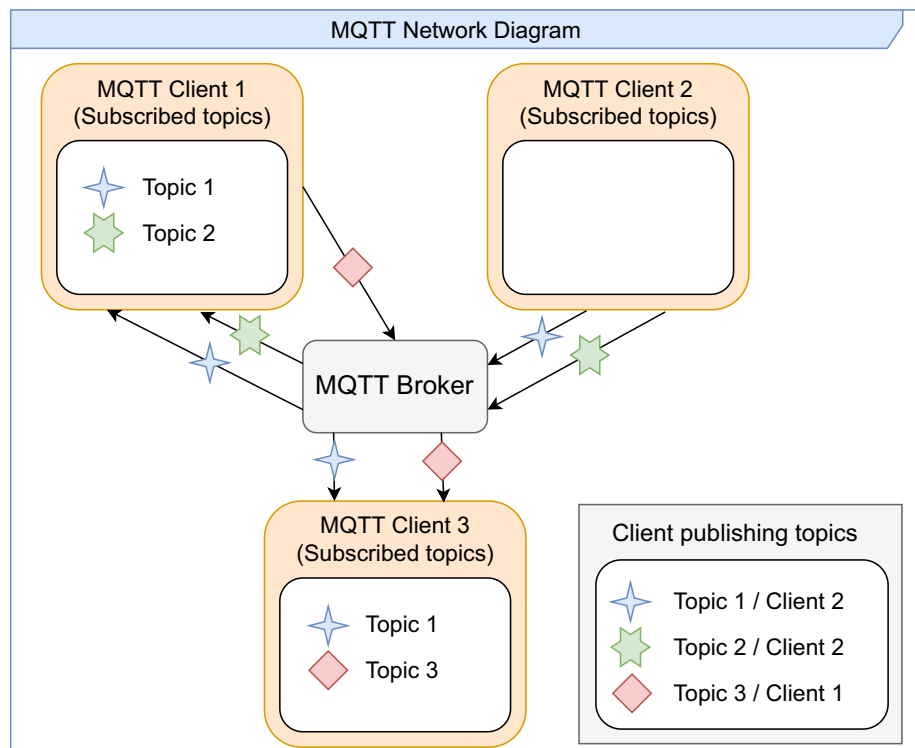
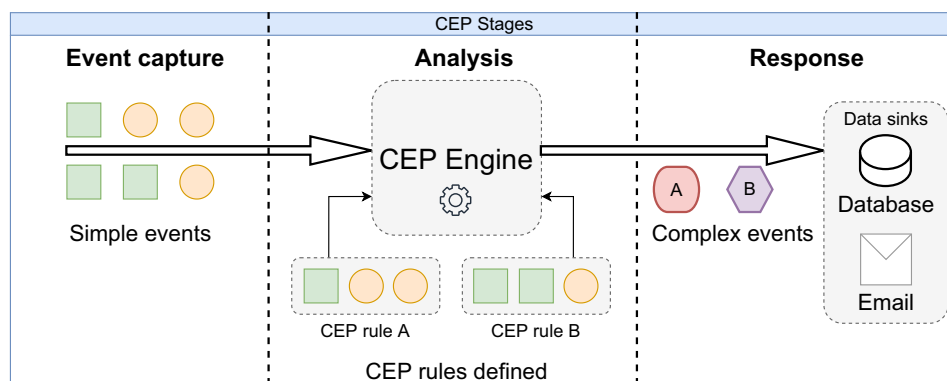


Fig. 2 CEP stages diagram



conditions defined in the CEP rules. When this occurs, a complex event is generated that represents the detected situation of interest.

- **Response:** The last stage consists of the actions to be taken once a situation of interest is detected, for example when an attack is detected, an email could be sent to the security manager. This is a phase that can be very heterogeneous depending on the context in which CEP operates.

3 Related work

There are several relevant works that address the problem of CEP rule generation from different perspectives. A detailed study of the different approaches is necessary in order to understand the intrinsic novelties of our approach.

For a better understanding of the different proposals, it is useful to classify them, and in this paper we will do so according to two criteria. The first criterion is the need to have prior rules for the generation of new CEP rules. The second criterion consists in the need to label the different events in the training data for the approach to learn, i.e. in a supervised or unsupervised way. Table 1 shows a comparison of all the papers analyzed in this section.

3.1 Supervised with prior rules

In this group we find proposals that require labeled training datasets and prior rules and aim to update existing patterns. This makes it possible to generate new rules that offer better results than the original ones. A work that fits in this category is the one proposed by Yunhao Sun et al. [20]. In this work, a historical set of training data and CEP rules are used. First, the authors apply a loss function, which is obtained from the error of the previous rule measurements with respect to the actual labelled results. A loss function and an activation function are then used to filter out rules that are considered bad on the basis of a manually defined

threshold. By using the remaining rules, a given set of support vectors is determined in order to build a coverage region for each class. Finally, updated rules are created by the projection of a regional boundary. This work is interesting despite addressing the problem of updating rather than generating new rules.

A different approach in this category is found in the work proposed by O-Joun-Lee and Jai E. Jung [21]. The authors propose the use of a history of rules defined by domain experts. Subsequently, by clustering the sequences that generate the simple events, the simple events and their relationships necessary to generate a complex event are determined. Finally, the rule is generated by means of a Markov transition probability model. One drawback of this approach is that it is highly dependent on the decision history generated by the domain experts. Furthermore, it is not possible to add new complex events if there is no data history where this complex event appears.

In the work proposed by Nathan Tri Luong et al. [22], CEP is used to preprocess the data, while Tensor Flow is used to implement an additional component that performs the training and classification of the different events. In this type of approach, CEP rules only perform the processes prior to training and classification. The limitation of this architecture is that the bottleneck can be transferred to the component in charge of performing the classifications. This results in not taking full advantage of the capacity of CEP engines to process a large amount of data.

3.2 Supervised without prior rules

In this group we find proposals that do not require prior rules, but label complex events based on historical data. One paper in this category is that of Bruns et al. [23], which succeeds in adapting the bat algorithm to the CEP rule search by structuring the different CEP operators, attribute values and time windows in the form of a tree. In this way the algorithm determines these values in the rule they represent. The results they obtain average an F1 score of .9923, and are achieved using an unusual algorithm in the

Table 1 Comparison of the works analyzed

References	Unsupervised	Need for prior rules	Metric	Novelty / Highlight
[20]	No	Yes	Accuracy=.90	Pre-filtering rules before training improves performance
[21]	No	Yes	–	Employ Clustering of simple event sequences
[22]	No	Yes	–	It uses CEP to perform data preprocessing
[23]	No	No	F1 score=.99	It uses Bat algorithm to generate new rules
[24]	No	No	Accuracy=.98	It uses PART algorithm to generate new rules
[25]	No	No	Accuracy=.93	It uses Evolutionary algorithm to generate new rules
[26]	No	No	F1 score=1	It compares the prediction of key features with their actual values
[27]	No	No	F1 score=.99	PCA allows rules to be generated with high performance
[14]	Yes	Yes	–	Defined CEP rules and pretrained neural networks will generate efficient CEP rules
[28]	Yes	No	F1 score=.89	It uses GRU and Furia to generate CEP rules in an unsupervised manner
[29]	Yes	No	Accuracy=.92	It use of reinforcement and active learning to mine and add new rules
[30]	Yes	No	F1 score=.96	It uses LSTM and decision tree used to detect anomalies
This work	Yes	No	F1 score=.98	PCA and GMM enable unsupervised generation of high-performance CEP rules

CEP context. The only limitation of the proposal, in addition to requiring labeling for training, is that it needs a definition of the complex events as a function of the simple events. This is not always easy without prior rules.

This is not the only proposal that employs tree diagrams to generate CEP rules. Another work that uses these diagrams is that of Mohammad Mehdi Naseri et al. [24]. In this paper the authors use the PART algorithm to generate CEP rules automatically in a supervised manner, and the CEP engine is deployed in a hospital to monitor different events. Its main limitation is the same as all the works in this category, namely that it needs a properly labeled dataset to work.

There is also a proposal based on evolutionary algorithms. The work presented by Jiayao Lv and Bihui Yu [25] uses evolutionary algorithms to generate CEP rules. In this case, the authors start from a history of simple events and the simple events generated by them. From these data, simple rules are generated which form part of the initial population of the evolutionary algorithm. The limitation of this work lies in the complexity of finding complex events associated with simple events without prior rules, which is not always possible.

