UNIVERSITY OF CASTILLA-LA MANCHA

Computing Systems Department

Intelligent threat detection in Internet of Things environments

A dissertation for the degree of Doctor of Philosophy in Computer Science to be presented with due permission of the Computing Systems Department, for public examination and debate.

Author: D. José Roldán Gómez
Advisor: Dr. D. José Luis Martínez Martínez
Dr. D. Juan Boubeta Puig

Albacete, March 2023

UNIVERSIDAD DE CASTILLA-LA MANCHA

Departamento de Sistemas Informáticos

Intelligent threat detection in Internet of Things environments

Tesis Doctoral presentada al Departamento de Sistemas Informáticos de la Universidad de Castilla-La Mancha para la obtención del título de Doctor en Tecnologías Informáticas Avanzadas.

Autor: D. José Roldán Gómez
Director: Dr. D. José Luis Martínez Martínez
Dr. D. Juan Boubeta Puig

Albacete, Marzo de 2023

A mis abuelos

Agradecimientos

Quisiera agradecer a mis padres, José y María Teresa, por su apoyo constante y por creer en mí desde el principio. Sin su amor y su guía, no habría sido posible llegar hasta aquí. También quiero agradecer a mi hermana Noelia por ser un apoyo.

Mis directores de tesis, Juan Boubeta y José Luis Martínez, han sido una fuente constante de sabiduría y orientación durante mi investigación. Les estoy muy agradecido por su paciencia y dedicación.

Mis compañeros de grupo, Javi, David, Juanma, Sergio, Rubén y Carlos, han sido un gran equipo y una fuente de inspiración durante el desarrollo de la tesis. A mis amigos, especialmente a Pedro, Fer, Panadero y David, por su apoyo y su amistad durante todos estos años.

También quiero agradecer a Jaime Y Jesús por su ayuda durante la estancia y a los compañeros de La universidad de Oviedo que me acogieron como a uno más desde el primer momento.

Por último, quiero agradecer a mis profesores de carrera, colegio e instituto por enseñarme tanto y por prepararme para este momento. Sin su dedicación y su pasión por la educación, no habría llegado hasta aquí.

Gracias a todos por estar a mi lado en este camino.

Este trabajo ha sido cofinanciado por el Ministerio de Economía y Competitividad y la Comisión Europea (fondos MINECO/FEDER), bajo el proyecto con referencia RTI2018-098156-B-C52, por la Consejería de Educación, Cultura y Deportes de la Junta de Comunidades de Castilla-La Mancha, mediante el proyecto con referencia SBPLY/17/180501/000353, y por el Ministerio de Educación, Cultura y Deporte a través de la beca FPU 17/03105.

Summary

The Internet of Things (IoT) has experienced a dizzying growth. The applications of the IoT are many and diverse, ranging from enabling a "smart" home to monitoring industrial processes to optimizing traffic patterns. This unprecedented growth has caused it to become a very attractive target for cyber criminals. This is especially worrisome in this paradigm because there are certain intrinsic limitations. Among these are the fact that the devices involved are often resource-constrained, which means that they have little processing power and memory. This makes them difficult to protect. Additionally, these devices are often not well-maintained, meaning that they might be using outdated software that is vulnerable to known exploits. It is therefore necessary to design new solutions or adapt traditional solutions that take into account the characteristics of the paradigm. The objective of this PhD thesis is to design, implement and evaluate an architecture that allows the detection of known and unknown threats in IoT environments in real time. Furthermore, it is intended that this architecture can detect such attacks with the least possible intervention.

Complex Event Processing (CEP) allows the processing and correlation of a large amount of data in real time. To achieve this, an expert defines a set of rules, called CEP rules, and when simple events, which contain information necessary to detect situations of interest, comply with these rules, a complex event is triggered.

First, an architecture capable of generating CEP rules to detect IoT threats in real time is designed, implemented and evaluated. However, this architecture requires an expert to specify the most important fields of the protocols to be monitored. It is also necessary that the network traffic, with which we train our architecture, is labeled, i.e., that the architecture knows which packets are attacks when training.

So next, the above architecture is improved by eliminating the need for a domain expert to identify key fields , and then it is updated to enable it to generate the rules without the need for tagged traffic. The results obtained throughout the Thesis support the viability of all the proposals we present, as they show that the different architectures achieve good results from a functional and performance point of view. We can conclude that the proposals described are viable.

Resumen

El IoT ha experimentado un crecimiento vertiginoso. Las aplicaciones del IoT son muchas y diversas, y van desde la habilitación de un hogar inteligente hasta la monitorización de procesos industriales o la optimización de patrones de tráfico. Este crecimiento sin precedentes ha hecho que se convierta en un objetivo muy interesante para los ciberdelincuentes. Esto es especialmente preocupante en este paradigma porque existen ciertas limitaciones intrínsecas que pueden complicar la implementación de medidas en este paradigma. Entre ellas, el hecho de que estos dispositivos suelen tener recursos limitados, lo que significa que tienen poca capacidad de procesamiento y memoria. Además, estos dispositivos no suelen estar bien mantenidos y pueden utilizar software obsoleto que es vulnerable a exploits conocidos. Por tanto, es necesario diseñar nuevas soluciones o adaptar las tradicionales teniendo en cuenta las características del paradigma. El objetivo de esta tesis doctoral es diseñar, implementar y evaluar una arquitectura que permita detectar amenazas conocidas y desconocidas en entornos IoT en tiempo real. Además, se pretende que esta arquitectura pueda detectar dichos ataques con la menor intervención posible.

CEP permite procesar y correlacionar una gran cantidad de datos en tiempo real. Para ello, un experto define un conjunto de reglas, denominadas reglas CEP, cuando los eventos simples, que contienen la información necesaria para detectar situaciones de interés, cumplen con estas reglas, se dispara un evento complejo.

Primero, se diseña, implementa y evalúa una arquitectura capaz de generar reglas CEP para detectar amenazas IoT en tiempo real; sin embargo, esta arquitectura requiere que un experto especifique los campos más importantes de los protocolos a monitorizar. También es necesario que el tráfico de red, con el que entrenamos nuestra arquitectura, esté etiquetado, es decir, que la arquitectura sepa qué paquetes son ataques a la hora de entrenar.

A continuación, se actualiza y mejora la arquitectura anterior. La mejora de esta implementación es que elimina la necesidad de un experto en el dominio para identificar los campos clave. Por último, se actualiza la arquitectura anterior para poder generar las reglas sin necesidad de tráfico etiquetado. Los resultados obtenidos a lo largo de la tesis avalan la viabilidad de todas las propuestas que presentamos. Los resultados obtenidos muestran que las diferentes arquitecturas consiguen buenos resultados desde el punto de vista funcional y de rendimiento. Podemos concluir que las propuestas descritas son viables.

Contents

List of Figures

List of Tables

List of Acronyms

CEP	Complex Event Processing
CSV	Comma Separated Values
EPL	Event Processing Language
ESB	Enterprise Service Bus
GGS	GII-GRIN-SCIE
GMM	Gaussian Mixture Models
IEEE	Institute of Electrical and Electronics Engineers
IF	Impact Factor
IoT	Internet of Things
IPCA	Incremental Principal Component Analysis
JCR	Journal Citation Reports
JNIC	Jornadas Nacionales de Investigación en Ciberseguridad
JSON	JavaScript Object Notation
KPCA	Kernel Principal Component Analysis
LR	Linear Regression
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
MQTT-SN	Message Queuing Telemetry Transport for Sensor Networks
PCA	Principal Component Analysis
SOA	Service Oriented Architecture
SVR	Support Vector Regression
TCP	Transmission Control Protocol

Telnet Teletype Network

UDP User Datagram Protocol

CHAPTER 1

Introduction

This chapter introduces this Doctoral Thesis, which is entitled "Intelligent threat detection in IoT environments". The current state of threat detection in Internet of Things (IoT) environments is presented first. This allows us to motivate the development of this work, formulate the hypotheses and to justify the decisions taken in this work. Furthermore, the general and specific objectives are defined together with the methodology followed to achieve them. Finally, the proposed solutions and the results obtained by them are briefly described.

1.1 Motivation and Justification

The IoT has grown rapidly in the last decade and it does not seem that this growth will stop or slow down any time soon, due to the obvious potential offered by this new paradigm. Proof of this is the growing number of interactions with certain applications oriented to this paradigm through devices such as smartphones or wearables. This new paradigm has proven to be useful in a myriad of contexts, for example, healthcare applications, home automation, intelligent resource management, among many others [1, 2, 3, 4].

Tangible proof of the paradigm's potential, which shows that it is already a reality, is that IoT devices now outnumber non-IoT devices connected to the Internet. This overtaking occurred for the first time in 2020 [5], even though the pandemic caused a delay in the paradigm's growth rate due to chip shortages. Nevertheless, there are currently 12,300 million connected devices and the number is expected to reach 27,000 million by 2025 [6].

Despite the dizzying growth of the paradigm, there are certain intrinsic characteristics that tend to limit the implementation or adaptation of traditional security solutions. Firstly, we can highlight the heterogeneity that IoT systems present. There are very different devices connected to each other, as well as protocols from different manufacturers and with different objectives. This results in a level of heterogeneity that must be taken into account when implementing solutions in this paradigm [7].

Another problem with this paradigm is the limited resources, at all levels, that devices tend to have. These devices usually have a low computational capacity, which can complicate the deployment of certain basic features in traditional systems. For example, it is often difficult to deploy communications encryption systems [8]. This limitation means that the solutions implemented in this paradigm must be especially light from a computational point of view.

Unfortunately, the limitations are not only present in the devices. It should be noted that there are IoT system deployments in very different locations. This results in a very common problem of low bandwidth in many networks in this paradigm [9]. This means that this peculiarity must be taken into account when deploying systems within the networks. Therefore, a common characteristic of many protocols intended for this paradigm is that they focus on reducing the overhead they introduce. However, this often comes at the expense of doing without other desirable features. The fact that IoT systems can be deployed in a wide range of applications makes it necessary to use batteries in some specific contexts [10]. Again, this is an important limitation, as the power consumption that these batteries may suffer when an IoT solution is deployed within these contexts must be taken into account. Another disruptive feature, with respect to the traditional paradigm, is the ubiquity of the devices. In many applications, devices can enter and leave the system periodically or change their position, and this must be taken into account when designing networks and systems in this paradigm [11].

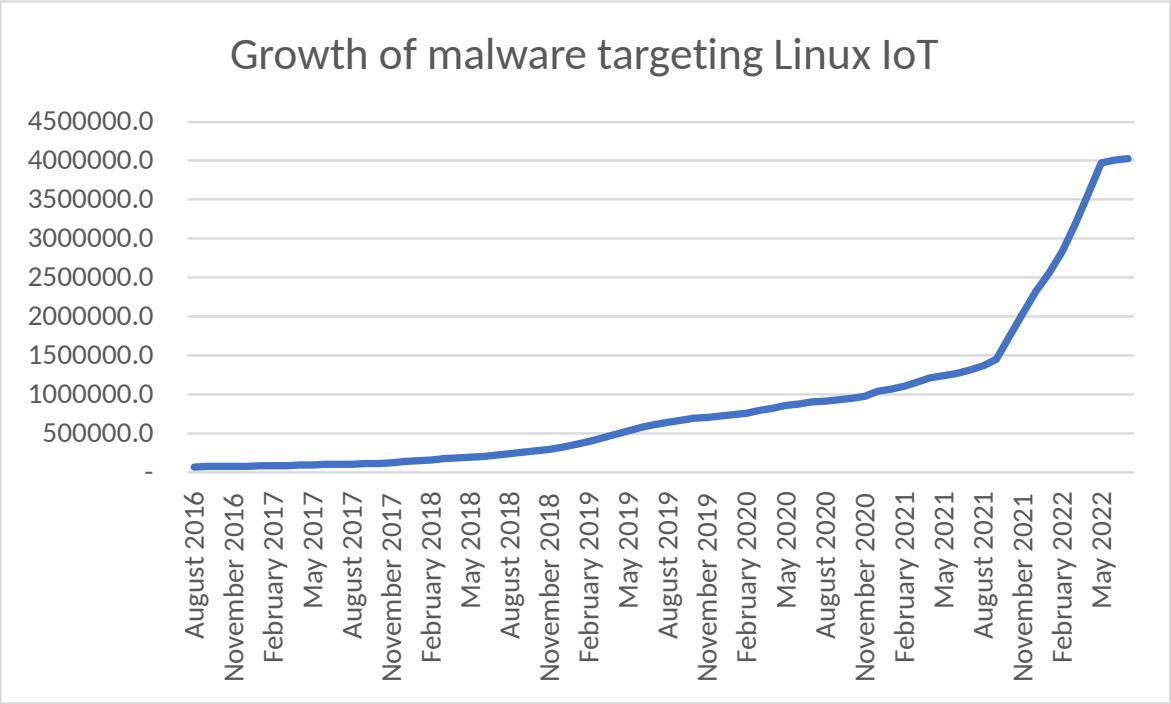


Figure 1.1: Quarterly growth of malware targeting Linux IoT.

As the number of IoT devices connected to the network has grown, so has the interest of criminals in this type of device. This is evidenced by the data shown in Figure 1.1, in which

we can see the quarterly growth in malware samples targeting Linux IoT devices. From August 2016 to May 2022 there was a huge increase, with fifty-nine times more malware samples targeting IoT [12].

This rapid growth is compounded by the aforementioned limitations of IoT devices. This disables many common solutions that require a lot of resources to function properly. Therefore, we can conclude that it is necessary to make an effort to find new solutions, or adapt typical solutions, to deal with these threats in this type of environment.

We can measure security within the IoT, and any paradigm, by defining three categories of cybersecurity. These are confidentiality, integrity and availability. Within the field of cybersecurity, confidentiality refers to the prevention of unauthorized access to data. In order to maintain confidentiality, data must be protected from unauthorized individuals. There are a variety of ways to protect data, including encryption, access control, and data masking. Integrity, in the same context, refers to the prevention of unauthorized modification of data. To maintain data integrity, data must be protected from unauthorized individuals who may attempt to alter it. Data integrity can be protected through the use of cryptographic hashes, digital signatures, and data validation. Availability, finally, refers to the prevention of data loss. To ensure data availability, data must be backed up and stored in a secure location.

One way to address the three categories of cyber security is by detecting threats in real time, and within the field of threat detection there are two different categories. The more classical detectors are based on signatures and focus on checking a static signature, which can be a hash, or any other element that univocally identifies a specific threat. In the other category we find the behavior-based threat detectors. These detectors learn from the behavior of different threats and are able to identify this behavior later to detect these threats. Although signature-based detectors are widely used, they are not as flexible as behavior-based detectors. This is because a small modification of the threat can cause the signatures to mismatch.

As a general rule, behavior-based threat detectors make use of techniques that allow them to generate models that represent the attacks to be detected. This set of techniques are grouped in a field known as Machine Learning (ML) [13]. Within ML techniques there are two different categories. The first category is supervised learning, where the ML algorithm is given a set of training data that includes the desired output, also known as the label. The algorithm then learns to generate the desired output from the labeled input data. The second category is unsupervised learning, where the ML algorithm is given a set of data but not told what the desired output should be. The algorithm then has to learn to find patterns and structures in the data in order to generate the desired output. It is therefore necessary to focus research on finding solutions to these problems.

This Doctoral Thesis focuses on network attack detection in IoT environments. These attacks can be generated by a malicious device or by malware instances. There are several works that try to solve this problem [14, 15, 16, 17, 18, 19, 20, 21, 22], the trend in the

related research is to make use of ML algorithms, which are used because they allow the detection of attacks on the basis of behavior. This offers a large degree of flexibility compared with signature-based solutions, especially in such a dynamic and changing environment. These works focus on applying different ML algorithms to detect IoT threats. The results obtained, from a functional point of view, are very promising. The models seem to work well and detect attacks correctly. However, deploying these models in IoT environments presents a number of challenges and issues. These problems are mainly due to the intrinsic limitations of IoT devices and systems. Deploying these models usually requires considerable computational capacity and, in general, they need a large amount of information to be trained correctly. IoT devices do not usually have a very large computational capacity and deploying a high volume of information can be problematic in networks with low bandwidth.

Due to the inherent limitations of IoT, it is necessary to look for a technology that can correlate a large amount of information in real time and be successfully deployed in IoT environments despite their limitations. A technology that meets these two requirements is Complex Event Processing (CEP). CEP has been designed to detect situations of interest by processing and correlating a large amount of information that can be extracted in a specific context. In this way this information, which has not been processed by CEP, is divided into simple events. The first step in deploying a CEP solution is to define the simple events and the information they contain. When this step has been carried out, it is necessary for a domain expert to define the patterns that these simple events follow in order to automatically detect each situation of interest. Each of these situations will be triggered by one or more CEP rules, which are defined by the domain expert and implemented by using an Event Processing Language (EPL). An EPL is a language that allow us to define the conditions of the rules. When the simple events meet the requirements defined in a CEP rule, a complex event is triggered by summarizing the situations of interest that have occurred. This technology is ideal for detecting threats in IoT environments, but it has one major limitation, namely the need for a domain expert who is able to define CEP rules manually to detect situations of interest. The challenge is to develop an architecture that can generate CEP rules automatically in order to overcome this limitation.

There are multiple papers that focus on defining CEP rules automatically [23, 24, 25, 26]. Most of these papers succeed in generating CEP rules satisfactorily. However, these works do not focus on detecting attacks in IoT environments in real time, so perhaps this is why they do not focus on the computational performance of the rules generated. In this thesis we focus on designing an architecture that generates rules that are computationally and functionally effective and efficient. We will then use these rules to detect known and unknown attacks. In this context we assume that an attack is known when a rule for that attack has been generated during the training process. An attack is unknown when there are no samples of that attack in the training process.

The hypothesis to be tested in this thesis is that it is possible to integrate CEP with ML to automatically generate rules to detect attacks in real time in IoT environments. This

hypothesis, if confirmed, generates some interesting questions. The first is straightforward since the hypothesis is posed for an IoT scenario with the constraints described above (RQ1): Are the rules generated by the architecture computationally efficient in a constrained environment, such as an IoT network? If the performance is efficient, it means that it is feasible to deploy a threat detector in an IoT environment. The heterogeneity of this paradigm makes it very complicated to train our detector models with all possible attacks. This leads to the second question (RQ2): Is the proposal capable of detecting unknown attacks? There is also some heterogeneity within CEP technologies, which leads us to the third question (RQ3): Is this integration of technologies capable of working in different CEP implementations? This ability to adapt to different technologies is desirable because of the different CEP implementations that exist. Finally, a question that arises from the use of ML techniques is the following (RQ4): Is it possible for the models used to generate the rules to be trained in an unsupervised way?

1.2 Objectives

As a result of the different challenges described above, and taking into account both the hypothesis and the subsequent research questions, we can define the main objective of this thesis as the design, implementation and evaluation of a novel architecture that integrates CEP and ML to generate CEP rules to detect real-time network attacks in IoT environments. Furthermore, such rules should be able to detect known and unknown threats.

To achieve this general objective, we will work incrementally on different specific objectives. In this way we focus on a specific problem in each iteration of the architecture. The specific objectives in question are the following:

- **Objective 1**. We will study and analyze the solutions proposed by other researchers. This objective is transversal and cyclical, so that in each advance proposed in this thesis a process of study of the state of the art will be carried out. This will allow us to know the results obtained by other researchers and to focus on working with the technologies and techniques that offer good results, comparing the advantages and disadvantages of the existing solutions with respect to our proposal. This way of proceeding is ideal to obtain a multiperspective view of our contributions within the state of the art.

- **Objective 2**. We will analyze common IoT protocols, design, implement and deploy a test environment. This step is fundamental to obtain an environment in which we will experiment with the different solutions that will be proposed later. It is necessary to generate a scenario with well-defined normal behavior and scenarios in which this behavior is attacked with different threats. This will provide us with the data to work with and a way to validate our solutions that we will use throughout the thesis. Thus, in an iterative way, we will use these data to define different experiments depending on the architecture we want to evaluate.

- **Objective 3**. We will design, implement, deploy and validate an architecture, that integrates cutting-edge technologies such as CEP and ML in order to automatically detect threats in IoT environments on the basis of one or more key features. These features refer to characteristics of the elements that may or may not be a threat. For example, the size of the network packet, or the protocol to which the packet belongs, among others. It is common for security experts to know the most common indicators of an attack in a protocol. But sometimes quantifying the values of these indicators is a complicated task, especially in the face of unknown attacks. Therefore, it is important to establish an initial architecture that solves this problem and is functional and efficient in IoT environments. This will not solve the problem in its entirety, but it will address many real situations and allow us to establish a baseline to work from.

- **Objective 4**. We will ensure the feasibility of the initial architecture in different CEP implementations. It is useful to ensure the independence of the proposed architectures for threat detection in IoT environments with respect to a specific CEP engine. In this way we will know that we can adapt our architectures to different CEP engines. An analysis of the performance of the different CEP engines in detecting network attacks will also be performed. This will allow us to ensure their efficiency from a computational performance point of view.

- **Objective 5**. We will redesign, implement and validate a new architecture to detect threats in IoT environments without the need to specify key features and achieve improved computational performance with respect to the initial architecture proposed. This will be an obvious improvement because it allows security experts to deploy an intelligent threat detector that is able to generate its own rules whenever there is labeled network traffic data. However, it will still be necessary to obtain labeled network traffic to train the system.

- **Objective 6**. We will update, improve, implement and validate a new architecture capable of detecting threats in IoT environments without the need to specify key features in an unsupervised way, i.e, with unlabeled data. This improvement will allow us to obtain an intelligent threat detector in a completely unsupervised way. In addition, the performance improvements achieved in the previous proposal will be maintained.

1.3 Methodology and Work Plan

This section describes the methodology that will be used to achieve each of the objectives proposed in Section 1.2. It describes the milestones that must be reached in order to achieve the main objective of the Thesis. The tasks to be carried out to achieve each objective are as follows:

- **Objective 1** — To study and analyze the solutions proposed by other researchers.

The main function of this milestone is to understand the state of the art of IoT threat detection. In this way we can understand what technologies, techniques and methodologies are followed to work on this issue. This objective is especially useful in the early stages of the Doctoral Thesis in order to plan how to achieve of the rest of the objectives. However, the state of the art is analyzed throughout the whole working process of the PhD Thesis in order to update the existing knowledge with respect to new solutions proposed by other researchers.

- **Objective 2** —To analyze common IoT protocols, and design, implement and deploy a test environment.

  This objective focuses on obtaining a set of data that allows us to correctly evaluate our proposals.

  The Message Queuing Telemetry Transport (MQTT) protocol is used for this purpose, because it is widely used in IoT scenarios. A baseline scenario and attack scenarios are then generated using virtualization. By using a sniffing process, the packets related to each scenario are obtained. These packets will allow us to obtain the data that will be vital for the development of the Thesis. But they also represent an important added value in themselves.

  The tasks necessary to achieve the first part of this objective are:

  1. Perform a security analysis of the MQTT and Message Queuing Telemetry Transport for Sensor Networks (MQTT-SN) protocol to detect possible vulnerabilities.
  2. Deploy a virtual MQTT network with a legitimate baseline scenario.
  3. Implement and perform attacks against the MQTT network.
  4. Obtain the packets of the base scenario and of the attacks by means of a sniffing process.

- **Objective 3** — To Design, implement, deploy and validate an architecture to detect threats in IoT environments on the basis of one or more key features.

  A major problem in threat detection is the need to process a significant amount of data, preferably in real time. To create a threat detector that is able to be deployed in an IoT environment and correlate a large number of events per second, CEP is used. CEP needs defined rules to function and this leads to limitations.

  Sometimes it is even complicated for domain experts to define rules [27, 28]. Even if they are able to identify the characteristics to be monitored in a protocol, it is not a trivial task to define acceptable thresholds for such features in threat detection. These thresholds are used to check whether an item belongs to a category, in this case to check whether a network packet is a threat and the specific threat type. Therefore,

an architecture capable of detecting threats in IoT environments is designed to define these thresholds automatically. For this purpose, the following tasks are performed:

1. Design and implement an architecture capable of generating rule thresholds based on their key features.
2. Feature selection and preprocessing of labeled datasets. These will be used for training and validation of the architecture.
3. Training of the model used by the threat detector.
4. Definition of thresholds and generation of CEP rules for threat detection.
5. Evaluation of the CEP rules generated for threat detection.

- **Objective 4** — To ensure the feasibility of the initial architecture on different CEP engines.

  Ensuring that the architecture works well on a different CEP engine is an important step forward. For this purpose, the above architecture will be deployed on both the: Esper [29] and WSO2 [30] CEP engines. Both are widely used and stand out for being open source. For their EPL they use, EsperEPL [31] and SiddhiQL [32], respectively. In addition, a comparison of the two CEP engines under situations of stress is performed. This also provides an insight into how it can perform in an IoT scenario. The steps followed to achieve this milestone are as follows:

  1. Training of the predictor in charge of predicting the value of the key features.
  2. Deploy the architecture on Esper CEP and WSO2 CEP.
  3. Deploy the detection patterns on both CEP engines.
  4. Implement a simulator to modulate the sending of events to the CEP engines. The objective is for both CEP engines to be in similar situations in the experiments.
  5. Measure the computational performance of both CEP engines and compare the results.

- **Objective 5** — To redesign, implement and validate a new architecture to detect threats in IoT environments without the need to specify key features and achieve improved computational performance over the initial proposal.

  In certain circumstances it is difficult to choose a set of key features for a given protocol. In addition, since it is a threat detector designed for IoT environments, computational performance is critical if we want it to work in real time. To achieve this computational improvement, it is useful to reduce the number of features of simple events. However, if we do this directly, we could hinder the generation of models to detect attacks. Therefore, we use Principal Component Analysis (PCA) [33], which is an algorithm used to reduce the dimensionality of a sample space. In this case,

it is used to reduce the size of the simple events while maintaining the maximum amount of information possible. This allows us to generate rules from simple events with fewer attributes. This in turn means that the CEP engine has to process smaller simple events and less information is sent to the network. This constitutes an improvement in computational performance and a more efficient use of the network.

The tasks required to complete this objective are:

1. Redesign the architecture to generate complete CEP rules for threat detection.
2. Feature selection and preprocessing of datasets for PCA model training.
3. Training of the PCA model used to reduce the size of simple events.
4. Definition of thresholds for each CEP rule generated for attack detection.
5. Generation of CEP rules obtained by our architecture.
6. Deployment of the generated CEP rules and evaluation of their performance.

- **Objective 6** — To update, improve, implement and validate an architecture that can detect threats in IoT environments without the need to specify key features in an unsupervised manner.

  The architecture obtained in Objective 5 is useful for generating complete rules when there is previously-labeled traffic. However, this requirement is not always easy to meet. Therefore, a further improvement in the architecture is proposed that allows completely unsupervised CEP rule generation. These rules are able to detect attacks in IoT environments in real time and eliminate the need to have labeled traffic.

  This unsupervised learning mechanism is achieved through a clustering process using Gaussian Mixture Models (GMM) [34]. GMM is an algorithm whose objective is to find different clusters by establishing the normal distributions that represent each cluster. This allows us to generate anonymous families that enable the generation of different CEP rules for each traffic class. In addition, the mechanism for defining the thresholds has also been updated and Incremental Principal Component Analysis (IPCA) has been used [35]. This version of PCA allows the original model to be fed back to generate new models iteratively. The following steps are followed to achieve this objective:

  1. Design of the additional module to generate anonymous families.
  2. Update of the module used to define thresholds.
  3. Feature selection and preprocessing of datasets for IPCA model training.
  4. Training of the IPCA model used to reduce the size of simple events.
  5. Definition of thresholds for each CEP rule generated for attack detection.
  6. Generation of CEP rules obtained by our architecture.

7. Deployment of the generated CEP rules and evaluation of their performance.

1.4 General Discussion and Description of the Proposals

As mentioned above, the fundamental objective of the thesis is to develop an architecture that can detect threats in IoT environments and do so with as little supervision by a domain expert as possible. This section offers a brief analysis of the different proposals made throughout the Doctoral Thesis. First, Section 1.4.1 describes the attacks used to evaluate the different architectures and the metrics used in these evaluations. Maintaining a certain homogeneity in the datasets and metrics used in all the architectures allows us to know the performance of each of the proposed designs. Section 1.4.2 describes the initial proposed architecture, which is able to define thresholds for key features, allowing us to generate CEP rules to detect attacks in IoT environments. Section 1.4.3 describes the comparison of Esper CEP and WSO2 to demonstrate the possibility of deploying our proposals on various CEP engines. Section 1.4.4 describes the architecture capable of detecting threats in IoT environments by defining complete CEP rules using PCA. Finally, section 1.4.5 describes the improvement made to the architecture of section 1.4.4 that enables us to generate complete CEP rules in a totally unsupervised way.

1.4.1 Threats and evaluation metrics

This section describes the attacks found in the datasets used to evaluate the different architectures proposed. In addition, the common metrics used during the evaluation of the different proposals are also described.

The dataset that has been generated and which we will use throughout the Thesis is obtained by obtaining the traffic of an MQTT network, which is composed of 3 MQTT clients and an MQTT broker in charge of coordinating the sending and receiving of messages. In this network, a legitimate behavior of the network is simulated when it is not under attack. To obtain this scenario, the clients periodically post on 3 different topics, one per client. These published data have a numerical format, so that they can simulate temperatures or decibels. Subsequently, a new malicious client is introduced in the network, and this client is in charge of performing the attacks described below:

- **Subfuzzing**. This attack is designed to find the different topics existing in the MQTT broker. The usefulness of this attack lies in those scenarios where the attacker is able to introduce a client into the network but is not able to subscribe to the root topic. The attack consists of iterating over a dictionary or using brute force to try to subscribe to different topics. This makes it possible to discover the different topics in the system.
- **Disconnect wave**. The objective of this attack is to perform a denial of service of the MQTT network by disconnecting all legitimate clients from the MQTT network. This attack works in configurations where a new client can connect with an existing client *ID* and the broker kicks out the original client when this happens. To do this

the attacker iterates over the various network client identifiers while connecting to the broker with those identifiers. This causes the broker to eject legitimate clients.

- **Transmission Control Protocol (TCP) SYN scan**. This scanner is used to determine which TCP ports are open. Although this is not a specific attack, it is often used in the early stages of targeted attacks. To perform this scan, the attacker sends TCP *SYN* packets to the different ports to be scanned. If the port is open the victim will respond with a *SYN/ACK* packet, otherwise it will respond with an *RST* packet.

- **User Datagram Protocol (UDP) scan**. The UDP port scanner is used to detect open ports with UDP applications. In this case, if the victim responds with a packet of type Unreachable, the port is considered closed. If the victim sends another error, the port is considered filtered. Finally, if the victim sends nothing or sends something else, the port is considered open.

- **Teletype Network (Telnet) scan**. This scanner simulates the first stage of Mirai, which consists in performing a dictionary attack over Telnet to try to gain access to the device.

- **Xmas scan**. This attack also tries to find open TCP ports, and it can be a good alternative when firewalls block the TCP SYN scan. To perform this scan, TCP packets are sent with the PSH, FIN and URG flags set. If the victim responds with an RST packet, the port is considered closed. If the response is an unreachable error packet, the port is considered filtered. If the victim does not respond, the port may be filtered or open.

The dataset containing all the attacks and normal traffic has been published in the Mendeley repository [36].

The metrics used to enhance the evaluations are as follows:

- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1Score = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$

where TP is the true positives, FP is the false positives and FN is the false negatives.

Thus, a high *Recall* score means that a CEP rule detects events that actually belong to that family, and a high *Precision* score means that that CEP rule does not detect many false positives. Finally, *F1 Score* uses the two scores to obtain a metrict hat is balanced between them. In a cybersecurity context, it is usually a good idea to maintain a high *recall* value to detect threats, but if the *precision* value is too low the detector will generate too many false positives.

1.4.2 Architecture to detect threats in IoT environments on the basis of one or more key features

This section describes the initial architecture developed in the Thesis [37]. As discussed above, this proposal is able to define the thresholds of the CEP rules within which the key features should be based on the rest of the features. In this way, traffic can be classified to detect attacks in IoT environments.

Figure 1.2 shows a diagram of the proposed architecture, which is composed of two different layers. The first layer includes the processes that are executed at runtime (upper half of Figure 1.2), i.e., the behavior of our system, while the second layer contains the actions that are carried out during the design stage (lower half of Figure 1.2).

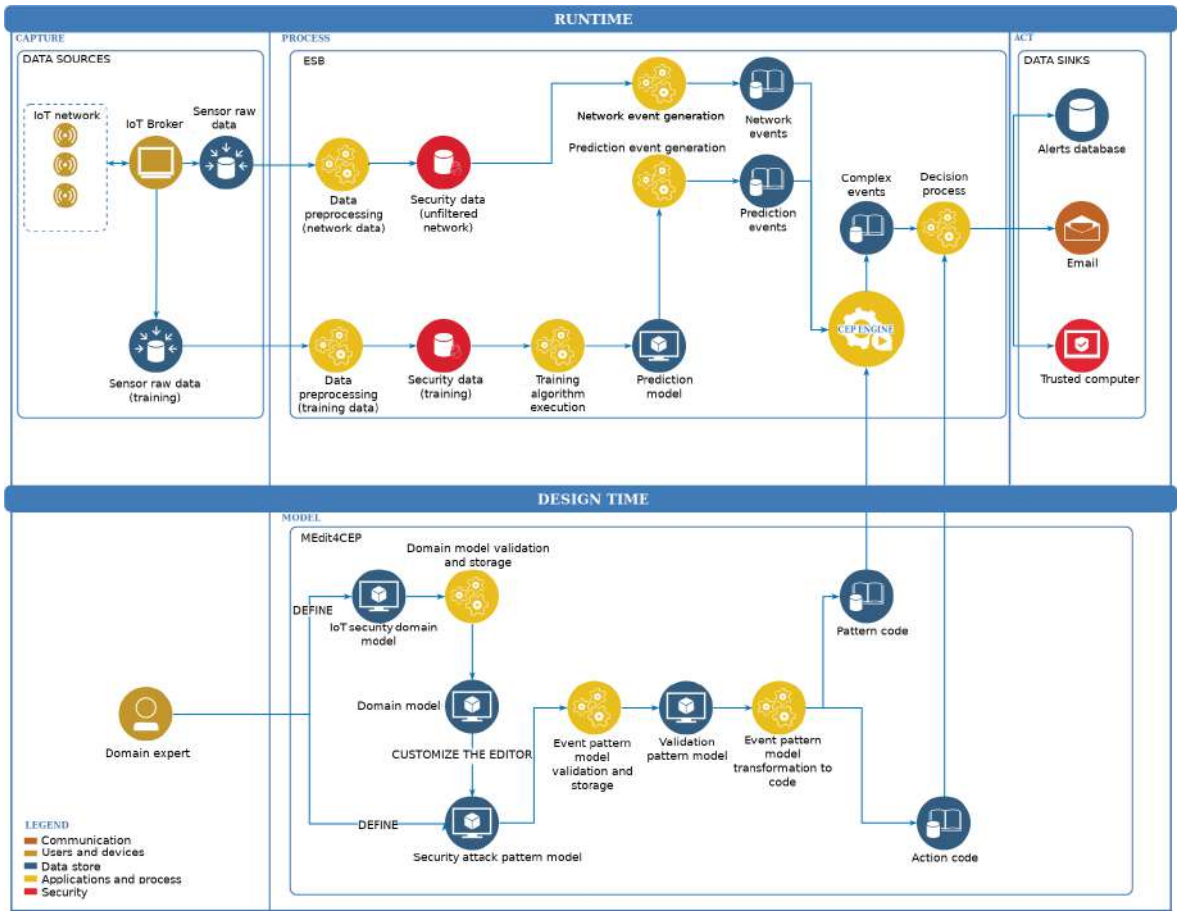


Figure 1.2: The proposed architecture for detecting IoT security attacks on the basis of key features.

Runtime Architecture Layer

The runtime layer is designed with a Service Oriented Architecture (SOA) 2.0 in mind, and this is responsible for: obtaining the data produced by the IoT network in real time; processing, analyzing and correlating such data to obtain relevant information; detecting

situations of interest and communicating these situations to the data sinks so that actions can be taken quickly.

