Engineering Applications of Artificial Intelligence 123 (2023) 106344

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

An automatic complex event processing rules generation system for the recognition of real-time IoT attack patterns

José Roldán-Gómez [a], Juan Boubeta-Puig [b,*], Javier Carrillo-Mondéjar [a], Juan Manuel Castelo Gómez [a], Jesús Martínez del Rincón [c]

[a] University of Castilla-La Mancha, Campus Universitario s/n, Albacete, 02006, Spain
[b] Department of Computer Science and Engineering, University of Cadiz, Avda. de la Universidad de Cadiz 10, Puerto Real, Cadiz, 11519, Spain
[c] Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Belfast, BT3 9DT, UK

ARTICLE INFO

Keywords:
CEP rules generation
Cybersecurity
Internet of Things
CEP
Machine learning

ABSTRACT

The Internet of Things (IoT) has grown rapidly to become the core of many areas of application, leading to the integration of sensors, with IoT devices. However, the number of attacks against these types of devices has grown as fast as the paradigm itself. Certain inherent characteristics of the paradigm, as well as the limited computational capabilities of the devices involved, make it difficult to deploy security measures. This is why it is necessary to design, implement and study new solutions in the field of cybersecurity. In this paper, we propose an architecture that is capable of generating Complex Event Processing (CEP) rules automatically by integrating them with machine learning technologies. While the former is used to automatically detect attack patterns in real time, the latter, through the use of the Principal Component Analysis (PCA) algorithm, allows the characterization of events and the recognition of anomalies. This combination makes it possible to achieve efficient CEP rules at the computational level, with the results showing that the CEP rules obtained using our approach substantially improve upon the performance of the standard CEP rules, which are rules that are not generated by our proposal but can be defined independently by an expert in the field. Our proposal has achieved an F1-score of 0.98 on average, a 76 percent improvement in throughput over standard CEP rules, and a reduction in the network overhead of 86 percent over standard simple events, which are the simple events that are generated when our proposal is not used.

1. Introduction

The Internet of Things (IoT) is a new paradigm that has been growing at a tremendous pace, especially over the last few years. It is clear that it can transform our way of life, but it is a mistake to think of the IoT as only something for the future. Proof of this is that we already interact with it via devices such as smartphones and wearables. Moreover, it can be applied in countless contexts, such as smart healthcare or smart cities among many others (Calvo et al., 2019; Asghari et al., 2019; AlZubi et al., 2021). The growth of this paradigm brings with it a series of new problems and challenges in many fields (Sadeeq et al., 2021), such as the one on which we focus in this paper: real-time network attack detection. The devices that are typically used in such environments have limited properties, the most relevant being the fact of having small memories, a low-bandwidth communication channel, a low-end processor, and cheap sensors. These factors make it difficult to directly migrate the solutions used in classical systems to the IoT without prior adaptation, as they consume a greater amount of resources than IoT devices can provide (Shah and Issac, 2018). These characteristics, coupled with the ever-increasing use of these devices (Stoyanova et al., 2020; Hassija et al., 2019), are leading to cybercriminals targeting this paradigm, with recent reports highlighting the high number of threats against it (Kaspersky, 2021).

In order to recognize attacks in real time it is necessary to use technologies that can be deployed in IoT environments, and one of these technologies is Complex Event Processing (CEP) (Corral-Plaza et al., 2020).

CEP is a technology that is able to work perfectly on devices with limited resources (Ortiz et al., 2022), which makes it ideal for use on IoT devices. A CEP engine is a software component designed to detect specific situations, and this is achieved through the analysis and correlation of a large number of previously defined simple events. Another fundamental advantage of CEP which makes it a very attractive option in the cybersecurity field is its ability to process a large number of events per second in real time (Volnes et al., 2021; Garcia-de Prado et al., 2017; Mondragón-Ruiz et al., 2021). This is essential in a field in which it is vital to detect threats quickly in order to act with the lowest possible latency.

* Corresponding author.
E-mail addresses: josalbacet@gmail.com (J. Roldán-Gómez), juan.boubeta@uca.es (J. Boubeta-Puig).

https://doi.org/10.1016/j.engappai.2023.106344
Received 23 May 2022; Received in revised form 30 March 2023; Accepted 15 April 2023
Available online 28 April 2023

Unfortunately, the main problem with using CEP to design an architecture capable of detecting attacks in real time is that we will never detect attacks for which we do not have CEP rules, which are usually defined by a domain expert.

This work attempts to remedy this problem by proposing an architecture which integrates CEP and Machine Learning (ML) technologies in order to recognize attacks in such a way that it is able to automatically generate the CEP rules necessary to detect different attacks in real time. In order to satisfy the requirement of being a low computational power solution, the machine learning component used is Principal Component Analysis (PCA) (Martinez and Kak, 2001). PCA has two objectives: the first one is reducing the size of simple events in the CEP engine; and the second is facilitating the characterization of different attacks, so that a CEP rule can be defined to detect each one of them. This architecture is also able to identify anomalies, meaning that CEP rules can be generated to detect and classify them later, and to find similar attacks when it is an unknown attack, or an anomaly. In the context of this work, we refer to unknown attacks as all those attacks that have not been seen by the model in the training phase, i.e., all attacks not modeled by the system. The latter is useful in order to deploy countermeasures quickly and take advantage of CEP's ability to detect attacks in real time. A solution based on the Mahalanobis distance has been employed to achieve this purpose (De Maesschalck et al., 2000).

In this work we formulate five Research Questions (RQs) in order to achieve our research objectives. These questions are as follows:

- **(RQ1)** Do the generated rules achieve adequate computational performance for the IoT paradigm?
- **(RQ2)** Are the rules generated efficient at the network traffic level?
- **(RQ3)** Can the CEP rule generator be integrated into a deployed IoT architecture?
- **(RQ4)** Can CEP rules detect unknown attacks?
- **(RQ5)** Is it possible to add an attack classification mechanism by proximity?

The deployment we propose in this paper is based on an extension of the algorithm presented in our previous paper (Roldán-Gómez et al., 2021). The novelty of this new work is that the algorithm has been extended to identify similar types of attacks when an anomaly is detected. In addition, the algorithm has been deployed in a network for the first time, allowing us to demonstrate the correct operation of our proposal in a full deployment. Experiments have also been performed to demonstrate the computational performance and network performance of the rules generated by our proposal. This was done with respect to the best possible rules, from a performance point of view, that can be achieved without using our proposal. Schematically, the main novelties proposed by this work are the following:

- The algorithm has been deployed, for the first time, in a complete environment to demonstrate that it can work seamlessly in an IoT environment.
- A comprehensive description of the complete architecture is provided. This enables an in-depth understanding of how the CEP rule generator behaves in an IoT network. In addition, this architecture is divided into three components to facilitate this understanding. These components are data production, data processing and data consumption.
- A new mechanism based on the Mahalanobis distance is proposed for creating classification rules. Apart from being more effective, the rules it generates also determine which attack category is the closest. This function is useful when an anomaly is detected, as unknown attacks can be linked to existing ones according to their behavior. From an incident response viewpoint, it may provide crucial information on what countermeasures may be effective when facing an unknown attack.
- A comparison baseline of simple rules, whose objective is to detect attacks, has been generated without using our architecture. These rules have been optimized for maximum possible computational performance in order to make a fair comparison with our proposal.
- Experiments have been carried out to measure the computational performance from the point of view of the productivity of the rules generated with our proposal, with respect to the simple rules generated. Obtaining an optimal productivity performance is necessary because of the need to detect attacks in real time and to be able to take actions quickly.
- Experiments have also been carried out to measure the bandwidth consumption of the CEP rules of our proposal with respect to simple CEP rules. This metric is important, since a low network overhead allows our proposal to be successfully deployed in IoT environments.
- The dataset used to perform the experiments has been published, and this provides additional value. The CEP rules generated by the proposal to perform the experiments are also made available.

The rest of the article is structured as follows. Section 2 defines the fundamental concepts necessary to understand the technical aspects of this work. Section 3 analyzes the proposals from the community that deal with the problem of automatic CEP rule generation and highlights the substantial differences between our work and the state of the art. Section 4 provides an in-depth description of the components of the proposal and explains how it is deployed in an IoT network. Section 5 lists all the steps performed for generating the CEP rules to be used in the experiments, which are then described in Section 6, which also introduces the metrics we used and discusses the results obtained. In addition, the research questions formulated above are answered in this section. Section 7 introduces certain aspects for improvement as part of future work. Finally, Section 8 presents the conclusions drawn from the experiments.

2. Background

This section describes the background to security in the IoT and CEP.

2.1. Security in the Internet of Things

Unfortunately, the development of security measures for IoT devices has not been as successful as their growth. This is evidenced by the number of cyberattacks detected in the first half of 2019, which exceeded 100 million (Kaspersky, 2019), a 7-fold increase compared with the 2018. The main attack vector used by criminals is still dictionary attacks, a technique which exploits weak credentials to gain access to devices (Demeter et al., 2019). Once access to the device has been gained using these credentials, the device can be infected with malicious code, allowing the attacker to easily manage it. In general, this malware is usually used to create bots, which are later sold or rented to carry out Distributed Denial-of-Service attacks (DDoS). One of the first and most prevalent pieces of malware is Mirai, which targets the Telnet protocol (Kaspersky, 2021).

But not only conventional protocols are the target of cybercriminals, as newly developed IoT ones are also of interest to them. A good example of this is Message Queuing Telemetry Transport (MQTT), which is a very popular network protocol that is used in the IoT (OASIS, 2019) and designed to reduce overheads as much as possible at the application layer. MQTT message sending is publish/subscribe-based, which is a scheme that proposes that clients publish information on a topic that is collected by a broker. This broker acts as a server that manages the flow of messages, which is organized as a hierarchy of topics, and sends these messages to all clients that are subscribed to this topic. Despite being widely used, this protocol has several

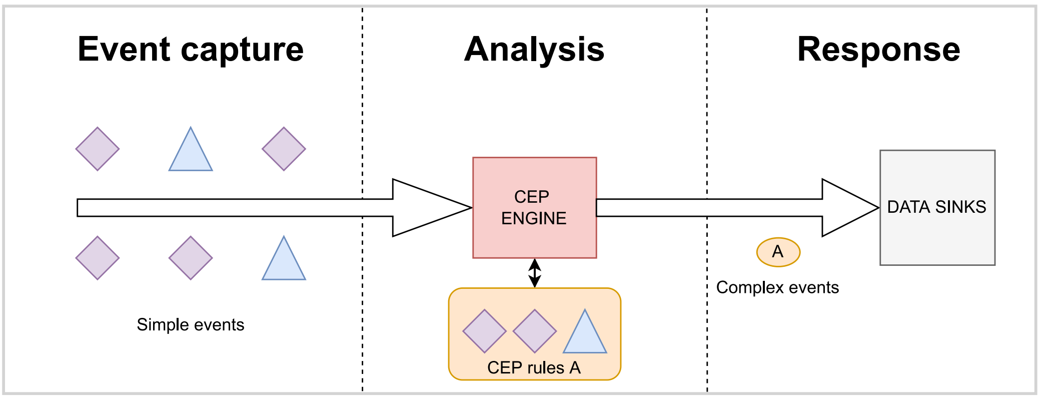


Fig. 1. CEP stages diagram.

weaknesses, which include unencrypted communications by default, the possibility of introducing devices into the network without the need for a username and password, and the ability to obtain all available topics regardless of the importance of the information they contain. It also allows the disconnection of legitimate devices by connecting another device with the same ID.