Another work that manages to extract rules automatically without prior rules is the one proposed by Roldán-Gómez et al. [26]. In this case, the rules are constructed from the prediction of the value of the most important feature for a category. If the difference between the actual value and the prediction exceeds a threshold, this simple event does not correspond to a category. This proposal is able to detect all attacks, although the main limitation of

this work is the difficulty that may exist in generating certain rules based only on a key variable and an expected value.

A natural evolution of the above-mentioned work is that of Roldán-Gómez et al. [27]. In this work authors reduce the dimensions of individual events by using Principal Component Analysis (PCA), thereby achieving two goals. The first is to simply characterize the individual events, and the second is to drastically improve the performance of the CEP engine and the system network by reducing the dimension of the individual events. From the labels of the individual events, the averages of the reduced events are calculated. The rule consists of a Euclidean distance weighted by the weights of each component of the reduced event. This difference is compared with the sum of errors of each component weighted again with the weights of each component and with the standard deviation of each component. The results obtained from this study show an average F1 score of .9878, in addition to a reduction in the event size and the consequent improvement in the performance of the network and the CEP engine. A small limitation of this work is that it is a supervised way to calculate the rule for each category.

3.3 Unsupervised with prior rules

This group is the least common as it requires unsupervised training and the existence of rules capable of detecting items of interest. However, we can find works such as that of Ren et al. [14]. This one focuses on optimizing performance in IoT environments, which is the main

differentiating factor with respect to other proposals. To achieve this goal, a micro CEP engine and a model based on Tensorflow Lite Micro with pre-trained neural networks are used. These neural networks (either supervised or unsupervised, such as autoencoders) can be updated to adapt to the changing behavior of a real system. The main difference of this proposal with respect to the others analyzed is that the output of these neural networks feeds the CEP engine, which has manually defined rules. It may seem that this proposal does not fall within the scope of automatic CEP rule generation. However, it is possible to generate simple rules that detect the output of neural networks. The main limitation of this proposal is that the CEP rules are defined manually, unlike with our proposal, in which they are generated automatically.

3.4 Unsupervised without prior rules

In this group we find proposals that do not require prior rules or labels on the data. Some works mainly focus on labeling simple events and then use known rule extraction algorithms. The work by Simsek et al. [28] performs a study using different classifiers to label simple events, and uses the most common algorithms for rule extraction. Their conclusions show that Gated Recurrent Unit (GRU) together with the FURIA algorithm obtain the best results in their experiments. The value of this work lies in the comparison made with different algorithms. The fundamental disadvantages of this approach lie in the large amount of data required for deep learning models and the computational cost involved.

Another work has been proposed in [29] on the use of reinforcement and active learning to mine and add new rules that were previously unknown. This approach, however, requires a human in the loop to confirm the adequacy of the added patterns.

There are also works within this category that attempt to generate CEP rules for unsupervised anomaly detection. As an example, the work by Liu et al. [30] proposes the use of LSTM neural networks with an attention mechanism to detect anomalies based on the model, and this also allows them to calculate a threshold for anomalies. Subsequently, to generate the CEP rules, the authors use a decision tree algorithm. The limitations of this work are that it focuses exclusively on anomaly detection, and the use of neural networks, with the consequences discussed above.

Our proposal would fall into this category. The novelty is that we achieve an unsupervised method without the need for prior rules while performing event dimension reduction, and this improves the computational performance. In addition, our proposal is able to work correctly while training with few samples, and this is an advantage over proposals based on deep learning. Finally, the

implementation used facilitates the creation and updating of new rules in a changing system.

4 Proposed architecture

This section describes the architecture for recognizing real-time IoT attack patterns. Figure 3 shows a graphical scheme of the architecture. Our proposal focuses on the automatic CEP rule generator, and the training data is obtained from the IoT network. As discussed above these packets are not labeled and feed the CEP rule generator.

The CEP rule generator is composed of four phases. First, after preprocessing, comes the PCA phase, which is responsible for generating the PCA model and reducing the dimensionality of network traffic. Next comes the GMM phase, which is in charge of performing the clustering process in order to obtain the different families of packets. Since it is necessary to establish a threshold for each family to differentiate them from anomalous traffic and/or other families, this is performed in the Threshold phase. Finally, the Sending phase sends the rule parameters to the CEP engine.

These phases are discussed in detail below.

4.1 Preprocessing

Before the first phase, it is necessary to preprocess the data so that it can be consumed by the PCA model for training. In our case we perform the following steps in the preprocessing:

- Filling of empty fields. The existence of different protocols results in certain characteristics that are not present in all network packets. PCA does not support these empty values, so it is necessary to fill them in. In our case these fields are filled with value “-1”. This is because there are no negative values in the features, and in this way we emphasize this empty feature.
- Categorization of non-numerical features. Non-numerical features that are represented by text or another type of label do not allow training a PCA model. To solve this problem a one-hot encoding scheme is used. This allows each category to be identified as a binary feature.
- Scaling of values. PCA is conditioned by the scales of the features. This means that variables with very high values have more weight in the model. To solve this problem, we use a min-max scaler. This allows us to equalize the scales of the different features.

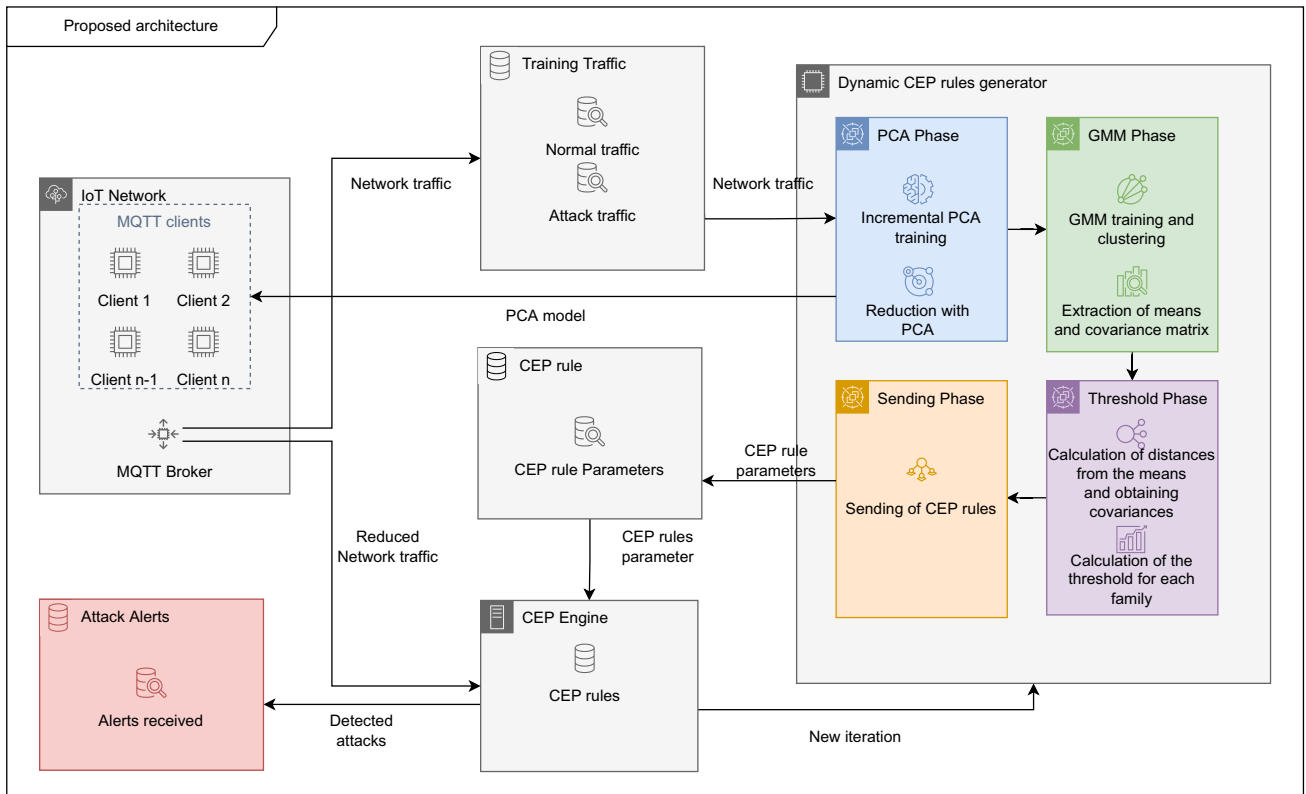


Fig. 3 Diagram of the proposed architecture to detect IoT attacks in real time

4.2 PCA phase

This phase is responsible for generating (or updating if it is not the first generation) the PCA model using the input traffic. PCA is a statistical method whose objective is to reduce the complexity of a sample space by reducing the dimensions of that space. Thus, if we have an element $x \in \mathbb{R}^n$ represented by n variables, the objective is to find a representation with m variables where $m \ll n$. These new variables are obtained by linear combinations of the original ones. Each new variable is known as a component and each component is linearly independent of the other components. The goal of PCA is to maximize the amount of information represented by each component. Thus, if an element $x \in X$ in a given dataset X is composed of the vector of variables $x = \{x_1, x_2, \dots, x_n\}$, the new variables of the vector $x' = \{x'_1, x'_2, \dots, x'_m\}$ with $x' \in X'$ will have the representation shown in Eq. 1:

$$X' = X \cdot W \tag{1}$$

where W is an n -by- m matrix of weights whose columns are the first m eigenvectors of $X^T \cdot X$, ordered according to their eigenvalues. An advantage of this model is the ease of converting an element from the original space to the reduced one when we have the PCA model trained.