In this layer the data from the dataset, which has been described in the previous section, is obtained and sent to the Enterprise Service Bus (ESB). Using the MQTT protocol. In this context, ESB is the centralizing component of the architecture that transports the data to the different stages of the architecture. In this way, the ESB receives the raw data from the sensors in real time, then preprocesses it and converts it into a common format for network events. The preprocessing also allows the training data to be consumable for our training ML algorithm (Linear Regression (LR) or Support Vector Regression (SVR)).

The ESB then produces a trained model, which allows us to generate *NetworkPrediction* events with a timestamp, a predicted value (of packet length in this case) and a threshold. Each packet in the network also has an associated *NetworkPrediction* event.

Thus, when the difference between the actual value and the prediction of a predicted feature is outside the threshold for this type of packet, the system will detect it. This will allow CEP to be used to detect attacks.

Subsequently, the CEP engine, which is integrated with the ESB, receives both network events and *NetworkPrediction* events, and this allows it to calculate the differences between actual values and predictions. Comparing this difference with the threshold allows it to detect security attacks (in the form of complex events) in real time. Once a complex event has been detected, the decision process quickly notifies the data sinks (trusted servers or databases, for example).

Design time architecture layer

The design time layer follows a model-driven approach similar to MEdit4CEP [27], and its main purpose is to define high-level models that are easy to understand for any user. These models can be transformed into code interpretable by CEP engines.

We can observe the design time layer in the lower half of Figure 1.2, as well as the image of the domain expert, who, in this context, is someone who has knowledge of the operation of the network and is able to detect attack patterns, although he or she does not necessarily have any knowledge of CEP and ML. Domain experts are in charge of precisely defining the CEP domain formed by the types of network events and *NetworkPrediction* events with their features, they also define the attack patterns using a graphical tool to abstract the implementation details of the CEP rules. This tool is responsible for transforming graphical models of attack patterns into EPL code, which can be deployed in the CEP engine.

The steps to define and generate the IoT security domain code and event patterns are as follows:

1. The domain expert designs the IoT security domain model using the graphical tool. This is achieved by defining the existing event types and their features.

2. The graphical editor will validate the model syntactically. If errors are detected, the domain expert is advised to correct these errors. Once they have a valid model, this model is saved.

3. These patterns are converted into code. This code consists of both the EPL, which is the code that implements the conditions that must be satisfied in order for the CEP engine to detect security attacks, and the code for the actions to be taken in the ESB when detecting possible security attacks.

4. The EPL code is added to the CEP engine, while the generated actions code will be added to the decision process component in the ESB at runtime.

At design time this architecture provides two fundamental advantages. First, the use of MEdit4CEP allows the domain expert to model the patterns he/she needs without the need for CEP knowledge. Second, the integration of a predictor allows us to generate dynamic patterns. These patterns are considered dynamic because our predictor calculates the estimated value for key features that a legitimate package would have using the features of these packages. The patterns make use of these predicted values to calculate the difference with respect to the actual value. The integration of MEdit4CEP and the predictor makes our architecture easy to use, and adaptable to new attacks and scenarios.

The fundamental advantage of our proposal over other rule-based detectors is that our proposal is capable of detecting attacks that are not modeled by the system. This allows us to define an anomaly detector pattern that triggers complex events when the predicted values do not match normality. Therefore, this allows us to detect unknown attacks.

1.4.3 Ensuring the feasibility of the initial architecture on different CEP engines

This section describes the process of comparing CEP engines when deploying the architecture proposed in section 1.4.2. A further explanation on this contribution can be found in [38]. As mentioned above, the objective of this process is to ensure that the mechanisms we propose can be implemented in different CEP engines. Furthermore, we intend to test the computational performance offered by these engines when deploying our proposal, or when employing common patterns in an IoT environment, both in common and stressful situations. In this case we compare Mule ESB with the Esper CEP engine against WSO2 with the Siddhi CEP engine.

There are differences between the two implemented architectures, since our proposal has to be adapted to the technologies used.

Architecture Implementation with Mule

The implementation used by Mule and Esper is composed of three data streams: *DataReceptionAndManagement*, *ComplexEventReceptionAndDecisionMaking* and *EventPatternAdditionToEsper*.

The *DataReceptionAndManagement* flow receives data from the IoT network, transforms it into an event format and then sends it to the Esper CEP engine. This flow is implemented with an MQTT input endpoint where a topic is in charge of receiving the data obtained from the data sources. The received JavaScript Object Notation (JSON) data is then transformed into Java Map events, which are sent to the Esper CEP engine through a message component.

The *ComplexEventReceptionAndDecisionMaking* flow captures the complex events generated by the CEP engine by detecting the patterns that have been previously displayed. These events are transformed to JSON format and stored in a log file.

Finally, the *EventPatternAdditionToEsper* flow allows adding new event patterns to the CEP engine at runtime. To do so, it periodically checks whether a new file with extension EPL exists. If such a file exists, its content is transformed into a textyo string that is displayed in the Esper CEP engine.

Architecture Implementation with WSO2

The WSO2-based architecture does not require integration of an ESB with a CEP engine, as is the case with the Mule and Esper-based architecture. WSO2 provides the Siddhi CEP engine by default as part of its framework.

The WSO2-based architecture receives the data obtained from the data sources. For this purpose, it employs an MQTT broker with two topics, one for *NetworkPacket* and another for *NetworkPrediction*. This data is checked against the event patterns implemented with SiddhiQL and displayed in the Siddhi CEP engine. Similarly to the Esper and Mule-based implementation, when a complex event is generated it is stored in a log file.

An attempt has been made to ensure that both implementations operate in similar conditions, although their intrinsic differences make an exact replication using both technologies impossible. In addition, in order to achieve a similar sending of events and to be able to reproduce the experiments, a simulator has been implemented.

Simulator

The simulator is designed to send network packets to an MQTT. For this purpose it makes use of different Comma Separated Values (CSV) files containing real network traffic that were previously stored, as discussed in Section 1.4.1. This simulator has been implemented in Python, and the paho-mqtt, pandas and JSON libraries are used for the implementation of this simulator. This allows us to have real traffic but maintain the reproducibility of

the experiments thanks to data that has been generated in a real MQTT network. The packets are sent while taking into account the original delay in order to perform realistic experiments. These delays can also be reduced or omitted altogether to allow experiments that simulate high-stress situations. These experiments will give us the key information to know whether our proposal is correctly adapted to IoT environments, as well as indicating the best CEP engine to use.

1.4.4 Architecture to detect threats in IoT environments without the need to specify key features

This section describes the second proposal developed in the course of the Thesis. This proposal tries to improve upon the previous one in two key points. First, it tries to avoid the need to define key features, thus automating the process much more. Secondly, it improves the computational performance of the CEP engine. A further explanation of this contribution can be found in [39].

To achieve this objective we will rely on the use of PCA to reduce the size of simple events. In this way, we achieve the following objectives:

- To reduce the size of the rules generated while characterizing such events with fewer features, thus eliminating the need to search for key feature sets.
- To reduce the architecture's use of the network. Smaller simple events reduce the amount of information sent to the network.

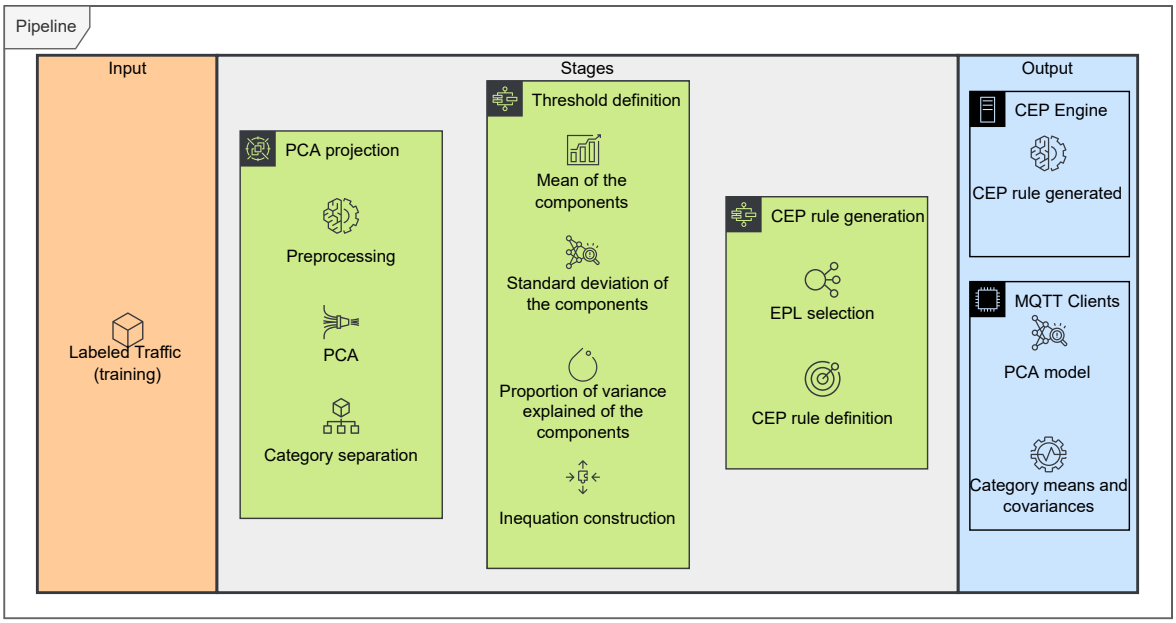


Figure 1.3: The proposed architecture to detect threats in IoT environments without the need to specify key features

Figure 1.3 shows the diagram of the new architecture in charge of generating rules. As we can see, as input, it receives the packets labeled by category. As outputs, it obtains the CEP rules, but also the PCA model, means and covariances.

The operation of the rule generation proposed in this architecture can be divided into the following stages:

- **PCA projection**. This stage is in charge of training the PCA model that will be in charge of reducing the dimensionality of the simple events. To train the PCA model, it is also necessary to perform a previous data preprocessing, which is also in charge of this stage. Finally, a separation of the training data by category is performed. This step is automatic because the data are previously labeled. The objective of PCA is to reduce the complexity of a sample space, and to achieve this it tries to reduce its dimensions. Given an element x, a simple event in this case, represented by n variables, PCA tries to find a representation with m variables where $m<<n$. Such variables are obtained as linear combinations of n variables, and the new m variables are called components. The components are linearly independent of the rest of the components. PCA thus obtains the largest amount of information represented by these components. Moreover, since they are linearly independent, it ensures that there is no redundancy in the information represented.

  In this way, an element x is represented as a vector of variables $n = n_1, n_2, ..., n_n$, thus the vector $m = m_1, m_2, ..., m_m$ will be obtained as we can observe in Equation 1.3, where each variable is weighted with a weight $alpha$. Variables with higher weights have higher weights in that component. This allow us to easily reduce the dimensionality of the events when the model is already trained. The amount of information in each component is not homogeneous. Each component has an explained variance ratio, represented by rv.

$$m_i = n_1 * \alpha_1 + n_2 * \alpha_2 + n_3 * \alpha_3 + ... + n_n * \alpha_n \tag{1.1}$$

- **Threshold definition**. This phase is responsible for generating the function that classifies the simple events into the different categories. This function is converted into a CEP rule in the next stage. Given a training data set $X = [x^1, x^2, ...x^t, ...x^T]$ of T samples and the corresponding category labels $Y = [y^1, y^2, ...y^t, ...y^T]$, each sample is a matrix composed of n components $x^t = [x_1, x_2, ...x_i, ...x_n]$. For each category $j \in [1, c]$ where c is the total number of known categories, the mean of each component i, defined as m_i^j with respect to its category, is calculated, then the standard deviation, std_i^j of each component with respect to its category, is calculated as follows:

  $m_i^j = 1/R \sum x_i^r \forall r \in R : y^r = j$
  $std_i^j = \sqrt{1/(R-1) \sum (x_i^r - m_i^j)^2} \forall r \in R : y^r = j$
  In addition, the proportion of variance explained, rv_i, is obtained for each component. There is a single rv_i for all categories since it is common to the model and allows us

to power the importance of each component. With these elements, the Equation 1.2 is constructed for n components. When 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 that has been used to generate the CEP rule. A corrector element ($alpha$) is also incorporated into the inequation to increase the value of the threshold. This element is intended for categories with very different events so that the original threshold is lower than that of the events that are further away from the mean.

$$f(x,j) = \begin{cases} 1 \text{ if } f(x) = \sum_i^n \sqrt{(x_i - m_i)^2} \cdot rv_i \leq (\sum_i^n std_i \cdot rv_i) + \alpha \\ 0 \text{ if } f(x) = \sum_i^n \sqrt{(x_i - m_i)^2} \cdot rv_i > (\sum_i^n std_i \cdot rv_i) + \alpha \end{cases} \tag{1.2}$$

- **CEP rule generation**. By using the inequation obtained, a CEP rule is generated. Before defining the rule it is necessary to choose a specific EPL depending on the CEP engine to be used. In this case, as we are dealing with WSO2, we use Siddhi as the CEP engine and Siddhi EPL as the EPL. Finally, it only remains to build the rule, adapting it to the syntax of the chosen EPL. This rule is constructed as a template so that the numerical values of the rule are filled with the elements calculated to obtain the inequation.

This architecture allows the generation of complete rules without the need to define features, and improves upon the computational and network performance of classical rules.

1.4.5 Improving, implementing and validating an architecture that can detect threats in IoT environments in an unsupervised manner

This section describes the final improvement developed in the course of the Thesis. This proposal builds on the architecture described in the previous section, and the fundamental improvement it focuses on is that of achieving an architecture that can be completely unsupervised, allowing us to carry out training with unlabeled datasets. In addition, the updating of rules at runtime is performed using dynamic tables, which makes it efficient, as the tables can be updated with parameters when new rules are to be generated, and the rules of each attack simply consult the parameters of its registry. This makes it unnecessary to include new EPL code when generating new CEP rules. A further explanation of this contribution can be found in [40].

Figure 1.4 shows the CEP rule generator deployed in an IoT network. We can see that the stages of the CEP rule generator have changed, and that it now generates parameters for the CEP rules, and also that the training data no longer requires labels.

The different phases of the CEP rule generator are as follows:

- **PCA phase**. This phase is responsible for generating (or updating in subsequent iterations) the PCA model, using the incoming traffic without the need for it to be

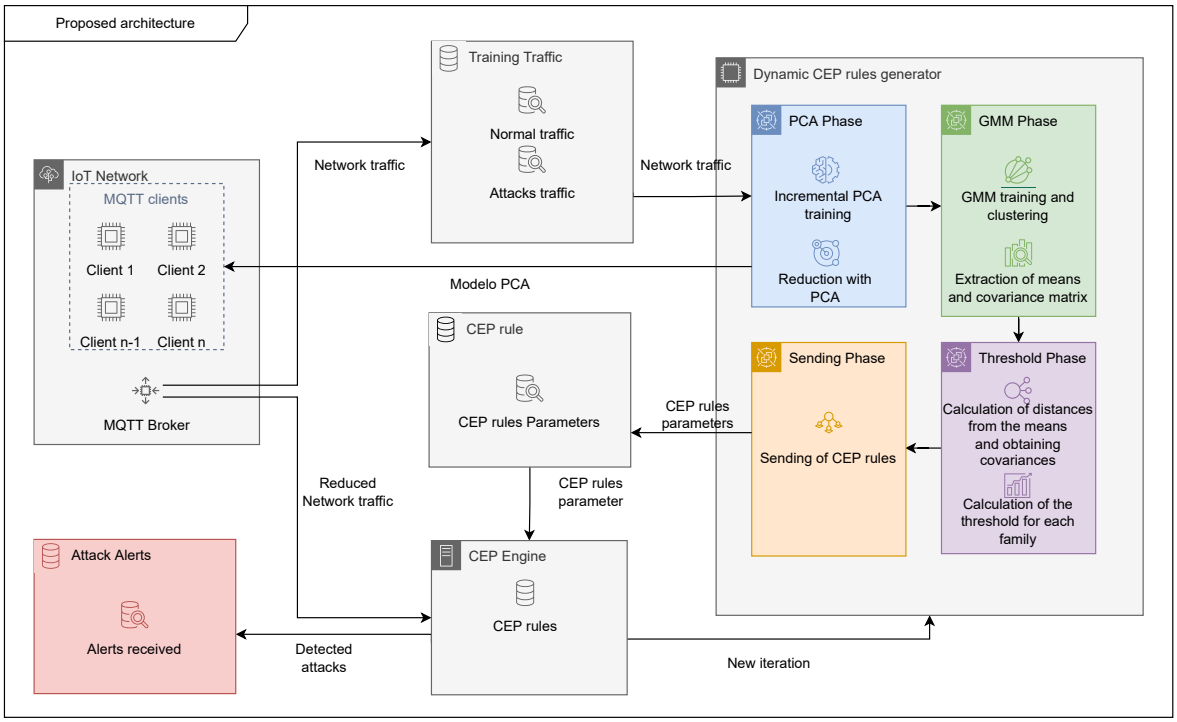


Figure 1.4: Updated architecture for generating CEP rules in an unsupervised manner for real-time threat detection in IoT environments

labeled. This phase is similar to the one found in the previous architecture, the only difference being that IPCA is used. As in the previous architecture, this stage requires preprocessing.

- **GMM phase**. This is the fundamental stage in the updating process and the one that allows training in an unsupervised manner. Gaussian mixture models (GMM) are used to group traffic into different anonymous families. GMM is a probabilistic model that assumes that in each data set X there are K normal distributions representing all C categories existing in the data. GMM tries to find the best combination of the K normal distributions. This allows the grouping of all elements into K anonymous families or different groupings.

$$p(x_i) = \Sigma_{k=1}^{K} p(x_i|c_k)p(c_k) \tag{1.3}$$

Equation 1.3 represents the probability of element $x_i \in X$ as the sum of the composite probabilities it has a set of each family, so that $p(x_i) = 1$. GMM assumes that all elements can be grouped into the various normal distributions.

$$p(x_i) = \Sigma_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k) \tag{1.4}$$

Equation 1.4 shows the GMM model as a linear combination of the K Gaussian distributions. π_k is the mixing coefficient that each normal distribution has. This coefficient provides an estimate over each normal distribution. $\mathcal{N}(x|\mu_k, \Sigma_k)$ is the mixture model component, this models each distribution, where μ_k is the mean and Σ_k is the covariance.

We use a variational version of the algorithm, whose objective is to infer the number of optimal distributions [41]. In this way, it is not necessary to indicate the number of K families a priori, and this allows the process to be completely unsupervised.

In conclusion, GMM offers anonymous families, thus avoiding the need for labeled training traffic.

- **Threshold phase**. This stage is in charge of calculating the threshold for each k family, using the Mahalanobis distance, which is a distance function that takes into account the covariance matrix to weight it [42]. The main advantage of this function is that it takes into account the differences in scale that may exist between the different components that make up the simple events reduced by PCA.

  In this case we have slightly modified the Mahalanobis distance. To account for differences in the explained variance ratio of the different components of the PCA, we improved the distance function by using the ratios as weights, as shown in Equation 1.5 .

$$d(x, \{\mu, \Sigma\}) = \sqrt{(x-\mu)^T \left(\Sigma^{-1} \times VE\right)(x-\mu))} \tag{1.5}$$

Equation 1.5 represents the difference between an element x and the mean of a category μ_k. The inverse covariance matrix is defined as Σ^{-1}. In this way, our distance function will give more weight to the components with a higher rv. The first step is to obtain the VE matrix as the diagonal matrix with the explained variance ratios of each component. Equation 1.6 shows how the matrix we use to weight the explained variance ratios is obtained.

$$VE = diag(rv_1, rv_2, ..., rv_m) \tag{1.6}$$

Equation 1.5 is used to compare each element with the mean of each family. These distances allow the threshold for that family to be calculated. For this purpose, the farthest element from the mean belonging to the family and the closest non-family element belonging to the family are obtained. With these distances we calculate the midpoint, which defines the threshold for that category k.

$$d_{max} = \max\{d(x, \{\mu_k, \Sigma_k\})\}, \forall x \in k \tag{1.7}$$

$$d_{min} = \min \{d(x, \{\mu_k, \Sigma_k\})\}, \forall x \notin k \tag{1.8}$$

$$Th_k = (d_{max} - d_{min})/2 \tag{1.9}$$

Equations 1.7 and 1.8 show how to obtain the element farthest from the mean of a family k and the nearest outside the family k, respectively. Equation 1.9 shows how the threshold is obtained with these two values.

1.5 Results

This section describes the results obtained from achieving the objectives described in Section 1.2. These provide a brief description together with the most important findings. The hypothesis that was put forward has been fulfilled, as can be seen from the results shown below.

Regarding Objective 1, no explicit results are included since the state of the art can be seen in the different articles published in the course of this thesis. The knowledge acquired during the achievement of this objective facilitates the realization of subsequent work, especially from the point of view of the methodology. Moreover, as this objective is transversal to the whole Thesis, the state of the art has been expanding with each new article.

Objective 2 has been achieved, as certain attacks specific to the MQTT and MQTT-SN protocols have been identified. The process is briefly described in Section 1.4.1. In addition, a test environment has been generated with a legitimate scenario and the different attacks were identified together with other very common ones. This allows us to maintain homogeneity in the experiments performed on the different architectures, and also exposes our proposals to a realistic case. This is not an objective that offers an evaluation as such. However, this milestone is crucial for the development of the proposed architectures, and it can also provide support to other research, since the datasets are published and visible from the different articles.

Objective 3 is consistently met. To achieve a successful evaluation of the proposed architecture, whose description is discussed in Section 1.4.2, an Esper CEP engine is deployed with ESB Mule, with CEP rules defined completely manually, just as a domain expert would do, together with a pattern to detect DoS attacks and anomalies using the proposal. In addition, two different models are used to predict the values of the key features. Linear regression, on the one hand, and SVR on the other, demonstrates that this approach is adaptable to different regressors.

The CEP rules defined by our proposal work very well. When the linear regressor is used, an *F1 score* of 1 is obtained. When the SVR regressor is used an *F1 score* of 0.99998 is obtained. Thus we can conclude that this approach is successful and detects modeled and unmodeled attacks consistently. An important detail to note is that the linear regression

obtains excellent results, because we found that there is a linear relationship between the key variables and the behavior of the packages. This was key to our decision to use PCA in subsequent improvements.

Despite the good results, this architecture has certain limitations. First of all, a domain expert has to choose key features, which is not always trivial. Secondly, it requires two different types of simple events, and we will see in Objective 4 that this is not optimal. Furthermore, we consider an attack detected when we identify a packet from that attack, although many packets belonging to the attack are not detected. This is not a problem with attacks in our dataset, but it could be a problem with attacks that have a very small network flow.

These limitations are what we try to correct with the following architectures and proposals. But first it is useful to perform a performance study of the proposal and see how it behaves on another CEP engine. This is the purpose of Objective 4.

Objective 4, which is achieved as described in Section 1.4.3, focuses on observing whether our proposal can be deployed on other CEP engines. In addition it also allows us to know how the system performs in situations with very high workloads, which CEP engine works best, and which types of events work best with CEP engines. As already mentioned, the evaluation of this work consists in comparing the CEP engines in different situations. The results obtained show that the proposal developed is perfectly capable of working in an IoT scenario with either of the two CEP engines. The conclusions we obtain are that WSO2 works better when there is only one type of event, and that WSO2 is better when there is a high workload. Mule performs better when comparing simple events of different types, as was the case in our architecture discussed in Section 1.4.2. We can conclude that our architecture is able to perform perfectly even in high workload situations. With the results of this work we design the following architecture with single type events and over WSO2 to improve performance.

Objective 5, whose architecture is summarized in Section 1.4.4, is evaluated from two different perspectives. On the one hand, we adopt the functional perspective, which evaluates whether the rules generated with this new architecture are able to detect attacks. It should be noted that in this new architecture we consider each packet of an attack as an attack event, and this may degrade the performance metrics in a fictitious way. On the other hand, the computational performance of the rules generated using PCA is evaluated with respect to the rules we saw in the original proposal.

Table 1.1 shows the results obtained in the detection of attacks with CEP rules using PCA. These results are very good and show that the proposal works well at the functional level. From a computational performance point of view, we achieved a 76% improvement in terms of throughput (measured in events/second). In addition, an 86% reduction in the size of events is achieved, as these events are sent over the network with a smaller amount of information, which also improves network performance.

Table 1.1: Metrics obtained by the CEP rules generated.

Metrics of rules generated with PCA	Precision	Recall	F1 score
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

We can conclude that this architecture allows the generation of complete CEP rules for detecting attacks without the need for the intervention of a domain expert while improving computational and network performance. However, it is still necessary to have labeled traffic. This limitation will be addressed in the final proposal.

The architecture to achieve Objective 6 is discussed in Section 1.4.5. In this case, it is not necessary to perform a computational performance evaluation, because simple events are also constructed with PCA so that we ensure optimal network and computational performances. In order to check the good performance of the architecture when unsupervised, two experimentation scenarios have been implemented. In both, attacks are introduced progressively, starting with the legitimate attack-free scenario. The difference is that in the first scenario each attack is divided into training and testing, with the training packets that are detected as anomalies training the model in the new iteration, while the testing packets validate the rules of each iteration. In the second scenario there are four datasets: the first training dataset is similar, while, the testing dataset is divided into three datasets that are mixed with different attacks in the new iterations. Moreover, in the second scenario, all of them feed back to the model (except the third testing dataset, which is exclusively for validation), even if they are correctly classified.

In both scenarios we obtain very good results. In the first experiment we obtain an *F1 score* of 0.9824, and in the second we obtain an *F1 score* of 0.9938. We can conclude that this proposal is able to generate CEP rules, in an unsupervised way, that are capable of detecting known and unknown IoT threats in real time.

Answers to the Research Questions

The results obtained and shown above allow us to answer the research questions posed in the Section 1.1.

- **(RQ1)**: Are the rules generated by the architecture computationally efficient in a constrained environment, such as an IoT network?
  We can affirm that yes, the rules generated by the proposals work perfectly well in low-resource environments. In fact, the rules generated in the proposal [39], present

a very significant performance improvement over the use of CEP rules without the reduction of simple events with PCA.

- **(RQ2)**: Is the proposal capable of detecting unknown attacks?
  The conclusion is that the proposals generated, in the course of the Thesis, are perfectly capable of detecting attacks that were not found in the training dataset initially.
- **(RQ3)**: Is this integration of technologies capable of working in different CEP implementations?
  The proposals implemented in the thesis can be easily adapted to other CEP technologies. Moreover, the rules work well in different CEP engines. The proof of this can be found in one of our papers [38].
- **(RQ4)**: Is it possible for the models used to generate the rules to be trained in an unsupervised way?
  The answer to this question is yes. This is demonstrated by the final version implemented in this Thesis [40].

Main Publications

All the objectives proposed in Section 1.2 have been achieved by following the tasks defined in Section 1.3. All the results of the work carried out in this Doctoral Thesis are summarized in the list of publications below:

- **Objective 2** — To analyze common IoT protocols, and design, implement and deploy a test environment.
  - *Roldán-Gómez, José, Carrillo-Mondéjar, Javier, Castelo Gómez, Juan Manuel and Ruiz-Villafranca, Sergio, Security Analysis of the MQTT-SN Protocol for the Internet of Things* , submitted in ***Applied Sciences*** . Journal paper. JCR2020 Q2, IF 2.838 [37].
- **Objective 3** — To design, implement, deploy and validate an architecture to detect threats in IoT environments on the basis of one or more key features.
  - *José Roldán, Juan Boubeta-Puig, José Luis Martínez, Guadalupe Ortiz, Integrating Complex Event Processing and Machine Learning: an Intelligent Architecture for Detecting IoT Security Attacks Volume 149, 2020, 113251, ISSN 0957-4174, doi: 10.1016/j.eswa.2020.113251.*, published in ***Expert Systems With Applications*** . Journal paper. JCR2020 Q1, IF 6.954 [37].
- **Objective 4** — To ensure the feasibility of the initial architecture on different CEP engines.
  - *Roldán-Gómez J, Boubeta-Puig J, Pachacama-Castillo G, Ortiz G, Martínez JL, Detecting security attacks in cyber-physical systems: a comparison of Mule and WSO2 intelligent IoT architectures Volume 7, 2021, e787, ISSN 2376-5992, doi:*

10.7717/peerj-cs.787, published in ***PeerJ Computer Science***. Journal paper. JCR2021 Q2, IF 2.411 [38].

- **Objective 5** — To redesign, implement and validate a new architecture to detect threats in IoT environments without the need to specify key features and achieve improved computational performance with respect to the first proposal.
  - *José Roldán-Gómez, Juan Boubeta-Puig, Juan Manuel Castelo Gómez, Javier Carrillo-Mondéjar, José Luis Martínez Martínez, Attack pattern recognition in the Internet of Things using complex event processing and machine learning, Date of Conference: 17-20 October 2021, ISSN: 2577-1655, doi: 10.1109/SMC52423.2021.9658711*, presented at the ***2021 Institute of Electrical and Electronics Engineers (IEEE) International Conference on Systems, Man, and Cybernetics***. Conference paper. GGS Rating A-, Class 2 [39].
  - *José Roldán-Gómez, Juan Boubeta-Puig, Juan Manuel Castelo Gómez, Javier Carrillo-Mondéjar, Jesús Martínez del Rincón, An Automatic Complex Event Processing Rules Generation System for the Recognition of Real-Time IoT Attack Patterns*, submitted in ***Engineering Applications of Artificial Intelligence***. Journal paper. JCR2021 Q1, IF 7.802.
- **Objective 6** — To Update, improve, implement and validate architecture that can detect threats in IoT environments without the need to specify key features in an unsupervised manner.
  - *José Roldán Gómez, Jesús Martínez del Rincón, Juan Boubeta-Puig and José Luis Martínez, Hacia la creacion de reglas CEP no supervisadas para la deteccion en tiempo real de ataques en entornos IoT, Date of Conference: 19-21 June 2022*, presented at the ***2022 Jornadas Nacionales de Investigación en Ciberseguridad (JNIC) Jornadas Nacionales de Investigación en Ciberseguridad***. Conference paper [40].
  - *José Roldán Gómez, Jesús Martínez del Rincón, Juan Boubeta-Puig and José Luis Martínez, An Automatic Unsupervised Complex Event Processing Rules Generation Architecture for Real-Time IoT Attacks Detection. 2023, ISSN 1572-8196, doi: 10.1007/s11276-022-03219-y*, published in ***Wireless Networks***. Journal paper. JCR2021 Q2, IF 2.701.
- **Objective 1** — To study and analyze the solutions proposed by other researchers.
  - All of the above publications meet this objective in a cross-cutting manner.

Other Publications

Here we include other findings resulting from collaboration with other researchers in works that are published or under review. These proposals aim to solve certain cybersecurity

problems but are not specifically focused on the objective of this Doctoral Thesis. These proposals are the following:

- Forensic analysis in IoT:
  - *Castelo Gómez, Juan Manuel, José Roldán Gómez, Javier Carrillo Mondéjar, and José Luis Martínez Martínez, Non-Volatile Memory Forensic Analysis in Windows 10 IoT Core, Volume 21, 2019, 1141, ISSN 1099-4300, doi: 10.3390/e21121141*, published in ***Entropy***. Journal paper. JCR2019 Q2, IF 2.494 [43].
  - *Castelo Gómez, J.M., Carrillo Mondéjar, J., Roldán Gómez, J., and Martínez Martínez J.L., A context-centered methodology for IoT forensic investigations, Volume 20, 2020, 647-673, ISSN 1615-5270, doi: 10.1007/s10207-020-00523-6* , published in ***International Journal of Information Security***. Journal paper. JCR2021 Q2, IF 2.427 [44].
  - *Castelo Gómez, Juan Manuel, Javier Carrillo Mondéjar, José Roldán Gómez, and José Luis Martínez Martínez, Developing an IoT forensic methodology. A concept proposal, Volume 36, 2021, 301114, ISSN 2666-2817, doi: 10.1016/j.fsidi.2021.301114*, published in ***Forensic Science International: Digital Investigation (continuation of the journal Digital Investigation)***. Journal paper. JCR2021 Q4, IF 1.805 [45].
- Vulnerability detection:
  - *Carrillo-Mondéjar, J., Castelo-Gómez, J.M., Roldán-Gómez, J., and Martínez Martínez J.L., An instrumentation based algorithm for stack overflow detection, Volume 16, 2022, 245-256, ISSN 2263-8733, doi: 10.1007/s11416-020-00359-7*, published in ***Journal of Computer Virology and Hacking Techniques***. Journal paper. Not indexed [46].
- IoT malware classification:
  - *Javier Carrillo-Mondejar, Juan Manuel Castelo Gomez, Carlos Núñez-Gómez, Jose Roldán Gómez, and José Luis Martínez, Automatic Analysis Architecture of IoT Malware Samples, Volume 2022, 2022, 8810708, ISSN 1939-0114, doi: 10.1155/2020/8810708*, published in Security and Communication Networks. Journal paper. JCR2020 Q4, IF 1.791 [47].

CHAPTER 2

Security analysis of the MQTT-SN protocol for the Internet of Things

- **Title**: Security analysis of the MQTT-SN protocol for the Internet of Things
- **Authors**: José Roldán-Gómez, Javier Carrillo-Mondéjar, Juan Manuel Castelo Gómez and Sergio Ruiz Villafranca
- **Type**: Journal paper.
- **Journal**: Applied Sciences
- **Publisher**:MDPI
- **ISSN**: 2076-3417
- **Status**: Published
- **Publication date**: October 2022
- **Volume**: 12
- **Paper Number**: 21
- **DOI**: 10.3390/app122110991
- **JCR IF/ranking**: 2.838/Q2 (JCR2021).

applied sciences

MDPI

Article

Security Analysis of the MQTT-SN Protocol for the Internet of Things

José Roldán-Gómez [1,*,†], Javier Carrillo-Mondéjar [2,†], Juan Manuel Castelo Gómez [2,†] and Sergio Ruiz-Villafranca [2,†]

[1] Department of Computer Science, University of Oviedo, 33003 Oviedo, Spain
[2] Institute of Informatics (I3A), University of Castilla-La Mancha, 02071 Albacete, Spain
* Correspondence: roldangjose@uniovi.es
† These authors contributed equally to this work.

Abstract: The expansion of the Internet of Things (IoT) paradigm has brought with it the challenge of promptly detecting and evaluating attacks against the systems coexisting in it. One of the most recurrent methods used by cybercriminals is to exploit the vulnerabilities found in communication protocols, which can lead to them accessing, altering, and making data inaccessible and even bringing down a device or whole infrastructure. In the case of the IoT, the Message Queuing Telemetry Transport (MQTT) protocol is one of the most-used ones due to its lightness, allowing resource-constrained devices to communicate with each other. Improving its effectiveness, a lighter version of this protocol, namely MQTT for Sensor Networks (MQTT-SN), was especially designed for embedded devices on non-TCP/IP networks. Taking into account the importance of these protocols, together with the significance that security has when it comes to protecting the high-sensitivity data exchanged in IoT networks, this paper presents an exhaustive assessment of the MQTT-SN protocol and describes its shortcomings. In order to do so, seven different highly heterogeneous attacks were designed and tested, evaluating the different security impacts that they can have on a real MQTT-SN network and its performance. Each one of them was compared with a non-attacked implemented reference scenario, which allowed the comparison of an attacked system with that of a system without attacks. Finally, using the knowledge extracted from this evaluation, a threat detector is proposed that can be deployed in an IoT environment and detect previously unmodeled attacks.

Keywords: Internet of Things; cybersecurity; protocols; MQTT-SN

Citation: Roldán-Gómez, J.; Carrillo-Mondéjar, J.; Castelo Gómez, J.M.; Ruiz-Villafranca, S. Security Analysis of the MQTT-SN Protocol for the Internet of Things. *Appl. Sci.* **2022**, *12*, 10991. https://doi.org/10.3390/app122110991

Academic Editors: Howon Kim and Thi-Thu-Huong Le

Received: 12 October 2022
Accepted: 27 October 2022
Published: 30 October 2022

1. Introduction

The Internet of Things (IoT) is a new technology paradigm that is on the path to change the way we interact with computers and machines. It can be explained as a global network composed of devices (also called *things*) capable of communicating with each other [1,2], which, by doing so, brings with it new possibilities in many fields such as health, the economy, engineering, resource management, and everyday life. As a result, a wide range of industries are researching applications that use this paradigm in the race to having the upper hand in a scenario that has the potential to become a key area in future technology.