Besides the weaknesses of MQTT, there are other common threats to IoT networks. The most recurrent technique used by cybercriminals for detecting visible devices and open ports is scanning. This action does not constitute an attack as such, but usually precedes it. In addition, there are volumetric attacks, such as Distributed Denial of Service (DDoS) attacks coming from captured IoT devices, which are usually performed by generating very high traffic flows to saturate the network or the target device. Common types of DDoS are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) flooding, Domain Name System (DNS) reflection or Lightweight Directory Access Protocol (LDAP) reflection, among many others (Warburton, 2021).

Among the main reasons why such simple attacks are effective against IoT devices, we can highlight the use of insecure services, networks and protocols, and the inherent limitations of the paradigm's devices, such as processors with low computational capacities, limited memory capacity or constrained network connectivity. These limitations also promote implementations that do not take into account security by default in the design. It is therefore necessary to develop, implement or adapt solutions to detect threats and mitigate their impact specifically for the IoT paradigm. The solutions to be implemented in this paradigm need to meet two requirements. First, that they can be deployed in systems in which there are devices with constrained resources, since these solutions are intended for IoT environments. Secondly, these solutions have to be able to detect threats with low latency even when there is a lot of network traffic. This is because in cybersecurity an early reaction can mitigate the magnitude of threats.

It is important to categorize the solutions designed for threat detection into two distinct categories. On the one hand, there are detectors for which rules are defined manually to detect specific attacks. Two widely used examples are Snort (Roesch, 1999) and Suricata (Wong et al., 2017). This type of threat detector is useful for detecting known threats by using rules defined by experts. However, they have the limitation that they are not able to detect attacks for which no rules exist, so they are unable to detect zero-day threats. The second type of solutions are behavioral-based, and try to model system behavior and possible attacks. In this way they are able to detect unknown or zero-day attacks, since by modeling the normal behavior, it is possible to detect anomalous behavior of the system. Our proposal falls into the second category, because the rules model the behavior of attacks and the normal behavior of the system. In this way we are able to detect unknown threats. Although threat detectors in the first category have been widely used in the last decade, there is evidence of a shift towards solutions of the second category in recent years (Kumar et al., 2021).

2.2. *Complex event processing*

CEP is a technology that aims to identify significant situations by collecting and correlating events (Luckham, 2012). This is achieved through the creation of CEP rules by domain experts, which monitor specific scenarios within event streams. Once a rule is triggered, a complex event is generated to indicate a situation of interest.

The CEP engine is a specialized piece of software used for real-time data processing. In this work, Siddhi CEP (Anon, 2022) is employed as the CEP engine.

CEP rules are defined using specific languages called Event Processing Languages (EPLs). SiddhiQL, provided by Siddhi CEP, is the EPL used in this case.

Simple events are the raw data input for the CEP engine. In the context of real-time network attack detection, these simple events are network packets. However, this can vary depending on the problem at hand.

CEP rules, which are created by domain experts, define the scenarios of interest. In this work, each CEP rule, which is written in SiddhiQL, can identify a category of attacks.

Fig. 1 shows the three stages of CEP. The event capture stage is the first stage and consists in sending simple events to the CEP engine for processing. In this case, these simple events contain network packets, although this may vary according to the context. The analysis stage is responsible for correlating the simple events in search of situations of interest. These events of interest occur when the simple events meet the conditions described in one or more CEP rules. When this occurs, a complex event is triggered that summarizes the situation of interest. The response stage is the last stage and is the one that encompasses the actions taken with the complex events generated by CEP. These are very heterogeneous as they depend on the context. For example, they may consist of saving the complex events in a database or cutting off a suspicious connection when an attack is detected. This last stage of the process is beyond the scope of this research.

Complex events are generated when simple events trigger CEP rules and serve to identify meaningful scenarios. In this work, a complex event represents the detection of an attack from a specific category. CEP provides many advantages over other possible implementations. First of all, it allows the correlation of simple events in real time and on the fly, without the need to do so from a database. CEP also allows new rules to be added without the need to stop the system. In addition, CEP is designed to handle a large amount of information quickly, and in real time (Cugola and Margara, 2012), and it is easy to deploy. Furthermore, CEP has been successfully tested in real IoT systems, demonstrating that it can be deployed on devices with low computational capacity (Corral-Plaza et al., 2020). This feature is essential if it is to be applied within the field of cybersecurity.

3. Related work

In this section we review the work related to our proposed work.

There are several works that deal with threat detection using machine learning algorithms (e.g. Martins et al., 2022; Zhang and Liu, 2022), and there are also a large number of studies in which different ML techniques are applied for other purposes (e.g. Lawal, 2021; Matkovic et al., 2022; Roy, 2022). However, our work address the open issue of using CEP engines to perform the detection by generating CEP rules automatically. We use CEP engines because of their high data processing capacity. Therefore, this section focuses exclusively on papers that attempt to generate CEP rules.

There are several relevant works that try to solve the problem of automatic rule generation via CEP, and it is possible to classify the proposals according to the need for prior rules.

3.1. Proposals that require previous rules

Sun et al. (2020) focus their work on generating new rules by using previous CEP rules that already exist. The requirements for using this proposal are that you have historical data to train the models and the CEP rules that generate these historical data. First, a loss function is defined to measure the difference between the results of the previous rules with respect to the real values, and an activation function based on the unipolar S-curve whose objective is to decide whether a new CEP rule is good. Then this new CEP rule is added to the rules set. With this new set of rules, clusters containing all the elements of each category are created and a rule is generated to represent these families. Although this work manages to update previous rules in a satisfactory way, it is not able to create them from scratch.

Another approach in which the authors employ CEP in a completely different way can be found in the work by Luong et al. (2020). In this work, the authors use CEP to perform the pre-processing of the data, and Tensor Flow, as an additional component, to classify the different events. The limitation of this architecture is that it does not use CEP to perform the classification, which may result in not fully utilizing the capability of the CEP engines to correlate a large amount of data.

Finally, Ren et al. (2021) use a completely different approach which is focused on optimizing performance in IoT environments. The advantage of this work is that it is not very common to find proposals focused on performance in IoT environments. This proposal employs a micro CEP engine and a model based on Tensorflow Lite Micro with pre-built neural networks. These neural networks are updated to adapt to the behavior of a real system. The difference of this work with respect to the state of the art is that the output of these neural networks feed the CEP engine, which uses simple rules defined manually. Thus, the classification is performed by these neural networks and the CEP engine simply processes the results of the neural networks. Therefore, the main limitation of this proposal is that the CEP rules are defined manually and it does not take advantage of the potential of CEP to process a large amount of data.

3.2. Proposals that do not require previous rules

In this group we find proposals that do not require prior rules, but label complex events based on historical data. Bruns and Dunkel (2022) adapt the bat algorithm to the CEP rule search by structuring the different elements forming a rule in a tree scheme. This tree structure allows the algorithm to generate new CEP rules. The results obtained are promising, but it needs a context in which complex events are known as a function of simple events. This requirement is not always easy to meet without prior rules.

There are other proposals that use tree diagrams to produce CEP rules. The study by Naseri et al. (2021) is another effort that makes use of these diagrams. In this study, the authors deploy a CEP engine at a hospital to track various events, and they utilize the PART algorithm to automatically generate CEP rules. Although the work is interesting, again no optimization was performed from the point of view of event dimensionality to optimize performance.

A proposal built on evolutionary algorithms has also been published. In the study by Lv et al. (2022), CEP rules are generated using evolutionary algorithms. In this instance, the authors begin with a history of straightforward occurrences and the straightforward events that were brought about by them. Simple rules are produced from these data and are included in the evolutionary algorithm's initial population. The difficulty of finding complicated events related to basic events without prior rules, which is not always possible, represents the work's main shortcoming.

Few papers have aimed to extract patterns to identify time series. As an example, Li et al. (2019) propose the generation of CEP rules on the basis of the order in which different shapelets appear and the use of the standardized Euclidean distance and a threshold to check whether a time series corresponds to that pattern. The authors also propose mechanisms to improve the performance of the patterns. One of the drawbacks is that the algorithm is designed specifically for time series and cannot be used with other types of events.

Another work that manages to extract rules automatically without prior rules is the one proposed by Roldán et al. (2020). The rules are constructed by predicting the value of the most important feature for each category. If the difference between the actual value and the prediction exceeds the calculated threshold, this event does not correspond to that category. The results obtained are very promising, 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.

Finally, Simsek et al. (2021) compare different algorithms for labeling simple events, and then use the most common algorithms for rule extraction. The paper concludes that Gated Recurrent Unit (GRU) (used for labeling) together with the FURIA algorithm (used for rule extraction) obtain the best results in their experiments. The comparison made in their paper is certainly of high academic value. However, the proposal requires an enormous amount of data to train deep learning algorithms.

Table 1 shows a comparison of the state-of-the-art papers analyzed, focusing on the need for prior rules, and on the performance of the papers.

To the best of our knowledge, there are no previous works corresponding to the state of the art that do not require prior rules for rule generation and try to optimize the rules from a computational point of view. The main novelty of our work, with respect to the rest of the state of the art, is the use of dimensionality reduction in simple events to achieve computationally efficient CEP rules, and the use of the Mahalanobis distance and normal distributions to create a model that is simple enough to be encoded into a CEP rule, yet powerful enough to provide an excellent level of detection. In addition, most of the studied works do not offer a computational performance analysis. This work does offer such an analysis, both with regards to the CEP engine and to the overhead that simple events introduce in the network. Furthermore, the integration of a proximity mechanism to classify anomalies using the closest category as a reference is also a novelty with respect to other state-of-the-art works. Although the latter feature is more useful in environments in which the non-detection of a simple event can be very detrimental, the cybersecurity context is one such environment.