Each resulting component collects an amount of information, with this amount being called the explained variance ratio rv . The first components always have a higher rv than the last ones. In an ideal and perfectly linear scenario, the sum of the explained variance ratios of all the components could be 1. In practice we seek to approximate this as closely as possible while keeping the dimension reduction as high as possible.

To implement our proposal we use incremental PCA [31]. This version allow us to recalculate the model, i.e., the new eigenvectors W_{n+1} if new data are added, using the existing eigenvectors W_n with their corresponding eigenvalues and covariance matrix of the current PCA model, plus the new samples X_{n+1} . In this way it is not necessary to generate a new model from scratch, and/or to store the previous samples X_n in memory, if new training data arrives. Instead, it is possible to obtain an estimate of the $n + 1$ iteration using the eigenvectors and eigenvalues of n . This makes it possible to obtain new PCA models incrementally from already trained PCA models. The advantage of this is that we do not have to train the PCA model from scratch in each iteration, thus achieving a lighter training in new iterations.

Once the trained PCA model is obtained, it is sent to the IoT network broker. This model is also used to reduce the input traffic. We also extract the variance ratios explained in each component, and we obtain the diagonal matrix of them that will

be further used for thresholding purposes in Sect. 4.4. This reduction is necessary for the following stages.

4.3 GMM phase

Once we have reduced the dimensionality of the traffic, Gaussian Mixture Models (GMM) are used to cluster the traffic into different families.

GMM is a probabilistic model that assumes that for a data set X there are K normal distributions representing all C categories present in the data, within which all X elements are found. The goal of GMM is to find the best combination of the parameters for the K normal distributions. In this way we can group the elements into K different families or groupings.

$$p(x_i) = \sum_{k=1}^K p(x_i|c_k)p(c_k) \quad (2)$$

Equation 2 describes the probability of element $x_i \in X$ as the sum of composite probabilities it has of belonging to each family, such that $p(x_i) = 1$. This means that GMM assumes that all elements lie within these distributions, as discussed above.

$$p(x_i) = \sum_{k=1}^K \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k) \quad (3)$$

Eq. 3 represents the GMM model as a linear combination of the K normal distributions, where π_k is the mixing coefficient for each distribution and provides an estimate for each of the normal distributions.

The term $\mathcal{N}(x|\mu_k, \Sigma_k)$ is called the mixture model component, which models and describes each of the normal distributions, where μ_k is the mean and Σ_k is the covariance.

The main advantage of GMM is that it allows some flexibility in each category, so that 2 normal distributions can be very different, and it does not have a bias for circular groups and works well even in certain non-linear distributions [32].

In this case a variational version of the algorithm is used [33], which allows us to infer an optimal number of normal distributions. The objective of using this version is not to have to indicate the number of K families a priori, thus allowing the process to be completely unsupervised, since we do not need to know a priori how many families or groupings make up the normal traffic or how many different types of attacks we may be exposed to.

In conclusion, GMM allows us to generate families without the need to label the training data previously, where each family is defined by its mean μ_k and covariance matrix Σ_k .

GMM has to be recalculated with training data from previous iterations on the new PCA model [33]. This is because each iteration modifies the PCA model, causing the original distributions to be useless in the new model.

4.4 Threshold phase

At this stage, the threshold is calculated for each family k using the Mahalanobis distance. The Mahalanobis distance is a distance function that takes into account the covariance matrix in order to weight it [34]. The fundamental advantage of the Mahalanobis distance is that it takes into account the scale differences that may exist between the different variables and families as well as the correlation that may exist between variables.

In this proposal we use the Mahalanobis distance to see the difference of each element reduced by PCA with respect to the categories previously obtained with GMM.

$$d(x, \{\mu_k, \Sigma_k\}) = \sqrt{(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)} \quad (4)$$

Equation 4 describes how the difference between the element x and the mean of a category μ_k is calculated. Σ^{-1} represents the inverse covariance matrix. Its inclusion in the distance equation implies a weighting of such a distance function, so that families with smaller covariances (more compact families) result in larger distances in relative terms with regard to more sparse families.

In our particular case, we apply the Mahalanobis distance to the reduced elements resulting from PCA. To account for the differences in the explained variance ratios of the different PCA components, we improve the distance function by using the ratios as weights, as indicated in Eq. 5.

$$d(x, \{\mu, \Sigma\}) = \sqrt{(x - \mu)^T (\Sigma^{-1} \times VE) (x - \mu)} \quad (5)$$

In this way, our distance function will give more weight to the components with a higher rv . The first step is to obtain the VE matrix as the diagonal matrix with the explained variance ratios of each component. Equation 6 shows how the matrix that we use to weight the explained variance ratios is obtained.

$$VE = \text{diag}(rv_1, rv_2, \dots, rv_m) \quad (6)$$

By using Eq. 5, each element is compared with the mean of each family. Once we have all the distances, we can calculate the threshold for that family by using the farthest element of the family with respect to the mean and the closest non-family element with respect to the family mean. With these distances we calculate the midpoint, which defines the threshold for that category k .

$$d_{max} = \max \{d(x, \{\mu_k, \Sigma_k\})\}, \forall x \in k \quad (7)$$

$$d_{min} = \min \{d(x, \{\mu_k, \Sigma_k\})\}, \forall x \notin k \quad (8)$$

$$Th_k = (d_{max} - d_{min})/2 \quad (9)$$

Equations 7 and 8 define how to obtain the element farthest from the mean of a family k and the closest one outside the

Table 2 Features description table

Feature name	Feature type	Feature description
PacketLength	Integer	Determines the size in bytes of the network packet
Info	Text type	Provides information about the packet. This information varies depending on the protocol
SourceMac	Text type	Identifies the MAC address of the sending device
DestMac	Text type	identifies the MAC address of the receiving device
SourceIp	Text type	Identifies the IP address of the sending device
destIp	Text type	Identifies the IP address of the receiving device
tcpSourcePort	Text type	Identifies the TCP port of the sending device
tcpDestPort	Text type	Identifies the TCP port of the receiving device
udpSourcePort	Text type	Identifies the UDP port of the sending device
udpDestPort	Text type	Identifies the UDP port of the receiving device
tcpFlags	Text type	Identifies the TCP flags enabled on that packet
tcpWindowSize	Integer	Identifies the TCP window size in bytes
mqtFlags	Text type	Identifies the MQTT flags enabled on that packet
mqtMessage	Text type	Identifies the MQTT message on that packet
mqtTopic	Text type	Identifies the MQTT topic on that packet
mqtMessageLength	Integer	Identifies the MQTT message size in bytes
mqtFrameCount	0–1 Integer	Identifies how many packets the original message has been fragmented into
Delay	Float	Identifies the milliseconds of separation that exist with respect to the reception of the previous packet

family k , respectively. With these elements, obtaining the threshold is simple, as we can see in Eq. 9.

4.5 Sending phase

At this stage the rule parameters are sent to the CEP engine. The parameters sent for each rule are the numerical identifier of the rule, the iteration number, the covariance matrix of the PCA model, the threshold for the specific rule family and the mean of each component of that specific family. The Siddhi code is sent the first time, but it is not necessary to send it again in the following iterations. This allows us to generate dynamic CEP rules, which is a very useful novelty of our proposal.