This is evidenced by the fast-paced growth rate of the IoT, whose number of devices connected to the Internet surpassed the total of the non-IoT ones [3] in the year 2020; even with the chip shortage caused by the pandemic, there are 12.3 billion IoT devices connected to the Internet, and the predictions are for this figure to grow and reach 27 billion by 2025 [4]. The IoT has several peculiarities. For example, networks are usually dynamic and heterogeneous, giving rise to many different protocols, such as MQTT [5], Bluetooth Low-Energy (BLE) [6], ZigBee (based on the IEEE 802.15.4 standard) [7], and the constrained Application Protocol (CoAP) [8], among others. However, this feature also applies to IoT devices, which can range from the simplest sensors to computing hardware with more resources that sends information to the cloud.

Although this environment has many applications and benefits for industry, from a security point of view, it constitutes a new vector for attacks from cybercriminals [9]. Most of the safety concepts that are widely accepted and applied in network communications have not been taken into account in the IoT. This is due to the fact that IoT systems are usually constrained by a lack of resources related to memories, processors, network bandwidths, and power consumptions, which do not support the implementation of security measures [10].

The diversity of IoT systems and their limitations cause new problems associated with this paradigm [11], for example the need to improve lightweight encryption algorithms and the weak defenses against Denial of Service (DoS) due to the lack of memory, processor power, and bandwidth. Battery consumption gains importance since there are many applications that require the use of devices that only have batteries. Furthermore, it is hard to keep IoT devices updated because they are very heterogeneous [12].

These vulnerabilities are being exploited by criminals to the extent that the evolution of malware specifically targeting IoT Linux has vastly increased in recent years. Figure 1 shows the quarterly growth of samples registered against IoT Linux since the appearance of the Mirai malware, a botnet that interrupted Internet access for millions of people. It can be seen that, from August 2016 to May 2022, there has been a huge increase in samples, fifty-nine-times higher in May 2022 compared to August 2016.

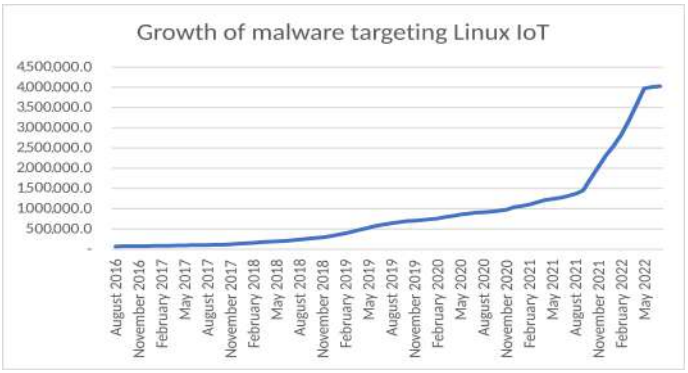


Figure 1. Growth of malware targeting IoT Linux.

In the IoT, most attacks are successful because the devices are misconfigured by default, and this indicates that old problems are present in the IoT as well. At this point, it is vital to evaluate the protocols involved in the IoT for a security audit in order to determine their deficiencies and, on that basis, propose improvements.

There are many challenges to be addressed by the research community in the field of IoT cybersecurity. Some of the main ones are listed in the paper *An overview of security and privacy in smart cities IoT communications* by Al-Turjman et al. [13], which identifies the main threats to IoT security. According to the authors, malware, false information, traffic modification, traffic eavesdropping, and identity theft are the most common IoT threats. There are other articles that study the challenges and trends posed by the IoT, and cybersecurity is a key area in all the analyses [14–16], highlighting the protocols and their heterogeneity as a crucial element that needs to be scrutinized. Consequently, this is the field on which we will focus in this work, showing that analyzing protocols from an experimental point of view and extracting their vulnerabilities is critical for IoT cybersecurity research, in addition to assessing the impact of different attacks that exploit the vulnerabilities exposed by Al-Turjman et al. in their work [13].

This research is focused on MQTT-SN [17], which is based on its predecessor, namely MQTT. While MQTT is a lightweight protocol, which uses a publish/subscribe messaging transport scheme, MQTT-SN is the newest version and has been designed to be even lighter than MQTT. The reasoning behind the choosing of this version is due to it being optimal for environments with resource-constrained devices such as the Industrial Internet of Things (IIoT) [18] and yet not having received the focus of detailed research from the security research field. Under these circumstances, this paper aims to firstly find deficiencies in MQTT-SN, secondly to exploit them, and thirdly, to evaluate their impact on the whole IoT scenario. After this evaluation, we recommend an attack detector designed to discover attacks in IoT environments. The advantage of this threat detector is its ability to detect anomalies and unmodeled attacks, which is crucial in combating novel attacks such as those proposed in this work.

This proposal is a substantial extension of the work *Security Assessment of the MQTT-SN Protocol for the Internet of Things* [19]. Schematically, the main contributions of this article are as follows:

- A practical analysis of MQTT-SN protocol vulnerabilities.
- An implementation of attacks based on the vulnerabilities discovered.
- An evaluation of the impact of the different attacks.
- A threat detector, which has already been published, is suggested and tested with the implemented attacks.

This paper is organized as follows. Section 2 describes the MQTT-SN protocol and the main differences between it and MQTT [5]. The state-of-the-art is reviewed in Section 3. A presentation of the MQTT-SN system deployed is made in Section 4. The attacks discovered are described and evaluated in Section 5. Section 6 describes a novel threat detector designed for IoT environments. Finally, we draw our conclusions in Section 7.

2. MQTT-SN Protocol

The MQTT-SN [17] protocol is a connectivity protocol based on MQTT [5], which operates at the application layer. MQTT is a topic-based protocol, which makes it possible to create a publish/subscribe-based topology in which each device can subscribe to or publish information about a topic. Thus, there is a broker that manages the topics and the connected devices in the network. This topology is very useful for the IoT, as it allows the transmission of information generated by sensors to central nodes.

MQTT-SN was designed to be lightweight (even more than MQTT [5]) and works with wireless communications. This means that it is characterized by link failures, a short message size, low bandwidth, and low overheads, among other features. MQTT-SN was designed to be similar to MQTT, but there are a few differences. Firstly, MQTT-SN includes *TopicId*, which replaces the topic name in MQTT. *TopicId* is a 16-bit integer, which acts as the topic name, with the *REGISTER* command being able to negotiate the mapping between *TopicId* and the topic name. Secondly, MQTT-SN provides a sleeping client mechanism. This feature enables clients to shut themselves down and save power for a while. MQTT-SN allows us to update or delete the *will* message, which allows devices to notify other clients about an ungracefully disconnected client. Finally, another important aspect is that MQTT-SN works over the User Datagram Protocol (UDP) [20].

UDP, which operates at the transport layer, introduces an 8-byte header. 6LoWPAN, on the other hand, has header compression mechanisms (in the best case, it can use 4-byte headers), and it also has a Maximum Transmission Unit (MTU) of 127 bytes, which is really small compared to other protocols; for example, IPv6 offers an MTU of 12,800 bytes [21]. Payload sizes will be conditioned by the MTU. In our implementation of the different attacks on a 6LoWPAN network, the maximum payload of 6LoWPAN is 38 bytes, and consequently, the UDP payload is 30 bytes. It is important to mention that MQTT-SN operates over UDP, but does not need to operate over 6LoWPAN. The study and implementation of the attacks described in this analysis focused exclusively on MQTT-SN, so the complete study of the protocols of other layers is outside the focus of this work. In Figure 2, which depicts

an MQTT-SN-based network, we can observe how one operates. The publication on a topic is represented with an arrow of the same color as that topic. The grey rectangles symbolize MQTT-SN clients, and the colors inside them represent the topics to which they are subscribed. This scheme allows us to easily understand MQTT and MQTT-SN. The client with *Id* 1 is subscribed to topics 1, 2, and 3; the client whose *Id* is equal to 2 is subscribed to topics 1 and 2; finally, the client whose *Id* is equal to 3 is subscribed to topics 1 and 3. If the client with *Id* number 2 publishes on topic 3, this message is received by clients 1 and 3. The same happens with topic 2 is published by client 2.

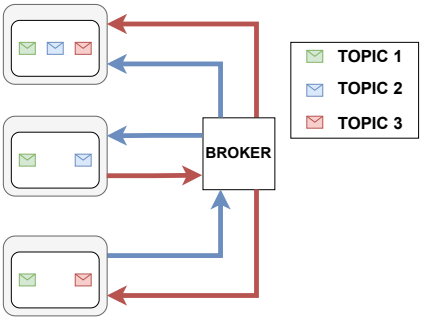


Figure 2. MQTT -SN scheme.

2.1. Format of the Main Packets

In order to properly understand the attacks to be implemented, a description of the format of an MQTT-SN packet is provided in Table 1, containing the field information common to all packets to be analyzed, with the MQTT-SN packets that are generated or modified in this experiment being detailed below.

Table 1. Brief description of the fields common to the different packets.

Common Fields	Description
Length	It determines the length of the packet.
MsgType	It defines the type of MQTT-SN packet.
Flags	It indicates additional options.

2.1.1. Connect Message

This type of packet is used to connect MQTT-SN clients to the broker.

Table 2 shows the format of a *Connect*-type packet, the different fields, and the octets they use. The function of these fields is as follows:

- **Flags**: The most relevant ones for a *Connect* packet:
  - *Will*: It defines a message and a topic. In case of an error in the connection between the device and the broker, this message is sent to the chosen topic.
  - *CleanSession*: Enabling this flag causes the client to forget previously subscribed topics. If it is not enabled, the broker computes the topics to which the client was subscribed and keeps them, using the *ClientID* field.
- **ProtocolId**: It determines the version of the protocol being used.
- **Duration**: It contains the value of the keep alive timer of the connection.
- **ClientId**: It uniquely identifies a client connected to the MQTT-SN broker.

Table 2. Connect message format.

	Length	MsgType	Flags	ProtocolId	Duration	ClientId
Octet number	0	1	2	3	4–5	6:n

2.1.2. Publish Message

Publish packets are used when a client sends a message on a topic.

Table 3 shows the format of a *Publish*-type packet, the different fields, and the octets they use. The function of these fields is as follows:

- **Flags**: The most relevant ones for a *Publish* packet:
  - *DUP*: It indicates that the message is a duplicated message, which is sent when no acknowledgment has been received after sending the original one. This behavior occurs with a Quality of Service (QoS) value greater than 0.
  - *QoS*: It determines the QoS level of the Publish message.
  - *Retain*: When a message is published with the *Retain* flag set, the broker saves it as a reference one. Consequently, when a client subscribes to the topic, it automatically receives this message, so it is not necessary for the original sender to publish it again.
  - *TopicIdType*: It indicates the type of identifier found in TopicId. This can be a short name, formed by two characters, or the topic Id formed by 2 bytes.
- **TopicId**: It contains the topic identifier in either of the two formats mentioned above.
- **MsgId**: It uniquely identifies a message when the QoS is greater than zero. It is encoded with 16 bits.
- **Data**: It contains the data of the message to be published.

Table 3. Publish message format.

	Length	MsgType	Flags	ProtocolId	Duration	ClientId
Octet number	0	1	2	3–4	5–6	7:n

2.1.3. Subscribe Message

Subscribe packets are used when a client wants to subscribe to a topic.

Table 4 shows the format of a *Subscribe*-type packet, the different fields, and the octets they use. The function of these fields is as follows:

- **Flags**: The most relevant ones for a *Subscribe* packet:
  - *DUP*: If enabled, the last message with an active DUP fag is received if it exists.
  - *QoS*: It indicates the QoS level required for that topic.
  - *TopicIdType*. It specifies the type of identifier found in TopicId. This can be a short name, topic name, or topic Id.
- **MsgId**: It is used to identify the acknowledgment of receipt, which is sent by means of a *Suback* packet.
- **TopicName or TopicId**: It contains the topic identifier in the format specified in the flag *TopicIdType*.

Table 4. Subscribe message format.

	Length	MsgType	Flags	MsgId	TopicName or TopicId
Octet number	0	1	2	3–4	(5:n) or (5–6)

2.1.4. Pingreq Message

This type of message is used to know whether a client is connected to the broker. Additionally, in MQTT-SN, it allows taking the device out of sleep mode.

Table 5 shows the format of a *Pingreq* type packet. This format is simpler than those described above:

- **ClientId**: It is an optional field that is used for changing the status of a client from *sleeping* to *awake*. It contains the client identifier.

Table 5. Pingreq message format.

	Length	MsgType	ClientId (Optional)
Octet number	0	1	2:n

3. State-of-the-Art

As far as the authors of this paper know, there are currently few published studies about the shortcomings of IoT protocols. One of the few is *A survey: Attacks on RPL and 6LoWPAN in IoT* by P.Pongle and G.Chavan [22], which is focused on attacks in the Routing Protocol for Low-Power and Lossy Network (RPL) [23,24]. The RPL protocol enables device routing in Low-power and Lossy Networks (LLNs), which are characterized by the fact that routers usually operate under severe power, memory, and network constraints. This protocol supports point-to-point routing, but also multipoint-to-point and vice versa. It is also highly scalable and can be successfully used with thousands of devices within the LLN. The attacks presented in this paper against RPL typically serve two distinct purposes. On the one hand, we have the attacks that allow a complete or partial denial of service of the network, for example the sinkhole attack. On the other hand, there are also attacks that allow attacking a specific device, e.g., the selective forwarding attack. Finally, there are others that allow the identity of devices to be hijacked to compromise network confidentiality, e.g., the alteration and spoofing attack. The authors also collected certain patterns so that a rule-based Intrusion Detection System (IDS) can be deployed with these patterns.

Another interesting work is [25]. In this paper, the authors conducted a brief study of the Constrained Application Protocol (CoAP), a web protocol designed specifically for devices and networks with limited capabilities. In this paper, the authors used BurpSuite to analyze CoAP request traffic and concluded that the it is not encrypted and also susceptible to attacks.

In [26], the authors describe several security issues with MQTT, but there were no implementations or evaluations of the described attacks. The attacks in this paper, moreover, made an ordinary use of the protocol, not presenting any use of it, which varies from what it is described in the standard. They also contemplated MQTT as a means to create a botnet, although this is not an attack against MQTT; it is the use of it as a means to establish a botnet.

There are several other pieces of research that focus on ZigBee, which is a protocol for low-data-rate and short-range wireless networking. One example is [27], which shows two example attacks: one tries to achieve an eavesdropping attack on a ZigBee network where AES-CCM is used. In order to achieve this, the authors tried to use the same key twice, for example by causing a reset of the nodes. The other attack consists of performing a denial of service, for which they suggested modifying the payload of ZigBee packets, even if there is encryption, to cause an error in the operation of the protocol.

In *Exploiting MQTT-SN for Distributed Reflection Denial-of-Service Attacks* by Sochor et al. [28], the authors suggested to make use of the topology used by MQTT-SN to perform a pro-reflective attack. In other words, they took advantage of the fact that the broker sends messages to all devices subscribed to a topic to perform denials of service against the MQTT-SN network.

A particularly interesting work is *IoT Content Object Security with OSCORE and NDN: A First Experimental Comparison* by Cenk Gundogan et al. [29], which proposes the use of OSCORE, which is a protocol designed to protect end-to-end communications in resource-constrained systems. Its interest lies in them making a comparison with the use of DTLS, which is the encryption protocol usually implemented over UDP. Another interesting proposal that aims to adapt a lightweight encryption method to MQTT and MQTT-SN is the *Lightweight Security Scheme for MQTT/MQTT-SN Protocol* by Ousmane Sadio et al. [30]. To achieve lightweight encryption over the MQTT and MQTT-SN protocol, the authors propose to use ChaCha20, which is a stream cipher protocol. In addition, they also propose to use Poly1305 as a one-time authenticator. In this way, they achieved a lightweight and secure encryption scheme.

Another proposal in the field of lightweight encryption is *Design and evaluation of a novel white-box encryption scheme for resource-constrained IoT devices* by Bang, A.O. and Rao, U.P. [31]. This work focused on generating a scheme capable of protecting IoT environments from white box attacks, those in which the attacker has full view of the execution environment and uses it to break the encryption. The authors, to achieve their goal, propose a scheme to hide the private key in ciphers with elliptic curves.

With regard to comparisons between already existing algorithms, we find the proposal *Safe MQTT-SN: a lightweight secure encrypted communication in IoT* by L.Kao et al. [32], which proposes a secure encryption scheme for MQTT-SN systems. They propose to use the digital signature (ECDSA), hash function, key exchange (ECDHE), ChaCha20, and Poly1305. The authors compared the delay time introduced by the MQTT-TLS handshake with respect to their proposed scheme.

Within the comparisons of the protocols, although not part of the analysis of security, we find a paper that makes a brief comparison from the performance point of view of the IoT protocols, namely *Survey on State of Art IoT Protocols and Applications* by Kumar N.V.R. and Kumar P.M. [33]. It is observed that the results of MQTT-SN are good, especially in the packet loss vs. bandwidth contribution.

A very interesting paper, although not focused on any IoT protocol in particular, is *A Large-Scale Empirical Study on the Vulnerability of Deployed IoT Devices* by Binbin Zhao et al. [34]. In it, the authors surveyed 1,362,906 IoT devices over 10 months. All the conclusions they drew were very interesting, but of special interest is the huge number of vulnerable MQTT servers, a fact that gives an idea of the importance of the matter at hand.

Table 6 shows a brief comparison of the different state-of-the-art works.

Table 6. Comparison of the state-of-the-art works.

Reference	Target Protocol	Highlights
[22]	RPL	Presents and evaluates different attacks against RPL.
[25]	CoAP	Brief analysis of the protocol; concludes that it is not safe due to the lack of encryption.
[26]	MQTT	Known attacks against MQTT analyzed.
[27]	ZigBee	They describe two different attacks against ZigBee.
[28]	MQTT-SN	Reflective attacks employing MQTT-SN to cause DoS.
[29–32]	General encryption	All of them propose improvements to achieve a lightweight encryption.
[33]	Various	Comparison of protocols; MQTT-SN obtains good results.
[34]	Various	Empirical analysis; there are many unsecured MQTT servers.
This work	MQTT-SN	Several attacks are implemented and evaluated, and countermeasures are proposed.

As we can see, the topic of analyzing the security of the MQTT-SN protocol is an unexplored one. Other novelties introduced in this research are the study of the protocol to detect vulnerabilities, as well as how to exploit them. In addition, we analyzed the impact that they have on the system, and we propose a threat detection system to detect such

attacks. Furthermore, as demonstrated by the high number of vulnerable MQTT servers currently deployed, working on the security of the lighter version MQTT-SN is necessary before this protocol is used globally and replaces or coexists with MQTT.

4. MQTT-SN Baseline Scenario

The first step in performing a security analysis of the MQTT-SN protocol in order to test attacks and evaluate their impact is to define a legitimate scenario without attacks and evaluate its performance. This scenario is referred to as the baseline. This is important because it makes it possible to compare the impact of each attack (which exploits a shortcoming in the protocol) on the system against the baseline. The evaluation can be focused on features such as packet size, packet rate, etc., and, on that basis, try to determine the impact of the attacks discovered.

The proposed baseline scenario is composed of six devices: one is defined as the broker and the rest as regular devices (which perform the function of a simple sensor to acquire data to be sent to the broker). Within this set of regular devices, the baseline scenario includes one device with certain privileges, which is subscribed to a confidential topic, and four others without privileges. The overall behavior is quite simple: each non-privileged mote is subscribed to a topic referred to as */topic1*. In addition, these non-privileged devices retrieve the information generated by other devices under the */topic1* label. The information generated as */topic1* consists of random floats with no practical meaning. This information could be, in a real scenario, the temperature taken in a room, the humidity in a vineyard, or the vital signs of a patient in a hospital. The generation and handling of this randomly generated information is beyond the scope of this work. In this case, the objective is to evaluate MQTT-SN as an information exchange protocol between the broker and the subscriber nodes.

As mentioned above, the baseline scenario also includes a privileged mote, which generates a random float and publishes it in the other topic referred to as */topic2* in each cycle. The privileged mote generates a symbolic message, which is confidential, and it is useful to evaluate a scenario that manages confidential information. These devices do not contain additional protection measures, but the differentiation of these measures makes it easier to observe the handling of the different topics and their contents. Finally, the broker manages all the connections, processes all the information collected by the devices, and centralizes the network.

Figure 3 shows the baseline scheme, which is composed of devices that generate information on both topics, namely */topic1* and */topic2*, and the broker. In this case, the device with an orange background color, which is located at the bottom left, is subscribed to topic 2, which simulates the sending of critical information. Including this device is useful to evaluate how the system can be altered when using two different topics. By establishing this scenario as the baseline, we can extract valuable information about the regular (not under attack) behavior of the network. Below, we compare the baseline scenario against scenarios with the system under attack.

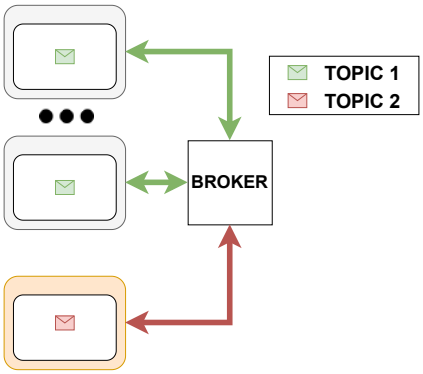


Figure 3. MQTT-SN baseline scenario.

5. MQTT-SN Attacks and Performance Evaluation

This section enumerates the shortcomings discovered in the MQTT-SN protocol, as mentioned in Section 4, which an attacker can exploit to compromise the information collected between nodes (affecting integrity and confidentiality) or to compromise the entire network and/or devices (affecting availability) in an IoT infrastructure. All the attacks discovered, which represent the main contribution of this paper, are described in the following sub-sections. In addition, these attacks are implemented and their impact is evaluated in this section. For this purpose, the baseline scenario was deployed. Once an attack has been successfully demonstrated, different countermeasures to prevent or mitigate it are presented.

The implementation of these attacks was performed using a network simulator called Cooja, which is Contiki's network simulator, as it allows the experiments to be much more reproducible and parameterizable than a real scenario. Cooja provides an intuitive User Interface (UI), which offers a graphical representation of our network and devices and a toolkit to perform different actions. Contiki [35] is an open-source operating system designed for the IoT, which provides a set of features and tools to design, implement, and deploy IoT applications and systems. It is built on a TCP/IP stack and offers lightweight preemptive scheduling on an event-driven kernel, which is a very motivating feature for the IoT.

This way, the combination of Cooja and Contiki allowed us to deploy a virtualized IoT network and study its behavior. It is widely used to develop applications, but it can be used to design, implement, and study application layer attacks over 6LoWPAN (network layer) on several different devices. In order to capture the traffic of the virtualized network, it was redirected from the broker to the host device, then Wireshark was used to obtain the traffic and analyze it later [36].

5.1. Trash-Inject Attacks

The first attack described aims to introduce unexpected elements into the legitimate MQTT-SN application.

Before performing this attack, it is necessary to introduce a malicious device into the network. In order to do so, it is either necessary not to have an authentication mechanism or to gain access to the broker by guessing or having access to the credentials. In our implementation, we assumed that we had access to the network due to a lack of authentication mechanisms, which is something very common in MQTT and MQTT-SN servers [34], so a realistic scenario is proposed.

Once that access to the broker has been gained, the attack consists of introducing malformed information in a specific topic or in a set of topics. The range of possibilities in this attack is very wide since, if the attackers know the logic of the application on which they are going to inject information, they may be able to modify its behavior arbitrarily. However, even without knowing the operation of the application into which malformed information is introduced, this attack can have a very negative impact.

This attack requires prior knowledge of the system's topics, but by default, it is possible to subscribe to all topics using the # character, so knowing all the topics in the system is not a difficult task. Another possible way is to take advantage of the short topic name format of MQTT-SN as it is easy to scan topics when they are encoded with two characters.

Figure 4 shows the format of the *Publish*-type packets. As can be seen, the packet fields that are key to carrying out the attack are marked in red. In this case, the *topic Id* field indicates the topic where the malicious information is to be inserted, and the *data* field contains the information to be entered. The content of these two fields may vary depending on the attacker's target. As mentioned above, if the attacker knows the logic of the system, he/she can inject specific data to cause a specific system failure. However, he/she can also introduce pseudo-random data (trash) to cause a system malfunction.

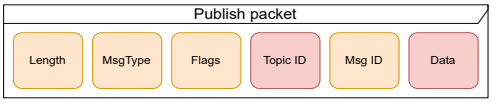


Figure 4. Publish packet format.

Figure 5 shows how the attack works, with the attacker having the ability to inject messages into a used topic. The device on the right, which has a red background color, represents the attacking device. As can be observed, in this case, the attacker injects trash into topic 1 and the broker distributes that trash to all clients subscribed to that topic. It should be noted that this attack can be carried out by contaminating several topics simultaneously in a similar way. In our implementation, we only focused on one topic, because it is easier to quantify the impact in this way.

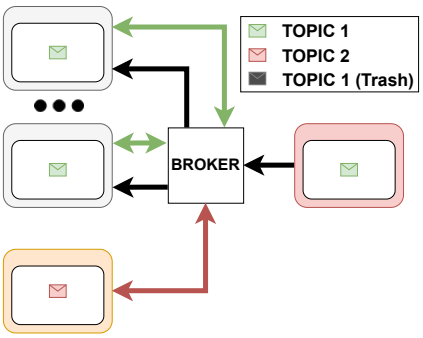


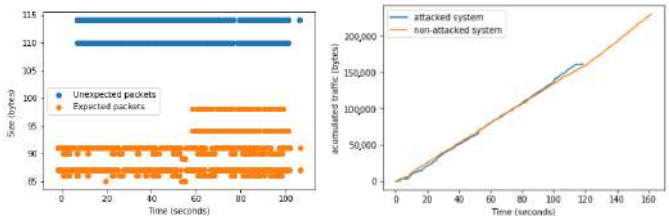
Figure 5. MQTT-SN trash injection scheme.

Evaluation of the Trash Injection Attack

Before diving into the evaluation, it is necessary to clarify that the baseline system works by publishing random numbers to simulate a sensor. Thus, our application expects a numerical value, and if it receives a different kind of data, it will not work properly.

Figure 6a shows the expected topics (numbers) versus the unexpected values (character strings). This type of graph is recurrent throughout the paper, each point representing a packet, so that we can easily observe the behavior of the system. As we can see, the malicious device easily sends many malicious packets. This ability to inject traffic can lead to a malfunctioning of the MQTT-SN application. In this experiment, the client application receives text when it expects numeric data, what causes the application to stop working. However, in other applications, the consequences can be more negative, since an attacker is capable of injecting inject information into an application.

Figure 6b shows the amount of accumulated bytes received by the broker over time, proving that this attack can be performed without significantly increasing the traffic.



(**a**) (**b**)

Figure 6. Trash injection attack evaluation diagrams. (**a**) Expected against unexpected packets. (**b**) Cumulative traffic of baseline scenario vs. trash injection attack scenario.

This kind of attack can be partially stopped if we enable the user/password request in our MQTT-SN network. The problem is that this solution will not work if the attacker infects a mote that has been previously authenticated. Another relevant aspect is that this attack is even easier to use against MQTT-SN than against MQTT, because the use of UDP allows an IP spoofing attack to be carried out easily.

5.2. Information Leaks

Another security weakness of MQTT-SN is that (by default) it might be possible to subscribe to all the topics, which, in many cases, can have serious consequences. The intrinsic problem is that the topics in MQTT-SN are defined as a structured hierarchy. For example, we could subscribe to or publish on the topic referred to as *house/bedroom2/sensor1*. This structure is perfectly valid for a house with several bedrooms and a few sensors inside each room. In this example, we can subscribe to all the topics from all the bedrooms by simply subscribing to *house*/#. The # operator will obtain a subscription to all the rooms in the house. Furthermore, if we subscribe to the /# topic, we will subscribe to all the topics in the broker. It is useful to understand that when the # character is used, it is not mapped to a numeric *Id*. In addition, when a client subscribes to a topic, you can use the name, the *Id* of the topic, or the short name of the topic, which can lead to obtaining sensitive information by using the # character. It could be possible to prevent subscription to /#, as this receives a simple solution in MQTT in which there are no short topics, with each topic being linked to a number. However, if the attack is against an MQTT-SN network, the attackers could use the short name format to iterate over possible topics easily. They could also use the topic *Id*, since, as it is encoded with two octets, its iteration is feasible. Once the attacker manages to subscribe to all the topics, they obtain access to all the messages in the network, even if we have sensitive information in them. This implementation is not a classic eavesdropping attack in which a Man in the Middle (MitM) scheme is sought as a general rule. In this case, it is necessary to subscribe to the topics, with the vulnerability consisting of the possibility of subscribing to unknown topics.

As we can see in Figure 7, packets of type *Subscribe* are used, and the attackers only need to manipulate the *topic Id* field to subscribe to the topic of their choice.

Figure 7. Subscribe packet format.

Figure 8 shows the scheme of the attack. As we can see, the attacking device tries to subscribe to all topics by iterating the numeric *Id*. In this way, the malicious device is able to extract information from all the topics in the system.

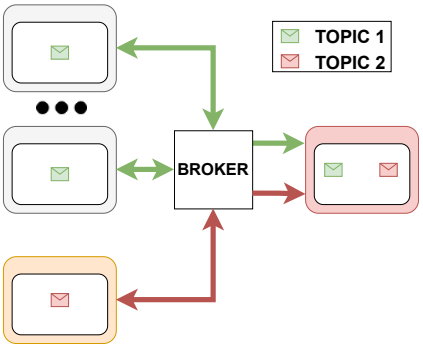


Figure 8. MQTT-SN information leak scheme.

Evaluation of Information Leak Attack

This attack can be dangerous if the system is misconfigured. In this case, an attacker can subscribe to any topic in the system and retrieve all the information generated in the IoT network. Figure 9 shows how easily an attacker can access confidential information. This protocol is highly vulnerable to this type of attack because the publish/subscribe scheme means that the attacker does not need a previous man in the middle attack to carry it out.

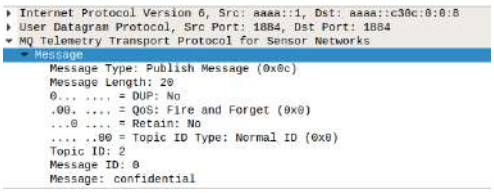


Figure 9. Malicious mote subscribed to confidential topic.

Although this attack does not modify the behavior or performance of the network, it greatly impacts the confidentiality of the system. This can be avoided by making use of MQTT-SN's built-in authentication.

5.3. Disconnect Wave

Disconnect wave is a DoS attack that exploits a big failing in the MQTT-SN protocol. The specification of MQTT-SN indicates that each client has a unique *Id*, so if a new client tries to register this *Id* again, the latter gains the *Id* and ejects the previously created connection. Once we know the *Id*, the attack can be carried out immediately. The *Id* field has a variable length between 1 and 23 characters, and it is sent by the client when making the connection. This means that if a bad security policy is used by the client application, it is possible to guess the *Id*. In our application, this *Id* is a device counter to ensure that it is unique.

Figure 10 shows the format of the *connect* packets used to steal the *Id* of legitimate clients and removethem from the system. In this case, the *client Id* must be modified depending on the targeted client.

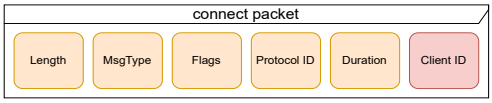


Figure 10. MQTT-SN connect packet format.

As can be seen in Figure 11a, the malicious device connects to the *Ids* of legitimate devices to eject them. To achieve this, it iterates over the *Ids* of the legitimate devices. If guessing this *Id* is not trivial, it is possible to generate an MitM scheme to know the identifiers of the legitimate devices. Figure 11b shows the result of the attack, and as we can see, the legitimate devices are expelled.

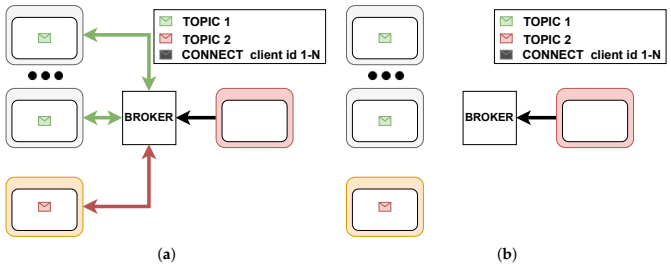
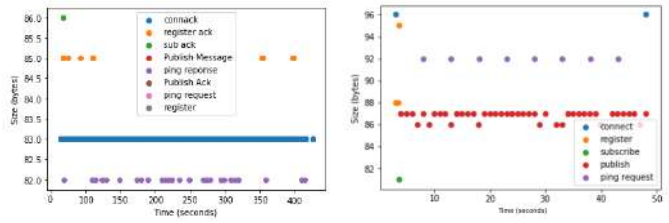


Figure 11. MQTT-SN disconnect wave scheme. (**a**) MQTT-SN disconnect wave, implementation of the attack. (**b**) MQTT-SN disconnect wave, result of the attack.

Evaluation of the Disconnect Wave Attack

As is shown in Figure 12a, the broker receives a flood of connect commands, but it does not send any publish packets. The attacker sends connect packets iterating over the *Id* to expel legitimate devices whose *Id* matches those of these packets. Therefore, we have many *connect ack* packets and zero *publish message* packets, which can ultimately completely override the network.

Even if the attacker performs this attack successfully, as we can see in Figure 12b, legitimate devices keep sending messages. This attack can be stopped with a rules-based IDS, which detects that the network is flooded with connect packets. This attack is an innovation introduced in this work, so it can be difficult to find rules if you want to use a rule-based threat detector, as is the case with other novel attacks due to the lack of knowledge about them.



(**a**) (**b**)

Figure 12. Disconnect wave attack evaluation diagrams. (**a**) MQTT-SN broker after disconnect wave attack. (**b**) MQTT-SN legitimate client after the disconnect wave attack.

5.4. Sniffing Attack

This attack is based on a well-known MitM scheme. It can be performed using layers below the application layer, where MQTT-SN operates. This case is different from the rest of the attacks discussed in this paper because the attackers need to achieve an MitM scheme, a process that requires no use of MQTT-SN features. This attack takes advantage of the unencrypted connection (by default) in MQTT-SN communications, but has the limitation (compared with the others presented in this paper) that it needs a third protocol to complete the MitM scheme. Normally, MQTT-SN is supported with other lower layers such as RPL [24] or 6LoWPAN [21]. For this attack, it is firstly necessary to compromise the communications by attacking these protocols, and once the attacker is in the middle of the communications, he/she has access to MQTT-SN information. The impact of this attack could be extensive: first of all, it can affect confidentiality; secondly, integrity suffers because the attacker can modify the messages; finally, availability is not guaranteed because the attacker can break the communication. Figure 13 depicts an MitM attack over the MQTT-SN protocol. We can see that the malicious device is placed between the broker and the legitimate devices.

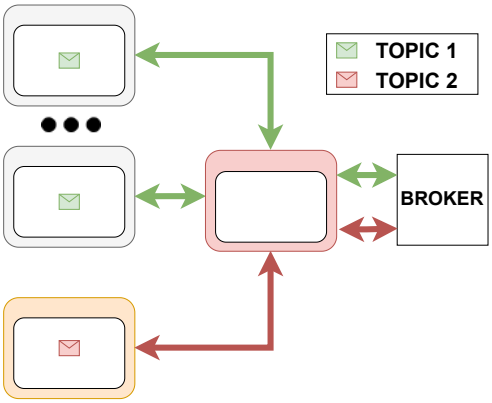


Figure 13. MQTT-MitM scheme.

Evaluation of the Sniffing Attack

This attack is successful when there is unencrypted communication, and it must be preceded by a man in the middle attack. In our experiment, a rank attack against RPL [37] was used to achieve the MitM scheme. The consequence of this attack is that the attacker gains access to the information exchanged between the broker and the devices (see Figure 14). This is not a pure IoT attack, but in this context, it can be more dangerous, because when there are constrained devices, it is more difficult to implement an encryption algorithm to protect the communication. One way to tackle this problem is to use Secure MQTT-SN (SMQTT-SN) [38], but this implies an overhead that is introduced in the communication and more resources employed on the devices. This attack does not affect network performance, but then again, it is an attack that compromises system confidentiality.