4. Architecture for IoT security

This section describes in detail the proposed architecture, which integrates the use of PCA, Mahalanobis and CEP in order to automatically generate CEP rules capable of detecting both modeled and unmodeled attacks. In this way we can see how the CEP rule generator is integrated and deployed in an IoT environment. Since PCA and the Mahalanobis distance are considered well-known algorithms, their details are given in Appendix A in the case of PCA and in Appendix B in the case of the Mahalanobis distance.

Table 1
Comparative table of state-of-the-art works.

Reference	Performance focused	Need for prior rules	Metrics	Novelty/Highlight
Sun et al. (2020)	Yes	Yes	Accuracy: 0.90	Pre-filtering rules before training improves performance.
Luong et al. (2020)	No	Yes	–	It uses CEP to perform data preprocessing.
Bruns and Dunkel (2022)	No	No	F1 score: 0.99	It uses the Bat algorithm to generate new rules.
Naseri et al. (2021)	No	No	Accuracy: 0.98	It uses the PART algorithm to generate CEP rules.
Lv et al. (2022)	No	No	Accuracy: 0.93	It uses genetic algorithms to generate CEP rules.
Li et al. (2019)	No	No	F1 score: 0.90	It Uses shapelets to generate CEP rules for time series.
Roldán et al. (2020)	No	No	F1 score: 1	It is based on comparing the prediction of key features with their actual values.
Ren et al. (2021)	Yes	Yes	–	Manually defined CEP rules and pretrained neural networks.
Simsek et al. (2021)	No	No	F1 score=.89	It uses GRU and Furia to generate CEP rules in an unsupervised manner.
This work	Yes	No	F1 score=.98	PCA enables the generation of high-performance CEP rules.

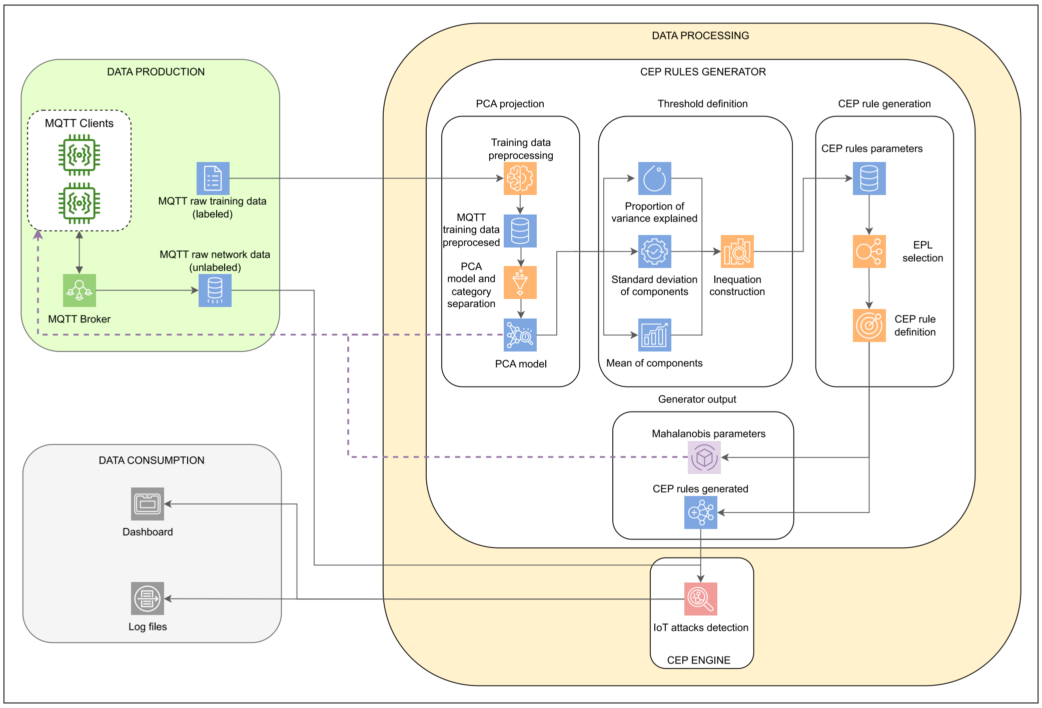


Fig. 2. Diagram of the proposed architecture with rule generator deployed.

4.1. Complete view of the architecture

Fig. 2 shows a diagram of our proposed architecture that includes the CEP rule generator and the rules integrated in an MQTT network. Our architecture is divided into the following 3 layers: a data production layer, which includes the data sources of the architecture; a data processing layer, composed of the elements in the architecture that are in charge of processing the data produced by the data sources; and finally, there is a data consumption layer, which includes the elements in the architecture that will receive the alerts and data generated by the CEP engine.

4.1.1. Data production

In this layer we can find the MQTT network, which is composed of MQTT clients and an MQTT broker. As explained above, these clients generate data and the broker is in charge of distributing the messages containing this data in a centralized way. This MQTT network generates the traffic that will be scanned for threats to the system. In addition, this layer also contains the labeled training dataset that we will use to train the models that will later allow us to generate CEP rules in an automated way. An important detail to note again is that each simple event contains a network packet. This means that we can quickly detect attacks with one or just a few network packets.

4.1.2. Data processing

This layer contains the elements that process the data. First, there is the CEP rule generator, which is responsible for generating the CEP rules to detect the different attacks in the IoT environment. Secondly, there is the CEP engine, which is able, through the CEP rules generated by the CEP rule generator, to correlate all the simple events, which are network packets in this case, and detect the different attacks that appear in the network.

4.1.3. Data consumption

This layer is responsible for managing the attack alerts generated by the CEP engine. In this case there is a dashboard that allows the monitoring the events generated by the CEP engine in real time, and there is a database that stores the events generated by the CEP engine together with additional information such as the packet that generated them. Within the context of IoT network security, regardless of the approach used to mitigate attacks, a rapid response is required. This makes it possible to implement measures that reduce the impact of a successful attack. As a general rule, these actions are typically as follows:

- Blocking and isolating the source of the attack, in this case an MQTT client. This is necessary to prevent the attack from continuing to have an effect or recurring.

- Analyzing the impact of the attack on the system, i.e., identifying which assets have been compromised and how they have been compromised.
- Restoring the system to its normal state. It is necessary to bring the system back up, once we have corrected the vulnerabilities that the attackers exploited to attack the system.

Note that this layer is not within the research focus of this paper. Nevertheless, we can conclude that early detection can make a difference in mitigating the impact of attacks.

4.2. CEP rules generator stages

Now that the architecture as a whole has been described, we focus on the operation of the CEP rules generator. To perform this task, this pipeline is divided into 3 stages, which are as follows:

- PCA projection. This stage generates a PCA model that is able to reduce the dimensionality of the different events in the system by projecting the input data into a space of reduced dimensionality. In order to obtain a successful projection, it is first necessary to carry out the preprocessing of the training dataset. This step requires an adaptation according to the training dataset, and in this case it consists of 3 phases. PCA was chosen to reduce dimensionality for several reasons. First of all, it is much lighter computationally and requires less data than other more expensive alternatives such as autoencoders (Bank et al., 2021). In addition, PCA can be updated quite easily if we make use of Incremental Principal Component Analysis (IPCA) to add new elements to the model, whereas other options such as T-distributed Stochastic Neighbor Embedding (t-SNE) (Van der Maaten and Hinton, 2008) are much more difficult to update. Finally, PCA is unsupervised, so it can be calculated and updated without labeled data, something that alternatives such as LDA cannot do (Xanthopoulos et al., 2013).
  - Filling of empty fields. Since features belonging to different protocols are used, we will find many packets in which there is no value for these features. It is necessary to fill in these values to train our model. The usual techniques, such as using the mean, median or mode are not valid because the values of these features simply do not exist in the packets in which they do not appear. That is why they have been filled with the value "−1", as using a negative value in a dataset in which there are no negative numbers makes it possible to highlight a difference in a significant way.
  - Categorization of non-numerical variables. PCA requires the values to be numeric, but there are some features such as IP addresses and ports that cannot be treated as such. 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. The scales of the dataset features are very different, which can be a problem for PCA because features with larger numerical values will have more weight than features with smaller numerical values. To solve this problem we use a min–max scaler, which makes it possible to equalize the scales of the different numerical features.

  Once the data have been correctly preprocessed, the PCA model can be trained. This results in a trained model that reduces the dimensionality of future events.
  Once the PCA model has been generated, the training dataset is separated by using the training labels so that each category or class can be mathematically modeled as a Normal distribution.

- Threshold definition. The objective of this stage is to generate a function to classify each sample, which is modeled as a simple event, in its corresponding category. This function will be translated into a CEP rule in the next stage.
  Given a training dataset $X = [x^1, x^2, \dots, x^t, \dots, x^T]$ of T samples and corresponding category labels $Y = [y^1, y^2, \dots, y^t, \dots, y^T]$, each sample is an array composed of n components $x^t = [x_1, x_2, \dots, x_i, \dots, x_n]$. The steps to follow to calculate the threshold are as follows:
  - Obtain the mean of the components. For each category $j \in [1, c]$, where c is the total number of known categories, the mean of each component i, which is defined as m_i^j with respect to its category, is obtained. Knowing the mean of each category will serve as a reference to check the distance of the simple events to that mean. Eq. (1) shows how the mean is calculated.

$$m_i^j = \frac{\sum x_i^r}{R} \forall r \in R : y^r = j \tag{1}$$

  - Obtain the standard deviation of the component. The standard deviation, defined as std_i^j, of each component with respect to its category is also obtained. If the mean makes it possible to obtain the reference with which to compare the simple events, the standard deviation will make it possible to obtain a threshold. Thus, a larger standard deviation will have a larger threshold, since a category with a large standard deviation will also have larger value variations. Eq. (2) shows how the standard deviation is calculated.

$$std_i^j = \sqrt{\frac{\sum (x_i^r - m_i^j)^2}{R-1}} \forall r \in R : y^r = j \tag{2}$$

  - Obtain the proportion of variance explained for the components. The proportion of variance explained, defined as rv_i for each component, is extracted. Note that the latter does not make use of the division by categories, so a single rv_i for all categories exists. The proportion of variance explained allows a weight to be assigned to each of the components on the basis of their importance, which is why it is used to weight the means and standard deviations assigned to each component. In this way, a component with a high level of importance will have a greater impact when calculating the threshold than another component of less importance.
  - Build the inequation. By using these elements, an inequation similar to the one shown in Eq. (3) is constructed for n components. If the left-hand side of the inequation is smaller than the right-hand side, which is the threshold that is defined, this event corresponds to the category used to construct this CEP rule. In addition, a corrective element (α) is added to the inequation to increase the threshold value. This element is intended for categories that have events that may be quite different from each other, so that the original threshold is below that of the events that are farther away from the mean. This parameter can be calculated by either heuristics or using an optimization process (Devi et al., 2021; Gupta et al., 2021; Ghasemi et al., 2022). In this proposal, this parameter was calculated heuristically, by trial and error.
    The obtained inequation allows us to implement a distance function with a threshold in the CEP rules.