Regarding the operation of dynamic CEP rules, when the CEP rule generator generates new rules, it is not necessary to generate a new Siddhi file, which is used to generate an application in the Siddhi engine. Instead, it makes use of dynamic tables containing the parameters of the current rules. This reduces the network data transfer when updating or generating new rules, and greatly facilitates the implementation, creation and updating.

Once in operation, the broker reduces the packets with PCA and sends them to the CEP engine. With these reduced packets, the distance of the same packet with respect to the average of each family is calculated with Eq. 5, and if this distance is less than the threshold of that family, the packet is considered to belong to that family. If

a packet does not fall within the threshold of any family, that packet is considered to be an anomaly.

The Siddhi application can be seen in Listing 1. There are 3 input streams, which can be identified with the directive *source*. The first one, called *ReducedEvent*, is used to receive the simple events previously reduced with the PCA model. The second, defined as *ClearEvent*, is used to clear the parameters of a particular iteration. The third, named *ThresholdParameters*, is used to add the parameters of a new iteration to the parameter table. The *MeanDiffEvent* and *ComputedMeanDiffEvent* streams are intermediate streams used to store the difference from the mean and the difference from the weighted mean, respectively. *DetectedEvent* stores the events detected by the rules. The implementation of Eq. 5 is carried out in the last three code blocks. Although they can be unified in a single block, this would worsen their readability.

The great advantage of this implementation is that creating or updating rules is simply a matter of updating the table because the structure is maintained. This, coupled with the unsupervised operation of the proposal, offers a solution that can be deployed without the need for a domain expert.

We can also observe that the CEP engine can request new iterations from the rule generator. In our experiments these new iterations are defined by the training datasets, and this allows us to generate reproducible experiments. In a real deployment, new iterations could be initiated when a certain number of anomalies are obtained, or when a specific time elapses. This will depend on the type of network and applications Table 2.

Listing 1 Siddhi application for real-time IoT attack detection

```

App:name("DynamicPCAIcremental")

@App:description("Dynamic PCA Test")

@source(type='mqtt', url='tcp://172.18.0.4:1883', topic = 'ReducedEvent',
@map(type = 'json'))
define stream ReducedEvent(c1 double, c2 double, c3 double, c4 double);

define stream MeanDiffEvent(idFam int, d1 double, d2 double, d3 double, d4 double,
c1 double, c2 double, c3 double, c4 double);

define stream ComputedMeanDiffEvent(idFam int, d1 double, d2 double, d3 double,
d4 double, cd1 double, cd2 double, cd3 double, cd4 double, c1 double, c2 double,
c3 double, c4 double);

@source(type='mqtt', url='tcp://172.18.0.4:1883', topic = 'ClearEvent',
@map(type = 'json'))
define stream ClearEvent(iterationNumber int);

define stream DetectedEvent(iteration int, idFam int, c1 double, c2 double,
c3 double, c4 double);

@sink(type='log')
define stream CountEvent(iteration int, idFam int, number long);

@source(type='mqtt', url='tcp://172.18.0.4:1883', topic = 'ParameterTable',
@map(type = 'json'))
define stream ThresholdParameters(idRule int, iteration int, m1 double, m2 double,
m3 double, m4 double, threshold double, x00 double, x01 double, x02 double,
x03 double, x10 double, x11 double, x12 double, x13 double, x20 double, x21 double,
x22 double, x23 double, x30 double, x31 double, x32 double, x33 double);

@primaryKey('idRule')
@index('idRule')
define table ParametersTable(idRule int, iteration int, m1 double, m2 double,
m3 double, m4 double, threshold double, x00 double, x01 double, x02 double,
x03 double, x10 double, x11 double, x12 double, x13 double, x20 double,
x21 double, x22 double, x23 double, x30 double, x31 double, x32 double, x33 double);

from ThresholdParameters
select *
insert into ParametersTable;

from ClearEvent
delete ParametersTable
on iterationNumber==ParametersTable.iteration;

from DetectedEvent
select iteration as iteration, idFam as idFam, count() as number
group by idFam, iteration
insert into CountEvent

from ReducedEvent as re left outer join ParametersTable as pt
select pt.idRule as idFam, re.c1-pt.m1 as d1, re.c2-pt.m2 as d2, re.c3-pt.m3 as d3,
re.c4-pt.m4 as d4, re.c1 as c1, re.c2 as c2, re.c3 as c3, re.c4 as c4
insert into MeanDiffEvent;

from MeanDiffEvent as md join ParametersTable as pt
on md.idFam==pt.idRule
select md.idFam, md.d1, md.d2, md.d3, md.d4,
((md.d1*pt.x00)+(md.d2*pt.x10)+(md.d3*pt.x20)+(md.d4*pt.x30)) as cd1,
((md.d1*pt.x01)+(md.d2*pt.x11)+(md.d3*pt.x21)+(md.d4*pt.x31)) as cd2,
((md.d1*pt.x02)+(md.d2*pt.x12)+(md.d3*pt.x22)+(md.d4*pt.x32)) as cd3,
((md.d1*pt.x03)+(md.d2*pt.x13)+(md.d3*pt.x23)+(md.d4*pt.x33)) as cd4,
md.c1 as c1, md.c2 as c2, md.c3 as c3, md.c4 as c4
insert into ComputedMeanDiffEvent;

from ComputedMeanDiffEvent as cm join ParametersTable as pt
on cm.idFam==pt.idRule
select pt.iteration, cm.idFam, cm.c1, cm.c2, cm.c3, cm.c4
having math:sqrt((cm.d1*cm.cd1)+(cm.d2*cm.cd2)+(cm.d3*cm.cd3)+(cm.d4*cm.cd4))
<pt.threshold
insert into DetectedEvent;

```

5 Experiments and results

This section describes the experiments performed, and the results are analyzed and discussed.

The scenario we propose is an MQTT network with three legitimate clients and a broker. The clients generate numerical data and send it to the broker, and this allows us to simulate a temperature sending scenario. To demonstrate that our proposal can correctly detect, different attacks, the following attacks were carried out:

- *Subscription fuzzing*: This attack consists of trying to subscribe to different topics. This can be used when the attacker has access to an MQTT system.
- *Disconnection wave*: This consists of spoofing the *id* of the MQTT protocol and launching the disconnect command. If it is not configured correctly, it is possible to steal the *id* of the legitimate device and expel it from the system. The goal of this attack is to disconnect all the devices from the system.
- *TCP syn scan*: This is the classic scanner used to check which TCP ports are open. The attacker starts with a SYN packet, and if they receive a SYN/ACK, they assume the port is open. If they receive an RST they assume it is closed.
- *UDP scan*: This involves sending UDP packets to each port to be scanned. If a UDP response is received, the port is considered open, and if no response is received, the position is open or filtered. A packet of type ICMP *port unreachable error* means that the port is closed, and any other type of ICMP error means that the port is filtered.
- *Xmas scan*: This is a rather unusual scanner nowadays, but we use it in the scenario because it is different from the UDP and TCP SYN scanner. It involves sending a packet to each TCP port with the FIN, PSH and URG flags set to 1. If no response is received, the port is considered open or filtered, and if an RST is received, it is considered a closed port. If any ICMP packet is received with *unreachable error*, it is considered a filtered port.

- *Telnet connection*: This involves packets that try to connect via Telnet with different users and passwords, to simulate the first stage of Mirai. The idea is to test the proposal in a very common scenario [35].

The training and testing datasets are generated by collecting the normal packets and the attacks. The dataset is accessible from the following repository <https://data.mendeley.com/datasets/pzhm3jnw6w/draft?a=1565272f-bc8b-4eac-a566-11ec45124a44> [36]. The distribution of the dataset can be seen in Table 3. The percentage of packets with which the model is trained for each attack is inversely proportional to that produced by such an attack, i.e. attacks that produce little number package of packages are used in a bigger proportion for training. This distribution has been chosen because it allows enough packages for the model to learn new rules and therefore evaluate the model's ability to generate new rules for detecting unknown attacks, while also detect them as early as possible once a minimum number of samples have been collected. If the experiments are successful, the detector's ability to generate rules with few attack samples will also be demonstrated. Modifying the distributions of the training datasets, a priori, should modify the generated CEP rules, however in these experiments, where a realistic scenario has been used, it has not been necessary to use additional techniques to augment the datasets or to balance them. This leads us to check that the architecture is robust to these details. Each event is considered a separate attack in our experiments so that we can measure the effectiveness of the CEP rules more accurately. This dataset has been used because it contains MQTT attacks of different types. To the best of our knowledge, there is no other public attack dataset containing a large variety of attacks and in sufficient number to allow incremental detection and addition of new rules to the CEP engine. Moreover, subscription fuzzing attacks or disconnection wave type attacks are not available in any existing dataset.