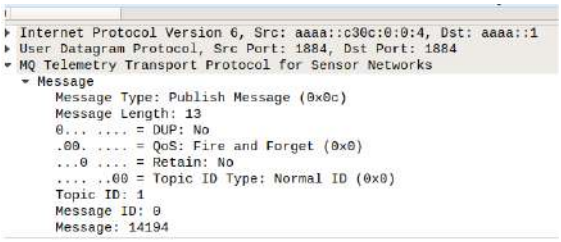
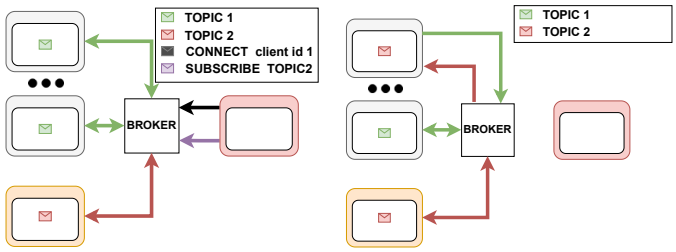


Figure 14. Sniffed pcap view.

5.5. Spoofing Connection Via Id

This attack has a more complex and circumstantial implementation than the previous ones. Its objective is to modify the topics to which a client is subscribed, which requires that the legitimate client application does not use the *CleanSession* flag each time it connects to a client.

As seen in other attacks, the client identifier is exploited to expel a client if it is connected when the message is sent, or simply to spoof in case that it is not connected. It is important to remark that, in this scenario, the *CleanSession* flag is enabled. This means that the broker deletes the subscriptions to the different topics of the client with that specific *Id*. The attackers can then subscribe the client to the topics they consider, or simply leave it without any subscription. This completely modifies the reception of information from one or more clients. Figure 15a shows a spoofing attack over MQTT-SN.



(**a**) Spoofing connection, implementation of the attack. (**b**) Spoofing connection, result of the attack.

Figure 15. MQTT-SN spoofing connection scheme.

In addition, Figure 16 shows the format of the connect-type packet, when the connection is to be spoofed. In this case, in addition to modifying the client *Id*, the attackers have to send the *CleanSession* flag active.

Finally, Figure 15b shows the result of the successful attack. The target *Id* is now subscribed to the topics that the attacker wishes. The malicious device does not need to interact with the broker again.

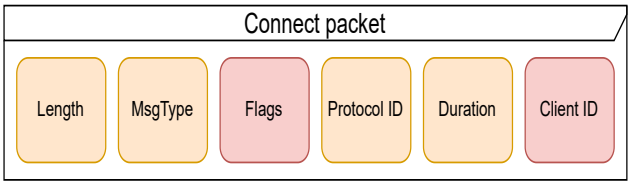
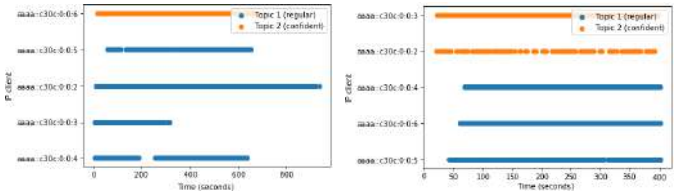


Figure 16. Connect packet format (spoofing connection).

5.6. Evaluation of the Spoofing Legitimate Clients Attack

This attack is implemented in Cooja by adding a malicious client that expels the target client and overwrites the subscribed topics. Figure 17a shows the normal topic setup for each legitimate device in the baseline scenario, while Figure 17b shows a malicious topic setup for each legitimate device in an attacked system. As can be seen in Figure 17b, an attacker could change the configuration of each device. We can see that, through this attack, we can change the topics to which each legitimate device subscribes. This attack could be avoided by including an IDS in the network, as this can detect when a new device is not authorized to join the topology.



(**a**) (**b**)

Figure 17. Spoofing connection attack evaluation diagrams. (**a**) MQTT-SN topics sent for each device in the baseline scenario. (**b**) MQTT-SN topics sent for each device after spoofing connections.

5.7. Congestion Attack

This attack is very common in many protocols, but there is a new factor to take into account when it is implemented over MQTT/MQTT-SN. As the broker must distribute the published messages, an attacker could publish massive messages on a common topic. This means the impact is multiplied by the number of devices that have been subscribed to this topic. There are several approaches to performing the attack. For example, it can create big packets (with 6LoWPAN, there is a maximum of 127 bytes for the maximum transmission unit), or it can send many packets as quickly as possible.

Figure 18 shows the scheme for performing the attack, illustrating how the flow of */topic1* increases with respect to the normal scenario. In this experiment, this was performed by sending messages without delay with the maximum allowed size.

We can see the format of the *Publish*-type packet (Figure 19) in the congestion attack. The only difference with respect to a trash injection attack is that, here, as a general rule, the packet will be sent with the largest possible size, so the *length* field gains relevance,

although it is actually calculated on the basis of the *data* field, which can be used to identify this attack quickly. The size may vary depending on the network protocols used.

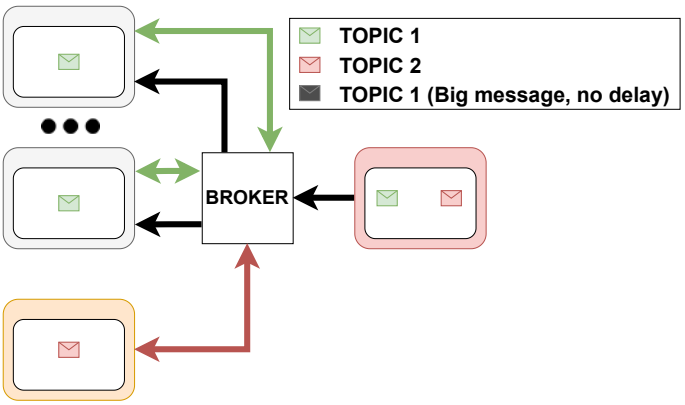


Figure 18. MQTT-SN congested accumulated traffic scheme.

Figure 19. Publish packet format (congestion attack).

Evaluation of the Congestion Attack

To demonstrate the impact of this attack, firstly, Figure 20a shows the packets that have been sent by the broker in the baseline scenario, and then, Figure 20b shows the traffic on a compromised IoT baseline system. It can be seen how the broker seems to send fewer messages.

In order to observe the impact more clearly, Figure 20c zooms in on the behavior of the system when it is not under attack, while Figure 20d zooms in on the behavior of the system when it is under attack. These images show an increase in the number of packets received by the broker when the system is under attack.

In this implementation, the attacker floods a topic with two devices that send messages uninterruptedly, and the broker's buffer seems to have overflowed because it stops sending other topics. In this sense, the attacker is easily able to modify the normal behavior of the system. Figure 21 shows the accumulated traffic volume difference between the baseline system and a congested system (flooding a topic with two devices). As we can observe, the attackers considerably increased the traffic of the overall system. This could be a serious issue in an IoT context, because in a small network, this increase will have a negative impact on battery consumption. A network-based IDS could identify congestion in the network and alert the administrator to a possible attack.

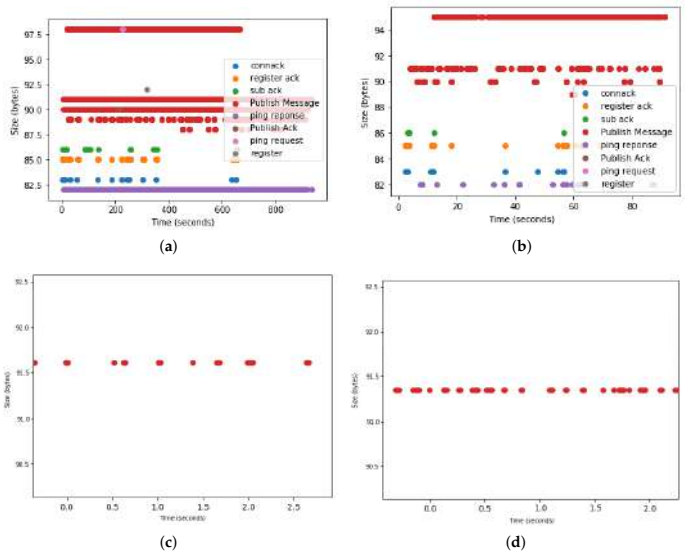


Figure 20. Congestion attack evaluation diagrams. (**a**) MQTT-SN traffic in the legitimate scenario. (**b**) MQTT-SN traffic in the attacked system. (**c**) MQTT-SN most-common topic frequency without attacks. (**d**) MQTT-SN most-common topic frequency on an attacked system.

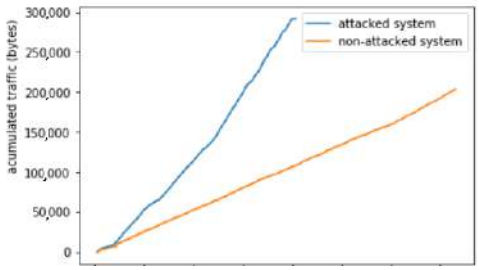


Figure 21. Accumulated traffic with topic flooding.

5.8. Wake up Wave Attack

The sleeping mode of MQTT-SN allows clients to remain in a sleeping state in order not to waste battery power. While in this state, they do not receive the published messages of the topics to which they are subscribed. This state is reached by sending a *disconnect* message in which a time of duration is indicated. If this time is not specified, the broker simply disconnects the client. When the client is ready to receive messages again, a *Pingreq* message is sent, indicating the client identifier. This sends the client to a state called *awake*, where it can receive messages again. A diagram of how this mechanism works can be seen in Figure 22.

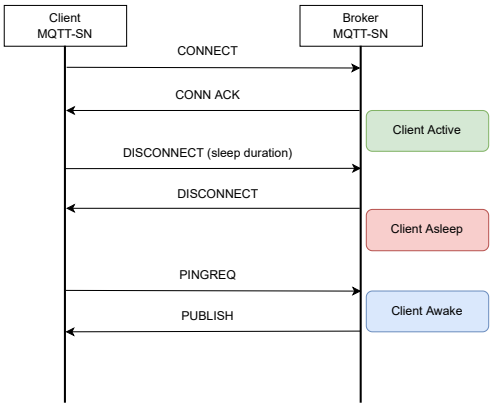


Figure 22. MQTT-SN sleeping mechanism diagram.

This attack, which exploits the mechanism introduced in MQTT-SN, aims to prevent clients from sleeping, so that the broker does not stop sending them the various publishes. This can be harmful, especially for devices that rely on batteries to operate. It also increases network traffic.

Figure 23 shows the format of the Pingreq packet. In this case, an identifier of the client we want to wake up is sent. In our implementation, we iterated over the different clients permanently with the aim of preventing any of them from going into sleep mode. A diagram of the attack can be seen in Figure 24.

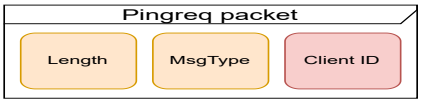


Figure 23. Pingreq packet format.

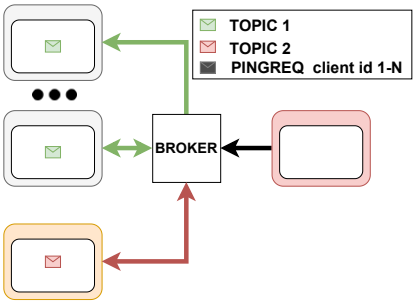


Figure 24. MQTT-SN wake up wave attack scheme.

Evaluation of the Wake up Wave Attack

As mentioned above, in this implementation of the attack, it iterates over the different clients indefinitely, preventing any of them from sleeping. This causes, as can be seen in

Figure 25, a huge increase in the system's network usage. The same occurs with the battery usage if the devices are dependent on it, which is usually the case when using sleep mode. To defend against this attack, a rule-based IDS could be interesting since detecting that a single IP address sends multiples Pingreqs with different Ids could be the signature to identify this particular attack.

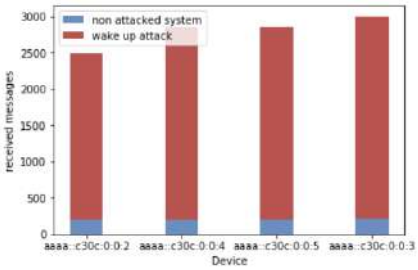


Figure 25. Received messages with attack versus those without attack.

6. Threat Detector Proposed

This section describes the deployment of the threat detector designed for IoT environments.

As described above, the impact of these attacks when successful is considerable, and deploying typical detection systems can be complicated due to the scarce resources that IoT devices possess. Therefore, we propose a threat detector that is specifically designed to operate in IoT environments [39] and is based on a Complex Event Processing (CEP) engine. CEP is a technology capable of processing and correlating an enormous amount of data. The CEP engine receives simple events, which in this case are network packets. When these simple events meet the requirements set in the CEP rules, a complex event is triggered, which defines a situation of interest, which in this case, is a particular type of attack. CEP is used as the base engine because it has been demonstrated that it is possible to deploy this type of engine in IoT environments with few resources [40–42].

As a general rule, it is necessary for a domain expert to define these CEP rules. However, an architecture capable of generating CEP rules automatically has been designed [39]. This architecture is ideal for detecting the attacks in this paper, as they are novel attacks. This means that, in a real environment, the domain experts cannot define rules for these attacks because they do not know them.

This architecture is based on the use of Principal Component Analysis (PCA) [43] and the Euclidean distance weighted with the explained variance ratio of each component of the PCA model. Each component condenses the information of different features of the events by means of linear relationships between them, and the components are linearly independent of each other. The explained variance ratio allows us to know the weight of each component. By doing so, we can weight each component more accurately when generating the rules. This allows us to reduce the dimensionality of simple events while characterizing the different families of attacks and to improve the performance of CEP rules in the network, computation, and memory domains. In addition, it also allows us to generate anomaly-detecting rules, which are ideal for detecting unknown attacks.

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

Equation (1) shows the definition of a threshold for a CEP rule generated by our proposal. Simple events, which are network packets in this case, are reduced with the PCA model. Once the reduced simple event x is available, it is sent to the CEP engine. The CEP engine calculates the difference of each component x_i of that reduced simple event with

the mean of that component for each family m_i and weights it with the explained variance ratio of each component rv_i. The α element makes it possible to add a bias for attacks with elements that are very far from the mean. If the sum of these weighted differences is less than the sum of the standard deviations std_i weighted with the explained variance ratio of each component, it means that this element is part of that family. Otherwise, it is not part of that family. This allows the generation of rules that detect anomalies by using a rule that detects legitimate system behavior and makes it possible to generate anomaly detection rules when a single event does not correspond to any type of attack and is not part of the system's legitimate behavior. It is important to understand that the real novelty of this threat detector does not lie in Equation (1), which defines the detection threshold. The real novelty is the use of PCA to reduce single events, which reduces the computational load of the CEP engine. This is why this approach is novel and ideal in IoT environments.

Table 7 shows the results of the proposed threat detector against the attacks that were presented in this paper. It should be noted that it was used to detect attacks that are more difficult to detect using rule-based threat detectors. In this case, a single rule was generated that tries to highlight all traffic that is malicious.

Table 7. Suggested threat detector results by packet type.

Packet Type	TP	TN	FP	FN
Normal traffic	0	653	24	0
Disconnect wave	2357	0	0	0
Trash injection	76	0	0	0
Congestion attack	824	0	0	0
Fake Id	8	0	0	0

We can see how the different types of traffic are detected by the detector (Table 8); in this case, it was analyzed packet by packet. The classifier achieves a True Positive (TP) when a packet, which belongs to an attack, is classified as an attack, a True Negative (TN) when a packet, which belongs to normal traffic, is classified as such, a False Positive (FP) when a packet, which belongs to normal traffic, is classified as an attack, and finally, a False Negative (FN) when a packet that belongs to the attack traffic is classified as normal traffic. To accurately assess the effectiveness of the classifier, the following metrics are used:

- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$
- F1 Score = $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$

Table 8. Overall results of the classifier.

Precision	Recall	F1 Score
0.9927	1	0.9963

All metrics are between 0 and 1, with 1 being a perfect score. A high precision metric score indicates that the classifier does not generate many false positives. A high recall score indicates that the classifier does not generate many false negatives, and the F1 score metric combines the two previous metrics to obtain the combined performance.

As we can see, this configuration causes some normal packets to be detected as attacks, but in contrast, allows all attacks to be detected. Prioritizing the recall metric over the precision metric is very common in the cybersecurity field.

In addition, the threat detector we suggest has been tested in other contexts to detect attacks in the IoT environment with very positive results [39], being able to detect attacks against the MQTT protocol with very good computational performance figures.

7. Conclusions

The growth of the IoT has meant a drastic change in the technological world. When a new paradigm arrives in a field, it brings many changes with it. In the case of the IoT, having new devices and systems with specific features and requirements has consequently meant the development of new communication protocols. Although it may not be the first element to pay attention to when it comes to discussing IoT security, their importance must not be underestimated, as they are the means to transport information. Data are exchanged in large quantities in the IoT, with some of them being extremely sensitive. For this reason, the research community has shown interest in analyzing some protocols to determine their security level and how they can be exploited. However, the MQTT protocol, while being one of the most-used ones in this environment, has not been studied in detail. The same goes for its lighter version, MQTT-SN, for which, although it has not been yet as successful as MQTT due to its novelty, it is reasonable to think that it may reach similar usage figures.

Under these circumstances, this paper provided an overview of the shortcomings of the MQTT-SN protocol, which is appropriate for IoT scenarios. By compromising MQTT-SN communications, the entire IoT infrastructure can be affected in terms of integrity, confidentiality, and availability. MQTT-SN has inherited security weaknesses from MQTT, its predecessor, which are related to authentication and encryption. In addition, this paper highlighted the fact that MQTT-SN includes new features, such as the sleeping device feature or the short name topic, which make it easier to attack MQTT-SN than MQTT. The performance evaluation demonstrated that security is not an intrinsic feature of MQTT-SN, and it is necessary to investigate, in a new specification, the way to solve security issues while keeping in mind that the devices are resource-constrained and thus limited to executing low-complexity algorithms, as well as most of the protocols are designed to be light and do not support excessive overheads. Finally, this paper showed the fragility of this protocol, meaning it is relatively simple to attack, and as we can see in the previous section, these attacks have a huge impact on the system. Therefore, we proposed a CEP-based IDS capable of operating in IoT environments and detecting unmodeled attacks. This threat detector was tested at MQTT with extremely good results.

The main conclusions and novelties drawn from this work are as follows:

- A practical analysis of the MQTT-SN protocol from a security point of view was carried out, noting that it has several weaknesses that can be exploited.
- Different attacks that exploit the vulnerabilities of the protocol were proposed, and their operation was explained.
- The attacks were implemented, and the impact they have on an MQTT-SN network was measured, allowing the impact of the attacks to be analyzed. This impact is quite large in some attacks, while other attacks have a more circumstantial impact.
- Countermeasures were proposed to mitigate the effect of these attacks. In addition, the use of a threat detector was suggested, which obtained good results, with an F1 score of 0.9963.

Author Contributions: Conceptualization: J.R.-G., J.C.-M. and J.M.C.G.; methodology: J.R.-G.; software: J.R.-G.; investigation: J.R.-G., J.C.-M. and J.M.C.G.; data curation: J.R.-G.; draft preparation: J.R.-G.; review and editing: J.M.C.G. and S.R.-V.; visualization: J.R.-G., J.C.-M. and S.R.-V.; supervision: J.M.C.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Spanish Ministry of Science, Innovation and Universities and the European Union FEDER Funds (Grant Numbers FPU 17/02007 and FPU 17/03105),by the Spanish Ministry of Economic Affairs and Digital Transformation under the project RTI2018-098156-B-C52, by the Spanish Ministry of Science and Innovation under the project PID2021-123627OB-C52, by the University of Castilla La Mancha (Grant Numbers DO20184364 and PI001482), and by the JCCM (Grant Number SBPLY/21/180501/000195).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Thierer, A.; Castillo, A. Projecting the growth and economic impact of the internet of things. *Georg. Mason Univ. Mercat. Center June* **2015**, *15*, 1–10.
2. Laghari, A.A.; Wu, K.; Laghari, R.A.; Ali, M.; Khan, A.A. A review and state of art of Internet of Things (IoT). *Arch. Comput. Methods Eng.* **2021**, *29*, 1395–1413. [CrossRef]
3. State of the IoT 2020: 12 Billion IoT Connections, Surpassing Non-IoT for the First Time. 2020. Available online: https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/ (accessed on 25 August 2022).
4. State of IoT 2022: Number of Connected IoT Devices Growing 18% to 14.4 Billion Globally. 2022. Available online: https://iot-analytics.com/number-connected-iot-devices/ (accessed on 25 August 2022).
5. Hunkeler, U.; Truong, H.L.; Stanford-Clark, A. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In Proceedings of the 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08), Bangalore, India, 6–10 January 2008; pp. 791–798.
6. Ghori, M.R.; Wan, T.C.; Sodhy, G.C. Bluetooth Low Energy Mesh Networks: Survey of Communication and Security Protocols. *Sensors* **2020**, *20*, 3590. [CrossRef] [PubMed]
7. Prakash, S. ZigBee based wireless sensor network architecture for agriculture applications. In Proceedings of the 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 20–22 August 2020; pp. 709–712.
8. Shelby, Z.; Hartke, K.; Bormann and C.; Frank, B.; *The Constrained Application Protocol (CoAP)*; Universitaet Bremen: Bremen, Germany, 2014.
9. Gupta, P. A Survey of Application Layer Protocols for Internet of Things. In Proceedings of the 2021 International Conference on Communication information and Computing Technology (ICCICT), Mumbai, India, 25–27 June 2021; pp. 1–6.
10. Mohanty, J.; Mishra, S.; Patra, S.; Pati, B.; Panigrahi, C.R. IoT Security, Challenges, and Solutions: A Review. *Progress in Advanced Computing and Intelligent Engineering*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 493–504.
11. Heer, T.; Garcia-Morchon, O.; Hummen, R.; Keoh, S.L.; Kumar, S.S.; Wehrle, K. Security Challenges in the IP-based Internet of Things. *Wirel. Pers. Commun.* **2011**, *61*, 527–542. [CrossRef]
12. Zhang, Z.; Cho, M.C.Y.; Wang, C.; Hsu, C.; Chen, C.; Shieh, S. IoT Security: Ongoing Challenges and Research Opportunities. In Proceedings of the 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, Matsue, Japan, 17–19 November 2014; pp. 230–234. [CrossRef]
13. Al-Turjman, F.; Zahmatkesh, H.; Shahroze, R. An overview of security and privacy in smart cities' IoT communications. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e3677. [CrossRef]
14. Javed, A.R.; Shahzad, F.; Rehman, S.u.; Zikria, Y.B.; Razzak, I.; Jalil, Z.; Xu, G. Future smart cities: Requirements, emerging technologies, applications, challenges, and future aspects. *Cities* **2022**, *129*, 103794. [CrossRef]
15. Dahiya, P.; Kumar, V. IOT Security: Recent Trends and Challenges. In *Emerging Technologies in Data Mining and Information Security*; Dutta, P., Chakrabarti, S., Bhattacharya, A., Dutta, S., Shahnaz, C., Eds.; Lecture Notes in Networks and Systems; Springer Nature: Singapore, 2023; pp. 3–10. [CrossRef]
16. Punia, A.; Tiwari, M.; Verma, S.S. The IoT in Security Architecture, Challenges, and Solutions. In *Optical and Wireless Technologies*; Tiwari, M., Ismail, Y., Verma, K., Garg, A.K., Eds.; Lecture Notes in Electrical Engineering; Springer Nature: Singapore, 2023; pp. 405–416. [CrossRef]
17. Stanford-Clark, A.; Truong, H.L. Mqtt for sensor networks (mqtt-sn) protocol specification. *Int. Bus. Mach. IBM Corp. Version* **2013**, *1*, 1–28.
18. Shakya, S.R.; Jha, S. Challenges in Industrial Internet of Things (IIoT). In *Industrial Internet of Things*; CRC Press: Boca Raton, FL, USA, 2022; pp. 19–39.
19. Roldán-Gómez, J.; Carrillo-Mondéjar, J.; Gómez, J.M.C.; Martínez, J.L.M. Security Assessment of the MQTT-SN Protocol for the Internet of Things. *J. Phys. Conf. Ser.* **2022**, *2224*, 012079. [CrossRef]
20. Postel, J. *User Datagram Protocol*; Technical Report; RFC: Sacramento, California, USA, 1980. .
21. Mulligan, G. The 6LoWPAN architecture. In *Proceedings of the 4th Workshop on Embedded Networked Sensors, EmNets '07, Cork, Ireland, 25–26 June 2007*; Association for Computing Machinery: New York, NY, USA, 2007; pp. 78–82. [CrossRef]
22. Pongle, P.; Chavan, G. A survey: Attacks on RPL and 6LoWPAN in IoT. In Proceedings of the 2015 International Conference on Pervasive Computing (ICPC), Pune, India, 8–10 January 2015; pp. 1–6. [CrossRef]
23. Winter, T.; Thubert, P.; Brandt, A.; Hui, J.W.; Kelsey, R.; Levis, P.; Pister, K.; Struik, R.; Vasseur, J.P.; Alexander, R.K. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. *RFC* **2012**, *6550*, 1–157.
24. Paszkowska, A.; Iwanicki, K. The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) under Network Partitions. In *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks, EWSN'18, Madrid, Spain, 14–16 February 2018*; Junction Publishing: Junction, TX, USA, 2018; pp. 90–101.

25. Arvind, S.; Narayanan, V.A. An overview of security in CoAP: Attack and analysis. In Proceedings of the 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 15–16 March 2019; pp. 655–660.
26. Andy, S.; Rahardjo, B.; Hanindhito, B. Attack scenarios and security analysis of MQTT communication protocol in IoT system. In Proceedings of the 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Yogyakarta, Indonesia, 19–21 September 2017; pp. 1–6. [CrossRef]
27. Farahani, S. ZigBee wireless networks and transceivers. *Newnes* **2011**, *4*, 2021.
28. Sochor, H.; Ferrarotti, F.; Ramler, R. Exploiting MQTT-SN for Distributed Reflection Denial-of-Service Attacks. *Commun. Comput. Inf. Sci.* **2020**, *1285*, 74–81. [CrossRef]
29. Gündoğan, C.; Amsüss, C.; Schmidt, T.C.; Wählisch, M. IoT Content Object Security with OSCORE and NDN: A First Experimental Comparison. In Proceedings of the 2020 IFIP Networking Conference (Networking), Paris, France, 22–26 June 2020.
30. Sadio, O.; Ngom, I.; Lishou, C. Lightweight Security Scheme for MQTT/MQTT-SN Protocol. In Proceedings of the 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), Granada, Spain, 22–25 October 2019; pp. 119–123. [CrossRef]
31. Bang, A.; Rao, U. Design and evaluation of a novel White-box encryption scheme for resource-constrained IoT devices. *J. Supercomput.* **2022**, *78*, 11111–11137. [CrossRef]
32. Kao, T.; Wang, H.; Li, J. Safe MQTT-SN: A lightweight secure encrypted communication in IoT. *J. Phys. Conf. Ser.* **2021**, *2020*, 012044. [CrossRef]
33. Kumar.N.V, R.; Kumar P, M. Survey on State of Art IoT Protocols and Applications. In Proceedings of the 2020 International Conference on Computational Intelligence for Smart Power System and Sustainable Energy (CISPSSE), Keonjhar, India, 29–31 July 2020. [CrossRef]
34. Zhao, B.; Ji, S.; Lee, W.H.; Lin, C.; Weng, H.; Wu, J.; Zhou, P.; Fang, L.; Beyah, R. A Large-Scale Empirical Study on the Vulnerability of Deployed IoT Devices. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 1826–1840. [CrossRef]
35. Dunkels, A.; Gronvall, B.; Voigt, T. Contiki—A lightweight and flexible operating system for tiny networked sensors. In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, Tampa, FL, USA, 16–18 November 2004; pp. 455–462. [CrossRef]
36. Orebaugh, A.; Ramirez, G.; Beale, J. *Wireshark & Ethereal Network Protocol Analyzer Toolkit*; Elsevier: Amsterdam, The Netherlands, 2006.
37. Boudouaia, M.A.; Ali-Pacha, A.; Abouaissa, A.; Lorenz, P. Security Against Rank Attack in RPL Protocol. *IEEE Netw.* **2020**, *34*, 133–139. [CrossRef]
38. Singh, M.; Rajan, M.A.; Shivraj, V.L.; Balamuralidhar, P. Secure MQTT for Internet of Things (IoT). In Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, India, 4–6 April 2015; pp. 746–751. [CrossRef]
39. Roldán-Gómez, J.; Boubeta-Puig, J.; Castelo Gómez, J.M.; Carrillo-Mondéjar, J.; Martínez Martínez, J.L. Attack Pattern Recognition in the Internet of Things using Complex Event Processing and Machine Learning. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 17–20 October 2021; pp. 1919–1926. [CrossRef]
40. Corral-Plaza, D.; Medina-Bulo, I.; Ortiz, G.; Boubeta-Puig, J. A stream processing architecture for heterogeneous data sources in the Internet of Things. *Comput. Stand. Interfaces* **2020**, *70*, 103426. [CrossRef]
41. Ortiz, G.; Boubeta-Puig, J.; Criado, J.; Corral-Plaza, D.; Garcia-de Prado, A.; Medina-Bulo, I.; Iribarne, L. A microservice architecture for real-time IoT data processing: A reusable Web of things approach for smart ports. *Comput. Stand. Interfaces* **2022**, *81*, 103604. [CrossRef]
42. Roldán-Gómez, J.; Boubeta-Puig, J.; Pachacama-Castillo, G.; Ortiz, G.; Martínez, J.L. Detecting security attacks in cyber-physical systems: A comparison of Mule and WSO2 intelligent IoT architectures. *PeerJ Comput. Sci.* **2021**, *7*, e787. [CrossRef] [PubMed]
43. Martinez, A.; Kak, A. PCA versus LDA. *IEEE Trans. Pattern Anal. Mach. Intell.* **2001**, *23*, 228–233. [CrossRef]

CHAPTER 3

Integrating Complex Event Processing and Machine Learning: an intelligent architecture for detecting IoT security attacks

- **Title**: Integrating Complex Event Processing and Machine Learning: an Intelligent Architecture for Detecting IoT Security Attacks.
- **Authors**: José Roldán-Gómez, Juan Boubeta-Puig, José Luis Martínez, Guadalupe Ortiz
- **Type**: Journal paper.
- **Journal**: Expert Systems with Applications.
- **Publisher**: Elsevier
- **ISSN**: 0957-4174.
- **Status**: Published.
- **Publication date**: July 2020.
- **Volume**: 2020.
- **Paper Number**: 113251.
- **DOI**: 10.1016/j.eswa.2020.113251
- **JCR IF/ranking**: 6.954/Q1 (JCR2020).

Expert Systems With Applications 149 (2020) 113251

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Integrating complex event processing and machine learning: An intelligent architecture for detecting IoT security attacks

José Roldán[a,*], Juan Boubeta-Puig[b], José Luis Martínez[a], Guadalupe Ortiz[b]

[a] Research Institute of Informatics (i3a), University of Castilla-La Mancha, Campus Universitario s/n, 02071, Albacete, Spain
[b] Department of Computer Science and Engineering, University of Cadiz, Avda. de la Universidad de Cádiz 10, 11519 Puerto Real, Cádiz, Spain

ARTICLE INFO

Article history:
Received 1 August 2019
Revised 22 December 2019
Accepted 25 January 2020
Available online 30 January 2020

Keywords:
Complex event processing
Machine learning
Software architecture
Intelligent decision making
Internet of Things
Security attack

ABSTRACT

The Internet of Things (IoT) is growing globally at a fast pace: people now find themselves surrounded by a variety of IoT devices such as smartphones and wearables in their everyday lives. Additionally, smart environments, such as smart healthcare systems, smart industries and smart cities, benefit from sensors and actuators interconnected through the IoT. However, the increase in IoT devices has brought with it the challenge of promptly detecting and combating the cybersecurity attacks and threats that target them, including malware, privacy breaches and denial of service attacks, among others. To tackle this challenge, this paper proposes an intelligent architecture that integrates Complex Event Processing (CEP) technology and the Machine Learning (ML) paradigm in order to detect different types of IoT security attacks in real time. In particular, such an architecture is capable of easily managing event patterns whose conditions depend on values obtained by ML algorithms. Additionally, a model-driven graphical tool for security attack pattern definition and automatic code generation is provided, hiding all the complexity derived from implementation details from domain experts. The proposed architecture has been applied in the case of a healthcare IoT network to validate its ability to detect attacks made by malicious devices. The results obtained demonstrate that this architecture satisfactorily fulfils its objectives.

1. Introduction

The Internet of Things (IoT) is expanding globally, providing diverse benefits in nearly every aspect of our lives, such as healthcare, entertainment, transportation, industry, smart cities and even our daily routine. Every single device is able to connect to the Internet and communicate with the web or mobile applications, or even share data with other objects (Atzori, Iera, & Morabito, 2010). However, it is important to bear in mind that these autonomous objects are provided with limited features, such as small memories, a low bandwidth communication channel, a small processor, and low cost sensors or actuators, so although this confers a form of intelligence (smart objects), it is limited.

Nowadays, the trend for the IoT market is for continued growth, as indicated by CISCO, and it is expected to be worth around 14.4 trillion dollars between 2013 and 2022 (Raynovich, 2017). Nevertheless, emerging IoT technologies face various security attacks and threats, which include malware, privacy breaches, Denial of Service (DoS) attacks, security vulnerabilities with available exploits, and disruption of IoT networks (Andrea, Chrysostomou, & Hadjichristofi, 2015). Taking into account the inherent computational limitations of IoT devices in addition to their vulnerabilities, as well as their expected proliferation worldwide, both the risks and the projected global impact of connecting IoT devices to the network in any modern environment become evident. Not only do IoT devices have to be protected, but also the communications between them: huge volumes of data can be exchanged between devices and between the latter and servers or consumers, and the compromised availability, integrity, or confidentiality of these data can have a great impact.

In this field, in recent years, Machine Learning (ML) has been used to detect anomalous behaviour in the IoT (Buczak & Guven, 2016; Gharibian & Ghorbani, 2007; Meidan et al., 2017). Basically, the proposals consist in trying to train a model to understand the normal behaviour of an infrastructure or system and to detect when a new security issue occurs. However, when testing the model, numerous false positives—considering false positives as legit packets detected as anomalies— can appear until the model is correctly adjusted. As an example, a pure ML-based Intrusion Detection System (IDS) usually fails when the scenario is very complex, producing many false positives. Therefore, there is a gap in

* Corresponding author.
E-mail addresses: jose.roldan@uclm.es (J. Roldán), juan.boubeta@uca.es (J. Boubeta-Puig), joseluis.martinez@uclm.es (J. Luis Martínez), guadalupe.ortiz@uca.es (G. Ortiz).

https://doi.org/10.1016/j.eswa.2020.113251
0957-4174/

Chapter 3. Integrating Complex Event Processing and Machine Learning: an Intelligent Architecture for Detecting IoT Security Attacks

this field with has not yet been covered: there are no tools or approaches which enable the prediction and detection of unknown attacks in the field of the IoT, bearing in mind that in this scenario resources may be limited and it is essential to detect such attacks in real time. In addition, since the handling of IoT devices is not limited to specially qualified users, as nowadays any citizen with a basic knowledge of the technology can install and use such devices, the tools that detect attacks should provide a graphical interface that allows such users to benefit from the knowledge of a computer expert.

In order to cover this gap, we propose to use Complex Event Processing (CEP) technology in conjunction with ML techniques. CEP is a technology that is designed to process, analyze and correlate big amounts of real-time data produced by IoT devices and systems with the aim of promptly detecting situations of interest in multiple domains (Kousiouris et al., 2018; Garcia-de Prado, Ortiz, & Boubeta-Puig, 2017; Terroso-Saenz, Gonzlez-Vidal, Ramallo-Gonzlez, & Skarmeta, 2019). Even though CEP has demonstrated benefits such as its ability for efficiently processing network security data on the fly without being previously stored (Gad, Boubeta-Puig, Kappes, & Medina-Bulo, 2012; Gad, Kappes, Boubeta-Puig, & Medina-Bulo, 2013), only some isolated works make use of this technology for detecting security attacks and threats (Gad et al., 2013; Vegh & Miclea, 2016).

Considering such benefits provided by CEP and ML, in this paper, we propose an intelligent architecture that integrates both CEP and ML in order to promptly detect IoT security attacks by defining event patterns whose conditions depend on values of the network packets predicted by ML algorithms. More specifically, this proposal is based on our previously proposed Event-Driven Service-Oriented Architecture (SOA 2.0) (Boubeta-Puig, Ortiz, & Medina-Bulo, 2015), which has been extended with ML capabilities and then applied to the IoT security domain.