$$f(x,j) = \begin{cases} 1 \text{ if } f(x) = \sum_i^n \sqrt{(x_i - m_i)^2} \cdot rv_i \leq (\sum_i^n std_i \cdot rv_i) + \alpha \\ 0 \text{ if } f(x) = \sum_i^n \sqrt{(x_i - m_i)^2} \cdot rv_i > (\sum_i^n std_i \cdot rv_i) + \alpha \end{cases} \tag{3}$$

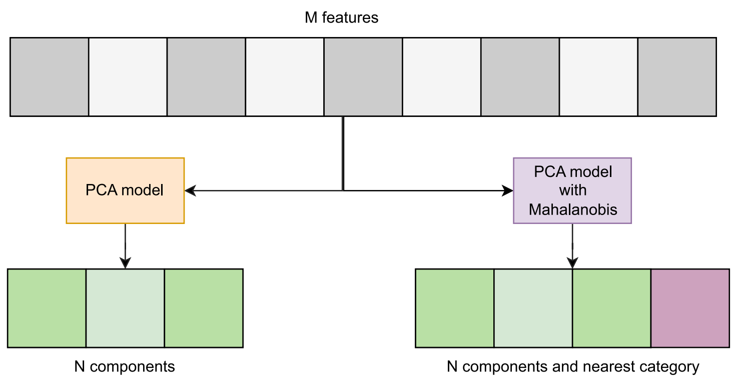


Fig. 3. Modification diagram of simple events with PCA and PCA with Mahalanobis.

- CEP rule generation. Once we have the inequation, it is necessary to transform it into a CEP rule, which is the objective of this stage. The first step is to choose a particular EPL, which will depend on the one provided by the CEP engine to be used. In this case WSO2 is used, and it provides Siddhi as the EPL. Finally, it only remains to build the rule by adapting it to the syntax of the chosen EPL. Listing 1 shows the template used to generate the CEP rules and the explanation of each part of the CEP rule generated with Siddhi. It is important to note that the rule-set template can be reused to generate any CEP rule, as long as the PCA model has three components. If the number of components varies, one simply has to modify the number of components in the template. This process is easily automated since the underlying syntax of the generated CEP rules is always the same.
These steps make it possible to generate rules capable of detecting known attacks. In addition, if non-malicious events are used in the training data, we can generate a rule capable of detecting unknown attacks, and this allows it to perform the function of an anomaly detector. This CEP rule is built with non-malicious traffic, and the threshold is inverted, so any sample above the threshold is considered an anomaly.

Listing 1: Template used for CEP rule generation

```
1  @info(name={attack name})
2  from ((every a1 = {pca input stream}
3  [((math:abs({first component} —
4  {mean of first component}) *
5  {explained variance ratio of first component}) +
6  ...
7  ...
8  (math:abs({n component}—
9  {mean of n component}) *
10 {explained variance ratio of n component})
11 <
12 ({standard deviation of first component} *
13 {explained variance ratio of first component}+
14 ...
15 ...
16
17 {standard deviation of n component} *
18 {explained variance ratio of n component}+
19 {corrective element})]))
20 select *
21 insert into {stream name};
```

The first two lines of the CEP rule, which can be seen in Listing 1, define the name of the rule (*attack name*), and also specify the name of the simple events to be checked by the rule (*pca input stream*). The code between lines 3 and 10 defines the left-hand side of the inequation. As can be seen, this part is variable, as it depends on the number of components extracted by the PCA model. The absolute value function is used to guarantee positive distances. Line 11 defines the sign of the inequality, which is inverted in the anomaly detection rules. The code between lines 12 and 19 defines the right-hand side of the inequation, which is the part that defines the threshold with which the left-hand side is compared. The standard deviations, the explained variance ratio and the corrective element, if any, are used. The last two lines define what data are sent to the output (all components in this case). It also defines the name of the output stream (*stream name*). As mentioned above, this template will be reused for any CEP rule on SiddhiQL. This means that the creation of new rules can be performed automatically.
- Proximity classification mechanism. As discussed above in the description of the complete architecture, the clients have the means and covariances of the different families of known attacks. This allows the clients to calculate the Mahalanobis distance of the simple events with respect to the means of the different families. The closest category is then sent to the CEP engine together with the reduced simple event. This is useful when there is no specific attack whose rule detects that simple event, but it falls within the anomaly CEP rule. In this way we can know that it is an anomalous attack together with the closest known class, which can be vital in being able to take countermeasures quickly, although such countermeasures are not within the scope of this paper. The main advantage of using the Mahalanobis distance is that it takes into account the scale differences that may exist between components for each category, as well as the correlations that may exist between components. Although the latter feature is not useful in our case, since the components are linearly independent, the former feature allows us to obtain a more accurate distance, especially when there are substantial differences in the values that each category has in the principal components. The Mahalanobis distance achieves this as it allows weighting the differences in each case on the basis of the category mean. In this way the same difference will have greater weight in a component where the mean is very small and the standard deviation is also small. Relativizing these distances is ideal in the case of PCA, where the component values may vary between categories.
Moreover, the choice of this distance follows a similar rationale to choosing PCA, since both work well with relatively small samples. Given that Mahalanobis assumes an underlying normal distribution modeled by mean and covariance, it will be possible to estimate viable values in small sample populations of 30 elements or more, as stated by the central limit theorem (Kwak and Kim, 2017).

A step-by-step description can be found in Algorithm 1.

Algorithm 1 CEP rules generation algorithm

```
Input: Labeled training data
Output: PCA trained model, Generated CEP rules

 1: Preprocess_dataset
 2: Train_pca_model
 3: Transform_preprocessed_dataset_with_pca
 4: Category_separation
 5: for each_component_of_pca do
 6:     Extract_proportion_of_variance_explained
 7:     for each_category do
 8:         Extract_mean
 9:         Extract_std
10:     end for
11: end for
12: Inequation_construction
13: Select_EPL
14: for each_category do
15:     Generate_Rule
16:     if proximity_classification_mechanism then
17:         Generate_proximity_classification_pattern
18:         Extract_covariance_matrix;
19:     end if
20: end for
```

The diagram in Fig. 3 shows how simple events are transformed into reduced simple events using the PCA model. It can also be seen how an indicator showing the closest category obtained using the Mahalanobis distance is added. This diagram clearly shows the large reduction in the size of simple events obtained by transforming simple events with PCA. In addition, PCA allows us to keep as much information as possible while reducing the size of simple events.

It is important to note that the only steps that need to be performed manually are the labeling of the samples for training and preprocessing, as these are dependent on the context in which the CEP rule generator is deployed. All other steps can be fully automated in a general way for future updates. This means that it is easy to retrain the model if new labeled samples are available. In summary, the training data reduced by PCA allows the generation of CEP rules for each packet category, including packets produced by normal system behavior and those produced by different attacks. Simple events, which contain the network packet information, are reduced with the same PCA model on the client and sent to the CEP engine. If the rules generated by the CEP rule generator are triggered by the simple events, a complex event is generated that identifies the attack corresponding to the attack category modeled by that CEP rule. This makes it possible to detect attacks, even unknown attacks if an anomaly-detecting rule is implemented, and to classify them if labeled data of those attack categories is available.

5. Test environment

This section describes the elements necessary to understand the experimental environment. These are: the dataset and its attacks, the feature selection and the experimental environment. In addition, different CEP rules generated with our proposal are shown and the main challenges faced when performing the experiments are listed.

5.1. Challenges of experimentation

The challenges we faced in the experimentation are as follows:

- Heterogeneity of the dataset. In order to carry out the experiments in such a way that they represent a real situation, protocols and attacks that can be found in real situations were chosen. We also tried to ensure that these attacks behaved differently, in order to avoid bias in the experimentation. However, this introduces the challenge of dealing with heterogeneous attacks in the generation of CEP rules. To overcome this challenge we employed PCA together with a Euclidean distance weighted by the explained variance ratios of the components. In addition, the Mahalanobis distance was also used for the proximity mechanism, the advantages of which we listed above.
- Choice of features. Although it is easy to collect the different features of the packets using Wireshark, the selection of these features may not be trivial. To make an optimal selection, we used the KD99 dataset, especially for the classical protocols, such as TCP, UDP and Telnet. We also added features that are heuristically, to the best of our knowledge, considered relevant for attack detection. Finally, Extremely Randomized Trees (ERT) was used to check that the initial feature selection was optimal, and to find out which features were the most relevant (Geurts et al., 2006).
- Distribution of the training dataset. While ideally a balanced dataset is preferred to train ML applications, this will result in an unrealistic dataset, since in reality each attack will produce a wildly different number of samples/packets, and this will also result in a different unbalanced set at testing time. Therefore, we followed the same distribution in both training and testing as the attacks in a real environment, resulting in an unbalanced set. While this is a potential challenge, it shows the potential of our approach to work even when training with unbalanced datasets and with few samples of some attacks.

5.2. Dataset

The dataset is obtained from an MQTT network composed of 3 clients and a broker. In this network, normal behavior is simulated in which the network is not under attack. This normal behavior consists in periodically publishing on 3 topics, one per client. The format of these data is numerical and simulates possible scenarios such as sending temperature or decibel readings. After this, an additional client is introduced which is in charge of performing the attacks mentioned below. This dataset was used because it contains a variety of MQTT attacks. To the best of our knowledge, there is not any publicly available attack dataset with such a wide range of attacks and a large enough sample size to enable incremental detection and the addition of further rules to the CEP engine. Moreover, no existing dataset contains subscription fuzzing attacks or disconnection wave attacks. The dataset, together with the CEP rules used in the experiments, has been published (Roldán-Gómez, 2023).

- Subfuzzing. The objective of this attack is to know the different topics that are registered in the MQTT broker. This makes sense when it is possible to introduce a client into the network, but the topic #, which is the root topic in the hierarchy, is disabled. The client tries to subscribe to all the topics to identify the existing ones. To do this, a dictionary is used, but it can also be done by pure brute force. This attack is not common, but it can be very dangerous as it allows the attacker to have a map of all the topics in the network. Depending on the network configuration, this can have a major impact on the confidentiality and integrity of the network. This attack was included in our work (Roldán-Gómez et al., 2022), and it is precisely the type of attack that cannot be detected by detection systems based on specific rules.
- Disconnection wave. This is an attack that attempts to disconnect all the clients from the broker, and works in configurations where the broker disconnects old clients when a new client tries to

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.
mqttFlags	Text type	Identifies the MQTT flags enabled on that packet
mqttMessage	Text type	Identifies the MQTT message on that packet.
mqttTopic	Text type	Identifies the MQTT topic on that packet.
mqttMessageLength	Integer	Identifies the MQTT message size in bytes.
mqttFrameCount	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.

connect with the same *id*, since this id is used to univocally identify a client in the network. The client sends several packets with the connect command with the identifiers of the clients to be disconnected. A sweep can be made through the different *ids* to eject all the legitimate devices from the network. This attack was also included in our work (Roldán-Gómez et al., 2022). The impact of this attack, when successful, is mainly on network availability, as it disables all legitimate clients and prevents the network from functioning.