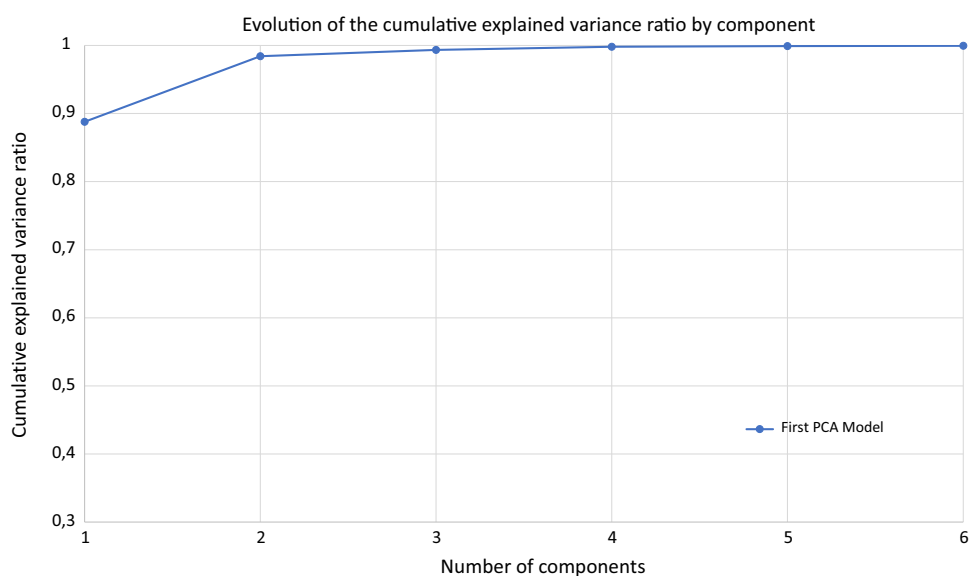
Table 2 shows a description of the dataset features and the data type that represents them.

Two different experimentation scenarios were generated. In both experiments, PCA models with $m = 4$ were generated, which means that 4 components are used. To

Table 3 Distribution of the dataset used

Traffic type	Training packets/events	Testing packets/events
Normal Traffic	7936 (50%)	7936 (50%)
Subscription Fuzzing	3277 (80%)	820 (20%)
Disconnection Wave	3000 (15%)	17000 (85%)
TCP Syn Scan	901 (90%)	101 (10%)
UDP Port Scan	530 (90%)	59 (10%)
Telnet	452 (90%)	51 (10%)
Xmas Scan	900 (90%)	100 (10%)

Fig. 4 Evolution of the cumulative explained variance ratio by component



choose the number of components of the PCA model, we rely on the first PCA model, which is generated with nonmalicious traffic. This is the most realistic approach, as in a real situation the attacks are not always available for the deployment of the threat detection system. Thus, the analysis is performed with the first model because it is the one we know a priori.

Figure 4 shows the ratio of variance accumulated by the different components of the first model and shows how a value $m=4$ preserves most of the variance present in the data. The graph shows how four components preserves most of the information present in the data, and values of $m>4$ contain negligible increments.

5.1 Experimental scenario 1: Detecting new attacks

The first scenario seeks to demonstrate that the proposed architecture is capable of detecting new attacks in an unsupervised and incremental manner. The experiments in this first scenario are performed in several iterations. In the first iteration we only train with normal packets, since this would be the usual case when the architecture is deployed for the first time. However, note that it is possible for our framework to carry out the first training with attack packets without any problem. From the first iteration onwards, new attacks are introduced in each iteration and the model is retrained with the packets that have not been classified as belonging to any of the existing GMM families (i.e., their distance to all families is larger than the learned thresholds) by any previous rule. Note also that the predictions made by our system are used in the following iterations for retraining and not the real groups, in order to preserve the

unsupervised setting and not to require annotated ground-truth by human experts. This means that high misclassification could potentially lead to contamination of the existing family models or the creation of incorrect rules if the performance of the system was poor. This is applicable to both experiments.

An important detail to take into account is that the first time an attack is detected, it will not be included in any CEP rule because it is an anomaly. With the subsequent training of the model with the new data, the new CEP rule will be generated. Table 4 shows the input of the different attacks in each iteration. Each row represents one type of traffic and each column one iteration of the experiment. The character X represents the testing dataset, and the character A represents the training dataset, which is an anomaly in that particular iteration.

The following conventional metrics are used to evaluate the results of these experiments:

Table 4 Data input at each iteration in experiment 1

Traffic type	Iteration number						
	1	2	3	4	5	6	7
Normal traffic	X	X	X	X	X	X	X
Subscription Fuzzing	A	X	X	X	X	X	X
Disconnection Wave		A	X	X	X	X	X
TCP Syn Scan			A	X	X	X	X
UDP Port Scan				A	X	X	X
Telnet					A	X	X
Xmas Scan						A	X

- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1Score = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$
- $MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}}$

Where TP are the true positives, FP are the false positives, FN are the false negatives and TN are the true negatives. In the context of this work we will evaluate each CEP rule independently, thus obtaining results for each of the CEP rules, since we are interested in detecting if the attacks are successful or not rather than classifying the type of attacks. This means that a TP indicates that a packet belonging to the rule's category has been correctly identified by the rule. A FP identifies a packet that has been detected by that rule, but does not belong to the family of that rule, i.e. a false positive. A TN indicates a packet that has not been detected by the rule and does not belong to the category modeled by that rule, i.e. it has been correctly classified. Finally, an FN indicates a packet not detected by the rule, but which does belong to the family modeled by the rule. This means that the sum of TP and TN identifies the total hits of the CEP rule, and the sum of FP and FN identifies its failures. In this way, each rule can be treated as a binary classifier from which we extract the confusion matrix, which makes it easy to obtain the accuracy, recall, F1 score and MCC metrics that allow us to understand the performance of these classifiers from different perspectives Tables 5 and 6.

Thus, a high *Recall* score means that a CEP rule detects many events that actually belong to that family, a high *Precision* score means that that CEP rule does not detect many false positives, and the *F1 Score* makes use of the two scores in order to obtain a balanced metric between the two.

In a security context where not detecting an attack can have a very negative impact on the system, it is advisable to maintain a very high recall value. However, if the precision is very low, the system will return many false positives. This complicates the task of the security managers, who have to decide whether the positive is real later on.

Finally, Matthew's correlation coefficient (MCC) is applied [37]. It is particularly robust when dealing with unbalanced datasets. This metric is bounded between -1 and 1, where 1 is a perfect classifier, 0 is a random

Table 6 Data input at each iteration in experiment 2

Traffic type	Iteration number								
	1	2	3	4	5	6	7	8	9
Normal traffic	X ₁	X ₂	X ₃	X ₃	X ₃	X ₃	X ₃	X ₃	X ₃
Subscription fuzzing	A	X ₁	X ₂	X ₃	X ₃	X ₃	X ₃	X ₃	X ₃
Disconnection wave		A	X ₁	X ₂	X ₃	X ₃	X ₃	X ₃	X ₃
TCP syn scan			A	X ₁	X ₂	X ₃	X ₃	X ₃	X ₃
UDP port scan				A	X ₁	X ₂	X ₃	X ₃	X ₃
Telnet					A	X ₁	X ₂	X ₃	X ₃
Xmas scan						A	X ₁	X ₂	X ₃

classifier and -1 is a classifier that classifies everything wrong. MCC weights true positives and true negatives equally, and also weights false positives and false negatives equally. This allows the evaluation to be fair in categories with more items and fewer items.

Table 7 shows the results of the first experiment scenario. We can observe how the proposal behaves when new attacks appear and how it converges to a system capable of detecting and correctly classifying the packets of the different attacks. An important detail that we can verify in the fifth iteration is that a single rule is generated to detect UDP and TCP scans. This is because they present very similar behavior and characterization. This unification of the two attacks demonstrates the capacity of the rules generated to classify attacks based on their behavior. An important detail which highlights the system's ability to detect anomalies, is that the first time a new attack is deployed, there are no CEP rules for that attack.