In such an architecture one of the key parts is the Enterprise Service Bus (ESB): the ESB is the middleware that makes it possible to create the data connection between IoT networks and both the CEP engine and ML algorithms, as well as to allow the CEP engine to communicate with data consumers such as trusted computers, email servers and alert databases. In this way, the CEP engine can receive both network packet events as a result of preprocessing IoT sensing data, and prediction network packet events generated upon training ML algorithm execution. By analyzing and correlating these events through the use of event patterns, the CEP engine is capable of detecting IoT security attacks in real time, as well as notifying them to data consumers through the ESB. Thanks to the integration of the ESB and the ML algorithms with the MEdit4CEP model-driven approach (Boubeta-Puig et al., 2015), security domain experts can graphically define which types of security attack must be analyzed and prevented, without having advanced knowledge about CEP, ML or IoT networks. The graphical security attack models are then automatically transformed into implementation code, which is automatically deployed in the proposed architecture at runtime. Therefore, the main contribution of this paper is an SOA 2.0 integrating CEP, ML, IoT and MEdit4CEP for detecting IoT security attacks and threats in a user-friendly way.

To sum up, the main contributions of this paper are twofold: firstly, the integration of an SOA 2.0 architecture with ML techniques which facilitates the real-time detection and prevention of security attacks in the IoT; secondly, the extension of MEdit4CEP to support the graphical definition of the security attacks to be detected and prevented in the new enhanced architecture. Furthermore, in order to validate our proposal, the architecture has been applied to an IoT network prototype constructed in a hospital with the aim of detecting attacks made by a malicious device. The results confirm that the integration of CEP with ML is powerful in this realistic scenario, and that this architecture works effectively when using a model that can be adapted to the context of the scenario in question. Since a model can be defined with many layers similarly to those proposed by the Open System Interconnections (OSI) and Transmission Control Protocol(TCP)/Internet Protocol (IP) models (Rondeau, Temple, & Lopez, 2019), our proposal is able to adapt to any layers from physical to enterprise network ones. To address it, the only requirement is to redefine the packet features.

The rest of the paper is organized as follows. Section 2 explains the basic concepts needed to understand the work presented. Section 3 deals with the main components of the proposed architecture integrating CEP and ML. Then, Section 4 presents the application of the architecture to detect IoT security attacks. Section 5 shows the results obtained, while Section 6 describes the related work. Finally, Section 7 draws some conclusions and describes lines for future research.

2. Background

The subject matters relevant to the content of this paper is introduced in this section.

2.1. Complex event processing

CEP is a technology that allows the capture, analysis and correlation of large amounts of simple events with the aim of detecting relevant situations in a particular domain (Luckham, 2012). Captured events are data that, for instance, flow through information systems or are provided by IoT devices, among others. They are called simple events because they are usually raw data. From the processing of such simple events, we can infer information with a greater semantic knowledge in real time, therefore obtaining the so-called complex events.

Thus, in a given context, we will be interested in detecting relevant situations according to the domain in question, that is, complex events. To do this, we will have to define a series of event patterns that specify the conditions that must be met by input simple events to detect such situations. These patterns are defined in a CEP, which is a software used to match these patterns over the incoming event streams, and capable of analyzing the data and highlighting detected situations of interest in real time. Each CEP engine has its own specific Event Processing Language (EPL) for the pattern definition.

The main advantage of CEP compared with other traditional event analysis software is the ability to process large amounts of data and notify the interested parties of the detected situations of interest in real time, thus considerably reducing the latency in the decision-making process, and therefore giving a competitive advantage to the user in question.

2.2. Event-driven service-oriented architecture

SOA 2.0, also called ED-SOA, evolved from the traditional Service-Oriented Architecture (SOA). While in SOA communications are performed following the request/response paradigm (Papazoglou, 2012), in SOA 2.0 communications are performed in reaction to events occurring in the system (Taylor, 2009), i.e., events trigger asynchronous messages that are sent to independent software components. To achieve this integration, a software abstraction layer is required that is capable of integrating various heterogeneous data sources with the software components to be invoked. These functionalities are offered by an ESB (Papazoglou & Heuvel, 2006), which permits interoperability among several communication protocols and heterogeneous data sources and sinks. In addition to being in charge of transforming messages to the necessary protocols and their correct routing, the ESB can provide a

series of added functions such as security support, monitoring, auditing, and many others, depending on the particular ESB.

In this way, through the use of ED-SOA and an ESB, applications and services can respond quickly to asynchronous events and allow server and client to be completely decoupled. Despite all the advantages provided by SOA 2.0, this type of architecture might not be ideal for analyzing and correlating big amounts of data in terms of events. Therefore, we have integrated SOA 2.0 with CEP technology in order to be able not only to respond quickly to events, but also to process, analyze, and correlate these events in order to promptly detect situations of interest for a particular domain.

2.3. MEdit4CEP

As mentioned above, the integration of SOA 2.0 with CEP considerably improves real-time decision-making. In order to benefit from such integration, we need to accurately define event patterns that allow the detection of the situations of interest based on the incoming event stream. This is undoubtedly the task of the domain expert, who does not necessarily have to be an expert in software programming. Therefore, we must provide experts with a graphical tool to help them define the patterns correctly. MEdit4CEP (Boubeta-Puig et al., 2015) was designed and implemented for this purpose.

Thus, MEdit4CEP is a model-driven solution for real-time decision making in SOA 2.0. whose main aim is to facilitate the definition of event patterns for domain experts. This solution provides a graphical modeling editor for CEP domain definition and a graphical modeling editor for event pattern definition. It also provides automatic code generation and deployment from the patterns modeled by the domain expert. In particular, MEdit4CEP currently generates the pattern code in Esper EPL, the language supported by the Esper CEP engine (EsperTech, 2019a), and deploys the code in the engine at runtime.

2.4. Machine learning

ML techniques refer to the study of algorithms and systems that are capable of learning or acquiring knowledge from experiences. ML uses statistics with different kinds of algorithms to solve a problem by studying and analyzing data. Due to its success, ML has been used in an extensive range of applications including search engines, medical diagnosis, Deoxyribonucleic acid (DNA) sequence classification, and object recognition in computer vision, and, more recently, it has been applied to improve network security in terms of authentication, access control, anti-jamming offloading, DoS and malware detection (Buczak & Guven, 2016; Narudin, Feizollah, Anuar, & Gani, 2016; Ozay, Esnaola, Yarman Vural, Kulkarni, & Poor, 2016; Tan, Jamdagni, He, Nanda, & Liu, 2011).

In general, there are two types of ML algorithms: supervised learning and unsupervised learning. On the one hand, supervised learning uses data samples with known measurements and class membership to create a set of rules to classify data samples with known measurements and assign class membership. On the other hand, unsupervised learning does not require labeled data, as in the supervised learning, and investigates the similarity between the unlabeled data to cluster them into different groups (Al-Garadi, Mohamed, Al-Ali, Du, & Guizani, 2018).

The ML techniques used in this work are based on supervised learning. In particular, this paper uses a Linear Regression (Mihescu, 2011) method to model the relationship between a scalar response and many explanatory variables (multiple linear regression). The proposed model can be used to fit a predictive model generated from an observed dataset of values extracted from a regular network scenario. The key point is identifying the information that can be used to build the predictive model. After developing such a model, if additional values of the explanatory variables are collected without an associated response value, the fitted model can be used to make a prediction of the response: normal behaviour or anomalous behaviour. This paper also uses the Support Vector Regression (SVR) regressor. Note that the comparison of ML algorithms is out of the scope of this paper. We are aiming to show that our proposal can be adapted to any model; we use the same features set and a linear kernel (Basak, Pal, & Patranabis, 2007).

In particular, we will use a novel architecture (rules with CEP together with linear or SVR regression) and define manual patterns for common attacks, as port scans, and linear regression in conjunction with CEP against attacks over Message Queuing Telemetry Transport (MQTT). Although we can use CEP together with linear regression against common attacks, our goal is to show the advantage of our architecture against unknown attacks (or attacks that we have not been included in our model). Therefore, the use of CEP in conjunction with regression (linear or SVR in this case) is used to detect MQTT attacks, when we do not know the expected value of the attack packets. With this architecture we have a twofold benefit: on the one hand, we can predict attacks that we do not know in advance; on the other we can create patterns that detect well-known attacks.

2.5. IoT network communication

The IoT (Gubbi, Buyya, Marusic, & Palaniswami, 2013) is a powerful paradigm in which many objects are able to obtain relevant information and communicate with each other.

Although the IoT allows us to upgrade interaction with our surroundings, it creates new challenges to be overcome, such as constrained devices, dynamic networks, battery autonomy and heterogeneous protocols. In the field of security management, all these challenges might be a handicap, so we need new approaches and tools to improve the current situation. As an example, the integration of the IoT paradigm with CEP technology allows us to extract important information, understand the situation, act accordingly and improve our living and working conditions (Boubeta-Puig, Ortiz, & Medina-Bulo, 2014).

MQTT is a very common protocol used in then IoT (OASIS, 2019). MQTT offers a small overhead (2 bytes from its fixed header, normally). Moreover, MQTT is a binary protocol that reduces the overhead compared with other application layer protocols. It is a publish/subscribe-based protocol in which a server (there can be more than one), known as the message broker, manages the information flow organized in a hierarchy of topics. Each client can be a subscriber and a publisher simultaneously. When a publisher device needs to send new information, it publishes a topic containing the information in the broker, then the broker distributes this information to all subscriber devices, which have been previously subscribed to this topic in particular. Although there are other IoT protocols, we use MQTT because it is consolidated and widely used. In any case, our solution can be adapted for different technologies and protocols.

2.6. Security attacks

Security attacks, which can be classified as critical or soft, are a constant threat for every system. Therefore, taking actions is necessary to avoid unrecoverable problems that could affect our confidentiality, integrity or availability. IoT systems have several constraints, such as a small memory, a dynamic and heterogeneous network and a limited battery, that prevent developers from using a usual security setup. Therefore, the actions for this kind of system need to be reconsidered.

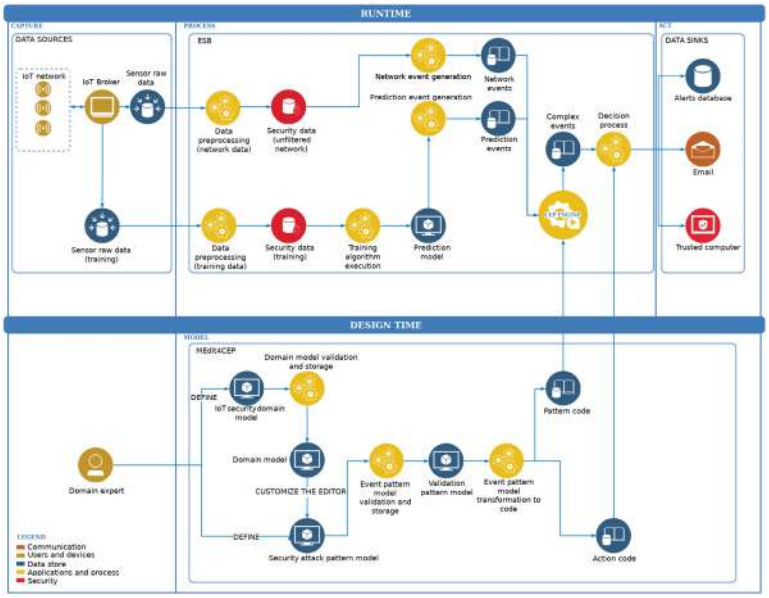


Fig. 1. The proposed architecture for detecting IoT security attacks.

A very common attack against IoT systems consists of producing a DoS of the system. There are many methods to perform this, such as, for example, the Mirai attack, which consisted in gaining access to the system using common credentials against Telnet, Secure Shell (SSH) or other services, thus allowing the attacker to take control of the device. The attacker, through the use of a big number of devices, usually uses this method to flood a target, which is the victim.

The first step before attacking the target is usually to carry out port scanning. This is not an attack per se but it makes it possible to be aware of the open ports. The attacker can craft and send packets and interpret the responses to perform the port scans.

Since DoS and port scans are very common, we have considered both of them in our application scenario.

3. Proposed architecture

In this section, our proposed architecture is described. This architecture is composed of two distinct parts. The first part includes the processes to be executed at runtime (top half of Fig. 1), i.e. the behaviour of our system, whereas the second part covers the processes that take place at design time (bottom half of Fig. 1).

3.1. Runtime architecture

In particular, the runtime part is based on an SOA 2.0, which is responsible for: (1) capturing the real-time IoT data produced by data sources; (2) processing, analyzing and correlating such data reached by the ESB; and (3) acting and making decisions about the situations of interest detected, as well as notifying data sinks of them.

In this architecture, IoT networks are used as data sources. IoT networks are composed of several IoT nodes connected to a message broker, which is in charge of sending all raw sensor data to the ESB through the MQTT protocol, as well as generating the raw sensor data for training purpose in an isolated scenario without any security attacks.

The ESB receives the real-time raw sensor data, preprocesses them and transforms them into a common format of network events. Additionally, the ESB receives the training raw sensor data and preprocesses them to make them consumable for our training ML algorithm (linear regression or SVR in this case). Then, the ESB produces a trained model, which will be used to generate *NetworkPrediction* events containing a timestamp, a predicted value (of packet length in this case) and a margin where a normal value should be. Each packet in the network will have an associated *NetworkPrediction* event. Note that the difference between a training packet and a normal packet is that the training packet is used during the training stage, while the normal packet is predicted by the trained model.

We should point out that the values of the key features (features essential for detecting a specific scenario) for validating a legitimate packet are not necessarily fixed, as the domain expert can use the predictor and generate a dynamic predictor with the knowledge obtained for a legit scenario. If a packet's real value obtained from a predicted feature is outside a prediction bound for this kind of packet, the system will detect it. Since the predictor has the ability to adapt to different cases, it works correctly not only with homogeneous scenarios but also with heterogeneous ones.

Afterwards, the CEP engine, which has been integrated with the ESB, receives both network events and *NetworkPrediction* events,

and it also analyzes and correlates these events in order to detect and predict security attacks (complex events) in real time, according to the event patterns previously defined and deployed in the engine.

Upon complex event detection, the decision-process component will promptly notify the data sinks (trusted computers, email servers and alerts databases, among others) previously subscribed to this type of complex events.

3.2. Design time architecture

The design time part of the architecture is based on the MEdit4CEP model-driven approach (Boubeta-Puig et al., 2015), whose main aim is the definition of high-level models, which are understandable to any user. These models are automatically transformed into code that is executable in the CEP engine and the decision-process component located in the ESB.

As shown in the bottom half of Fig. 1, the design time part includes the domain expert, representing people who possess a wide knowledge of network security and so understand the normal behaviour of the system to protect, but do not necessarily have knowledge of CEP or ML. These experts are suitable candidates to accurately define the CEP domain composed of the network event and *NetworkPrediction* event types with their properties, as well as to graphically define the security attack event patterns through a graphical tool, hiding them from the implementation details.

The steps involved for the definition and code generation of the IoT security domain and event patterns are as follows:

1. The domain expert creates a graphical IoT security domain model from scratch, defining event types with their properties.
2. Once the domain model is defined, the editor will be responsible for its syntactic validation. If it is not valid, the domain expert will be advised to correct the detected errors. This model will then be saved.
3. The domain expert will create security attack pattern models. To do this, the graphical editor will be customized with the previously defined domain model. One should take into account that the defined patterns might include the values obtained by the ML algorithm dynamically.
4. Once pattern models have been defined, the editor will be responsible for their validation. If any of them are invalid, the user will be asked to correct the detected errors. These models will then be saved.
5. Each of these pattern models will be automatically transformed into code, which consists of both the EPL the code implementing the conditions to be satisfied so that the CEP engine can detect the security attacks, and code of actions to be carried out in the ESB when detecting prospective security attacks.
6. The event pattern code generated will be added to the CEP engine while the action code generated will be added to the decision-process component in the ESB at runtime.

It is worth noting that the design time part of the architecture provides two advantages. Firstly, the use of MEdit4CEP allows the domain expert to easily design the IoT security domain model and security attack patterns without requiring any programming language. Secondly, the connection with a predictor makes it possible to model dynamic patterns, in which the definition of some event properties depends on the values automatically computed by the predictor. We use the term **dynamic** patterns because our predictor computes the estimated value for a legitimate packet by using the features of these packets, and the patterns make use of such prediction values. Therefore, the combination of MEdit4CEP and the predictor makes our architecture user-friendly, customized and adaptable to new attacks and security scenarios.

The main advantage of our proposal against existing rule-based IDS is that our architecture is able to detect attacks that have not been defined in our model, as explained in Section 5. So, we could define an "unknown attack pattern" to be triggered when the incoming packets contain uncommon values. Additionally, this could be used for a smart definition of thresholds for known attacks, such as a DoS. Therefore, our architecture is also very versatile.

4. Applying the architecture to detect IoT security attacks

The aim of this section is to provide a real-world scenario in which our proposed architecture can be applied in order to check that it works correctly.

In particular, we have applied our architecture to an IoT network prototype constructed in a hospital. This network is composed of noise sensors located in different rooms of the hospital. The measurements taken by the sensors could be used to detect noise levels in real time as well as to notify nurses when a noise level is too high in a particular room. In this way, a nurse could go to this room as soon as possible to recommend that the patient's visitors speak more quietly, thus improving the patient's well-being.

In this section, we focus on detecting IoT security attacks in this network by using our proposed architecture. Note that the detection of noise level is beyond the scope of this paper.

4.1. Data sources

To test our proposal we designed and constructed such an IoT network as the data source for our architecture.

Fig. 2 depicts the network topology, which is composed of three legitimate devices, a broker and a malicious device for implementing attacks. The devices use the MQTT protocol to share information with the broker; in this case, they share information about noise.

Although this network is simple, it is effective in illustrating the proposed architecture. Note that the same network topology could be applied to other application domains in which devices share other types of information.

Additionally, this IoT network has been emulated with Virtual Box (Oracle, 2019), obtaining similar results to those when using real devices. Specifically, an application in which each device generates random numbers and publishes them in its own topic has been emulated. We have used virtual machines because these make it possible to modify the set up quickly and change the test cases easily.

4.2. ESB

The functionalities provided by the ESB (see Section 3) have been implemented by using Mule ESB 3.9 (MuleSoft, 2019). Our ESB application is composed of four data flows, as detailed in the following paragraphs.

The first data flow is responsible for gathering the raw sensor data through the MQTT protocol endpoint, transforming them into network event format and then sending the events to the Esper 7.1 CEP engine (EsperTech, 2019a).

The second data flow is in charge of predicting events from the training sensor data received through another MQTT protocol endpoint. To do this, we have implemented a predictor in Python 3.6 (Python Software Foundation, 2019) using the Numpy (NumPy, 2019), Scikit-learn (Scikit-learn, 2019) and Pandas (Pandas, 2019) libraries. This predictor uses traces of packages captured using Wireshark (Wireshark, 2019) in the broker to fit the model. Since Pandas works easily with data in the Comma-Separated Values

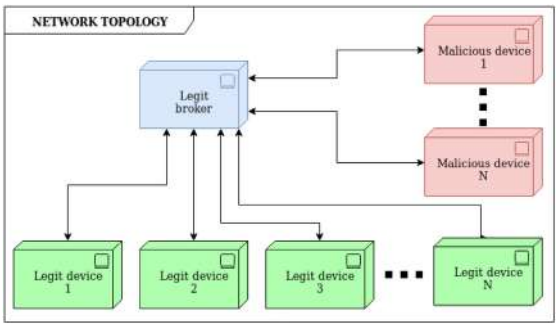


Fig. 2. IoT network topology for the real-world scenario.

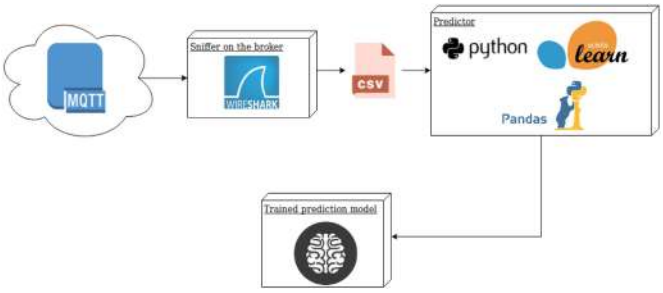


Fig. 3. Dataflow scheme.

(CSV) format and Wireshark makes it possible to export the captured data in CSV, the predictor also works with this format. Fig. 3 illustrates the dataflow scheme. Note that, if needed, Wireshark could be replaced by another packet analyzer because our architecture is entirely modular.

In particular, our predictor currently works with linear regression (or SVR algorithms in some experiments), which are essential to fit our model by using a *normal case* in which the system works correctly without attacks. Additionally, we have defined port scans and attacks (see Section 4.5) to test both our predictor and the system behaviour when it suffers an attack. We should point out that, although we have used linear regression and SVR because of their suitability for our features, other ML methods could be used in our architecture if necessary.

The third data flow is responsible for both receiving the Esper EPL pattern code automatically generated by MEdit4CEP and deploying it in the Esper CEP engine, while the fourth data flow receives the action code generated by MEdit4CEP and makes the decisions about which data sinks will be notified about the situations of interest detected in real time.

4.3. Data sinks

Three types of data sinks have been considered in this architecture: (1) alert databases, on which the detected situations of interest are stored; (2) emails with their body containing the description of the detected situations to be notified to the subscribed users; and (3) trusted computers in charge of displaying such situations on dashboards in real time.

Since our architecture is flexible, other data sinks could be included in the future, according to the domain expert's needs.

4.4. IoT security domain

The *IoTSecurity* domain has been graphically modeled by using MEdit4CEP, as shown in Fig. 4. This domain is composed of two types of event types: *NetworkPacket* and *NetworkPrediction*.

In particular, *NetworkPacket* defines the event information required for traces of packets captured by network protocol analyzers, such as Wireshark. As depicted in Fig. 4, the event properties have been modeled according to the OSI reference model (Zimmermann, 1980), i.e. grouping the properties by OSI layers. Therefore, *NetworkPacket* contains the time (in epoch format) at which the packet is captured, the packet length in bytes, the packet info that is defined by Wireshark and depends on the protocol, and also the following nested properties: *networkInterface*, *internet*, *TCP*, User Datagram Protocol (*UDP*) and *MQTT*.

The *networkInterface* property is composed of:

- *sourceMAC*: sender's MAC address.
- *destinationMAC*: receiver's MAC address.

The *internet* property is composed of:

- *sourceIP*: sender's IP address.
- *destinationIP*: receiver's IP address.

Fig. 4. IoT security domain modeled with MEdit4CEP.

- *protocol*: upper layer protocol.

The *TCP* property is composed of:

- *sourcePort*: TCP port of the sender device.
- *destinationPort*: TCP port of the receiver device.
- *flags*: flags provided by the TCP protocol indicating the following actions:
  1. Synchronization *(SYN)*: used for connection establishment. It is only used in the first packet from the sender and the receiver.
  2. Acknowledgement *(ACK)*: used to acknowledge the packets that have been received.
  3. Push *(PSH)*: used to send a packet immediately to the application layer, without requiring the receiver to wait for the maximum segment size buffering the packet.
  4. Urgent *(URG)*: allows marking packets as urgent, these being processed first.
  5. Finish *(FIN)*: used to request TCP to terminate the connection gracefully without data loss.
  6. Reset *(RST)*: used to request TCP to terminate the connection abruptly—data loss could occur.
- *calculatedWindowSize*: indicates the size of the connection window.

The UDP property is composed of:

- *sourcePort*: UDP port of the sender device.
- *destinationPort*: UDP port of the receiver device.

The MQTT property is composed of:

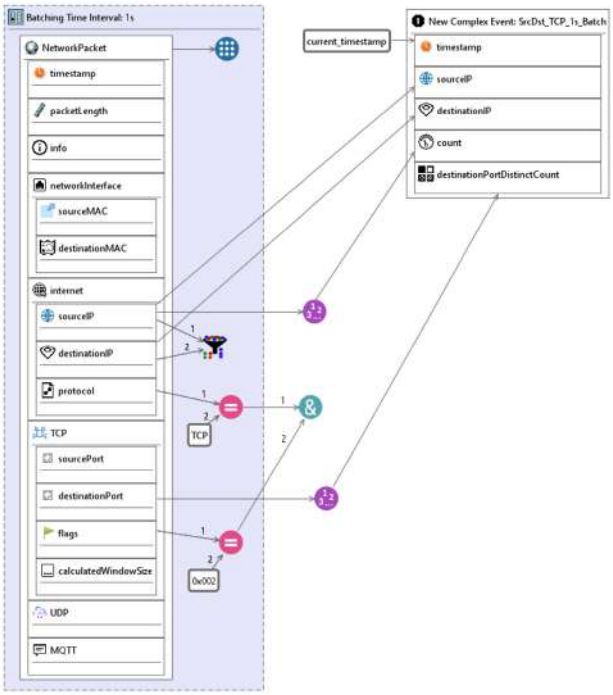
- *flags*: these indicate the kind of MQTT packet—there being 14 different packets, but their study is beyond the scope of this paper. Also, the publish packet provides three more flags, namely:
  1. Duplicate delivery of a publish Control Packet (*DUP*): indicates whether a publish message has been resent. This happens when there is a QoS greater than 0 and there is no ACK for the original packet.
  2. Publish Quality of Service (*QoS*): indicates the quality of service level. Three levels are represented by using 2 bits. Level 0 fulfils the *at most once* scheme; in this level there is no resending. Level 1 ensures that the packet meets the *at least once* schema, but duplicated packets at the receiver could exist. Level 2 ensures that the packet fulfils the *exactly once* schema.
  3. Publish Retain flag (*RETAIN*): This flag makes setting a fixed message (only one) per topic possible. In this way, when a new client subscribes to the topic, it will receive the retained message. Each client can manage the message as needed, although this flag is usually used to update the state of new devices.
- *message*: the message contained in the MQTT packet.
- *topic*: topic's name.
- *messageLength*: MQTT packet's length.
- *frameCounter*: the number of fragmented packets that make up a full packet.

In addition, the *NetworkPrediction* event type contains the time (in epoch format) at which the prediction is calculated, the *packetLengthPredict* property, which represents the value predicted by our predictor based on the destination port, protocol, frame counter and calculated windows size, and the *packetLengthPredictSquaredError* property, which indicates the squared error committed by our predictor in the training stage.

4.5. Security attack patterns

Once the *IoTSecurity* domain has been designed, the event pattern editor is automatically reconfigured for this domain, i.e., the event types modeled in the domain are included in the tool palette of the editor.

In order to validate our proposal, we consider the modeling of the following five patterns of attacks, in which a malicious device generates attacks against the MQTT network.



(a) Pattern model.

```
@Name("SrcDst_TCP_1s_Batch")
@Tag(name="domainName", value="IoTSecurity")
insert into SrcDst_TCP_1s_Batch
select current_timestamp() as timestamp,
   a1.internet.sourceIP as sourceIP,
   a1.internet.destinationIP as destinationIP,
   count(a1.internet.sourceIP) as count,
   count(distinct a1.TCP.destinationPort) as destinationPortDistinctCount
from pattern [(every a1 = NetworkPacket((a1.internet.protocol = 'TCP' and
   a1.TCP.flags = '0x002'))).win:time_batch(1 seconds)
group by a1.internet.sourceIP, a1.internet.destinationIP
```

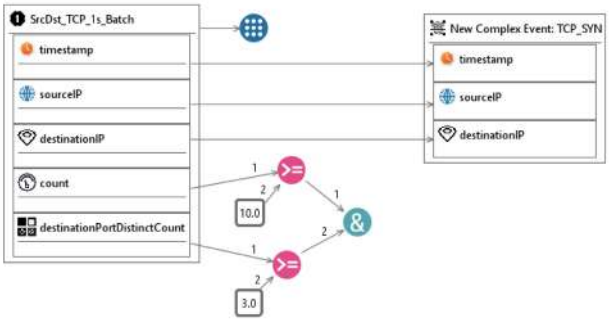
(b) Pattern implementation in Esper EPL.

Fig. 5. *SrcDst_TCP_1s_Batch* pattern modeled and generated with MEdit4CEP.

4.5.1. TCP/SYN port scan

In the *TCP/SYN port scan* attack, the malicious device sends a round of 10 or more TCP packets with SYN flag to 3 or more different ports of the broker in 1 s. If the port is open, the broker sends a SYN/ACK packet, otherwise it sends an RST packet.

By using the MEdit4CEP tool, this attack has been modeled as two patterns. The *SrcDst_TCP_1s_Batch* pattern (see Fig. 5)(a) is responsible for obtaining all the network packet events whose protocol is TCP and whose flags are 0x002 in 1-s batch windows; then, the number of packet events as well as the number of distinct TCP destination ports in every batch window are calculated and grouped by source IP and destination IP. This information, together with the current timestamp, the source IP and the destination IP, is inserted into complex events named *SrcDst_TCP_1s_Batch*. Once this pattern has been syntactically validated, the Esper EPL



(a) Pattern model.

```
@Name("TCP_SYN")
@Tag(name="domainName", value="IoTSecurity")
insert into TCP_SYN
select a1.timestamp as timestamp,
   a1.sourceIP as sourceIP,
   a1.destinationIP as destinationIP
from pattern [(every a1 = SrcDst_TCP_1s_Batch((a1.count >= 10 and
   a1.destinationPortDistinctCount >= 3)))]
```

(b) Pattern implementation in Esper EPL.

Fig. 6. *TCP_SYN* pattern modeled and generated with MEdit4CEP.

code, which is illustrated in Fig. 5(b), is automatically generated by MEdit4CEP.

Afterwards, for each *SrcDst_TCP_1s_Batch* complex event created, the *TCP_SYN* pattern modeled (see Fig. 6)(a) checks whether both its count property is greater or equal to 10 and the destination port distinct count is greater or equal to 3. Once this pattern has been syntactically validated, the Esper EPL code, which is depicted in Fig. 6(b), is automatically generated.

4.5.2. UDP port scan

In the *UDP port scan* attack, the malicious device sends a round of 10 or more empty UDP packets to 3 or more different ports of the broker in 5 s. If the broker sends any response, then the port is open, but if the broker does not send a response, the port could be open. If the broker sends ICMP unreachable, the port should be closed. And if it sends another error (not unreachable), the port should be filtered.

This attack has been modeled in a similar way to the *TCP/SYN port scan* attack, but replacing the 1-s batch window by a 5-s one, and the TCP protocol property by the UDP one. The modeled *SrcDst_UDP_5s_Batch* and the *UDP_Port_Scan* patterns are shown in Fig. 7(a) and Fig. 7(b), respectively.

4.5.3. Xmas port scan

In the *Xmas port scan* attack, the malicious device sends a round of 10 or more TCP packets with PSH, FIN and URG flags to 2 or more different ports of the broker in 1 s. If the broker does not respond, the port should be open or filtered. If the broker sends an RST packet, it should be closed. If the broker sends an ICMP unreachable error, it should be filtered. Note that although this attack/scan does not work frequently, it is useful for evaluating our proposal.

This attack has been modeled in a similar way to the *TCP/SYN port scan* attack, but replacing the *0x002* TCP flag property by *0x029*, and updating the condition of destination port distinct count to greater or equal to 2, instead of 3. The modeled *SrcDst_Xmas_1s_Batch* and the *Xmas_Scan* patterns are shown in Fig. 8(a) and Fig. 8(b), respectively.

4.5.4. DoS with big messages

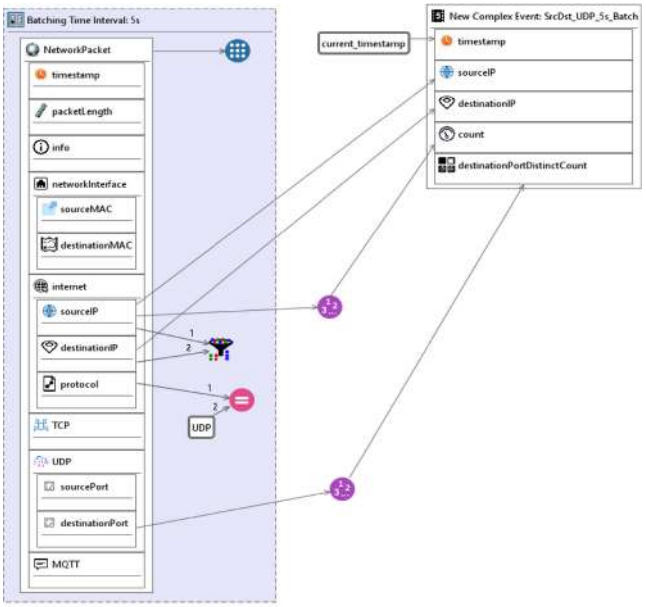
In the *DoS with big messages* attack, the malicious device publishes a heavy message to the broker, producing a DoS.

It is worth noting that this attack has been defined as a *dynamic* pattern (see Fig. 9) through the combining of CEP and ML. As can be seen, this pattern searches MQTT publish messages (publish is defined by flag *0x30*) that are beyond the upper or lower limit. We do not need to define the limit, since our regressor does it. Once the regressor has been trained with a part of the legitimate case (the normal MQTT case), it can predict a value for each MQTT packet by using its features, thus generating a *NetworkPrediction* event. This pattern is automatically transformed into Esper EPL code, as shown in Fig. 9(b).

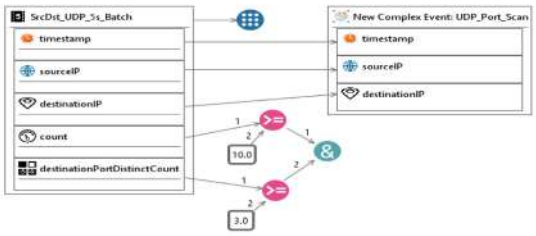
4.5.5. Anomaly

The *Anomaly* pattern's goal is to detect unknown attacks. This makes it possible to detect attacks which we have not modeled and our predictor does not know.

This pattern has been also defined as a *dynamic* pattern (see Fig. 10)(a). As can be seen, this pattern is triggered when a packet contains a non-known protocol (a non-known protocol is any protocol that we do not have in our normal scenario). This is also



(a) *SrcDst_UDP_5s_Batch* pattern model.



(b) *UDP_Port_Scan* pattern model.

Fig. 7. *SrcDst_UDP_5s_Batch* and *UDP_Port_Scan* patterns modeled with MEdit4CEP.

triggered when a packet has an anomalous length value. Note that we have used this value because it is linearly and directly correlated with many important features to detect anomalies (see Fig. 11) and we make use of the same predictor used in the DoS pattern. It is important to highlight that this pattern can be defined in many domains by adapting the features and the model.

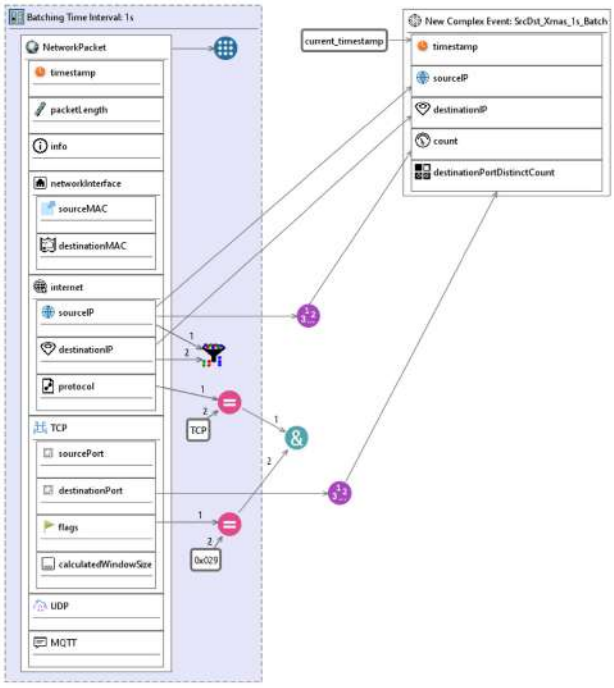
This pattern is automatically transformed into Esper EPL code, as shown in Fig. 10(b).