- TCP SYN scan. This type of scanner is very common, and is used to know the open ports on a computer, in this case the broker. If a port is open, the broker will send a Synchronization (SYN) or Acknowledgment (ACK) packet, and if it is closed, it will send a Reset (RST) packet. This type of scanner is widely used for TCP port detection. It provides information on the status of these ports to the attacker even though it has no direct impact.
- UDP scan. The UDP scanner is used to search for open UDP ports on a computer, which is the broker in this case. If the broker sends an Internet Control Message Protocol (ICMP) packet of type Unreachable, the port is closed. If the broker sends another error, the port is considered filtered. If it sends nothing, or sends something else, the port is considered open. This scanner is often used for detecting open UDP ports. Although it does not have a direct impact, it provides key information to the attacker.
- Telnet scan. This attack attempts to simulate the first stage of Mirai. It consists in a dictionary attack against Telnet to try to gain access to the computer, which in this case is the broker. Mirai is an extremely common malware in IoT environments (Antonakakis et al., 2017), so it is useful to include this attack in the dataset.
- Xmas scan. This is a scan targeting TCP ports. To perform this attack, TCP packets are sent with the Push (PSH), Finish (FIN) and Urgent (URG) flags set. If the response is an RST packet, the port is considered closed. If the response is an unreachable error, the port is considered filtered, and if the port does not respond, the port may be filtered or open. This scanner is rarely used nowadays, but it is beneficial to include it in the dataset because it is very different from the other scans.

The dataset is composed of a mix of attacks, which were chosen to represent a realistic scenario while comprising a variety of different attack types.

Table 2 shows the description of the features that the dataset has, as well as the type to which it belongs.

5.3. Collection and feature selection

The data collection to generate this dataset is performed using a sniffer on the broker. With the sniffer, Wireshark in this case, a PCAP file is obtained and converted into an easily manageable Comma-Separated Values (CSV) file.

Table 3
Feature importance ranking.

Feature name	Feature importance
Destination port (1883)	0.259
Calculated window size	0.240
Protocol (TCP)	0.122
Protocol (MQTT)	0.100
IP source (192.168.1.11)	0.092
Information (Publish message)	0.032
Source port (59662)	0.030
IP source (192.168.1.7)	0.029
Source port (62463)	0.027
Source port (52588)	0.016
Packet length	0.005

For the feature selection, we follow the selection made in KDD99 (Kayacik et al., 2005), since it is a very well-known set, and features specific to our MQTT dataset were also added. These additional MQTT features were chosen on the basis of our knowledge. However, to ensure that our selection was adequate to characterize our dataset we employed ERT (Geurts et al., 2006), which makes it possible to calculate the weight of each feature in the dataset quite accurately. ERT generates random trees and assigns weights to the dataset features on the basis of the importance of that feature in the correct classification of the dataset samples. From this a ranking of the importance of the features is obtained. This algorithm works similarly to Random Forest (RF) but with two fundamental differences. Firstly, ERT does not make use of bootstrapping, although there are implementations that do allow it, but in our case bootstrapping has not been used. The second is that ERT does not choose the optimal cut points as RF does; ERT performs this process randomly, although it does choose the best division later, as does RF. In practice, not choosing optimal cut points makes ERT computationally faster than RF. There are also studies that show that ERT obtains better results than RF (Savakar and Kannur, 2019), especially in low-dimensional datasets, such as the one used in this work. There are works that show that ERT performs well in feature selection compared with other algorithms (Zhang et al., 2023).

This allows us to affirm that our selection of features is correct. Table 3 shows a ranking of different features.

5.4. Experimental setting

Once the dataset has been preprocessed, it can be used to train our PCA model. The division between training and testing is as shown in Table 4.

Table 4
Splitting training/testing dataset.

	Percentage of training	Percentage of testing
Normal	3174(20%)	12696(80%)
Discwave	19999(20%)	80000(80%)
Subfuzzing	819(20%)	3277(80%)
TCP SYN scan	700(70%)	301(30%)
UDP scan	411(70%)	177(30%)
Telnet (Mirai)	351(70%)	151(30%)
XMAS scan	900(90%)	100(10%)

As we can see in Table 4, not all categories have the same percentage in the net training set. These attack distributions were selected to mimic a real attack's package distribution. Some attacks generate few packets, such as scanners, while others generate many. To ensure the successful representation of attacks with low representation, a higher proportion is used for training, while still preserving enough for validation. For attacks that are abundant, a smaller percentage is used for training to show they can be detected early and learned without waiting for a large number of samples to be captured. As we can see, our proposal is able to generate CEP rules with a small number of packets from different attacks. This is very useful and means that the model is able to generalize correctly without the need for a large amount of network traffic.

5.5. CEP rules generated

Once we have the PCA model, all that remains is to build CEP rules. To do this we use a template in which we fill in the values that we have already obtained, and which can be seen in Eq. (3).

In this way, a rule is generated to detect each of the different attacks described in the dataset, and an additional rule is generated to detect anomalies that do not fit any of the previous rules. We have 7 categories: 6 of them correspond to attacks, and the last one to anomalies.

Some of the CEP rules obtained are shown in Listings 2, 3 and 4.

Listing 2: Generated CEP rule for detecting subfuzzing

```
@info(name="subfuzzing")
from ((every a1 = MinimizedPacket
[((math:abs(a1.c1 − −0.08089453) *
0.45243782) +
(math:abs(a1.c2 − −0.93639381) *
0.25893292) +
(math:abs(a1.c3 − 0.1890189) *
0.09149742)) <
(0.050084+0.1 )]))
select *
insert into subfuzzingStream;
```

Listing 3: Generated CEP rule for detecting discwave

```
@info(name="discwave")
from ((every a1 = MinimizedPacket
[((math:abs(a1.c1 − −0.5369006) *
0.45243782) +
(math:abs(a1.c2 − −0.95389636) *
0.25893292) +
(math:abs(a1.c3 − −0.16195309) *
0.09149742)) <
(0.0048987)]))
select *
insert into discwaveStream;
```

Listing 4: Generated CEP rule for detecting anomalies

```
@info(name="anomaly")
from ((every a1 = MinimizedPacket
[((math:abs(a1.c1 − −1.08021604)*
0.45243782) +
(math:abs(a1.c2 − 1.17180894) *
0.25893292) +
(math:abs(a1.c3 − 0.0342363) *
0.09149742)) >
((0.14454024 * 0.45243782)+(0.17986563 *
0.25893292)+
(0.93275888 * 0.09149742) )]))
select *
insert into anomalyStream;
```

6. Results and discussion

This section analyzes the performance results of the proposed architecture. This analysis can be divided into three different parts: a first part in which the ability of the proposal to classify correctly is analyzed, a second part that focuses on the throughput of the proposal, and a last part in which the bandwidth usage is analyzed. Then, a discussion about the research questions posed is presented.

6.1. Standard CEP rules against generated CEP rules

One of the advantages associated with the use of PCA for defining CEP rules is the performance gain that can be obtained. In order to measure the improvement offered by using our architecture, basic rules were manually generated for comparison. These rules rely on the study of only one feature to decide whether the corresponding attack is taking place. Although in practice their feasibility is quite low due to the high number of false positives that they generate, they are ideal for being used as a testing reference, as they represent a scenario of maximum efficiency. Setting them as a benchmark means that if our proposal is capable of obtaining better results than the manually-generated rules, it will outperform any non-PCA generated rule. Some standard attack CEP rules are shown in Listings 5, 6 and 7. These rules were built manually without using our proposal.

Listing 5: Standard CEP rule for detecting subfuzzing

```
@info(name="sub_fuzz")
from (every a1 = NetworkPacket[(a1.protocol
== 'MQTT') and (a1.info=='Subscribe Request')
and (a1.mqttMessageLength==7)])
select a1.id
insert into  subStream;
```

Listing 6: Standard CEP rule for detecting discwave attack

```
@info(name="discwave")
from (every a1 = NetworkPacket[(a1.protocol
== 'MQTT')
and (a1.info=='Connect Command')])
select a1.id
insert into  discwaveStream;
```

Listing 7: Standard CEP rule for detecting UDP port scan

```
@info(name="udp")
from (every a1 = NetworkPacket
[(a1.protocol == 'UDP')])
select a1.id
insert into  udpStream;
```

As mentioned above, the objective is to make the rules as simple as possible so that they can obtain an optimal performance in order to make the comparison with the CEP rules generated by our architecture.

Table 5
Metrics obtained by the generated CEP rules.

Metrics of generated rules without Mahalanobis	Precision	Recall	F1
Discwave	1	0.9002	0.9474
Subfuzzing	1	0.9017	0.9483
Subfuzzing (with corrector)	1	1	1
TCP	1	1	1
UDP	1	0.9772	0.9885
XMAS	1	1	1
TELNET	1	0.8733	0.93236
Anomaly	0.9968	1	0.9984
Metrics of generated rules with Mahalanobis	**Precision**	**Recall**	**F1**
Discwave	1	1	1
Subfuzzing	1	1	1
TCP	1	1	1
UDP	1	1	1
XMAS	1	1	1
TELNET	1	1	1

6.2. Metrics used

To measure the effectiveness of the different CEP rules, we analyze each of them as a binary classifier. Thus, a true positive is obtained when the CEP rule detects a simple event that corresponds to the category modeled by that CEP rule. A false positive is obtained when the CEP rule detects a simple event, but it does not belong to the category modeled by that rule. A true negative is obtained when a CEP rule does not detect a simple event that does not really belong to the category modeled by that CEP rule. And a false negative is obtained when a CEP rule does not detect a simple event that does belong to the category modeled by that CEP rule. In order to correctly measure the results of the experiments we use the following metrics:

- Average throughput. This metric, which is used to measure the computational performance of CEP rules, is defined as the number of capable events that the CEP engine is able to process per second (events per second). In our case we have used time windows of 10 s, which means that the metric is calculated by dividing all the events processed during this interval by 10. This metric, which is widely used to measure performance in CEP, provides an insight into the computational performance of the CEP engine with different CEP rules.
- Average event size. This metric allows us to know the network usage for each CEP rule. It is determined as the average of the sizes (bytes) of the simple events, and it tells us how efficient each type of rule is in relation to the use of the network by means of the simple events it uses. Fewer bytes per simple event means that less overhead is introduced into the network.
- Precision. This metric measures the ability of the CEP engine to avoid false positives, so in this case it measures the ability of a CEP rule to avoid false positives, which occur when a packet is classified in a category that does not correspond to it. This happens when a normal packet is detected as an attack, or when an attack is classified with a rule that does not correspond to the category of that attack. Eq. (4) shows how precision is calculated.

$$(\frac{TruePositive}{TruePositive + FalsePositive}) \tag{4}$$

- Recall. This metric measures the ability of the CEP rules to avoid false negatives. Recall is a very important metric in the context of cybersecurity, as it measures the ability of a CEP rule to prevent the attacks that this rule models from not being detected by that rule. This is fundamental because it determines the system's capacity for allowing attacks to occur without being detected, something that usually entails a very high risk to the integrity, confidentiality and availability of the system. Eq. (5) shows how recall is calculated.

$$(\frac{TruePositive}{TruePositive + FalseNegative}) \tag{5}$$

- F1-score. This metric averages precision and recall in order to provide a more global view of the performance of the CEP rules. The F1-score metric is very useful because it represents a balance between the two previous metrics. It is therefore considered the most balanced metric. Moreover, in this case, it has the additional advantage of working very well on unbalanced datasets, such as with our testing data. Eq. (6) shows how the F1 score is calculated.

$$2 \cdot (\frac{Precision \cdot Recall}{Precision + Recall}) \tag{6}$$

Since this is an architecture intended for IoT environments, in which resources are very limited, it is desirable that the architecture can provide optimal classification capacity, with high throughput, while minimizing network bandwidth usage as much as possible.