Good results are obtained for all the types of packets handled. Among these good results, we can observe that threat detection shows a slight drop in performance compared with detection with specific CEP rules. We also note that the disconnection wave attack obtains excellent results. This may be due to the fact that it has very homogeneous packets. The rest of the attacks produce very similar results, with the exceptions that we have mentioned above. In contrast, we find that the normal traffic, although it also obtains very good results, is slightly worse than in

Table 5 Performance of CEP rules with and without PCA

Attacks / CEP rules performance	Average throughput no PCA	Average throughput PCA
Subscription fuzzing	4575.40	7645,30
Disconnection wave	4242.20	7747,50
TCP SYN scan	4142.60	7604,60
UDP scan	4157	7539,20
XMAS scan	4534.20	7831,10
Telnet (Mirai first stage simulation)	4574.80	7802,10

Table 7 Results of the first experiment scenario

Iteration number	Traffic type	TP	FP	TN	FN	Precision	Recall	F1 score	MCC
1	Normal traffic	7780	0	3277	156	1	.9893	.9900	.9674
1	Subscription fuzzing (Anomaly)	3277	156	7780	0	.9545	1	.9767	.9674
2	Normal traffic	7936	0	3820	0	1	1	1	1
2	Subscription fuzzing	819	0	10936	1	1	.9987	.9993	.9993
2	Disconnection wave (Anomaly)	3000	1	8755	0	.9996	1	.9998	.9998
3	Normal traffic	7780	0	18721	156	1	.9803	.9900	.9860
3	Subscription fuzzing	819	0	25837	1	1	.9987	.9993	.9994
3	Disconnection wave	16999	0	9657	1	1	.9999	.9999	.9999
3	TCP syn scan (Anomaly)	901	158	25598	0	.8508	1	.9193	.9195
4	Normal traffic	7780	0	18451	156	1	.9803	.9900	.9859
4	Subscription fuzzing	819	0	25567	1	1	.9987	.9993	.9993
4	Disconnection wave	16999	0	9387	1	1	.9999	.9999	.9999
4	TCP syn scan	101	1	26285	0	.9901	1	.9950	.9950
4	UDP port scan (Anomaly)	530	157	25700	0	.7714	1	.8709	.8756
5	Normal traffic	7780	0	18432	156	1	.9803	.9900	.9859
5	Subscription fuzzing	819	0	25548	1	1	.9987	.9993	.9993
5	Disconnection wave	16999	0	9368	1	1	.9999	.9999	.9999
5	TCP syn + UDP port scan	160	2	26206	0	.9876	1	.9937	.9937
5	Telnet (Anomaly)	451	157	25759	1	.7417	.9977	.8509	.8576
6	Normal traffic	7780	0	18931	156	1	.9803	.9900	.9859
6	Subscription fuzzing	819	0	26047	1	1	.9987	.9993	.9993
6	Disconnection wave	16999	0	9867	1	1	.9999	.9999	.9999
6	TCP syn + UDP port scan	160	1	26706	0	.9937	1	.9968	.9968
6	Telnet	51	0	26816	0	1	1	1	.9903
6	Xmas scan (Anomaly)	900	157	25810	0	.8514	1	.9197	.9199
7	Normal traffic	7780	0	18131	156	1	.9803	.9900	.9858
7	Subscription fuzzing	819	0	25247	1	1	.9987	.9993	.9993
7	Disconnection wave	16999	0	9067	1	1	.9999	.9999	.9999
7	TCP syn + UDP port scan	160	1	25906	0	.9937	1	.9968	.9968
7	Telnet	51	0	26016	0	1	1	1	.9900
7	Xmas scan	100	0	25967	0	1	1	1	1

the different attacks. This is due to the fact that within the dataset of normal packets there are some very circumstantial packets, which means that there are few of them and GMM does not assign them their own family. Good results are also obtained with attacks for which there is little representation in the model. For example, Xmas scan has a training representation of 900 samples and obtains a perfect result, with an MCC of 1. Something similar happens with other attacks such as Telnet, so we can conclude that the model is able to find new attacks with few samples of them, although if their behavior is similar it tends to group them together, as it does with UDP scan and TCP scan.

Table 5 shows the performance measured in throughput. This metric allows us to know the number of events per

second that the CEP engine is able to process when deploying rules with CEP and when doing so with classic ones as well. As it can be seen, the improvement when using the rules generated by our system is quite significant. This is probably due to the reduction in the size of the simple events, which also reduces the amount of information sent over the network. On average, the size of a simple event in a network without PCA is 418.7 bytes, while with PCA with 4 components it is 75.3 bytes. This evidences the performance improvement of our proposal, which is a relevant factor when deploying it in IoT environments.

Table 8 Results of the second experiment scenario

Iteration number	Traffic type	TP	FP	TN	FN	Precision	Recall	F1 score	MCC
1	Normal traffic (Test 1)	2593	0	3277	52	1	.9803	.9900	.9823
1	Subscription fuzzing (Training)	3277	52	2593	0	.9843	1	.9921	.9823
2	Normal traffic (Test 2)	2592	0	3273	53	1	.9799	.9898	.9820
2	Subscription fuzzing (Test 1)	273	0	5645	0	1	1	1	1
2	Disconnection wave (Training)	3000	53	2865	0	.9826	1	.9912	.9822
3	Normal traffic (Test 3)	2594	0	6840	51	1	.9807	.9902	.9866
3	Subscription fuzzing (Test 2)	273	0	9212	0	1	1	1	1
3	Disconnection wave (Test 1)	5665	0	3819	1	1	.9998	.9999	.9997
3	TCP SYN scan (Training)	901	52	8532	0	.9454	1	.9719	.9693
4	Normal traffic (Test 3)	2594	0	6502	51	1	.9807	.9902	.9864
4	Subscription fuzzing (Test 3)	273	0	8874	0	1	1	1	1
4	Disconnection wave (Test 2)	5665	0	3481	1	1	.9998	.9999	.9997
4	TCP SYN scan (Test 1)	33	0	9114	0	1	1	1	1
4	UDP port scan (Training)	530	52	8565	0	.9106	1	.9532	.9513
5	Normal traffic (Test 3)	2594	0	6443	51	1	.9807	.9902	.9864
5	Subscription fuzzing (Test 3)	273	0	8815	0	1	1	1	1
5	Disconnection wave (Test 3)	5665	0	3422	1	1	.9998	.9999	.9997
5	TCP SYN scan (Test 2)+UDP Port Scan (Test 1)	52	0	9036	0	1	1	1	1
5	Telnet (Training)	452	52	8584	0	.8968	1	.9456	.9441
6	Normal traffic (Test 3)	2594	0	6908	51	1	.9807	.9902	.9866
6	Subscription fuzzing (Test 3)	273	0	9280	0	1	1	1	1
6	Disconnection wave (Test 3)	5665	0	3887	1	1	.9998	.9999	.9997
6	TCP SYN scan (Test 3)+UDP Port Scan (Test 2)	52	0	9501	0	1	1	1	1
6	Telnet (Test 1)	17	0	9536	0	1	1	1	1
6	Xmas scan (Training)	900	52	8601	0	.9453	1	.9719	.9693
7	Normal traffic (Test 3)	2594	0	6041	51	1	.9807	.9902	.9861
7	Subscription fuzzing (Test 3)	273	0	8413	0	1	1	1	1
7	Disconnection wave (Test 3)	5665	0	3020	1	1	.9998	.9999	.9997
7	TCP SYN scan (Test 3)+UDP Port Scan (Test 3)	52	0	8634	0	1	1	1	1
7	Telnet (Test 2)	17	0	8669	0	1	1	1	1
7	Xmas scan (Test1)	33	0	8653	0	1	1	1	1
8	Normal traffic (Test 3)	2594	0	6041	51	1	.9807	.9902	.9861
8	Subscription fuzzing (Test 3)	273	0	8413	0	1	1	1	1
8	Disconnection wave (Test 3)	5665	0	3020	1	1	.9998	.9999	.9997
8	TCP SYN scan (Test 3)+UDP port scan (Test 3)	52	0	8634	0	1	1	1	1
8	Telnet (Test 3)	17	0	8669	0	1	1	1	1
8	Xmas scan (Test2)	33	0	8653	0	1	1	1	1
9	Normal traffic (Test 3)	2594	0	6041	51	1	.9807	.9902	.9861
9	Subscription fuzzing (Test 3)	273	0	8413	0	1	1	1	1
9	Disconnection wave (Test 3)	5665	0	3020	1	1	.9998	.9999	.9997
9	TCP SYN scan (Test 3)+UDP port scan (Test 3)	52	0	8634	0	1	1	1	1
9	Telnet (Test 3)	17	0	8669	0	1	1	1	1
9	Xmas scan (Test3)	33	0	8653	0	1	1	1	1

5.2 Experimental scenario 2: Detecting new attacks and updating existing ones

It has been shown that rules generated in one iteration are able to detect attacks of the same type in following iterations. Furthermore, our system is able not only to retrain when new events arrive, but also to incrementally improve the model for existing events/attacks when more data is available. This means that we can also update previous rules to make them more accurate. This experiment tries to check what happens when we keep iteratively feeding the model with events, regardless of whether they are classified in existing or new attacks. The objective is to check whether there is an improvement when new events of each family are introduced progressively.