4.5.6. Discwave attack

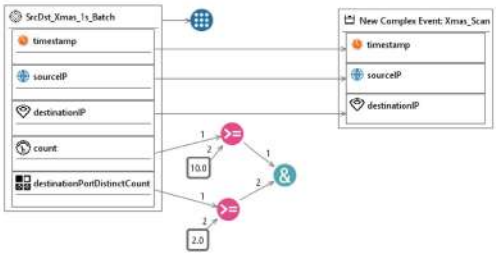
In the *DoS with big messages* attack, the malicious device sends many connects commands with different identifications (MQTT broker uses the id to identify each client) against the broker. This attack can be very dangerous in many systems, where the broker is configured for expel devices with old connections (when a new device with the same id sends a connect command). It can shut down all the MQTT network. Notice that this attack is very different from the DoS attack.

4.5.7. Subfuzzing attack

Subfuzzing attack has been designed to map the topics hierarchy when MQTT wildcards are disabled. The wildcards allow us to subscribe multiple topics simultaneously. The attack is quite simple, only a list of possible topics is required. The attacker sends multiple topics subscription attempts with different topic names. If it receives a *suback* packet, then it knows that the topic exists.



(a) *SrcDst_Xmas_1s_Batch* pattern model.



(b) *Xmas_Scan* pattern model.

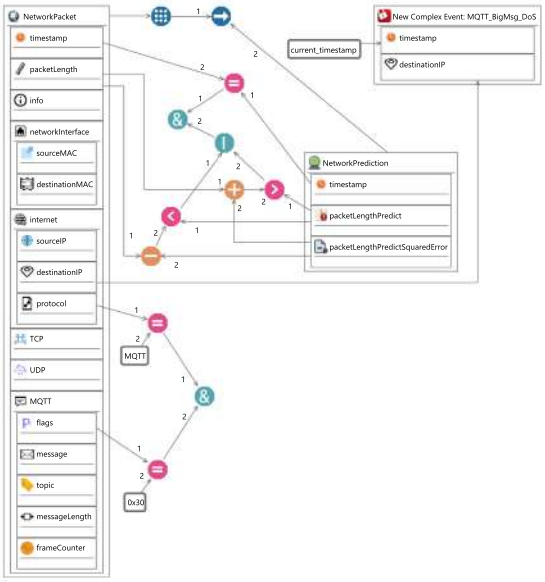
Fig. 8. *SrcDst_Xmas_1s_Batch* and *Xmas_Scan* patterns modeled with MEdit4CEP.

4.6. Mirai

Mirai is a well-known malware, although our predictor has not been trained with it. In this case, we try to detect the first Mirai stage (we want detect the attack before it can infect us). This first stage sends multiple usernames and passwords over Telnet to gain access, after that it runs (or downloads) the malware payload.

5. Experiments and results

This section describes and discusses the experiments performed and the results obtained in order to demonstrate the viability of the proposed architecture for detecting IoT attacks. In particular, we have conducted three types of experiments.



(a) Pattern model.

```
@Name("MQTT_BigMsg_DoS")
@Tag(name="domainName", value="IoTSecurity")
insert into MQTT_BigMsg_DoS
select current_timestamp() as timestamp, a1.internet.destinationIP as destinationIP
from pattern [((every a1 = NetworkPacket((a1.internet.protocol = 'MQTT' and
   a1.MQTT.flags = '0x30')))) -> a2 = NetworkPrediction((a2.timestamp = a1.timestamp
   and (a2.packetLengthPredict < (a1.packetLength -
   a2.packetLengthPredictSquaredError) or a2.packetLengthPredict >
   (a1.packetLength + a2.packetLengthPredictSquaredError)))))]
```

(b) Pattern implementation in Esper EPL.

Fig. 9. *MQTT_BigMsg_DoS* pattern modeled and generated with MEdit4CEP.

First, we have conducted a features selection process, which allows us to find representative features. Moreover, in this case, it can be used to determine the correct ML model.

Second, we have modeled the events patterns for detecting the well-known attacks such as—TCP, UDP and Xmas port scans (see Sections 4.5.1–4.5.3) and also DoS attack (see Section 4.5.4)—against our network (see Section 4.1). The simpler port scans are modeled by using the CEP engine without the predictor and the DoS attack by using a linear regression predictor to determine the threshold depending of the understanding of the network.

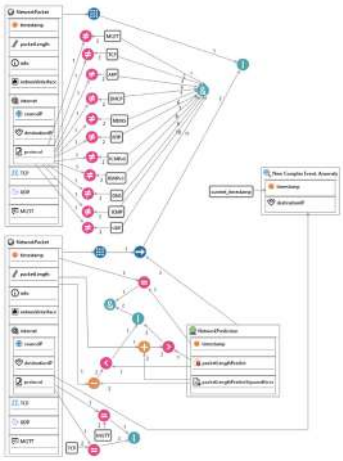
Third, departing from the model latter, it has been also modeled the event pattern Anomaly (see Section 4.5.5) against the broker by making use of the CEP engine together with the linear regression predictor (the same used in DoS pattern). Furthermore, in order to demonstrate that other predictors are also valid, the SVR predictor has been also include in the experiments.

Finally, in order to check the Anomaly detection pattern, we have included in the network three new attacks which have not been identified before by the proposed model such as –Discwave, Subfuzzing and Mirai (depicted in Sections 4.5.6, 4.5.7 and 4.6)–. All of them are detected by the model as anomalies and, after an analysis, they can be included and modeled as new patterns to further classify them in the future.

All these patterns have been validated and automatically transformed into Esper EPL code. Then, this code has been deployed in the CEP engine at runtime, producing the following results that are compared in Section 5.5.

5.1. Feature selection

The set of features have been selected with a simple and solid criteria proposed by KDD99 (Kayacik, Zincir-Heywood, & Heywood, 2005) (those which are adaptable to a MQTT scenario) and we have also added features extracted from MQTT. The MQTT features have been selected heuristically; although, it is adaptable to any protocol. We have used Extremely Randomized Trees with



(a) Pattern model.

```
@Name("Anomaly")
@Tag(name="domainName", value="IoTSecurity")
insert into Anomaly
select current_timestamp() as timestamp, a2.internet.destinationIP as destinationIP
from pattern [(
   (every a1 = NetworkPacket(a1.internet.protocol != 'MQTT' and
   a1.internet.protocol != 'TCP' and a1.internet.protocol != 'ARP' and
   a1.internet.protocol != 'DHCP' and a1.internet.protocol != 'MDNS' and
   a1.internet.protocol != 'NTP' and a1.internet.protocol != 'ICMPv6' and
   a1.internet.protocol != 'IGMPv3' and a1.internet.protocol != 'DNS' and
   a1.internet.protocol != 'ICMP' and a1.internet.protocol != 'UDP'))
   or ((every a2 = NetworkPacket((a2.internet.protocol = 'MQTT' or
   a2.internet.protocol = 'TCP'))) -> a3 = NetworkPrediction((a3.timestamp = a2.timestamp
   and (a3.packetLengthPredict < (a2.packetLength -
   a3.packetLengthPredictSquaredError) or a3.packetLengthPredict >
   (a2.packetLength + a3.packetLengthPredictSquaredError))))))]
```

(b) Pattern implementation in Esper EPL.

Fig. 10. *Anomaly* pattern modeled and generated with MEdit4CEP.

datasets composed by MQTT regular traffic, background and malicious attacks over MQTT (brute force, broker spoofing and malformed MQTT packets). This allows us measure the importance of each feature.

As we can see in Fig. 11, our features fits perfectly in this domain. It is important to highlight that we have not used particular datasets to fit the feature selection process or system evaluation.

We emphasize that some features are correlated with others ones; for example, packet length is correlated with others features such as calculated window size, protocol used and the information field. Thus, packet length feature is linearly and directly correlated with other features and fits well with a linear regressor and it is useful to detect unknown attacks.

Obviously, in other cases we will need a more complex predictor with other features, but this is not a limitation for our architecture: the modular architecture allow us to modify any component easily.

5.2. CEP engine without the predictor

As Fig. 12 shows, there are 2 complex events that have detected UDP port scans. However, why are there only 2 complex events although the scan is prolonged in time? This has happened because UDP/SCAN is a very slow scan and the pattern conditions require that there are at least 10 packets in 5 seconds. These conditions are satisfied only in the first part of our scan, which generated several packets in few seconds, as illustrated in this figure. After that, there is a slower flow of UDP packets, and no more complex events are detected.

Similarly to the UDP port scan, the results of the Xmas port scan (see Fig. 13) show that, at the beginning, the TCP flow is more dense.

Fig. 14 shows the detection of TCP/SYN port scans. As can be seen, TCP/SYN is very fast (less than 0.5 s). Therefore, only one complex event has been triggered after finishing the full scan.

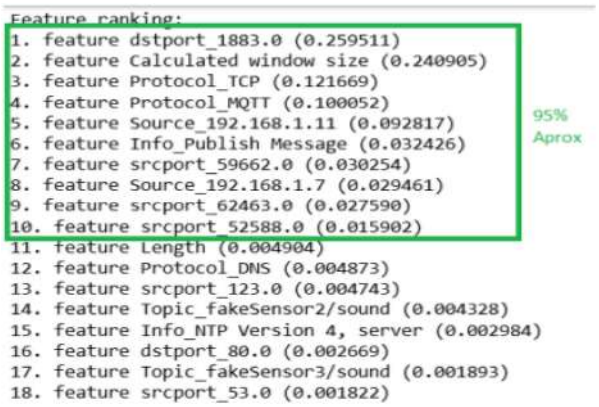


Fig. 11. Feature importance.

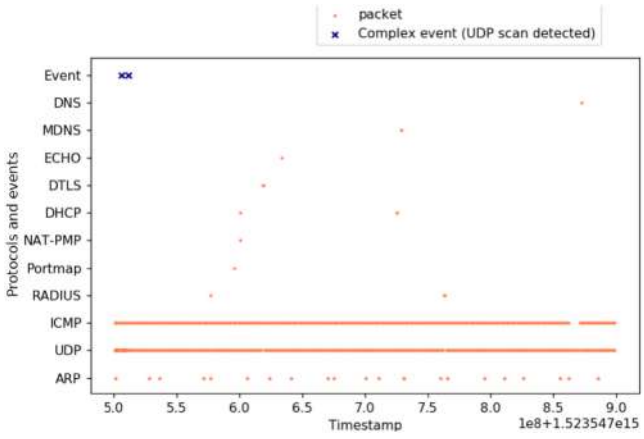


Fig. 12. UDP port scan detection.

With theses experiments we confirm that CEP is a suitable technology to detect such port scanning patterns in real time. However, in order to define such patterns appropriately, we have needed to study the different attacks to set the condition values correctly.

5.3. CEP engine together with the linear regression predictor

To evaluate the integration of CEP with linear regression, we have analyzed the exactitude of our regressor, studied the exactitude of our predictor for the DoS scenario and checked whether our predictor is able to set predicted values for legitimate packages and the DoS scenario properly, as explained in this section.

Regarding the exactitude of our regressor, we can affirm that it works correctly, as shown in Fig. 15. This situation is normal because the legitimate case is similar to the case used for training our regressor.

In the graphs in Fig. 15 and 16 we can see blue squares and orange circles. A blue square represents the real size of the packets, and an orange circle represents the predicted values. Therefore, if our architecture works correctly, there will be circles in the center of the squares when the system is not under attack, but these circles will be out of the square when there are strange packets in our network.

Regarding the exactitude of our predictor in the DoS scenario, as depicted in Fig. 16, many predictions failed. The difference between predicted values and real values is greater compared with Fig. 15 since there are more packets and they are bigger.

Tables 1 and 2 summarize the numerical results illustrated in Figs. 15 and 16 respectively, which represent the distribution of the

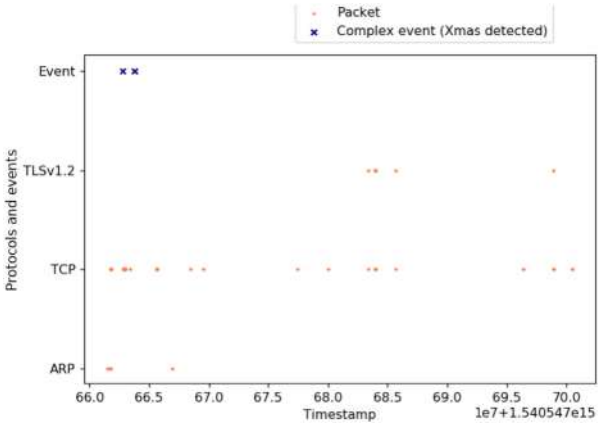


Fig. 13. Xmas port scan detection.

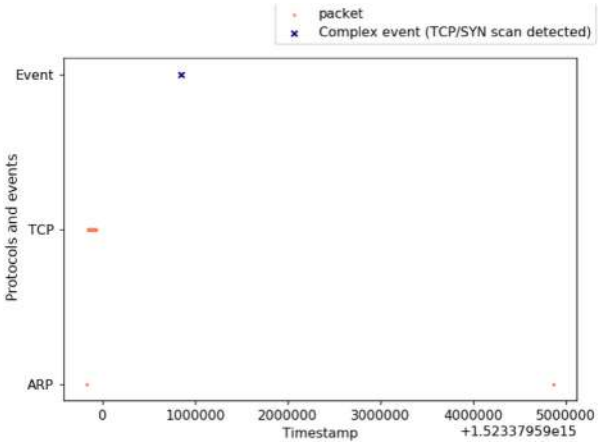


Fig. 14. TCP/SYN port scan detection.

errors (real packet length against predicted packet length). While in the legitimate MQTT scenario the error values are small, these are much bigger in the DoS scenario.

Finally, in regard to whether our predictor is able to set predicted values within the upper and lower limit for legitimate cases and for our DoS cases, we obtained the following results. The legitimate case prediction errors are illustrated in Fig. 17. Note that the error margin of our predictor is computed using the error of our linear regression with our training dataset (in our case), and the prediction error is computed as the difference between the

Table 1
Legitimate MQTT packet length prediction error.

Quartile	Quartile value	Number of packets
Q1	x <0.00166298	4214
Q2	0.00166298 <x <0.01077055	3648
Q3	0.01077055<x <0.01086934	7696
Q4	0.01086934 <x	1295

Table 2
DoS MQTT packet length prediction error.

Quartile	Quartile value	Number of packets
Q1	x <201.10600944	27,106
Q2	358.89399057 <x <201.10600944	26,977
Q3	761.10600943<x <358.89399057	26,735
Q4	761.10600943 <x	26,632

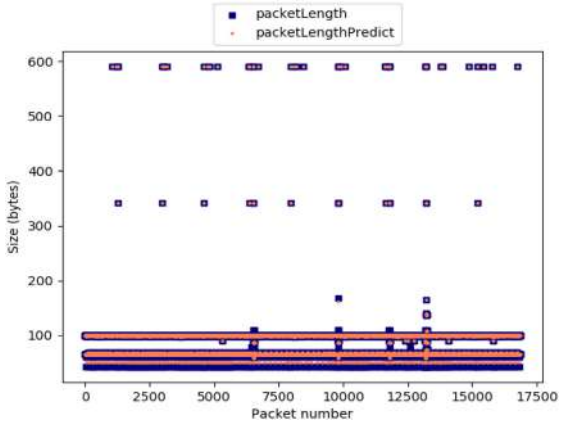


Fig. 15. Legitimate MQTT predictions.

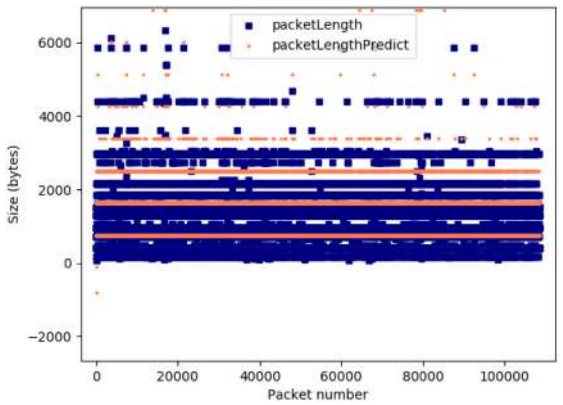


Fig. 16. DoS MQTT predictions.

predicted value and the real value (in our case), and if a prediction error (represented with orange points) is over the error margin (represented with the blue line), there is a missprediction. In the previous figures we only compared the predicted value with the real one for our prediction. If our architecture works correctly, there will just be a few of orange points over the blue line when we have a legitimate case (this means that these packet values, represented with orange points, are very different compared with the predicted value). However, there should be many points over the line when we receive an attack, as this lets us detect these attacks correctly. In Fig. 17 only 5 packets have been miss-predicted, but none of them are MQTT packets. This happens because our training dataset does not address packets other than MQTT protocol ones and the regressor cannot deal with them. Such additional protocols cannot trigger our patterns anyway, and they are not particularly relevant to this paper.

Fig. 18 shows the prediction errors when grouping the packets by protocol. The biggest errors are produced on ICMPv6 and ARP. This is because we do not have many packets of these protocols in our dataset and we are using linear regression, so the model does not give much importance to these packets. Table 3 presents the number of packets by protocol, showing that our predictor is worse when it needs to predict a very different value in packets with few samples. We should point out that this model is appropriate for our goal—detecting MQTT DoS—, but it could be replaced by a more complex one if we needed to make more exact predictions.

According to Fig. 19, our regressor's prediction is so far from the correct value when it receives an abnormal packet that, makes it possible to discover the malicious packets. This is exactly what we intended to achieve, i.e. we have a feature to distinguish legitimate packets from malicious packets. Of course, our architecture

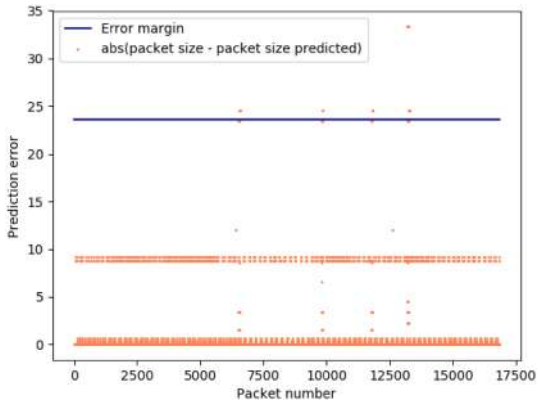


Fig. 17. Legitimate MQTT margin predictions.

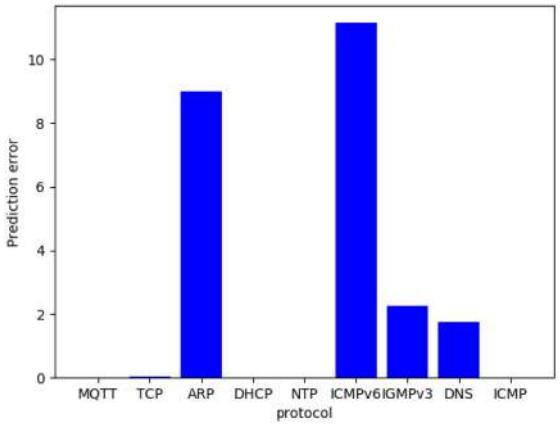


Fig. 18. Errors by protocol.

could be adapted to achieve other objectives. As an example, we could add a more complex model to predict different values. Note that the aim of this paper is to present a proof of concept demonstrating that our proposal can work appropriately by using *dynamic* patterns in which a model is in charge of calculating their prediction and margin values. In this way, the definition of dynamic pattern' conditions is more exact than static pattern' conditions; the latter being fixed and manually defined by domain experts.

Table 3
Number of packets by protocol for the legit case.

Protocol	Number of packets
MQTT	8098
TCP	8105
ARP	481
DHCP	58
NTP	14
ICMPv6	59
IGMPv3	16
DNS	20
ICMP	2

Fig. 20 highlights the main benefits of our novel proposal, namely; (1) the dynamic pattern is able to detect all the DoS packets; (2) the pattern has not been triggered by any legit packet: (3) there are no false positives or undetected DoS packets; (4) the value and the error margin for the pattern have not been defined manually; and so (5) the pattern conditions are adaptable at runtime.

As a result, we can affirm that the integration of the CEP engine with linear regression prediction permits the detection of security attacks in real time with a low error margin.

Besides, we have conducted a performance evaluation. Results have also demonstrated that the architecture performance is appropriate. Our normal scenario generated 17,040 packets in 2

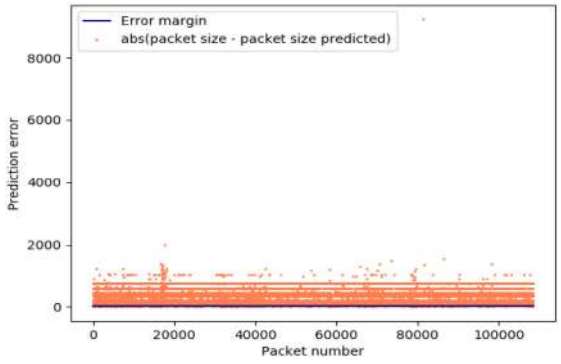


Fig. 19. DoS MQTT margin predictions.

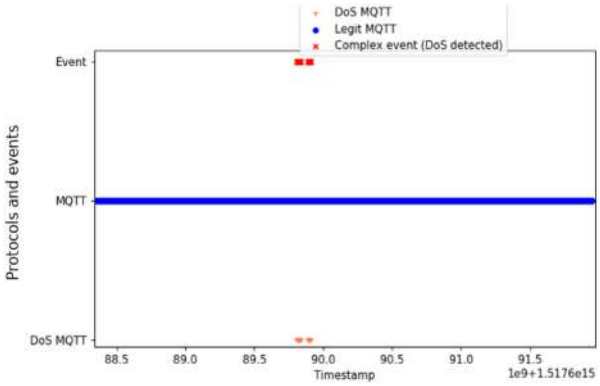


Fig. 20. MQTT DoS predictions.

hours, in other words, 2.3 packets each second. In the worst case (TCP/SYN scan), 2006 packets were generated in 5 seconds, i.e., 401 packets per second. Our predictor just needed 0.47 seconds to pre-process the training data and fit with the legit scenario. In addition, this process can be done offline when necessary. Note that online training is not recommended in a cybersecurity scenario since an attacker could poison the data to modify the model so as to evade our IDS (Jagielski et al., 2018). In addition, Esper 7.1 is able to process up to 6 millions of events per second (EPS) (EsperTech, 2019b); thereby the number of packets in a normal scenario can be processed in near-real time. Therefore, we can conclude that our architecture can work with a massive network since Esper is able to scale easily.

5.4. CEP engine together with the linear regression predictor against non-modeled threats

In this section we evaluate the integration of CEP with linear regression against non-modeled attacks. A non-modeled attack is an attack that our system unknowns. This means that we have not previously defined a pattern to detect that attack and, thus, the predictor has not been trained against this attack.

Table 4 shows the number of attacks, the packets produced for each attack and the packets detected. Note that one attempt of attack can generate various packets, we say that we are detecting them if we detect at least 1 packet of the attack. As we can see, all attacks are detected and there are no false positives.

Since we can detect Mirai and its derived malware (Discwave and Subfuzzing attacks), and all of them are pure IoT attacks—Mirai is one of the more common attack against IoT systems Antonakakis et al. (2017))—, we can conclude that our pattern works appropri-

Table 4
Results of CEP and Linear Regression against non-modeled attacks.

Category	Attempts	Simple events	Anomaly complex events
Normal traffic (new)	7930	7930	0
Discwave	100,000	700,025	400,027
Subfuzzing	4097	4097	4097
Mirai	126	787	504

Table 5
Results of Linear Regression as predictor.

	Predicted as negative	Predicted as positive
Is negative	7930 (True negative)	0 (False positive)
Is positive	0 (False negative)	104,223 (True positive)
Precision = 1	Recall = 1	F1-score= 1

ately. So, we are able to detect attacks that our architecture does not know in advance (they are not previously modeled).

Other interesting metrics are True Negatives (TN), False Positives (FP), True Positives (TP) and False Negatives (FN). True negative is a well predicted normal packet. False positive is a bad predicted normal packet (predicted as an anomalous packet). False negative is a bad predicted anomalous packet (predicted as a normal packet). True positive is a well predicted packet that is an anomalous packet. Moreover we will use 3 more metrics.

Precision:

$$\left(\frac{TP}{TP+FP}\right)$$

Recall:

$$\left(\frac{TP}{TP+FN}\right)$$

and F1 score:

$$2 \cdot \left(\frac{Precision \cdot Recall}{Precision + Recall}\right)$$

All these metrics take values between 0 and 1. A high precision score shows that the model does not detect many false positives, a high recall shows that the model detect many threats. And F1 score is a general metric.

Table 5 shows that, in this context, we get a perfect score. Notice that we have not modeled Mirai, discwave and subfuzzing attacks. We can conclude that our proposal works very well in a real scenario against unknown attacks.

5.5. Proposal comparison

This section compares our proposal against other commercial alternatives.

First, we will use Snort, a well-known rule-based IDS, to detect MQTT attacks and scans. By using Snort, we need to define our rules to detect each attack. Snort have 2 main problems. One problem is that it cannot detect novel attacks, i.e. if Snort has not a specific rule for a particular attack, then it cannot detect such attack. The second problem is that the definition of Snort rules can be hard, since you need to indicate the exact byte for each field (for example, MQTT flags). Our proposal solves both problems. On the one hand, we can detect novel attacks, as previously shown. On the other, MEdit4CEP allows us to define patterns easily with a powerful graphical tool.

In this comparison we use these standard Snort rules: *'attack-responses.rules'*, *'backdoor.rules'*, *'bad-traffic.rules'*, *'chat.rules'*, *'ddos.rules'*, *'dns. rules'*, *'dos.rules'*, *'experimental.rules'*, *'exploit.rules'*, *'finger.rules'*, *'ftp.rules'*, *'icmp-info.rules'*, *'icmp.rules'*, *'imap.rules'*, *'info.rules'*, *'misc.rules'*, *'multimedia. rules'*, *'mysql.rules'*, *'netbios.rules'*, *'nntp.rules'*, *'oracle.rules'*, *'other-ids.rules'*, *'p2p. rules'*, *'policy.rules'*, *'pop2.rules'*, *'pop3.rules'*, *'rpc.rules'*, *'rservices.rules'*, *'scan.rules'*, *'smtp.rules'*, *'snmp.rules'*, *'sql.rules'*, *'telnet.rules'*, *'tftp.rules'*, *'virus.rules'*, *'web-attacks.rules'*, *'web-cgi.rules'*, *'web-client.rules'*, *'web- coldfusion.rules'*, *'web-frontpage.rules'*, *'web-iis.rules'*, *'web-misc.rules'*, *'web-php. rules'*, *'x11.rules'*, *'community-sql-injection.rules'*, *'community-web-client.rules'*, *'community-web-dos.rules'*, *'community-web-iis.rules'*, *'community-web-misc. rules'*, *'community-web-php.rules'*, *'community-sql-injection.rules'*, *'community-web-client. rules'*, *'community-web-dos.rules'* and *'community-web-iis.rules'*. Moreover, we have added a rule that is able to detect MQTT packets with more than 400 bytes of length; we use it to detect DoS attacks over MQTT.

Table 6
Our proposal against Snort.

Attack	Snort	Our proposal
TCP Syn scan	YES	YES
UDP port scan	YES	YES
Xmas port scan	YES	YES
MQTT DoS bigmsg	YES	YES
MQTT Discwave	NO	YES
MQTT subbfuzzing	NO	YES
MIRAI	YES	YES

Table 6 shows the advantages of our proposal compared to Snort. As we mentioned before, our proposal does not need specific rules for each attack; we can detect anomalous packets without a specific rule. Anyway, we could create new rules for each new anomaly. This is very interesting and allows us to filter attacks better. We can conclude that purely rule-based IDSs are dependent on the rules created from known threats; therefore, this kind of IDS is useless against non-known threats and new malware (Hammarberg, 2014; Holm, 2014).

In addition to evaluating the advantages of our approach against Snort, we have done a comparison with very popular IDS/SIEMs. Table 7 summarizes the different features, hardware requirements and performance of common IDS/SIEMs with respect to our proposal. In particular, we have evaluated the following features:

- F1: this provides rule-based detection.
- F2: this supports anomaly detection.
- F3: this provides full ARM support.
- F4: minimum core requirements.
- F5: minimum RAM requirements.
- F6: disk capacity requirements.
- F7: EPS/Gbps (depending on the developer).

As we can see, most approaches provide anomaly detection and rule-based detection, but their main problem is that they require too many hardware resources and are not compatible with ARM architectures (or any non x64/x86 architecture), so these cannot be deployed on an IoT context. In contrast, our proposal can be deployed in a Raspberry or other constrained devices. Regarding performance comparison, we have obtained the measurements from the IDS/SIEM official websites and we can conclude than our proposal has a better performance rate (measured in EPS) than other alternatives (Mathew, 2014).

Table 7 also shows that the options examined cannot be deployed on a Rapberry Pi 3 or other constrained devices, with the exception of Snort. Anyway, Snort does not provide anomaly detection features. In general, all these systems have a poor resource-performance relationship. In the past we performed several performance tests on an architecture integrating the Esper CEP engine with Mule ESB to provide COLLECT, a collaborative context-aware SOA for intelligent decision-making in the Internet of Things (Garcia-de Prado et al., 2017). A case study with several of the collaborative nodes was provided, and two nodes were deployed on two Raspberry Pi3's (Model B), with 1GB of RAM memory and a 2 GHz 64-bit quad-core ARMv8, with 8GB and 16GB cards, respectively. Even though the implementation was not focused on IoT attacks and did not provide prediction but only detection of relevant situations for the domain question, the results of the performance and stress evaluations carried out for the architecture are applicable in this paper. As we can see in Garcia-de Prado et al. (2017),

Table 7
Our proposal against other IDS/SIEMs.

	F1	F2	F3	F4	F5	F6	F7
Snort	YES	NO	YES	–	–	–	70,000 aprox
Splunk (Splunk, 2019)	YES	YES	NO	12 cores x64/x86	12GB	600GB	154,000 low density
Qradar 1699 (IBM, 2019)	YES	YES	NO	4 cores x64	12GB	256GB	Up to 20,000
LogRhythm LR-NM3400 (LogRhythm, 2017)	YES	YES	NO	12 cores x64/x86	64GB	4 TB aprox	1 Gbps
Alienvault USM (AlienVault, 2018)	YES	YES	NO	8 cores x64/x86	24GB	4.2 TB	Up to 2500
Our proposal (constrained device)	YES	YES	YES	< 4 cores ARM	< 1GB	< 8GB	Up to 900 aprox
Our proposal (no constrained device)	YES	YES	YES	1 core (2Gh)	4GB	–	Up to 120,000

the Raspberry Pi CPU throughput and memory usage remain within reasonable values maintaining the response time under 25 ms with a load of up to 300 events per second. Therefore, we can conclude that comparatively, taking into account the resources of the machine, the existing alternatives (AlienVault, 2019; LogRhythm, 2019; Pechta, 2017; Shah & Issac, 2018) have a poorer performance or require many more resources.

Therefore, these results show that our proposal is more adaptable and lighter than the other analyzed ones. Additionally, it can achieve the best performance compared with other SIEMs.

Moreover, we have measured the performance using a new dataset that has 1,424,903 simple events (as we mentioned, each simple event is a packet or a prediction). By using this dataset as input data, 404,627 complex events have been generated by the CEP engine. With this dataset, a mean processing rate of 39,565.96 events/second is obtained when the architecture is deployed on an Intel Core i7-8550U CPU @ 1.80GHz 1.99 GHz, 16GB RAM. Note that, although this is a high mean rate, this is not a peak rate, i.e. our architecture is capable of managing a higher traffic data rate. Thus, depending on the necessities of the system, we can deploy it on a resource constrained device, on a computer or on a large server, obtaining a reasonably good performance, in accordance with the device used.

5.6. CEP with support vector regression

As previously mentioned, linear regression works adequately in our context. However, to demonstrate that our system can manage other regressors, we have conducted other experiments integrating it with the Epsilon-Support Vector Regression, a kernel-based regressor that allow us to probe other model (Basak et al., 2007).

In particular, we have used Mirai, subfuzzing and discwave as anomalous packets/events, and added more normal packets as normal behaviour packets. Tables 8 and 9 show that all attacks have been correctly detected again, and only 2 false positives have been obtained. Even though these results obtained by using the Support Vector Regression (SVR) regressor are worse than those by using the lineal one, they are still good enough for the IoT context.

Table 8
Results of CEP and SVR against non-modeled attacks.

Category	Attempts	Simple events	Anomaly complex events
Normal traffic (new2)	7930	7930	2
Discwave	100,000	700,025	323,584
Subfuzzing	4097	4097	4097
Mirai	126	787	787

Table 9
Results of SVR as predictor.

	Predicted as negative	Predicted as positive
Is negative	7928 (True negative)	2 (False positive)
Is positive	0 (False negative)	104,223 (True positive)
Precission = 0.99998	Recall = 1	F1-score= 0.99998

We can, therefore, conclude that our architecture can work with different regression models.

Table 10 shows the difference of linear regression against SVR in our context. As we can see, in this scenario (and with our features selection) SVR with a polynomial kernel works worse than linear regression. We have a lower precision (0.99998), which means that a few legit packets have been detected as anomalous packets with SVR. It is important to point out that these results are dependent on the feature selection and the scenario to be protected. With another scenario, SVR could provide better results than linear regression.

Table 10
Linear regression against SVR in our system.

	Linear regression	SVR
Recall	1	1
Precision	1	0.99998
F1-score	1	0.99998

6. Related work

In recent years, ML has been used to improve the security of computer systems. As an example, a set of techniques for each security attack class has been proposed by Xiao, Wan, Lu, Zhang, and Wu (2018).

Another interesting survey is the one conducted by Al-Garadi et al. (2018), providing a taxonomy scheme of ML and deep learning for IoT security. This paper discusses whether ML is more appropriate than deep learning, and vice versa, for each security issue. Additionally, this work explains the importance of choosing where the ML algorithm is executed. Li, Ota, and Dong (2018) propose a theoretical approach integrating ML into an edge computing scheme; however, real-world applications are not analyzed. Kulkarni and Venayagamoorthy (2009) implement an ML algorithm for securing a wireless sensor network against DoS attacks.

Thus, there are many papers that propose the use of ML for tackling IoT security issues, but they do not address the challenge of managing dynamic patterns. We propose a novel model combining ML and CEP in which an ML algorithm calculates heuristic values from observations in order to define event pattern conditions dynamically. These predicted values allow the modeling of flexible patterns for detecting IoT security attacks, avoiding many false positive alerts, which are very common on ML-based IDS.

There are a few papers that propose the integration of ML with CEP as a differentiating advantage. For instance, Raj, Sahu, Chaudhary, Prasad, and Agarwal (2017) defines a CEP-based approach for detecting meaningful patterns in smart buildings. Although the authors state that the CEP engine could be optimized by using efficient learning algorithms, this is considered as future work.

Baldoni, Montanari, and Rizzuto (2015) present an architecture combining CEP and ML algorithms designed using hidden

Markov models to detect anomalous behaviors of a safety-critical system, thus predicting a system failure before it happens. While this work focuses on safety-critical detection, our proposal tackles security-critical detection (attacks) in IoT networks. Furthermore, their model is state-based and this could be a problem for attack detection because an attack can be registered at any time (our system does not need to be in a specific state for the attack to be detected).

In addition, Petersen, To, and Maag (2017) add an ML technique together with CEP to the detection engine of a Mobile Ad Hoc Network Intrusion Detection System (MANET IDS), which is capable of recognizing DoS attacks as well as re-writing IDS rules on the fly. Similarly to our work, their work proposes the integration of ML and CEP for detecting attacks. However, while Petersen et al.'s work focuses on the MANET protocol, our work focuses on MQTT. Moreover, they propose SVM, but this algorithm requires finding a specific kernel, a task that is hard if the domain expert does not know the data distribution. SVM requires tuning other parameters such as gamma, degree, etc. Our proposal does not need to know the data distribution or to find the correct parameters. Unlike that work, we propose a complete architecture for detecting different types of IoT security attacks (not only DoS ones) in real time. Additionally, our architecture provides a model-driven graphical tool for event pattern definition and automatic code generation, hiding all implementation details from domain experts.

Therefore, taking into account the evaluation of known IDS/SIEMs provided in Section 5.5, together with the description of other research proposals described in this section, we can affirm that currently there are no tools or proposals which can be deployed on devices with limited resources, such as a Raspberry Pi, and which can detect and predict IoT unknown attacks in real time. However, when deploying the architecture on a more powerful device, we obtain a significantly good performance and, to the best of our knowledge, there are no existing model-driven proposals integrating CEP and ML for detecting IoT security attacks and threats. Thus, we are providing a unique graphical tool which will permit user, who do not necessarily have any programming knowledge to graphically design and automatically deploy means for attack detection.