6.3. Effectiveness results

This first experiment is used to measure the accuracy of the CEP rules generated by our approach. It consists of sending all the events from the testing to the CEP engine, which has the CEP rules defined via our proposal. The accuracy of both the CEP rules with the threshold and the CEP rules of classification by proximity are measured. In addition, the results obtained with the CEP rule used to detect anomalies are also measured.

The upper part of Table 5 shows the results obtained by the proposal for classifying each attack. The results are very good, as in the worst case scenario (Telnet) an F1-score of 0.93 is obtained. It is interesting to see how the discwave, Telnet and subfuzzing scenarios are the ones in which the most simple events fall outside the range of the CEP rules generated for those categories. This is because the 3 cases present some uncommon packets that are very different from the rest of their category. That is why they are not detected by the rule, as the fact that they are far from the mean, and that they constitute few packets, causes them to only slightly affect the mean and the standard deviation. However, the results are excellent even in these cases, since it must be taken into account that by detecting a single packet, we are already able to detect an attack, and in the worst case we have a recall of 0.93.

We can conclude that the architecture is able to classify attacks correctly without problems.

Table 5 also shows the results obtained by the anomaly detection CEP rule. As we can see, all the anomalies (attacks described above) are detected, and very few non-anomalous packets are misclassified as anomalies.

The bottom part of Table 5 shows the results obtained by the proximity classification mechanism. As we can see, when the proximity mechanism is activated, the results obtained are those of a total detection of all simple events in their respective categories. In these experiments where all the categories are defined, what is achieved with

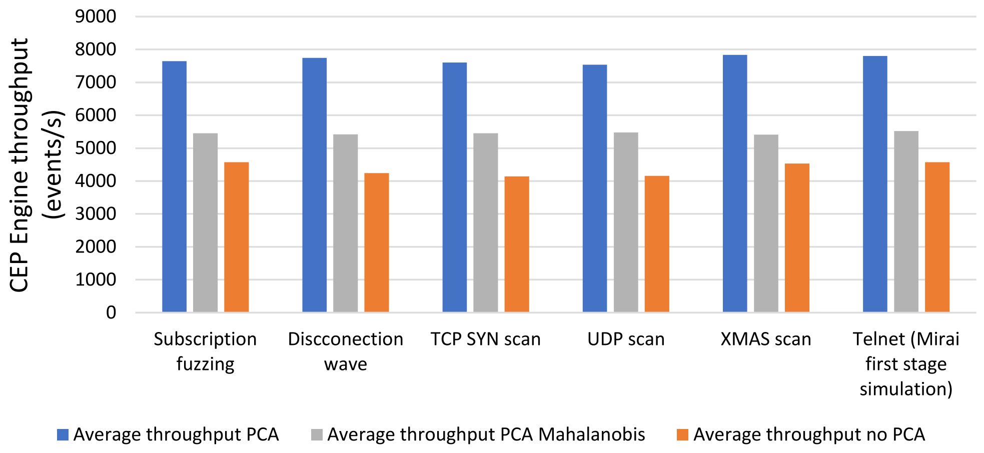


Fig. 4. Average throughput for PCA vs no PCA.

Table 6
Average throughput per attack measured in events per second (events/s).

	No PCA throughput	PCA throughput	PCA and Mahalanobis throughput
Subscription fuzzing	4575.4	7646.6	5451.4
Disconnection wave	4242.2	7746.2	5417.8
TCP SYN scan	4142.6	7605.8	5453.4
UDP scan	4157	7537.4	5474.2
XMAS scan	4534.2	7833.2	5410
Telnet (Mirai first stage)	4574.8	7806.8	5519

this mechanism is to correctly detect the cases of packet which, despite belonging to a category, are rare and scarce. In contexts where an attack is a zero-day attack, a quick action reference would be achieved by having the closest existing category. In any case, the improvement obtained with this mechanism is considerable.

6.4. Computational performance results

This experiment attempts to demonstrate the computational performance improvement that can be obtained by using the CEP rules of our proposal. In this experiment the testing datasets are sent in a loop without interruption for 50 s. The objective is to send a high workload to the system and observe how it behaves. This allows us to compare common CEP rules against CEP rules generated with our proposal, and measure their computational performance in terms of throughput and network event sizes.

As we can see in Fig. 4 and Table 6, the performance when using the rules generated by our proposal is far superior to that obtained when using common CEP rules, representing an average improvement of approximately 76 percent when all attacks are taken into account. This means that the rules generated by our proposal are capable of processing a higher number of events per second, which provides the system with a greater capacity for high workload situations. When the nearest category classification mechanism is used, the performance improvement is limited to 24 percent over common CEP rules. Another interesting observation is the homogeneity of performance of the different CEP rules in all categories. This makes sense, given that all rules are structurally the same and maintain the number of operations they perform. This allows us to know in advance the amount of resources we need based on the packets received. Something very useful in some implementations, especially if we intend to take the CEP engine to the cloud, although it has already been demonstrated that this is not necessary.

The last part of the results focus on the size of the events, and in this context a smaller event size means less saturation of the network.

As we can see in Fig. 5 and Table 7, the improvement in event size is enormous. We reduced the event size by 86% on average thanks to the dimensionality reduction. If the nearest category classification mechanism is used, the event sizes vary very little, because only one label is sent with the nearest category of that simple event. This means that only one more field is added to the simple event to be sent, so the number of elements increases from 3 to 4, thus implying only a very small additional overhead. The reduction in event size means that we can greatly reduce network usage. This improvement is very beneficial, especially in an architecture aimed at the IoT paradigm, in which network utilization is a critical resource.

6.5. Answers to the research questions

With the results obtained, we can answer all the research questions we posed above.

- **RQ1** Do the generated rules achieve adequate computational performance for the IoT paradigm?
  - The performance of the CEP rules generated is superior to that of simpler CEP rules. This is due to the use of PCA to characterize the events and reduce the dimensionality of the events. This allowed us to achieve a throughput increase of 76% in the experiments performed.
- **RQ2** Are the rules generated efficient at the network traffic level?
  - The simple events that are sent with the generated CEP rules are much lighter than the simple events that are used with standard rules, which have been generated manually. In our experiments the average event size was reduced by 86%. This drastically reduces network usage by improving the effectiveness of attack detection and facilitates real-time detection.
- **RQ3** Can the CEP rule generator be integrated into a deployed IoT architecture?
  - The results obtained indicate that this is the case, since the rules that are generated perform well from a computational point of view and also with regards to network utilization. In addition, the functional results of the rules are very good, as they are able to detect attacks systematically.

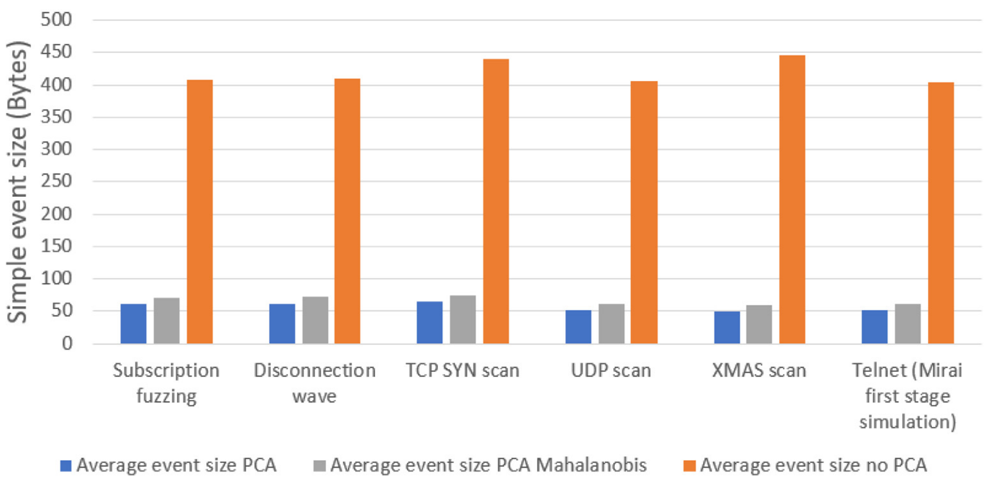


Fig. 5. Average event size for PCA vs no PCA.

Table 7
Average event size per attack measured in bytes.

	No PCA event size	PCA event size	PCA and Mahalanobis event size
Subscription fuzzing	407.7271	60.7644	70.7644
Disconnection wave	408.8412	61.917	71.917
TCP SYN scan	439.8443	64.8732	74.8732
UDP scan	406.1018	50.786	60.786
XMAS scan	445.8791	48.989	58.989
Telnet	403.8608	51.7634	61.7634

- **RQ4** Can CEP rules detect unknown attacks?
  - The results obtained from the experiments show that anomaly detection CEP rules can be created. These rules are capable of detecting unknown attacks, for which one or more rules that model the normal behavior of the system must be identified, thus generating the CEP rules capable of detecting anomalies.
- **RQ5** Is it possible to add an attack classification mechanism by proximity?
  - The results of the experiments show that it is possible to classify by proximity, although in this case the performance improvement is considerably lower.

7. Limitations and future work

The results obtained by the CEP rules generated by our architecture are very good, but there are certain aspects of it that could be improved with future contributions.