In the second experiment the testing dataset X is divided into 3 datasets: X_1 , X_2 and X_3 . These datasets have the same size, which is one third of the size of the testing dataset X , as shown in Table 3. The training dataset A of each type of traffic is similar to the previous experiment. Figure 6 shows the datasets entering each iteration. The training dataset A of each type of traffic is detected as an anomaly, as in experiment 1. The novelty of this scenario is that this training now continues with the first and second testing datasets (X_1 and X_2), which go on to train the model once they have been detected by the CEP rules. The third testing set of each dataset X_3 never trains the model. This is done in order to be able to correctly evaluate the CEP rules at each iteration.

Table 8 shows the results of the second set of experiments. An average F1 score of .9938 was obtained, even slightly better than those obtained in the first scenario. The first detection of each attack is the most improved in this new scenario. These results seem to indicate that a training reinforcement for previously learned rules can improve the classification of CEP rules, while keeping the ability to add unseen attacks to the rule base.

From the point of view of the different attacks, the analysis is similar to the one we performed in scenario 1. We still see that homogeneous attacks such as Disconnection Wave obtain very good results (F1 score = .9998). The detection of other attacks does improve slightly, with this being the case of Subscription Fuzzing and Telnet.

6 Conclusions and future work

This paper proposes an architecture focused on the IoT paradigm that is capable of generating and updating CEP rules, in an unsupervised manner, for detecting and classifying network IoT attacks in real time without the need of a domain expert. The integration of CEP and PCA to

reduce packet size makes the architecture optimal for IoT environments.

The rules generated by the proposed architecture work very well, as the results obtained are very good (F1 score of .9890). These rules are generated in an unsupervised manner, which allows the system to constantly learn without the need for an expert.

The architecture can successfully detect unseen attacks and anomalies, and these detected anomalies can be used to retrain the model, so that the new attacks are progressively better defined. Thus, the architecture generates new CEP rules automatically and incrementally.

All this demonstrates that our proposal can be successfully deployed in IoT environments with significant constraints and generate dynamic unsupervised CEP rules that are able to detect network attacks in real time.

As future work, we plan to increase the number of protocols and attacks in the experiments in order to test the architecture's performance in other contexts. In addition, it would also be useful to create an ontology to classify new unknown attacks in predetermined families. Finally, we intend to check whether this architecture can be made robust against model poisoning attacks and complex attacks that may involve multiple steps.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work was supported by the Spanish Ministry of Science and Innovation and the European Union FEDER Funds [grant numbers FPU 17/02007 RTI2018–093608-B-C33, RTI2018–098156-B-C52 and PID2021–122215NB-C33]. This work was also supported by JCCM [grant numbers SB-PLY/17/180501/000353 SBPLY/21/180501/000195], and the Research Plan from the University of Cadiz and Grupo Energético de Puerto Real S.A. under project GANGES [grant number IRTPO3_UCA].

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Langley, D. J., van Doorn, J., Ng, I. C. L., Stieglitz, S., Lazovik, A., & Boonstra, A. (2021). The internet of everything: Smart things and their impact on business models. *Journal of Business Research*, 122, 853–863. <https://doi.org/10.1016/j.jbusres.2019.12.035>.

2. Shilpa, A., Muneeswaran, V., Rathinam, D.D.K., Santhiya, G.A., & Sherin, J. (2019) Exploring the Benefits of Sensors in Internet of Everything (IoE). In: 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 510–514 . <https://doi.org/10.1109/ICACCS.2019.8728530>
3. AlZubi, A. A., Al-Maitah, M., & Alarifi, A. (2021). Cyber-attack detection in healthcare using cyber-physical system and machine learning techniques. *Soft Computing*, 25(18), 12319–12332. <https://doi.org/10.1007/s00500-021-05926-8>.
4. Asghari, P., Rahmani, A. M., & Javadi, H. H. S. (2019). Internet of things applications: A systematic review. *Computer Networks*, 148, 241–261. <https://doi.org/10.1016/j.comnet.2018.12.008>.
5. Calvo, I., Merayo, M. G., & Núñez, M. (2019). A methodology to analyze heart data using fuzzy automata. *Journal of Intelligent & Fuzzy Systems*, 37(6), 7389–7399. <https://doi.org/10.3233/JIFS-179348>.
6. Sajid, M., Harris, A., & Habib, S. (2021) Internet of Everything: Applications, and Security Challenges. In: 2021 International Conference on Innovative Computing (ICIC), pp. 1–9 . <https://doi.org/10.1109/ICIC53490.2021.9691507>
7. Sadeeq, M. M., Abdulkareem, N. M., Zeebaree, S. R., Ahmed, D. M., Sami, A. S., & Zebari, R. R. (2021). IoT and cloud computing issues, challenges and opportunities: A review. *Qubahan Academic Journal*, 1(2), 1–7.
8. Stoyanova, M., Nikoloudakis, Y., Panagiotakis, S., Pallis, E., & Markakis, E. K. (2020). A survey on the internet of things (IoT) forensics: Challenges, approaches, and open issues. *IEEE Communications Surveys Tutorials*, 22(2), 1191–1221. <https://doi.org/10.1109/COMST.2019.2962586>.
9. Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., & Sikdar, B. (2019). A survey on IoT security: Application areas, security threats, and solution architectures. *IEEE Access*, 7, 82721–82743. <https://doi.org/10.1109/ACCESS.2019.2924045>.
10. Mousavi, S. K., Ghaffari, A., Besharat, S., & Afshari, H. (2021). Security of internet of things based on cryptographic algorithms: A survey. *Wireless Networks*, 27(2), 1515–1555. <https://doi.org/10.1007/s11276-020-02535-5>.
11. Ferraz Junior, N., Silva, A., Guelfi, A., & Kofuji, S. T. (2019). IoT6Sec: reliability model for internet of things security focused on anomalous measurements identification with energy analysis. *Wireless Networks*, 25(4), 1533–1556. <https://doi.org/10.1007/s11276-017-1610-2>.
12. Corral-Plaza, D., Medina-Bulo, I., Ortiz, G., & Boubeta-Puig, J. (2020). A stream processing architecture for heterogeneous data sources in the Internet of Things. *Computer Standards & Interfaces*, 70, 103426. <https://doi.org/10.1016/j.csi.2020.103426>.
13. Ortiz, G., Boubeta-Puig, J., Criado, J., Corral-Plaza, D., Garcia-de-Prado, A., Medina-Bulo, I., & Iribarne, L. (2022). A microservice architecture for real-time IoT data processing: A reusable Web of things approach for smart ports. *Computer Standards & Interfaces*, 81, 103604. <https://doi.org/10.1016/j.csi.2021.103604>.
14. Ren, H., Anicic, D., & Runkler, T.A. (2021) The synergy of complex event processing and tiny machine learning in industrial IoT. In: Proceedings of the 15th ACM International Conference on Distributed and Event-based Systems. DEBS '21, pp. 126–135. Association for Computing Machinery, New York, NY, USA . <https://doi.org/10.1145/3465480.3466928>
15. Roldán-Gómez, J., Boubeta-Puig, J., Pachacama-Castillo, G., Ortiz, G., & Martínez, J. L. (2021). Detecting security attacks in cyber-physical systems: A comparison of Mule and WSO2 intelligent IoT architectures. *Peer Journal of Computer Science*, 7, 787. <https://doi.org/10.7717/peerj-cs.787>.
16. Lima, M., Lima, R., Lins, F., & Bonfim, M. (2022). Beholder - A CEP-based intrusion detection and prevention systems for IoT environments. *Computers and Security*. <https://doi.org/10.1016/j.cose.2022.102824>.
17. Soni, D., & Makwana, A. (2017) A survey on MQTT: A protocol of internet of things(iot). In: 2021 IEEE International Conference on Telecommunication Power Analysis and Computing Techniques (ICTPACT - 2017)
18. Rosa-Bilbao, J., & Boubeta-Puig, J. (2022). Model-driven engineering for complex event processing: A survey. *The Journal of Object Technology*, 21(4), 1–13. <https://doi.org/10.5381/jot.2022.21.4.a10>.
19. Query Guide - Siddhi. <https://siddhi.io/en/v5.1/docs/query-guide/> Accessed 2022-07-05
20. Sun, Y., Li, G., & Ning, B. (2020) Automatic rule updating based on machine learning in complex event processing. In: 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), pp. 1338–1343. <https://doi.org/10.1109/ICDCS47774.2020.00176>
21. Lee, O.-J., & Jung, J. E. (2017). Sequence clustering-based automated rule generation for adaptive complex event processing. *Future Generation Computer Systems*, 66, 100–109. <https://doi.org/10.1016/j.future.2016.02.011>.
22. Luong, N.N.T., Milosevic, Z., Berry, A., & Rabhi, F. (2020) An open architecture for complex event processing with machine learning. In: 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC), pp. 51–56. <https://doi.org/10.1109/EDOC49727.2020.00016>
23. Bruns, R., & Dunkel, J. (2022). Bat4CEP: A bat algorithm for mining of complex event processing rules. *Applied Intelligence*, 52(13), 15143–15163. <https://doi.org/10.1007/s10489-022-03256-2>.
24. Naseri, M.M., Tabibian, S., & Homayounvala, E. (2021) Intelligent Rule Extraction in Complex Event Processing Platform for Health Monitoring Systems. In: 2021 11th International Conference on Computer Engineering and Knowledge (ICCCKE), pp. 163–168. <https://doi.org/10.1109/ICCCKE54056.2021.9721525>
25. Lv, J., Yu, B., & Sun, H. (2022) CEP Rule Extraction Framework Based on Evolutionary Algorithm. In: 2022 11th International Conference of Information and Communication Technology (ICTech), pp. 245–249. <https://doi.org/10.1109/ICTech55460.2022.00056>
26. Roldán, J., Boubeta-Puig, J., Luis Martínez, J., & Ortiz, G. (2020). Integrating complex event processing and machine learning: An intelligent architecture for detecting IoT security attacks. *Expert Systems with Applications*, 149, 113251. <https://doi.org/10.1016/j.eswa.2020.113251>.
27. Roldán-Gómez, J., Boubeta-Puig, J., Castelo-Gómez, J.M., Carrillo-Mondéjar, J., & Martínez, J.L. (2022) Attack Pattern Recognition in the Internet of Things using Complex Event Processing and Machine Learning. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1919–1926. <https://doi.org/10.1109/SMC52423.2021.9658711>
28. Simsek, M. U., Yildirim Okay, F., & Ozdemir, S. (2021). A deep learning-based CEP rule extraction framework for IoT data. *The Journal of Supercomputing*, 77(8), 8563–8592. <https://doi.org/10.1007/s11227-020-03603-5>.
29. Shapira, G., & Schuster, A. (2022) Unsupervised Frequent Pattern Mining for CEP. arXiv. <https://doi.org/10.48550/arXiv.2207.14017>
30. Liu, Y., Yu, W., Gao, C., & Chen, M. (2022). An Auto-extraction framework for CEP rules based on the two-layer LSTM attention mechanism: A case study on city air pollution forecasting. *Energies*, 15(16), 5892. <https://doi.org/10.3390/en15165892>.
31. Ross, D. A., Lim, J., Lin, R.-S., & Yang, M.-H. (2008). Incremental learning for robust visual tracking. *International Journal*