7. Conclusions and future work

In this work, we have proposed and implemented an intelligent architecture combining CEP and ML that is capable of easily managing dynamic patterns for detecting IoT security attacks. Dynamic patterns have some event properties that depend on the values that are automatically computed by a linear regression and SVR predictor, a component that is also proposed in this paper.

Thanks to the use of the MEdit4CEP model-driven tool, domain experts (non experts in CEP and ML) can graphically model these patterns, which are then automatically transformed in Esper EPL code and deployed into the CEP engine at runtime.

The integration of CEP and ML has several benefits over using pure CEP without dynamic prediction. Firstly, we do not need to find correct fields, values and margins manually to compare features or event properties, since we make use of dynamic prediction. Secondly, if the network environment changes, our architecture lets the CEP engine quickly adapt to the new scenario. Therefore, using linear regression and SVR to calculate expected pattern values enables the detection of new attacks in real time.

In order to validate our proposal, the architecture has been applied to an IoT network prototype constructed in a hospital with the aim of detecting attacks (TCP, UDP and Xmas port scans, and DoS) made by a malicious device. The results confirm that this architecture works effectively if it uses a model that can be adapted to the context.

As future work, we plan to define more event patterns for detecting other types of attacks. Additionally, we will extend the predictor with the ability to automatically find which packet features are essential for our domain context. Moreover, we will automate alerts of "necessary training" when our network changes significately.

Funding

This work was supported by the Spanish Ministry of Science, Innovation and Universities and the European Union FEDER Funds [grant numbers FPU 17/02007, RTI2018-093608-B-C33, RTI2018-098156-B-C52 and RED2018-102654-T]. This work was also supported by the JCCM [grant number SB-PLY/17/180501/ 000353].

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.

Credit authorship contribution statement

José Roldán: Conceptualization, Software, Validation, Investigation, Resources, Writing - original draft, Visualization. **Juan Boubeta-Puig:** Conceptualization, Methodology, Software, Investigation, Resources, Writing - original draft, Writing - review & editing, Supervision, Funding acquisition. **José Luis Martínez:** Conceptualization, Methodology, Investigation, Writing - original draft, Writing - review & editing, Funding acquisition, Supervision, Project administration. **Guadalupe Ortiz:** Conceptualization, Methodology, Investigation, Writing - original draft, Writing - review & editing, Funding acquisition.

Acknowledgments

Boubeta-Puig would like to thank the High-Performance Networks and Architectures Research Group for their hospitality when visiting them at the University of Castilla-La Mancha, Spain, where part of this work was developed.

References

Al-Garadi, M. A., Mohamed, A., Al-Ali, A., Du, X., & Guizani, M. (2018). A survey of machine and deep learning methods for Internet of Things (IoT) security. arXiv preprint arXiv:1807.11023.

AlienVault (2018). Usm appliance deployment requirements. https://www.alienvault.com/documentation/usm-appliance/sys-reqs/hardware-spec.htm. Accessed 16 October 2019.

AlienVault (2019). Alienvault usm appliance. https://www.alienvault.com/documentation/usm-appliance/initial-setup/ossim-installation.htm. Accessed 16 October 2019.

Andrea, I., Chrysostomou, C., & Hadjichristofi, G. (2015). Internet of things: Security vulnerabilities and challenges. In *2015 IEEE symposium on computers and communication (ISCC)* (pp. 180–187). doi:10.1109/ISCC.2015.7405513.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., … Zhou, Y. (2017). Understanding the mirai botnet. In *26th USENIX security symposium (USENIX security 17)* (pp. 1093–1110). Vancouver, BC: USENIX Association.

Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. Computer Networks, *54*(15), 2787–2805. doi:10.1016/j.comnet.2010.05.010.

Baldoni, R., Montanari, L., & Rizzuto, M. (2015). On-line failure prediction in safety-critical systems. *Future Generation Computer Systems, 45*, 123–132. doi:10.1016/j.future.2014.11.015.

Basak, D., Pal, S., & Patranabis, D. C. (2007). Support vector regression. *Neural Information Processing, 11*, 203–224.

Boubeta-Puig, J., Ortiz, G., & Medina-Bulo, I. (2014). Approaching the Internet of Things through integrating SOA and complex event processing. In Z. Sun, & J. Yearwood (Eds.), Handbook of research on demand-driven web services: Theory, technologies, and applications. In *IGI Global book series Advances in Web Technologies and Engineering (AWTE)* (pp. 304–323). IGI Global.

Boubeta-Puig, J., Ortiz, G., & Medina-Bulo, I. (2015). MEdit4CEP: A model-driven solution for real-time decision making in SOA 2.0. *Knowledge-Based Systems, 89*, 97–112. doi:10.1016/j.knosys.2015.06.021.

Chapter 3. Integrating Complex Event Processing and Machine Learning: an Intelligent Architecture for Detecting IoT Security Attacks

Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials, 18*(2), 1153–1176. doi:10.1109/COMST.2015.2494502.

EsperTech (2019a). Esper - Complex Event Processing. http://www.espertech.com/esper/ Accessed 6 November 2019.

EsperTech (2019b). Esper - Complex Event Processing Scalability. http://www.espertech.com/2019/03/07/6-million-events-per-second/ Accessed 6 November 2019.

Gad, R., Boubeta-Puig, J., Kappes, M., & Medina-Bulo, I. (2012). Hierarchical events for efficient distributed network analysis and surveillance. In *Proceedings of the 2nd international workshop on adaptive services for the future internet and 6th international workshop on web APIs and service mashups*. In *WAS4FI-Mashups '12* (pp. 5–11). Bertinoro, Italy: ACM. doi:10.1145/2377836.2377839.

Gad, R., Kappes, M., Boubeta-Puig, J., & Medina-Bulo, I. (2013). Employing the CEP Paradigm for Network Analysis and Surveillance. In *Proceedings of the ninth advanced international conference on telecommunications* (pp. 204–210). Rome, Italy: IARIA.

Gharibian, F., & Ghorbani, A. A. (2007). Comparative study of supervised machine learning techniques for intrusion detection. In *Fifth annual conference on communication networks and services research (CNSR '07)* (pp. 350–358). doi:10.1109/CNSR.2007.22.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems, 29*(7), 1645–1660. doi:10.1016/j.future.2013.01.010.

Hammarberg, D. (2014). The best defenses against zero-day exploits for various–sized organizations. *SANS Institute InfoSec Reading Room, 21*.

Holm, H. (2014). Signature based intrusion detection for zero-day attacks: (not) a closed chapter? In 2014 47th Hawaii international conference on system sciences (pp. 4895–4904). doi:10.1109/HICSS.2014.600.

IBM (2019). System requirements for virtual appliances. https://www.ibm.com/support/knowledgecenter/en/SS42VS_7.3.2/com.ibm.qradar.doc/c_qradar_ha_vrt_ap_reqs.html Accessed 16 October 2019.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., & Li, B. (2018). Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In 2018 IEEE symposium on security and privacy (SP) (pp. 19–35). doi:10.1109/SP.2018.00057.

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: 94* (pp. 1722–1723).

Kousiouris, G., Akbar, A., Sancho, J., Ta-shma, P., Psychas, A., Kyriazis, D., & Varvarigou, T. (2018). An integrated information lifecycle management framework for exploiting social network data to identify dynamic large crowd concentration events in smart cities applications. *Future Generation Computer Systems, 78*, 516–530. doi:10.1016/j.future.2017.07.026.

Kulkarni, R. V., & Venayagamoorthy, G. K. (2009). Neural network based secure media access control protocol for wireless sensor networks. In *2009 International joint conference on neural networks* (pp. 1680–1687). doi:10.1109/IJCNN.2009.5179075.

Li, H., Ota, K., & Dong, M. (2018). Learning IoT in edge: Deep learning for the internet of things with edge computing. IEEE Network, *32*(1), 96–101. doi:10.1109/MNET.2018.1700202.

LogRhythm (2017). Product overview network monitor. https://logrhythm.com/pdfs/datasheets/lr-network-monitor-datasheet.pdf Accessed 16 October 2019.

LogRhythm (2019). Logrhythm netmon freemium. https://logrhythm.com/products/logrhythm-netmon-freemium/ Accessed 6 November 2019.

Luckham, D. (2012). *Event processing for business: Organizing the real-time enterprise*. New Jersey, USA: John Wiley & Sons.

Mathew, A. (2014). Benchmarking of complex event processing engine - ESPER. https://www.cse.iitb.ac.in/internal/techreports/reports/TR-CSE-2014-61.pdf Accessed 16 October 2019.

Meidan, Y., Bohadana, M., Shabtai, A., Guarnizo, J. D., Ochoa, M., Tippenhauer, N. O., & Elovici, Y. (2017). Profiliot: A machine learning approach for iot device identification based on network traffic analysis. In *Proceedings of the symposium on applied computing*. In *SAC '17* (pp. 506–509). New York, NY, USA: ACM. doi:10.1145/3019612.3019878.

Mihescu, M. C. (2011). Classification of learners using linear regression. In *2011 federated conference on computer science and information systems (FedCSIS)* (pp. 717–721).

MuleSoft (2019). Mule ESB. https://www.mulesoft.com/resources/esb/what-mule-esb Accessed 27 March 2019.

Narudin, F. A., Feizollah, A., Anuar, N. B., & Gani, A. (2016). Evaluation of machine learning classifiers for mobile malware detection. *Soft Computing, 20*(1), 343–357. doi:10.1007/s00500-014-1511-6.

NumPy (2019). NumPy. http://www.numpy.org/ Accessed 27 March 2019.

OASIS (2019). MQTT Version 5.0. http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html Accessed 27 March 2019.

Oracle (2019). Virtualbox. https://www.virtualbox.org/ Accessed 27 March 2019.

Ozay, M., Esnaola, I., Yarman Vural, F. T., Kulkarni, S. R., & Poor, H. V. (2016). Machine learning methods for attack detection in the smart grid. *IEEE Transactions on Neural Networks and Learning Systems, 27*(8), 1773–1786. doi:10.1109/TNNLS.2015.2404803.

Pandas (2019). Pandas. https://pandas.pydata.org/ Accessed 27 March 2019.

Papazoglou, M. (2012). *Web services and SOA: Principles and technology* (2nd). Essex, England; New York: Pearson Education.

Papazoglou, M., & Heuvel, W. V. D. (2006). Service-oriented design and development methodology. International Journal of Web Engineering and Technology, *2*(4), 412–442. doi:10.1504/IJWET.2006.010423.

Pechta, J. (2017). What are the limitations/restrictions for qradar community edition? https://developer.ibm.com/answers/questions/411519/what-are-the-limitationsrestrictions-for-qradar-co/ Accessed 16 October 2019.

Petersen, E., To, M. A., & Maag, S. (2017). A novel online CEP learning engine for MANET IDS. In 2017 IEEE 9th Latin-american conference on communications (LATINCOM) (pp. 1–6). doi:10.1109/LATINCOM.2017.8240196.

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 Systems with Applications, *85*, 231–248. doi:10.1016/j.eswa.2017.05.034.

Python Software Foundation (2019). Python. https://www.python.org/.

Raj, R., Sahu, R. K., Chaudhary, B., Prasad, B. R., & Agarwal, S. (2017). Real time complex event processing and analytics for smart building. In 2017 conference on information and communication technology (CICT) (pp. 1–6). doi:10.1109/INFOCOMTECH.2017.8340593.

Raynovich, R. (2017). What is the growth rate of iot markets? http://www.futuriom.com/articles/news/iot-growth-rate/2017/04 Accessed 15 March 2019.

Rondeau, C. M., Temple, M. A., & Lopez, J. (2019). Industrial iot cross-layer forensic investigation. *Wiley Interdisciplinary Reviews: Forensic Science, 1*(1), e1322. doi:10.1002/wfs2.1322.

Scikit-learn (2019). Scikit-learn. https://scikit-learn.org/stable/ Accessed 27 March 2019.

Shah, S., & Issac, B. (2018). Performance comparison of intrusion detection systems and application of machine learning to snort system. *Future Generation Computer Systems, 80*, 157–170. doi:10.1016/j.future.2017.10.016.

Splunk (2019). Reference hardware. https://docs.splunk.com/Documentation/Splunk/8.0.0/Capacity/Referencehardware Accessed 16 October 2019.

Tan, Z., Jamdagni, A., He, X., Nanda, P., & Liu, R. P. (2011). Denial-of-service attack detection based on multivariate correlation analysis. In B.-L. Lu, L. Zhang, & J. Kwok (Eds.), *Neural information processing* (pp. 756–765). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1145/2490428.2490450.

Taylor, H. (Ed.). (2009). Event-driven architecture: How SOA enables the real-time enterprise. Upper Saddle River, NJ: Addison-Wesley.

Terroso-Saenz, F., Gonzlez-Vidal, A., Ramallo-Gonzlez, A. P., & Skarmeta, A. F. (2019). An open IoT platform for the management and analysis of energy data. *Future Generation Computer Systems, 92*, 1066–1079. doi:10.1016/j.future.2017.08.046.

Vegh, L., & Miclea, L. (2016). Complex event processing for attack detection in a cyber-physical system. In *2016 ieee international conference on automation, quality and testing, robotics (AQTR)* (pp. 1–6). doi:10.1109/AQTR.2016.7501296.

Wireshark (2019). Wireshark. https://www.wireshark.org/ Accessed 17 March 2019.

Xiao, L., Wan, X., Lu, X., Zhang, Y., & Wu, D. (2018). IoT Security techniques based on machine learning: how do IoT devices use AI to enhance security? IEEE Signal Processing Magazine, *35*(5), 41–49. doi:10.1109/MSP.2018.2825478.

Zimmermann, H. (1980). OSI Reference model - The ISO model of architecture for open systems interconnection. IEEE Transactions on Communications, *28*(4), 425–432. doi:10.1109/TCOM.1980.1094702.

CHAPTER 4

Detecting security attacks in cyber-physical systems: a comparison of Mule and WSO2 intelligent IoT architectures

- **Title**: Detecting security attacks in cyber-physical systems: a comparison of Mule and WSO2 intelligent IoT architectures.
- **Authors**: José Roldán-Gómez, Juan Boubeta-Puig, Gabriela Pachacama-Castillo, Guadalupe Ortiz, Jose Luis Martínez
- **Type**: Journal paper.
- **Journal**: PeerJ Computer Science.
- **Publisher**: PeerJ.
- **ISSN**: 2376-5992
- **Status**: Published.
- **Publication date**: November 2021.
- **Paper Number**: 787.
- **DOI**: 10.7717/peerj-cs.787
- **JCR IF/ranking**: 2.411/Q2 (JCR2021).

PeerJ Computer Science

Detecting security attacks in cyber-physical systems: a comparison of Mule and WSO2 intelligent IoT architectures

José Roldán-Gómez[1], Juan Boubeta-Puig[2], Gabriela Pachacama-Castillo[3], Guadalupe Ortiz[2] and Jose Luis Martínez[1]

[1] Research Institute of Informatics (i3a), Universidad de Castilla La Mancha, Albacete, Spain
[2] Department of Computer Science and Engineering, University of Cadiz, Cadiz, Spain
[3] School of Engineering, University of Cadiz, Cadiz, Spain

ABSTRACT

The Internet of Things (IoT) paradigm keeps growing, and many different IoT devices, such as smartphones and smart appliances, are extensively used in smart industries and smart cities. The benefits of this paradigm are obvious, but these IoT environments have brought with them new challenges, such as detecting and combating cybersecurity attacks against cyber-physical systems. This paper addresses the real-time detection of security attacks in these IoT systems through the combined used of Machine Learning (ML) techniques and Complex Event Processing (CEP). In this regard, in the past we proposed an intelligent architecture that integrates ML with CEP, and which permits the definition of event patterns for the real-time detection of not only specific IoT security attacks, but also novel attacks that have not previously been defined. Our current concern, and the main objective of this paper, is to ensure that the architecture is not necessarily linked to specific vendor technologies and that it can be implemented with other vendor technologies while maintaining its correct functionality. We also set out to evaluate and compare the performance and benefits of alternative implementations. This is why the proposed architecture has been implemented by using technologies from different vendors: firstly, the Mule Enterprise Service Bus (ESB) together with the Esper CEP engine; and secondly, the WSO2 ESB with the Siddhi CEP engine. Both implementations have been tested in terms of performance and stress, and they are compared and discussed in this paper. The results obtained demonstrate that both implementations are suitable and effective, but also that there are notable differences between them: the Mule-based architecture is faster when the architecture makes use of two message broker topics and compares different types of events, while the WSO2-based one is faster when there is a single topic and one event type, and the system has a heavy workload.

Submitted 2 June 2021
Accepted 28 October 2021
Published 23 November 2021

Corresponding author
José Roldán-Gómez,
jose.roldan@uclm.es

Academic editor
Anand Nayyar

Additional Information and Declarations can be found on page 32

DOI **10.7717/peerj-cs.787**

Subjects Data Mining and Machine Learning, Embedded Computing, Security and Privacy
Keywords Internet of things, Complex event processing, Machine learning, Pattern detection, Security attack

INTRODUCTION

Over the past few years, expectations regarding the use of IoT devices have risen significantly. According to data published by the IoT Analytics company, since 2015 there has been a significant increase in the use of IoT devices, with 7,000 million of them being

How to cite this article Roldán-Gómez J, Boubeta-Puig J, Pachacama-Castillo G, Ortiz G, Martínez JL. 2021. Detecting security attacks in cyber-physical systems: a comparison of Mule and WSO2 intelligent IoT architectures. **PeerJ Comput. Sci. 7:e787 DOI 10.7717/peerj-cs.787**

registered in 2018, and this figure is estimated to reach 21,500 million in 2025 (*Lueth, 2018*). With this increase in the use of such devices, new security challenges also arise, such as ensuring the security of IoT devices (*Bertino et al., 2016*). Although there are quite a number of works in the literature addressing this problem, further research and implementation is still needed within the realm of the Internet of Things. An example of this is the attack in 2016 in which cybercriminals exploited the vulnerabilities of thousands of IoT devices to convert them into Domain Name System (DNS) request generators and carry out a Distributed Denial of Service (DDoS) attack, causing an Internet service disruption that affected several companies such as Amazon, PayPal, Netflix, Spotify and Twitter (*Moss, 2016*). It is also worth mentioning that the analysis published by the Gartner company indicated that in 2020 more than 25% of the attacks identified in companies would involve IoT devices (*Moore, 2018*). Several studies show the magnitude of the problem, revealing that, in just the first half of 2019, a hundred million attacks were carried out against smart devices, a figure seven times higher than the number detected in 2018. The Mirai malware was responsible for 39% of them (*Demeter, Preuss & Shmelev, 2019*). In 2020 and 2021 this problem has worsened; the most common threat remains Mirai, but new variants have also been created (*Gutnikov et al., 2021*; *Kaspersky, 2021*).

Considering all of the above, it is clear that there are currently significant security problems in IoT devices; if these problems are not addressed, it is certain that they will be even more damaging in the future. Therefore, it is imperative to examine new ways of identifying attacks on IoT devices in a timely and efficient manner, and to enable notification and alarm submission in critical attacks. In other words, it is essential to propose an Intrusion Detection System (IDS) for IoT devices. Such a system must be able to receive, analyze and process a large number of records in real time. Also, it must immediately notify security experts of attacks in progress in order to give them more reaction time to mitigate the attacks. However, the ability of traditional rule-based IDSs to detect security attacks in the IoT domain is limited, as they cannot detect novel attacks. Since current malware is not static, it is highly desirable to have the ability to detect previously-unknown attacks.

With the aim of addressing this challenge, in the past we proposed a software architecture that integrates Complex Event Processing (CEP) and Machine Learning (ML), and has the ability to detect, and provide notification of, security attacks on IoT devices in real time (*Roldán et al., 2020*). This architecture permits the detection of not only static but also dynamic security attacks in the IoT thanks to the use of both CEP technology (*Luckham, 2012*; *Boubeta-Puig, Ortiz & Medina-Bulo (2015)*) and ML techniques (*Buczak & Guven, 2016*).

Once we had proved the viability of building this architecture by combining CEP and ML, we observed a number of potential limitations that should be studied; in particular, we were concerned with the fact that the architecture is necessarily linked to the technologies of specific vendors, and also that other alternative implementations may not achieve the desired performance in this field of application. The architecture we proposed consists of the integration of ML with a CEP engine, and the ESB of two specific

vendors, namely Esper (*EsperTech, 2021*) and MuleSoft (*MuleSoft, 2021b*). We thus thought it might be advisable to be able to implement this architecture on other platforms; for example, on the well-known WSO2 suite (*WSO2, 2021c*).

This gave rise to our first research question (RQ1): can a real-time data stream processing architecture be implemented with the WSO2 ESB (*WSO2, 2021d*) together with the WSO2 Siddhi CEP engine (*WSO2, 2021b*) and be integrated with ML techniques? Assuming that it is feasible to implement the architecture with the CEP engine and the ESB of other vendors, in particular with those offered by WSO2, we are necessarily concerned about what impact this may have on the performance of the system, given that, as we have explained above, a real-time response is required to stop security attacks on the IoT.

This leads us to our second research question (RQ2): can a streaming data processing architecture based on the integration of ML techniques with the WSO2 CEP engine and ESB achieve or improve upon the performance of the previously proposed architecture (*Roldán et al., 2020*)? In addition, we consider the possibility that various implementations of the integration architecture of CEP, ESB and ML may present a more or less advantageous performance depending on the type of attack to be detected, that is, the type of pattern necessary for each attack. Likewise, there may be variations in how these systems support situations of stress.

This inevitably leads us to the third research question (RQ3): what kind of event patterns are processed faster with WSO2/Siddhi and which ones with Mule/Esper, and which of the two architectures is more suitable for supporting high-stress situations?

Once all this analysis has been carried out, we undoubtedly arrive at the question in which the domain experts are most interested (RQ4): which of these architecture implementations is the best to be deployed in an IoT security attack detection environment? To be able to answer these research questions requires the implementation of the architecture analogous to the one presented in *Roldán et al. (2020)* and replacing the technologies by the ones in the WSO2 suite. It also requires the implementation of a realistic security attack environment in an IoT network by carrying out various attacks against the TCP, UDP and MQTT protocols, as well as analyzing the response of the architectures in terms of performance and stress tests.

Therefore, the main aim of this paper is twofold: firstly, we aim to demonstrate that our intelligent architecture, which integrates CEP and ML in order to detect IoT security attacks in real time, can be implemented with different integration platforms such as Mule and WSO2, different CEP engines such as Esper and Siddhi and different ML algorithms such as linear regression (*Montgomery, Peck & Vining, 2021*). Secondly we aim to provide a comprehensive analysis of the performance and benefits of the architecture depending on the different vendor technologies used for its implementation; in particular, a comparison of the architecture implementation with Mule and Esper *versus* WSO2 and Siddhi is included. In this way, we provide a comparative analysis that can be very useful for the developer when choosing between one technology and another for the implementation of the architecture, depending on the requirements of the specific application domain and case study.

In addition to the research questions and the objectives to be achieved, in this work we rely on a series of assumptions that can be extracted from different works, These are:

- CEP works successfully in IoT environments. There are different works in which CEP architectures are successfully deployed in IoT environments (*Roldán et al., 2020*; *Corral-Plaza et al., 2020*).
- CEP engines and ESBs from different vendors can be integrated with our architecture to detect cybersecurity threats in real time: this architecture has already been deployed with Mule (*Roldán et al., 2020*) and there are works describing how to deploy WSO2 in an IoT environment (*Fremantle, 2015*).

The rest of the paper is organized as follows. The *Background* section describes the background to the paradigms and technologies used in this work. The *Related work* section describes the most relevant works in the literature, and the *Architecture for IoT security* section presents the architecture we propose for detecting attacks on IoT devices and how the implementation with the WSO2 suite differs from that of Esper CEP and Mule ESB. The *Comparing architecture performance and stress* section explains the comparison of the performance and stress tests conducted for these architectures, which have been implemented with Esper/Mule and WSO2. Then, the *Results* section presents the experiments and results obtained, the *Discussion* section discuss and answer the four research questions. Finally, the *Conclusions and future work* section contains our conclusions and some lines for future research.

BACKGROUND

This section describes the background to security in the IoT, ML, SOA 2.0 and CEP.

Security in the Internet of things

The IoT and cyber-physical system devices are increasingly present in our lives. The features offered by these devices are very attractive and they can be used for many different purposes, among which, we can highlight domotics, the automation and control of production processes, video surveillance and security, and medicine and health care. The various uses that have been given to these devices and the ability to access them *via* the Internet have attracted the interest of hackers. Unfortunately, the approach followed by developers in the design of security measures for IoT devices has not been as successful as their growth, and this is made evident by the number of cyber-attacks detected in the first half of 2019, which surpassed a hundred million, which is seven times higher than the previous year (*Demeter, Preuss & Shmelev, 2019*). The vector used by attackers in those attacks was mainly brute force, taking advantage of the weak default configuration of the devices and gaining access to them with the default credentials (*Demeter, Preuss & Shmelev, 2019*). These attacks took advantage of the vulnerabilities of the IoT devices to infect them with malicious code and then manipulate them to achieve their goal. The idea behind that malware focused on the creation of bots to be marketed for the carrying out of Denial of Service (DoS) attacks. One of the most widely-spread (and also the first)

pieces of malware specially designed for these devices was called Mirai, which is a botnet that inserts malicious code into IoT devices so that they initiate a DoS attack against a certain target. This caused shock and aroused the interest of hackers in these devices.

Another weakness of IoT and cyber-physical system devices is the use of unsafe network services and protocols, due mainly to these devices having several constraints, such as a small memory and a limited battery, which prevent developers from using a usual security setup. These vulnerabilities have been exploited to carry out several attacks that could have been prevented if the necessary measures had been taken. A lack of security in the storage and transfer of data that allows the observation and analysis of the information transmitted by these devices is another critical weakness in the security of IoT devices. In this regard, Message Queuing Telemetry Transport (MQTT) is a very common protocol in the IoT (*OASIS, 2019*). MQTT is a binary protocol that reduces the overhead compared with other application layer protocols. It is a publish/subscribe-based protocol in which a server (there can be more than one), known as the message broker, manages the flow of information, which is organized as a hierarchy of topics. Each client can be a subscriber and a publisher simultaneously. This protocol is similar to MQTT-SN and has several weaknesses, such as allowing the sending of many MQTT packets of a massive size, which overloads the broker. This attack causes a DoS in the MQTT network. Furthermore, an MQTT subscription fuzzing attack could gain information about the available topics because nodes are not authenticated and the information is not ciphered. Moreover, an MQTT disc-wave attack can exploit a failing in several implementations of the MQTT protocol. The specification of MQTT establishes that each client has a unique ID, so if a new client tries to register this ID again, the broker should reject it. However, many implementations allow a new client to connect with a registered ID, causing the existing client with that ID to be ejected from the previously-created connection.

Finally, a very common attack that can appear in an IoT-based network is scanning. Attackers can perform this procedure to discover devices and open ports in the network. By extending the scanning, attackers can cause a DoS in the network by sending large numbers of reconnaissance packets and congesting the network. The attack generates a large volume of traffic to try to saturate the network and so prevent users from accessing the system. The attack can also take advantage of flaws in the code of an application or part of the open-source code that uses the application. Two of the most common attacks of this type are TCP and UDP flood attacks (*Warburton, 2021*). When the connection is established through the TCP protocol, the client and the server exchange flags to initiate, close or restart the connection, or indicate that the request is urgent; the attacker sends several SYN flags asking to initiate a connection with the server, which is blocked when there are too many ACK requests waiting and the server runs out of resources to serve legitimate clients. A UDP port scan attack consists of sending a UDP packet to multiple ports on the same destination system, then analyzing the response and determining service and host availability. The attacker can determine whether the port is open, closed or filtered through a firewall or packet filter.

Machine learning

Machine Learning (ML) can be described as a set of techniques, technologies, algorithms and methodologies used to predict, cluster and classify entities, which can be events, objects, or anything else that can be described with attributes, also known as features, and entity behaviors. Broadly speaking, the best way to obtain these predictions is to model the behavior and attributes of these entities. There are many different algorithms to model these entities using functions which are plotted with these algorithms, and datasets of entities. The behavior, features and context of each entity are different. Therefore, the best algorithm does not exist, as each entity type has its correct algorithm or algorithms, if they even exist. For this reason, it is necessary to analyze these entities and their contexts, preprocess the datasets to allow them to be managed by these algorithms, and perform a feature selection (if it is necessary) to discover the most descriptive set of features. Sometimes, once the feature selection has been made, we can easily obtain the distribution of the entities, which is very useful for choosing the algorithm in a more precise way.

There are different types of machine learning techniques and algorithms, which can be classified as follows:

- Supervised learning. In this approach, the model is trained with labelled entities, *i.e.* the model knows the type of each entity in the training dataset. Also, it is possible to find regression techniques that aim to predict a numeric value.
- Unsupervised learning. This set of techniques does not require labelled entities, so the model learns how to group or classify them with similarity measures.
- Reinforcement learning. This kind of ML uses a prize/penalty approach. When our model performs a correct action, we can provide it with good feedback. When it fails, then it receives a penalty.

In this paper, we have used linear regression (*Montgomery, Peck & Vining, 2021*) because our dataset has a linear distribution. We would like to highlight that our approach can be adapted to other mathematical models, if needed.

Event-driven service-oriented architectures

Service-Oriented Architecture (SOA) is a paradigm for the design and implementation of loosely-coupled distributed system architectures whose implementation is fundamentally based on services. SOA services offer a well-defined interface in accordance with standards and facilitate communications between the service provider and the consumer in a decoupled way by using standard protocols. Thus, these architectures provide easy interoperability between third party systems in a flexible way, and therefore facilitate system maintenance and evolution when changes are required (*Papazoglou, 2012*).

ED-SOA, or SOA 2.0, has evolved from the traditional SOA. The distinguishing feature of SOA 2.0 is that it facilitates communication between users, applications and services through events, instead of using remote procedure calls (*Luckham, 2012*). With the growth of service components and processes, and the inclusion of events in event-driven

service-oriented applications, a new infrastructure is required to support the decoupled communications and to maintain applications flexibly. These requirements are fulfilled by an ESB, which permits interoperability among several communication protocols and heterogeneous data sources and targets (*Papazoglou, 2012*). In this way, an ESB provides and supports interoperability among diverse applications and components through standard interfaces and messaging protocols, also reinforcing the reliability of the communication as well as ensuring their scalability. There are several ESBs available, and in this paper we have selected two well-known ones for their evaluation, namely Mule and WSO2.

The Anypoint platform offers support for the design, implementation and management of APIs and integration (*MuleSoft, 2021a*). It includes Mule (*MuleSoft, 2021b*), an integration and ESB platform that provides assistance to developers in interconnecting applications, and provides support for various transport protocols, as well as for the transformation of different data formats. It delivers message routing as well as IoT and cloud integration. In addition, it provides a graphical interface for the development of business-to-business integration applications.

WSO2 is an open-source decentralized approach which provides support for building decoupled digital products that are ready to market, with a main focus on APIs and microservices, and a wide range of complementary products and solutions (*WSO2, 2021c*). WSO2 offers WSO2 Enterprise Integrator, an integration platform which consists of a centralized integration ESB with capabilities for data, process and business-to-business integration. WSO2 ESB (*WSO2, 2021d*) provides support for multiple transport protocols, data formats and flow integration, as well as IoT and cloud service integration.
The product also includes an analysis system for comprehensive monitoring.

As we can see, both ESBs provide similar features and can be used in conjunction with their integration platform with many plugins and solutions for further functionalities, such as stream and event processing.

Complex event processing

Despite all the advantages of SOA 2.0 mentioned in the previous subsection, this type of architecture requires the use of an additional technology that makes it possible to analyze and correlate the vast amounts of data that are present in the field of the IoT in real time. CEP (*Luckham, 2012*) fulfills this functionality appropriately as it is a technology that allows the analysis and correlation of heterogeneous data streams in real time in order to detect situations of interest in the domain in question. In particular, the software that is capable of analyzing the data in real time is known as the CEP engine. In order to detect situations of interest, a series of event patterns are defined in the CEP engine (*Valero et al., 2021*). These patterns represent the conditions that allow us to detect that such a situation has occurred. These rules are applied to the engine's incoming data, which are known as simple events, while the situations of interest detected by the pattern are named complex events. Thus, with CEP we can improve and speed up the decision-making process (*Boubeta-Puig, Ortiz & Medina-Bulo, 2015*; *Benito-Parejo, Merayo & Núñez (2020)*; *Corral-Plaza et al., 2021*).

There are several CEP engines available, and in this paper we have selected two well-known ones to be evaluated, namely Esper and WSO2 CEP.

Esper (*EsperTech, 2021*) is an open-source Java-based software engine for CEP, which can quickly process and analyze large volumes of incoming IoT data. Esper comes with the Esper Event Processing Language (EPL), which extends the SQL standard and permits the precise definition of the complex event patterns to be detected. The Esper compiler compiles EPL into byte code in a JAR file for its deployment, and at runtime this byte code is loaded and executed. Esper performs real-time streaming data processing, using parallelization and multithreading when necessary, and it is highly scalable. In addition, it provides the option of implementing distributed stream processing over several machines as well as horizontal scalability, should it be necessary. According to its documentation, Esper 8.1.0 can process around 7.1 million events per second (*EsperTech, 2019*).

WSO2 CEP is provided within the WSO2 Stream Processor. WSO2 CEP is an open-source CEP engine that facilitates the detection and correlation of events in real time, as well as the notification of alerts, counting in addition with the support of enriched dashboard tools for monitoring. It can be deployed in standalone or distributed modes, and is highly scalable. It uses a streaming processing engine with memory optimization, being able to find patterns of events in real time in milliseconds. According to its specification, a single WSO2 CEP node can handle more than 100 K events per second on a regular 4-core machine with 4 GB of RAM and several million events within the JVM (*WSO2, 2021c*). The cornerstone of the WSO2 CEP is Siddhi (*WSO2, 2021b*). It uses a language similar to SQL that allows complex queries involving time windows, as well as pattern and sequence detection. In addition, CEP queries can be changed at runtime through the use of templates.

ESB has been used in our system as a tool for transport and information management. This use is quite simple to implement but if the parameters are not specified properly, it could cause problems.

RELATED WORK

There is an interesting comparison between Mule, WSO2 and Talend conducted by *Górski & Pietrasik (2017)*. Note that Talend is beyond the scope of this work. The authors implemented seven different use cases and tested them with 5, 20 and 50 users simultaneously. Moreover, their work provides measurements of throughput, standard deviation and CPU usage for each experiment, and their results are closely aligned with ours, *i.e.*, WSO2 is always faster except when the output message is enormous (221,000 bytes of output message in this case). This is not a problem for our proposal because an IDS does not need big output messages. Moreover, WSO2 obtains a better throughput, whereas the CPU usage is similar in both cases. On the other hand, Mule provides a lower standard deviation, *i.e.*, Mule is more constant than WSO2 when processing different types of events.

Bamhdi's work (*Bamhdi, 2021*) is also interesting. In contrast to our work, his paper does not show an active performance comparison between WSO2 and Mule, but instead

provides a feature comparison between four ESB platforms (WSO2 and Mule are included among them). Although Bamhdi's work is focused on comparing open source platforms against proprietary ones, it allows us to compare specific features of Mule and WSO2. This comparison, which analyzes 15 features, shows that WSO2 supports the 15 listed capabilities, whereas Mule supports. The only feature which Mule cannot provide is web migration from 5.0 to 6.0; note that WSO2 is the only one that satisfies this feature.

Dayarathna & Perera (2018) compare WSO2 with other ESBs, but Mule is not considered in their work, which provides a brief feature comparison between the Esper (basic version) and Siddhi CEP engines. According to the authors, each language provided by a CEP engine has its pros and cons. On the one hand, Esper (basic version) provides nested queries and debugging support, while Siddhi registered a higher performance than Esper: 8.55 million events/second *versus* 500,000 events/second.

Another work which is focused on CEP engines is that of *Giatrakos et al. (2020)*. It does not directly compare WSO2 against Mule or Siddhi against Esper, but instead describes different CEP paradigms. In particular, it explains different selection policies, consumption policies and windows. Moreover, the paper describes the scalability and parallelization of several CEP engines. Although it is quite different from our work, it can be useful in order to understand our work and learn about other CEP engines.