If there is an element that could be improved in our proposal, it is in the automation of the selection of features. In this work we started from a prior selection where previous knowledge was used to choose a selection of features. The nature of ERT makes it capable of inferring the importance of the features in a dataset, so it would be useful to fully automate the feature selection by taking advantage of this feature.

A current limitation of the architecture is that the calculation of the Mahalanobis distance is performed on the clients. Although this is a simple operation, certain devices may have very limited computational capabilities. Therefore, it may be beneficial to move this feature to the server or to the CEP rules themselves. A first step towards improving the architecture may be to calculate the Mahalanobis distance on the server, or even to replace the weighted Euclidean distance, which is used as a threshold in the CEP rules, by the Mahalanobis distance used by the proximity mechanism. However, the experiment of adding fields to the simple events is useful in this work in order to see how the computational and network performance varies when adding one more field to the simple event.

Another possible improvement with respect to this architecture is a development focused on the explainability of the CEP rules. In this work we have focused on obtaining functional rules that perform well from a computational and network point of view, as well as on their evaluation when deployed in an IoT network. However, explainability for forensic analysis is a useful feature in this type of system. Using PCA to reduce simple events allows to obtain a measure of the weights of each feature in each component, and also of the weight of each component in the final model. This, together with the possibility of reconstructing the simple events into an unreduced version or storing the unreduced event, can allow us to generate an ontology that explains each of the categories.

An inherent limitation of PCA is the need for this algorithm to find linear combinations of the different features to form orthogonal components. Although in the case of network attacks this is not a problem, given the distribution we have observed in these data, it may be a problem in other use cases. One way to overcome these limitations may be to rely on autoencoders (Bank et al., 2021) or Kernel Principal Component Analysis (KPCA) (Schölkopf et al., 1997) in cases where PCA does not work properly. Although both solutions introduce other limitations, it may be useful to compare of them in future work.

Another aim of future work could be to deploy the architecture again with a different topology, and with a completely different set of protocols. Additionally, we also plan to apply our proposed architecture to other real-world IoT domains, such as water supply networks and air quality monitoring networks.

8. Conclusions

In this paper, we have presented an architecture capable of generating CEP rules automatically by integrating CEP and ML technologies.

The CEP rules generated by our proposal are functional and accurate, and are able to detect attacks perfectly. Using the PCA algorithm to characterize events is a fundamental part of the proposal, as this novelty reduces network usage and improves the computational performance of the CEP engine. The results show that we reduce the average event size by 86% and obtain a 76% improvement in throughput, which means that we can process 76% more packets per second than common CEP rules.

Our architecture also allows the creation of CEP rules which are capable of detecting anomalies and unknown attacks. Furthermore, CEP rules have been implemented to classify events by proximity to the different attacks. These CEP rules obtain a very good precision performance, but it is true that they considerably reduce the improvements in network usage and the performance obtained by the original CEP rules.

Therefore, we can affirm that the architecture is a success in terms of detection and performance thanks to the integration of CEP rule generation and dimensionality reduction. This combination makes it particularly useful in resource-constrained environments, such as an IoT network. To summarize, our achievements are as follows:

- The CEP rule generator has been successfully deployed in an IoT architecture. This demonstrates that the CEP rule generator algorithm can operate in an IoT network successfully.
- It has been demonstrated that our proposal generates CEP rules capable of obtaining higher productivity than standard rules without PCA under optimal conditions. This is a very important feature, especially in IoT environments, in which devices have low computational capacity.
- The difference in overhead introduced by the standard rules compared with the rules generated by our proposal has been measured. The rules generated with our algorithm achieve a reduction in the size of simple events of 86% on average. This reduction of the overhead introduced in the network makes our proposal very viable in environments where there is not a very large bandwidth. This is often the case in IoT environments.
- A proximity mechanism has been introduced that is especially useful when anomalies are detected. This mechanism allows the marking of the simple event with the most similar category available at that moment. This has two advantages, the first one being that it can be used to correctly classify a simple event that may escape the CEP rule of that category. This can occur with extreme cases in which a simple event is very different from the average of its category. The second advantage is that it approximates an unknown, or unmodeled, attack to the nearest category. This offers the possibility to act fast even with zero-day attacks, since we can act, by default, as we would with that closest category.

CRediT authorship contribution statement

José Roldán-Gómez: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Juan Boubeta-Puig:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition. **Javier Carrillo-Mondéjar:** Software, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Juan Manuel Castelo Gómez:** Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Jesús Martínez del Rincón:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Visualization, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The dataset can be accessed and will be published upon acceptance of the article.

Funding

This work was partly supported by the Spanish Ministry of Science and Innovation and the ERDF Funds [grant numbers FPU 17/02007 and FPU 17/03105, PID2021-122215NB-C33, PID2021-123627OB-C52, RTI2018-098156-B-C52], by the University of Castilla-La Mancha [grant number DO20184364], by the JCCM, Spain [grant number SB-PLY/17/180501/ 000353, SBPLY/21/180501/000195], and by the Andalusian Plan for Research, Development and Innovation and ERDF [grant number P20_00865].

Appendix A. Principal component analysis

PCA is a statistical method that tries to reduce the complexity of a sample space by reducing its dimensions. When using PCA, if we have an element x, a simple event in this case, represented by n variables, the objective is to find a representation with m variables where $m \ll n$. These new variables are calculated as linear combinations of the original ones, and are called components. Each component is linearly independent of the rest. In this way, PCA manages to maximize the amount of information represented by each component, without redundancy in the information of different ones. That is, if an element x is composed of the vector of variables $n = \{n_1, n_2, \dots, n_n\}$, the new variables of the vector $m = \{m_1, m_2, \dots, m_m\}$ will have the representation that we can observe in Eq. (A.1). We can observe that each variable is weighted with a weight α. A variable with a higher weight has more importance in that component. 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.

$$m_i = n_1 * \alpha_1 + n_2 * \alpha_2 + n_3 * \alpha_3 + \cdots + n_n * \alpha_n \tag{A.1}$$

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

In this work, PCA will allow us to reduce the sizes of simple events and generate accurate CEP rules with a reduced sample space. This is a novelty that offers promising results.

Appendix B. Mahalanobis distance

The Mahalanobis distance is a distance function that takes into account the covariance between the variables. The main advantage of the Mahalanobis distance is that it weights the scale differences that may exist between the different variables as well as the correlation that may exist between them (although this last property is not needed in our case because the components are linearly independent). In this proposal the Mahalanobis distance allows us to include an additional mechanism that determines the similarity of a simple event with respect to the different families of known attacks.

$$d(x-\mu) = \sqrt{(x-\mu)^T\, \Sigma^{-1}\, (x-\mu)} \tag{B.1}$$

Eq. (B.1) represents the calculation of the difference between the element x and the mean of a category μ. Σ^{-1} represents the inverse covariance matrix, which allows a transformation to be performed so that the covariances have a weight in the distance function.

In this case we apply the Mahalanobis distance to simple events reduced with PCA, and this allows us to modify the distance function to take into account the explained variance ratio of the different components. 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. We can see how it is obtained in Eq. (B.2). Eq. (B.3) shows the Mahalanobis distance weighted with the explained variance ratios.

$$VE = diag(rv_1, rv_2, \ldots, rv_m) \tag{B.2}$$

$$d(x-\mu) = \sqrt{(x-\mu)^T\,(\Sigma^{-1} \times VE)\,(x-\mu)} \tag{B.3}$$

Thus, Eq. (B.3) allows us to know the category closest to a new simple event that does not fit any of the current CEP rules. It is important to understand that this function does not define the threshold used for the CEP rules, but a proximity function that provides an additional mechanism to know the closest category of an event that may be an anomaly. This mechanism can be useful to act against that anomaly on the basis of the closest known category.

References

AlZubi, A.A., Al-Maitah, M., Alarifi, A., 2021. Cyber-attack detection in healthcare using cyber-physical system and machine learning techniques. Soft Comput. 25 (18), 12319–12332. http://dx.doi.org/10.1007/s00500-021-05926-8.

Anon, 2022. Query guide - Siddhi. URL https://siddhi.io/en/v5.1/docs/query-guide/. (Accessed 01 August 2022).

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.

Asghari, P., Rahmani, A.M., Javadi, H.H.S., 2019. Internet of Things applications: A systematic review. Comput. Netw. 148, 241–261. http://dx.doi.org/10.1016/j.comnet.2018.12.008.

Bank, D., Koenigstein, N., Giryes, R., 2021. Autoencoders. arXiv:2003.05991 [cs, stat].

Bruns, R., Dunkel, J., 2022. Bat4CEP: a bat algorithm for mining of complex event processing rules. Appl. Intell. http://dx.doi.org/10.1007/s10489-022-03256-2.

Calvo, I., Merayo, M.G., Núñez, M., 2019. A methodology to analyze heart data using fuzzy automata. J. Intell. Fuzzy Systems 37 (6), 7389–7399. http://dx.doi.org/10.3233/JIFS-179348.

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. Comput. Stand. Interfaces 70, 103426. http://dx.doi.org/10.1016/j.csi.2020.103426.

Cugola, G., Margara, A., 2012. Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. 44 (3), 15:1–15:62. http://dx.doi.org/10.1145/2187671.2187677.

De Maesschalck, R., Jouan-Rimbaud, D., Massart, D.L., 2000. The Mahalanobis distance. Chemometr. Intell. Lab. Syst. 50 (1), 1–18. http://dx.doi.org/10.1016/S0169-7439(99)00047-7.

Demeter, D., Preuss, M., Shmelev, Y., 2019. IoT: a malware story - Securelist. https://securelist.com/iot-a-malware-story/94451/. (Accessed 23 December 2021).

Devi, R., Manoharan, P., Jangir, P., Elkotb, M., Madurai Elavarasan, R., Nisar, K., 2021. IRKO: An improved Runge-Kutta optimization algorithm for global optimization problems. Comput. Mater. Contin. 70, 4803–4827. http://dx.doi.org/10.32604/cmc.2022.020847.

Geurts, P., Ernst, D., Wehenkel, L., 2006. Extremely randomized trees. Mach. Learn. 63 (1), 3–42. http://dx.doi.org/10.1007/s10994-006-6226-1.

Ghasemi, M., Akbari, M.-A., Jun, C., Bateni, S.M., Zare, M., Zahedi, A., Pai, H.-T., Band, S.S., Moslehpour, M., Chau, K.-W., 2022. Circulatory system based optimization (CSBO): an expert multilevel biologically inspired meta-heuristic algorithm. Eng. Appl. Comput. Fluid Mech. 16 (1), 1483–1525. http://dx.doi.org/10.1080/19942060.2022.2098826, Publisher: Taylor & Francis.