of *Computer Vision*, 77(1), 125–141. <https://doi.org/10.1007/s11263-007-0075-7>.

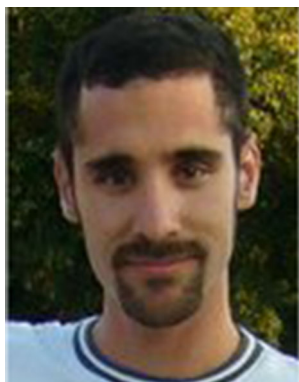
32. Patel, E., & Kushwaha, D. S. (2020). Clustering cloud workloads: K-means vs Gaussian mixture model. *Procedia Computer Science*, 171, 158–167. <https://doi.org/10.1016/j.procs.2020.04.017>.
33. Blei, D. M., & Jordan, M. I. (2006). Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1), 121–143. <https://doi.org/10.1214/06-BA104>.
34. De Maesschalck, R., Jouan-Rimbaud, D., & Massart, D. L. (2000). The Mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1), 1–18. [https://doi.org/10.1016/S0169-7439\(99\)00047-7](https://doi.org/10.1016/S0169-7439(99)00047-7).
35. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., & Kallitsis, M., et al. (2017). Understanding the Mirai botnet. In: 26th USENIX Security Symposium (USENIX Security 17), pp. 1093–1110
36. Roldán-Gómez, J. (2022) Dataset for an automatic unsupervised complex event processing rules generation architecture for real-time iot attacks detection. Mendeley data. <https://data.mendeley.com/datasets/pzhm3jnw6w/draft?a=1565272f-bc8b-4eac-a566-11ec45124a44>. <https://doi.org/10.17632/pzhm3jnw6w.1>
37. Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21, 6. <https://doi.org/10.1186/s12864-019-6413-7>.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



José Roldán-Gómez obtained a degree in computer engineering from the University of Castilla-La Mancha in 2017, obtained a university master's degree in computer engineering from the University of Castilla-La Mancha in 2018, and is currently a doctoral candidate at the University of Castilla-La Mancha, he is also an interim professor at the University of Oviedo. His main interests are artificial intelligence applied to threat detection in IoT environ-

ments and automatic rule generation in CEP engines.



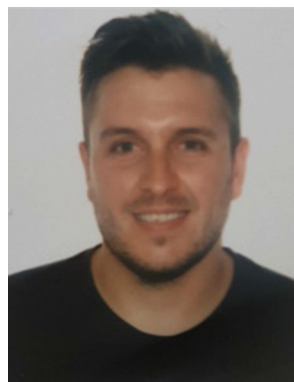
Jesus Martinez del Rincon is presently a Senior Lecturer in the School of Electronics, Electrical Engineering and Computer Science at the Queen's University of Belfast. He received a BSc in Telecommunication Engineering in 2003 and was awarded a PhD in Computer Vision in 2008 from the University of Zaragoza for his work into the development of tracking algorithms for video surveillance and human motion analysis. He also worked as

DIRC Research Fellow at Kingston from 2009 to 2012, leading research on human pose estimation



Juan Boubeta-Puig received the Ph.D. degree in computer science and engineering from the University of Cadiz (UCA), Cádiz, Spain, in 2014. He is an Associate Professor with the Department of Computer Science and Engineering, UCA. His research interests include real-time big data analytics through complex event processing, event-driven service-oriented architecture, Internet of things, blockchain and model-driven development of

advanced user interfaces, and their application to smart cities, industry 4.0, e-health, and cybersecurity. Dr. Boubeta-Puig was honored with the Extraordinary Ph.D. Award from UCA and the Best Ph.D. Thesis Award from the Spanish Society of Software Engineering and Software Development Technologies.



Jose Luis Martinez received his M.Sc. and Ph.D. degrees in Computer Science and Engineering from the University of Castilla-La Mancha (Spain) in 2007 and 2009, respectively. In 2005, he joined the Department of Computer Engineering at the University of Castilla-La Mancha, where he was a researcher in the Computer Architecture and Technology group at the Albacete Research Institute of Informatics (I3A). After completing his Ph.D. in 2010, he

joined the department of Computer Architecture at the Complutense University in Madrid, where he was an assistant lecturer. In 2011, he rejoined the Department of Computer Engineering of the University of Castilla-La Mancha, where he is currently full professor. His research interests include topics related to cybersecurity such as threats detection and classification techniques, with a particular focus on IoT. He has also been a visiting researcher at the Florida Atlantic University, Boca Raton (USA), and the Centre for Communication System Research (CCSR), at the University of Surrey, Guildford (UK). He has over 110 publications in international refereed journals and conference proceedings.