Gupta, D., Dhar, A.R., Roy, S.S., 2021. A partition cum unification based genetic-firefly algorithm for single objective optimization. Sādhanā 46 (3), 121. http://dx.doi.org/10.1007/s12046-021-01641-0.

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. http://dx.doi.org/10.1109/ACCESS.2019.2924045.

Kaspersky, 2019. IoT under fire: Kaspersky detects more than 100 million attacks on smart devices in H1 2019. www.kaspersky.com, URL https://www.kaspersky.com/about/press-releases/2019_iot-under-fire-kaspersky-detects-more-than-100-million-attacks-on-smart-devices-in-h1-2019. (Accessed 23 December 2021).

Kaspersky, 2021. Kaspersky security bulletin 2020–2021. EU statistics. URL https://securelist.com/kaspersky-security-bulletin-2020-2021-eu-statistics/102335/. (Accessed 23 December 2021).

Kayacik, H.G., Zincir-Heywood, A.N., Heywood, M.I., 2005. Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets. In: Proceedings of the Third Annual Conference on Privacy, Security and Trust, Vol. 94. Citeseer, pp. 1–6.

Kumar, S., Gupta, S., Arora, S., 2021. Research trends in network-based intrusion detection systems: A review. IEEE Access 9, 157761–157779. http://dx.doi.org/10.1109/ACCESS.2021.3129775, Conference Name: IEEE Access.

Kwak, S.G., Kim, J.H., 2017. Central limit theorem: the cornerstone of modern statistics. Korean J. Anesthesiol. 70 (2), 144–156.

Lawal, M.O., 2021. Tomato detection based on modified YOLOv3 framework. Sci. Rep. 11 (1), 1447. http://dx.doi.org/10.1038/s41598-021-81216-5, Number: 1 Publisher: Nature Publishing Group.

Li, Y., Zhang, T., Song, C., 2019. Research on generation algorithm of complex event processing rules based on time series. In: 2019 Eleventh International Conference on Advanced Computational Intelligence. ICACI, pp. 124–128. http://dx.doi.org/10.1109/ICACI.2019.8778586.

Luckham, D., 2012. Event Processing for Business: Organizing the Real-Time Enterprise. John Wiley & Sons, New Jersey, USA.

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, (ISSN: 2325-6362) pp. 51–56. http://dx.doi.org/10.1109/EDOC49727.2020.00016.

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. http://dx.doi.org/10.1109/ICTech55460.2022.00056.

Van der Maaten, L., Hinton, G., 2008. Visualizing data using t-SNE. J. Mach. Learn. Res. 9 (11).

Martinez, A., Kak, A., 2001. PCA versus LDA. IEEE Trans. Pattern Anal. Mach. Intell. 23 (2), 228–233. http://dx.doi.org/10.1109/34.908974.

Martins, I., Resende, J.S., Sousa, P.R., Silva, S., Antunes, L., Gama, J., 2022. Host-based IDS: A review and open issues of an anomaly detection system in IoT. Future Gener. Comput. Syst. 133, 95–113. http://dx.doi.org/10.1016/j.future.2022.03.001.

Matkovic, F., Ivasic-Kos, M., Ribaric, S., 2022. A new approach to dominant motion pattern recognition at the macroscopic crowd level. Eng. Appl. Artif. Intell. 116, 105387. http://dx.doi.org/10.1016/j.engappai.2022.105387.

Mondragón-Ruiz, G., Tenorio-Trigoso, A., Castillo-Cara, M., Caminero, B., Carrión, C., 2021. An experimental study of fog and cloud computing in CEP-based real-time IoT applications. J. Cloud Comput. 10 (1), 32. http://dx.doi.org/10.1186/s13677-021-00245-7.

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. ICCKE, pp. 163–168. http://dx.doi.org/10.1109/ICCKE54056.2021.9721525.

OASIS, 2019. MQTT Version 5.0. http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. (Accessed 23 December 2021).

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. Comput. Stand. Interfaces 81, 103604. http://dx.doi.org/10.1016/j.csi.2021.103604.

Garcia-de Prado, A., Ortiz, G., Boubeta-Puig, J., 2017. COLLECT: COLLaborativE ConText-aware service oriented architecture for intelligent decision-making in the Internet of Things. Expert Syst. Appl. 85, 231–248. http://dx.doi.org/10.1016/j.eswa.2017.05.034.

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, Association for Computing Machinery, New York, NY, USA, pp. 126–135. http://dx.doi.org/10.1145/3465480.3466928.

Roesch, M., 1999. Snort – Lightweight intrusion detection for networks.

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 Syst. Appl. 149, 113251. http://dx.doi.org/10.1016/j.eswa.2020.113251.

Roldán-Gómez, J., 2023. An Automatic Complex Event Processing Rules Generation System for the Recognition of Real-Time IoT Attack Patterns (Dataset). Mendeley data, https://data.mendeley.com/datasets/f6sknjshzy/draft?a=eff59f7b-a7e6-4b20-8673-b9c8ab1861d8.

Roldán-Gómez, J., Boubeta-Puig, J., Castelo Gómez, J.M., Carrillo-Mondéjar, J., Martínez Martínez, J.L., 2021. 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, (ISSN: 2577-1655) pp. 1919–1926. http://dx.doi.org/10.1109/SMC52423.2021.9658711.

Roldán-Gómez, J., Carrillo-Mondéjar, J., Castelo Gómez, J.M., Ruiz-Villafranca, S., 2022. Security analysis of the MQTT-SN protocol for the Internet of Things. Appl. Sci. 12 (21), 10991. http://dx.doi.org/10.3390/app122110991, Number: 21 Publisher: Multidisciplinary Digital Publishing Institute.

Roy, A.M., 2022. Adaptive transfer learning-based multiscale feature fused deep convolutional neural network for EEG MI multiclassification in brain–computer interface. Eng. Appl. Artif. Intell. 116, 105347. http://dx.doi.org/10.1016/j.engappai.2022.105347.

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 Acad. J. 1 (2), 1–7.

Savakar, D., Kannur, A., 2019. An extremely randomized trees method for weapons classification based on wound patterns of sharp metals using ultrasound images. Iran J. Comput. Sci. 2, http://dx.doi.org/10.1007/s42044-019-00036-z.

Schölkopf, B., Smola, A., Müller, K.-R., 1997. Kernel principal component analysis. In: Gerstner, W., Germond, A., Hasler, M., Nicoud, J.-D. (Eds.), Artificial Neural Networks — ICANN'97. In: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 583–588. http://dx.doi.org/10.1007/BFb0020217.

Shah, S.A.R., Issac, B., 2018. Performance comparison of intrusion detection systems and application of machine learning to Snort system. Future Gener. Comput. Syst. 80, 157–170. http://dx.doi.org/10.1016/j.future.2017.10.016.

Simsek, M.U., Yildirim Okay, F., Ozdemir, S., 2021. A deep learning-based CEP rule extraction framework for IoT data. J. Supercomput. 77 (8), 8563–8592. http://dx.doi.org/10.1007/s11227-020-03603-5.

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 Commun. Surv. Tutor. 22 (2), 1191–1221. http://dx.doi.org/10.1109/COMST.2019.2962586.

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. http://dx.doi.org/10.1109/ICDCS47774.2020.00176.

Volnes, E., Plagemann, T., Goebel, V., Kristiansen, S., 2021. EXPOSE: Experimental performance evaluation of stream processing engines made easy. In: Nambiar, R., Poess, M. (Eds.), Performance Evaluation and Benchmarking. In: Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 18–34. http://dx.doi.org/10.1007/978-3-030-84924-5_2.

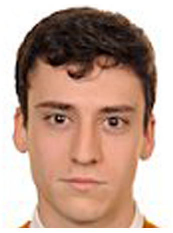
Warburton, D., 2021. DDoS Attack Trends for 2020. F5 Labs, URL https://www.f5.com/labs/articles/threat-intelligence/ddos-attack-trends-for-2020. (Accessed 23 December 2021).

Wong, K., Dillabaugh, C., Seddigh, N., Nandy, B., 2017. Enhancing Suricata intrusion detection system for cyber security in SCADA networks. In: 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering. CCECE, pp. 1–5. http://dx.doi.org/10.1109/CCECE.2017.7946818.

Xanthopoulos, P., Pardalos, P.M., Trafalis, T.B., Xanthopoulos, P., Pardalos, P.M., Trafalis, T.B., 2013. Linear discriminant analysis. Robust Data Mining 27–33.

Zhang, Y., Gao, S., Cai, P., Lei, Z., Wang, Y., 2023. Information entropy-based differential evolution with extremely randomized trees and LightGBM for protein structural class prediction. Appl. Soft Comput. 136, 110064. http://dx.doi.org/10.1016/j.asoc.2023.110064.

Zhang, Y., Liu, Q., 2022. On IoT intrusion detection based on data augmentation for enhancing learning on unbalanced samples. Future Gener. Comput. Syst. 133, 213–227. http://dx.doi.org/10.1016/j.future.2022.03.007.

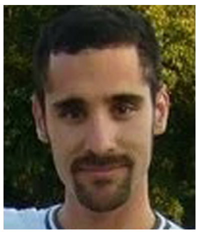


José Roldán-Gómez obtained a bachelor's degree in Computer Engineering from the University of Castilla-La Mancha in 2017, a master's degree in Computer Engineering from the University of Castilla-La Mancha in 2018, and is currently a doctoral candidate at the University of Castilla-La Mancha. He is also an interim professor at the University of Oviedo. His main interests are artificial intelligence applied to threat detection in IoT environments and automatic rule generation in CEP engines.

Juan Boubeta-Puig received a 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.

Javier Carrillo-Mondéjar received a B.Sc. and M.Sc. degrees in Computer Science and Engineering and a Ph.D. degree in Advanced Computing Technologies from the University of Castilla-La Mancha, Spain, in 2016, 2017 and 2022, respectively. In 2016, he joined the High-Performance Networks and Architectures (RAAP) group of the Informatics Research Institute of Albacete (I3A) as a Research Assistant. His research interests are related to malware detection and classification techniques, with a particular focus on IoT/firmware cybersecurity. He has also been a visiting researcher at King's College London for 5 months and the University of Jyväskylä for 3 months.

Juan Manuel Castelo Gómez joined the Computer Architecture and Technology research group at the Albacete Research Institute of Informatics in 2016. In 2017, he received his M.Sc degree in Computer Science from the University of Castilla La Mancha (Spain), and in 2021 obtained his Ph.D in Advanced Information Technologies from the aforementioned university. His research interests are related to cybersecurity, especially digital forensics, as well as malware detection.



Jesús Martínez del Rincón 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 B.Sc. in Telecommunication Engineering in 2003 and was awarded a Ph.D. 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.