Freire, Frantz & Roos-Frantz (2019) adopt a different approach in which they do not conduct performance experiments directly, but experts enumerate the features of different ESBs. These features are grouped into three dimensions: message processing, hotspot detection and fairness execution. Additionally, Freire et al.'s work defines two types of features: subjective and objective. The authors assign values for each feature, which allows them to obtain a score for each ESB. According to their paper, Mule should be faster than WSO2, but the problem is that this is not demonstrated through experiments. This approach is useful because it allows the measuring of different ESB platforms without implementing experiments; however, it would have been more useful if they had carried out experiments to support their results.

Our paper provides a real performance comparison between Mule and WSO2, following a similar methodology to the one proposed in the papers mentioned, *i.e.*, executing and deploying the proposed platforms under equal conditions and measuring events in relation to time. More specifically, we have analyzed different pattern types, namely time-window-based patterns and prediction patterns. The latter are a novelty with respect to other works, as each network event is compared with a prediction event, and this acts as an anomaly detector.

ARCHITECTURE FOR IOT SECURITY

This section describes our proposed SOA 2.0 architecture, which integrates CEP and ML paradigms in order to detect attacks on IoT devices. Then, two implementations of this architecture, one using Mule and the other WSO2, are presented with the aim of comparing them under the same conditions in order to find the strengths and benefits of each, which is the novelty and contribution of this research.

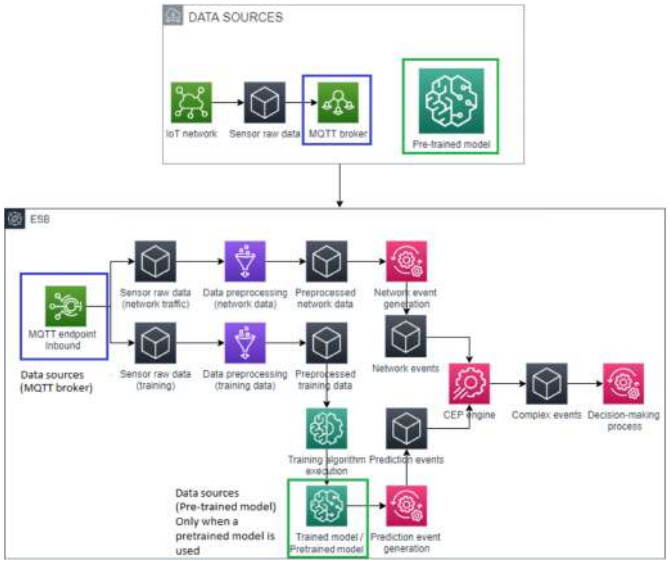


Figure 1 **Generic architecture to detect attacks on IoT devices.**
Full-size DOI: 10.7717/peerj-cs.787/fig-1

Architecture proposal

Our proposed architecture for detecting attacks on IoT devices is described below. This architecture, which is an improved version of the architecture we presented in *Roldán et al. (2020)*, is composed of three different parts.

The first module of the architecture, the data sources, consists of the data obtained from the network and the pre-trained model, if available. Otherwise, the model would have to be trained. As shown in Fig. 1, this module may be detached from the rest of the architecture, because it can be replaced by any computer network with an MQTT broker as collector. However, we consider it useful to analyze the whole system to understand its behavior. Note that in this new version of our architecture, an MQTT broker can be used with different topic numbers, with the aim of managing data grouped by type. Additionally, this new version permits the use of pre-trained models as data sources, which allows us to migrate our model from our architecture to other deployments. In addition, pre-trained models provide greater flexibility because they allow training the model outside (or inside) our deployed architecture.

The second module of the architecture, which is in fact the main module, receives raw network data and, optionally, a pre-trained model. This module is responsible for making

decisions on the basis of the network data analyzed in real time. This new version of our architecture is more flexible since different CEP engines can be used according to the user's needs.

At this point, the pipeline of the second module should be explained in detail. Through an MQTT inbound endpoint, the raw network data produced by data sources can reach the ESB. These data are preprocessed to make them consumable for the network event generator. The event generator provides network events which can be received and processed in real time by the CEP engine. Moreover, our architecture needs a trained model to predict the network event values. In particular, this model can be used to predict the type of network packet *via* a predicted value and a threshold, which is computed using the training data. In this case, our model has been built using a linear regression, and is used to predict values and a threshold from a key feature, or features, which is the packet length in our case. These features will vary with each case.

The last module is composed of data sinks which receive the notifications about the decision-making process conducted by the second module. Databases, event systems, emails, logs, or any other system required by end users to receive such notifications are examples of data sinks. Due to its simplicity, an explanatory diagram is not included.

We would like to point out that our architecture allows us to fit the model with raw sensor data; this traffic should be isolated and without any security attacks. There are two ways to obtain prediction patterns: the first is to set a pre-trained model, while the second is to train the model with the isolated network traffic. Regardless of the method which is selected, the architecture uses this model to predict the expected value of each incoming network packet. This prediction is used to create a prediction event which is compared with its corresponding network event. In this way, our architecture is able to obtain patterns which can detect anomalous packets by using the real value, the predicted value and a calculated threshold, since the absolute value of the subtraction of the real value and the predicted value must be smaller than the threshold; otherwise, the packet is anomalous.

Equation (1) describes our predictor in a formal way, where the number 1 means that the network packet belongs to the category used to train the model and obtain the ERROR.

$$f(x) = \begin{cases} 1 & if \quad (abs(real\ Value - predicted\ Value) \leq ERROR) \\ 0 & if \quad (abs(real\ Value - predicted\ Value) > ERROR) \end{cases} \tag{1}$$

It is important to note that we can fit the model with more attacks; for example, if we have traffic from a DoS attack, we can refit our model to detect this attack. The best way to generate patterns is to attack the architecture or obtain traffic from attacks. As mentioned above, we have improved the architecture to accept pre-trained models.

An initial deployment of the architecture could be composed of a few patterns that can be proposed and designed by the domain expert, and the anomaly detector, which uses legitimate traffic. When an anomalous pattern is triggered, anomalous packets can be used

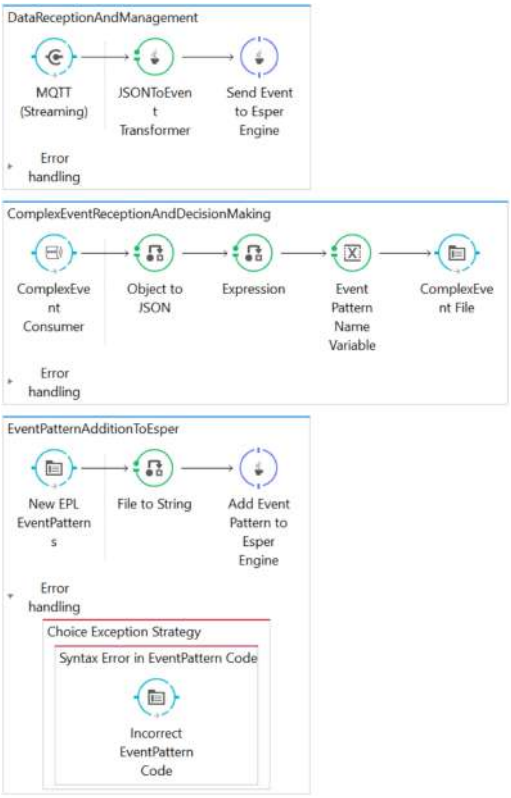


Figure 2 **Screenshot of the implemented Mule-based architecture.**
Full-size DOI: 10.7717/peerj-cs.787/fig-2

to generate a new pattern to detect this kind of anomaly again. This means that our architecture can improve and gradually become more accurate over time.

Architecture implementation with Mule

In this subsection we explain how our architecture for IoT security has been implemented by using the Mule ESB together with the Esper CEP engine.

The Mule-based architecture is composed of three data flows: *DataReception AndManagement*, *ComplexEventReceptionAndDecisionMaking* and *EventPattern AdditionToEsper* (see Fig. 2).

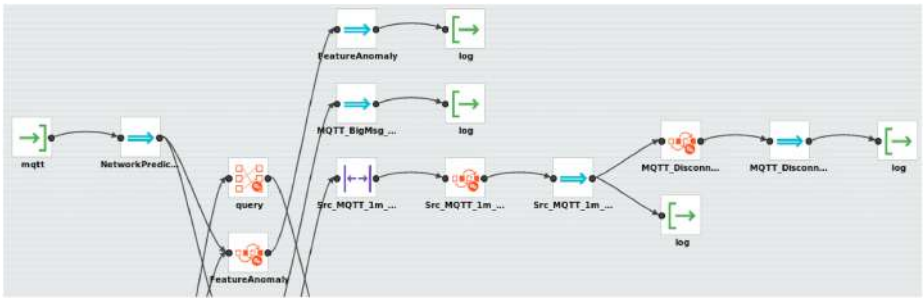


Figure 3 **Screenshot of the implemented WSO2-based architecture.** Full-size DOI: 10.7717/peerj-cs.787/fig-3

The *DataReceptionAndManagement* flow is responsible for receiving data from IoT data sources, transforming them into an event format and then sending them to the Esper CEP engine. Specifically, this flow is implemented with an MQTT inbound endpoint in which a topic is defined to receive the data obtained from data sources. Then, a Java transformer allows the transformation of the received JSON data into Java Map events, which are sent to an Esper CEP engine through a customized message component.

The *ComplexEventReceptionAndDecisionMaking* flow receives the complex events that are automatically generated by the CEP engine upon detection of previously deployed patterns, and transforms these complex events into JSON format. These are then saved in log files, which are a type of data sink for the architecture.

Finally, the *EventPatternAdditionToEsper* flow allows the runtime deployment of new event patterns in the CEP engine. To this end, a file input endpoint frequently checks whether there is a new file with an EPL extension, and if there is the event pattern code contained in this file is transformed into a string, which is then deployed in the Esper CEP engine.

Architecture implementation with WSO2

Figure 3 depicts our architecture for IoT security that is implemented with the WSO2 ESB. Unlike the implementation of the Mule-based architecture, which was integrated with the external Esper CEP engine, the implementation of the WSO2-based architecture does not require integration with an external CEP engine since WSO2 provides the Siddhi CEP engine by default.

As shown in Fig. 3, the architecture receives the data obtained from data sources (*NetworkPacket* and *NetworkPrediction*) by using an MQTT broker with two topics. Then, these data are matched through the different event patterns (queries) implemented with SiddhiQL and previously deployed in the Siddhi CEP engine. When a complex event is automatically created upon a pattern detection, it is saved in a log file, which is a data sink for the architecture.

COMPARING ARCHITECTURE PERFORMANCE AND STRESS

This section presents our comparison of the performance and stress tests conducted for the two architectures implemented with Mule and WSO2.

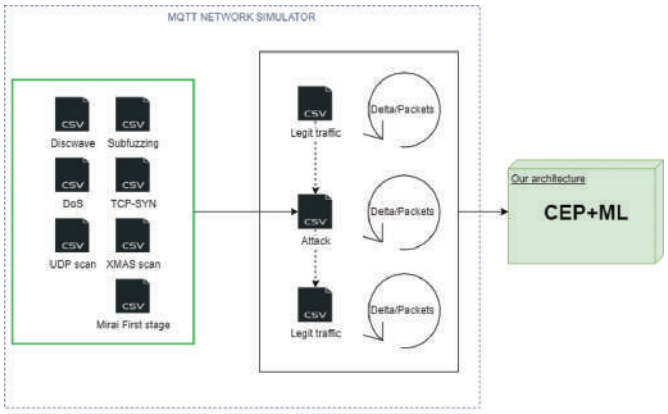
Proposed approach

Before analyzing each architecture component in depth, a schematic overview of the steps followed to address this comparison are explained below:

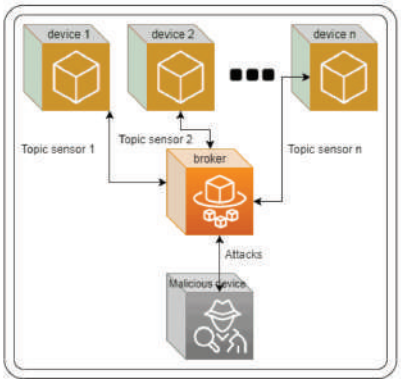
- First, a virtualized MQTT network, in which clients publish periodically, is deployed.
- Then, packets are collected from that network to define a *normal* scenario, in which the system is not under attack.
- Afterwards, a malicious client is introduced into the network and this client launches the attacks. Packets that generate attacks are collected to perform the experiments.
- A number of these packets are preprocessed and used to train the linear regression model. The mean square error for each category to be predicted with the regressor is also extracted.
- The values of the packets that were not used for training are predicted and saved. They will be used to perform the experiments.
- Then, event patterns are defined. To perform a complete comparison, we create a pattern per attack that will be able to detect the attack, as a domain expert would do, except for *DoS*, which is detected with a regressor because in practice it is difficult to establish a specific pattern for this type of attack. In addition, we create the *FeatureAnomaly* pattern, which is able to detect anomalies using the linear regressor. This pattern is used to detect unknown attacks, such as *Subfuzzing*, *DoS* or *Discwave*. And then there is the *ProtocolAnomaly* pattern, which detects any unknown protocol that should not be present in the network.
- Both platforms, Mule and WSO2, are deployed with their corresponding patterns.
- The simulator is used to perform the experiments (see next subsection) in such a way that these experiments are reproducible.
- Finally, the metrics of the experiments are extracted for comparison.

Simulator

To ensure the reproducibility of the experiments, we implemented an MQTT network simulator. We chose an MQTT simulator because, as mentioned above, it is a widely-used protocol in the IoT paradigm. Moreover, MQTT networks, by the nature of the protocol, are usually centralized because the broker acts as a centralizer, so that all MQTT packets pass through it. This makes it very easy to set up a network-based IDS in the broker, because there is no need to redirect traffic to another device. This simulator is capable of sending network packets to an MQTT broker, taking as data source different CSV files which contain real network traffic that was previously generated and stored. This is essential because it allows us to use real traffic and to combine the reproducibility of



(a) MQTT network simulator



(b) MQTT network diagram

Figure 4 MQTT network (A) and MQTT simulator (B).
Full-size DOI: 10.7717/peerj-cs.787/fig-4

the experiments with data that have been generated in a real MQTT network.
The main advantages of our simulator are that it can reliably send such network packets while taking the delay between packets into account, and it allows us to generate several scenarios to test both the proposed architectures.

Figure 4A outlines this MQTT network simulator. Note that when we wish to generate heavy workloads, we can use the sum of deltas from the packets or the number of packets as the threshold which is used to stop the generation of packets.

The behavior of the simulator is quite simple. First, it reads the CSV files, which first allows us to avoid the delay that is due to reading each row of the CSV while we are sending them.

When the simulator has read both CSV files (legitimate traffic, and the specific attack), it starts to send packets with MQTT, these being sent using JSON format. The number of packets is defined as described above.

Figure 4B shows an MQTT network diagram where there are a certain number of legitimate devices, 4 in our case, and 1 malicious device which attacks the network in different ways. This network is similar to the network used to obtain the network traffic.

Event patterns

In our previous work (*Roldán et al., 2020*), we defined and implemented twelve event patterns in Esper EPL for detecting the following security attacks:

- *TCP/SYN port scan*: the malicious device sends a round of 10 or more TCP packets with the SYN flag to three or more different ports of the broker in 1 s. If the port is open, the broker sends a SYN/ACK packet, otherwise it sends an RST packet. The *TCP_SYN* pattern implements this attack by making use of an intermediate pattern called *SrcDst_TCP_1s_Batch*.
- *UDP port scan*: the malicious device sends a round of 10 or more empty UDP packets to three or more different ports of the broker in 5 s. If the broker sends any response, then the port is open, but if the broker does not send a response, the port could be open. If the broker sends ICMP unreachable, the port should be closed. And if it sends a different error (not unreachable), the port should be filtered. The *UDP_Port_Scan* pattern implements this attack by making use of an intermediate pattern called *SrcDst_UDP_5s_Batch*.
- *Xmas port scan*: the malicious device sends a round of 10 or more TCP packets with PSH, FIN and URG flags to 2 or more different ports of the broker in 1 s. If the broker does not respond, the port should be open or filtered. If the broker sends an RST packet, it should be closed. If the broker sends an ICMP unreachable error, it should be filtered. The *Xmas_Scan* pattern implements this attack by making use of an intermediate pattern called *SrcDst_Xmas_1s_Batch*.
- *TELNET Mirai*: the malicious device simulates the first stage of Mirai, sending connect packets with different tuples (username/password). The pattern can be detected if the attacker sends more than 5 TELNET packets in 1 min. The *TELNET_Mirai* pattern implements this attack by making use of an intermediate pattern called *Src_TELNET_1m_Batch*.
- *MQTT disconnect wave*: the malicious device sends many MQTT packets with the *connect command*. As sending more than 1 *connect command* is strange, the pattern can be detected if the broker receives more than 5 MQTT *connect commands* in 1 min from a single IP address. The *MQTT_Disconnect_Wave* pattern implements this attack by making use of an intermediate pattern called *Src_MQTT_1m_Batch*.

- *MQTT subscription fuzzing*: the malicious device tries to subscribe to all topics, so the pattern can be detected if an MQTT client subscribes to more than 20 topics in 5 min. The *MQTT_Subscription_Fuzzing* pattern implements this attack by making use of an intermediate pattern called *Src_MQTT_5m_Batch*.

In the present work, we have used these twelve event patterns implemented in Esper EPL to test the Mule-based architecture. Moreover, we have implemented analogous patterns but in SiddhiQL to test the WSO2-based architecture.

Additionally, in this work we have split the *Anomaly* pattern, proposed in *Roldán et al. (2020)*, into 2 new patterns: *ProtocolAnomaly* and *FeatureAnomaly*. The first pattern allows us to detect protocols which are not expected because this may suggest that the system is under attack. The second pattern allows us to detect anomalous packets in expected protocols. Thus this pattern split allows us to classify anomalies more accurately.

Listing 1 shows the *FeatureAnomaly* pattern implemented in Esper EPL, while Listing 2 contains the implementation of the same pattern but using the SiddhiQL language. This pattern implements Eq. (1) and allows us to detect unmodeled attacks, such as the *DoS with big messages*. Moreover, it will detect other attacks, such as disconnect wave or subscription fuzzing, even if we do not define specific patterns to detect them. The *ProtocolAnomaly* pattern implemented in Esper EPL is shown in Listing 3, while Listing 4 contains the same pattern using the SiddhiQL language.

We have implemented two types of event patterns to detect such attacks. The first type uses a time batch window (*SrcDst_TCP_1s_Batch*, *SrcDst_UDP_5s_Batch*, *SrcDst_Xmas_1s_Batch*, *Src_TELNET_1m_Batch*, *Src_MQTT_1m_Batch* and Src_MQTT_5m_Batch) to trigger a complex event when a condition is met. The second type of pattern allows the comparison of messages coming from two broker topics, one that manages prediction and threshold data while the other topic manages real packet information. In this case, the pattern is activated when the difference between the prediction and the real values is higher than a certain threshold; this is useful because we can compare the performance for different attacks but also with different types of patterns.

Machine learning model

Selecting a machine learning model is a very important step in effectively deploying the architecture. Although this paper does not focus on ML processes, it is important to give a brief explanation of the model we have used.

The first step in defining our ML model was to select the most important features. For this purpose we applied the criteria proposed by KDD99, which are adaptable to our MQTT dataset. In addition, we also added features obtained from MQTT.

Once the features have been selected, they are normalized and binarized when necessary. Then we used Extremely Randomized Trees with our dataset to arrange the features by importance. After that, we selected the most important features (*Geurts, Ernst & Wehenkel, 2006*).

Table 1 show the importance of the binarized features. One or several features are chosen to be the key feature/s, and these are predicted with the rest of the features

Listing 1 ***FeatureAnomaly*** **pattern implemented in Esper EPL.**

```
@Name("FeatureAnomaly")
@Tag(name="domainName", value = "IoTSecurityAttacks")
insert into FeatureAnomaly select a2 . id as id,
current time stamp( ) as time stamp, a1 . destIp as destIp
from pattern[((every a1 = NetworkPacket((a1 . protocol = 'MQTT' or
a1 . protocol = 'TCP')))
-> a2 = NetworkPrediction((a2 . i d = a1 . id and
(a2 . packetLengthPredict < (a1 . packetLength
- a2 . packetLengthPredictSquaredError) or a2 . packetLengthPredict >
(a1 . packetLength
+ a2 . packetLengthPredictSquaredError)))))]
```

Listing 2 ***FeatureAnomaly*** **pattern implemented in SiddhiQL.**

```
@info(name= "FeatureAnomaly")
from ((every a1 = NetworkPacket[(a1 . protocol == 'MQTT'
or a1 . protocol == 'TCP')]) -> a2
= NetworkPrediction[(a2 . i d == a1 . id and
(a2 . packetLengthPredict < ( a1 . packetLength
- a2 . packetLengthPredictSquaredError) or
a2 . packetLengthPredict > (a1 . packetLength
+ a2 . packetLengthPredictSquaredError))])
select a2 . id a s id,
time : timestampInMilliseconds( ) as time stamp,
a1 . destIp as destIp
insert into FeatureAnomaly;
```

obtained. Furthermore, this prediction will be compared with the real value for each event, with the error threshold being defined using the mean square error obtained when we train the model.

By using pre-processed features, we can select the model. In this case, our data features fit a linear distribution very well. Therefore, we chose a linear regression to predict these key features. This model can change depending on the whole IoT network.

Tests

By implementing a network simulator, we were able to measure the performance of our proposed architecture implemented with WSO2 and Mule, and compare them. We designed 14 experiments with seven different attacks against MQTT, and each test was composed of legitimate traffic and one specific attack. Specifically, we carried out seven

Listing 3 ***ProtocolAnomaly*** **pattern implemented in Esper EPL.**

```
@Name("ProtocolAnomaly")
@Tag(name="domainName", value= "IoTSecurityAttacks")
insert into ProtocolAnomaly
select a1.id as id,
current_timestamp() as timestamp,
a1.destIp as destIp
from pattern[(every a1 = NetworkPacket((a1.protocol != 'TCP' and
a1.protocol != 'UDP'
and a1.protocol != 'MQTT' and
a1.protocol != 'ARP' and a1.protocol != 'DHCP'
and a1.protocol != 'MDNS' and
a1 . protocol != 'NTP' and a1.protocol != 'ICMP'
and a1.protocol != 'ICMPv6' and
a1.protocol != 'DNS' and a1.protocol != 'IGMPv3')))]
```

Listing 4 ***ProtocolAnomaly*** **pattern implemented in SiddhiQL.**

```
@info(name="ProtocolAnomaly")
from (every a1 = NetworkPacket[(a1.protocol != 'TCP' and
a1.protocol != 'UDP'
and a1.protocol != 'MQTT' and
a1.protocol != 'ARP' and a1.protocol != 'DHCP'
and a1.protocol != 'MDNS' and
a1.protocol != 'NTP' and a1.protocol != 'ICMP'
and a1.protocol != 'ICMPv6' and
a1.protocol != 'DNS' and a1.protocol != 'IGMPv3')])
select a1.id as id,
time:timestampInMilliseconds() as timestamp,
a1.destIp as destIp
insert into ProtocolAnomaly;
```

experiments (one per attack) which used the delay of each packet in order to simulate a network realistically, and seven experiments without a delay, which allowed us to measure the performance with heavy workloads. Thus, the proposed tests were as follows:

- TCP-SYN scan (with delay/without delay)
- UDP port scan (with delay/without delay)
- XMAS port scan (with delay/without delay)

- Mirai first stage (with delay/without delay)
- MQTT disconnect wave (with delay/without delay)
- MQTT subscription fuzzing (with delay/without delay)

RESULTS

This section presents and discusses the experiments and the results obtained when comparing the performance of our architecture implemented with WSO2 and Mule, as well as the limitations of each implementation.

These experiments were carried out under similar conditions for the WSO2-based architecture, composed of the WSO2 ESB and the WSO2 CEP engine, and the Mule-based architecture that integrates Mule ESB with the Esper CEP engine. We would like to point out that WSO2 provides some extra performance features such as multiworkers and PMML models(*WSO2, 2020*, *2021a*), which could enhance the architecture's performance. However, we did not integrate these features in our proposed architecture in order to create similar conditions for both systems.

The results obtained for the two types of tests conducted in this work (performance and stress tests) are discussed below. The implementation code can be accessed in the *Roldán-Gómez et al. (2021)* repository.

Performance tests

The results for the performance tests are presented in the following subsections.

Estimated computational complexity

Although it is difficult to give an exact figure for computational complexity because of the internal operations performed by the CEP engines, we estimate the computational complexity on the basis of the steps that we can calculate. Note that this estimation assumes that the model and preprocessing steps are as mentioned. Obviously, this will change if another model or steps are used during the preprocessing step.

To define the computational complexity, we consider the following variables: n, which defines the number of packets, which F is the number of variables where each step is applied (this value will be constant for each step); and v, which defines the different values of each category and is used only for the binarization of categorical attributes. First, we calculate the computational complexity of each step, then the total for the training stage, and then the total at runtime.

The estimated computational complexities are as follows: min-max scaler $O(2nF_1)$, fill empty values $O(nF_2)$, binarization of categorical attributes $O(2nF_3v)$, and training linear regression model $O(nF_4^2 + F_4^3)$. All these steps only have to be carried out once. In addition: predict a value with the regressor and create n events $O(F_5n)$. In summary, the estimated computational complexity in training is as follows:

$$O(2nF_1 + nF_2 + 2nF_3v + nF_4^2 + F_4^3).$$

Table 1 Feature importance.

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

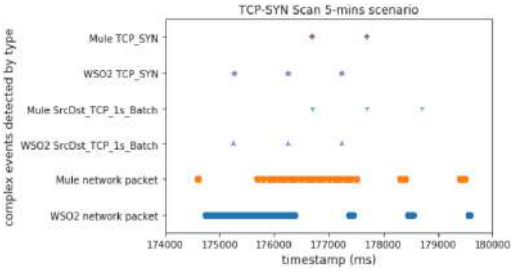


Figure 5 TCP_SYN attack comparison. Full-size DOI: 10.7717/peerj-cs.787/fig-5

And the estimated computational complexity at runtime is:

$O(F_5 n)$.

Since we do not know the exact inner workings of CEP engines, it is difficult to calculate the remaining steps. That is why performance experiments, such as those carried out in this paper, are so important.

TCP SYN scan

The first experiment performed was composed of legitimate traffic (a simple MQTT network) and a TCP SYN scan. We used our architecture as an IDS to detect attacks or scans.

Figure 5 shows the results obtained for the TCP-SYN scan test executed for 5 min on both the WSO2-based architecture and the Mule-based one. The X-axis represents the execution time in milliseconds, while the Y-axis shows the different complex event types detected in real time during the execution of the test.

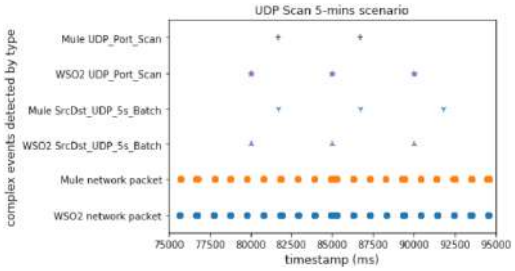


Figure 6 UDP scan attack comparison. Full-size DOI: 10.7717/peerj-cs.787/fig-6

As we can see, the WSO2 implementation triggers the TCP_SYN complex event first. Therefore, we can conclude that the WSO2 achieves an earlier detection than the Mule-based one. In this case, TCP_SYN starts sooner in the WSO2 scenario, but this delay is shorter than the detection time difference.

UDP port scan

The UDP scan is slower than the TCP one, and it is useful to study the performance in a different way. This experiment allowed us to compare the performance when the attack has a low packet sending ratio. As in the case of the TCP SYN Scan experiment, there was normal traffic and a UDP port scan.

Figure 6 shows the results obtained for this UDP port scan experiment. In conclusion, we can say that WSO2 was faster than Mule again. Mule generated a null window, not being able to detect the third *UDP_Port_Scan* complex event. It is important to note that the difference is smaller than in Fig. 5; this may be because the attacks, in both cases, started at the same time.

Xmas port scan

This scan is not very common and shows how our architecture is able to detect more unusual attacks. From the point of view of the experiment, it should be like the TCP_SYN scan, as both have similar packet sending ratios and event generation characteristics.

Figure 7 shows that WSO2 was faster than Mule, even though the Xmas port scan attack started sooner in the Mule scenario. This experiment is useful because it allowed us to confirm the superiority of WSO2 when there is not a comparison between different events.

Mirai first stage

This scenario simulates the first stage of Mirai. This attack tries to connect with Telnet using a username/password list. The main aim of this experiment was to check the behavior of our system under common IoT attacks. Figure 8 shows a comparison of the results for the Mirai scenario, executing it on the WSO2-based architecture and the ESB-based one. Again, WSO2 detected the first complex event faster than Mule.

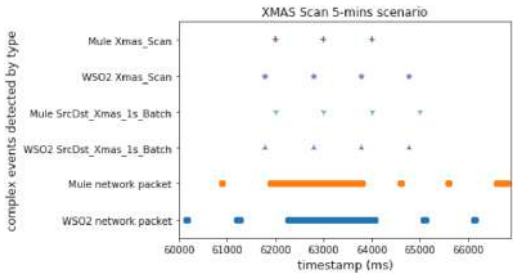


Figure 7 **XMAS scan attack comparison.** Full-size DOI: 10.7717/peerj-cs.787/fig-7

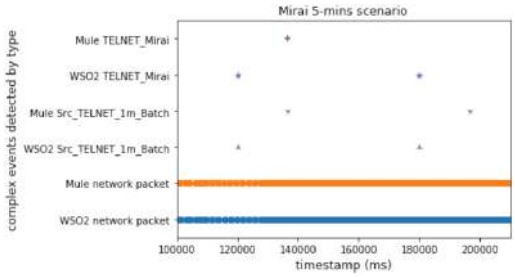


Figure 8 **Telnet-Mirai attack comparison.** Full-size DOI: 10.7717/peerj-cs.787/fig-8

DoS big message

This scenario simulates a common DoS attack in which the attacker sends big messages quickly to the broker.

This experiment is different to the other ones because time windows are not used. Instead of time windows, each packet is matched with its prediction. As we mentioned above, there are two different ways to detect attacks using our predictor. In this case, the system trains the model with legitimate and isolated traffic, allowing us to detect anomalous packets. Note that each packet which does not match with its prediction, and whose difference exceeds the threshold, can be classified as anomalous. Additionally, we could have fitted a model to detect each specific attack.

Figure 9 illustrates that, in this case, Mule was faster than WSO2, since WSO2 needed more time to detect all the malicious packets. Therefore, we can conclude that Mule offers better performance when we need to compare different events (network events and prediction events in this case).

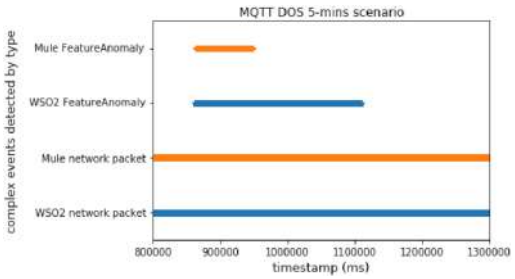


Figure 9 **DoS big message attack comparison.** Full-size DOI: 10.7717/peerj-cs.787/fig-9

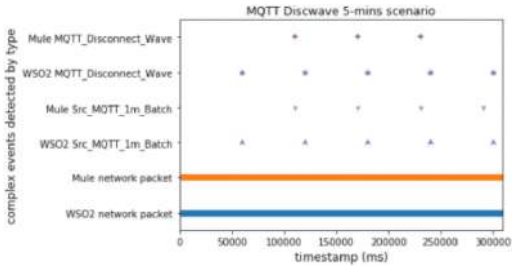


Figure 10 **Discwave attack comparison (using time windows).** Full-size DOI: 10.7717/peerj-cs.787/fig-10

MQTT disconnect wave

This scenario provides useful knowledge about both platforms. Here there are time windows and an anomalous packet detector, which works by matching each packet with its prediction, as we did in the DoS experiment. The advantage of this experiment is that it allowed us to check the behavior of the whole proposal deployed with the predictor working. Note that in a real scenario we would not use both methods (time windows and prediction), but it was useful and appropriate to test the performance.

As we can conclude from Figs. 10 and 11, Mule again worked better with predictions (by using two topics) than WSO2. On the other hand, WSO2 again detected the first complex event earlier than Mule when using time windows.

Subscription fuzzing

In this scenario, we used both methods again (time windows and predictions), but this attack is slower than the discwave one, which meant that the delay between packets was longer than in the discwave attack. This experiment shows the behavior of our

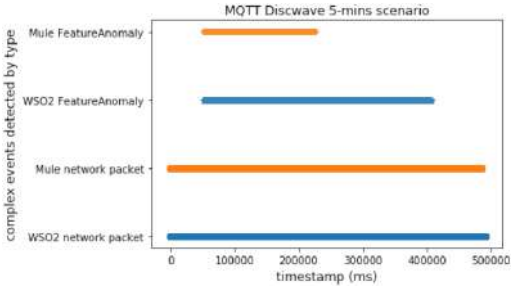


Figure 11 **Discwave attack comparison (using FeatureAnomaly pattern).**
Full-size DOI: 10.7717/peerj-cs.787/fig-11

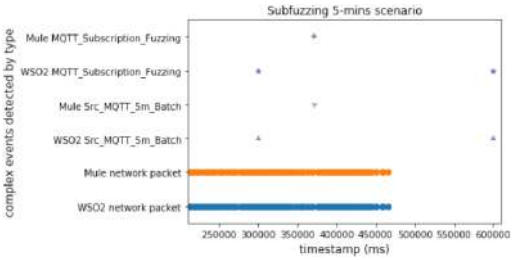


Figure 12 **Subfuzzing attack comparison (using time windows).**
Full-size DOI: 10.7717/peerj-cs.787/fig-12

proposal when the system receives an attack with a lower packet sending ratio than DoS or discwave.

Figures 12 and 13 show that there are two interesting facts that we can extract from this experiment. The first is that WSO2 detected the second complex event very late when using time windows. As it uses a 5-min window, the second time window was closed after the attack finished. But the important thing is that, in this case, WSO2 and Mule presented a similar performance with predictions. This is due to the long delay between packets in this experiment. WSO2 again registered the first detection sooner. It seems that Mule was processing a heavy workload when we compared two different events, but WSO2 provided a better brute performance when the system compared features/properties in the same event.

Stress test

Additionally, we carried out 7 more experiments in which the network packets had no delay. Although this is not a realistic case, it is very useful because it allows us to study the

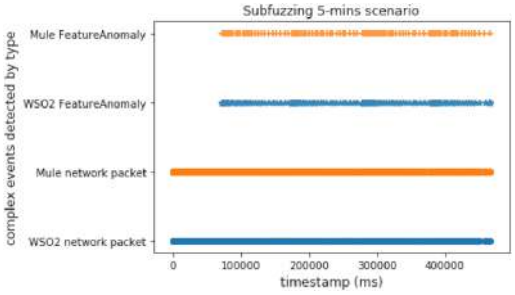


Figure 13 **Subfuzzing attack comparison (using FeatureAnomaly pattern).**
Full-size DOI: 10.7717/peerj-cs.787/fig-13

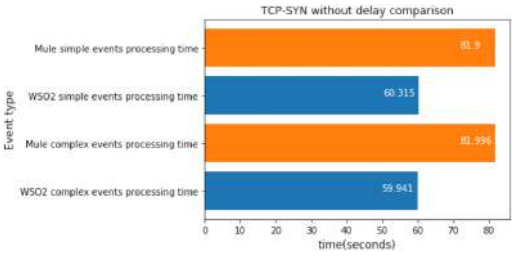


Figure 14 **TCP-SYN without delay comparison.** Full-size DOI: 10.7717/peerj-cs.787/fig-14

difference in performance between the two architecture implementations in greater depth. The figures in this subsection compare the last simple event detected with the first one, as well as the last complex event detected with the first one, measuring the time differences between these events.

TCP-SYN without delay

For each attack mentioned above, we implemented a stress scenario.

In this case, we executed the TCP-SYN scan 100 times, which took about 1 min.

Figure 14 shows the difference between the last and the first simple events detected, as well as that for the last and first complex events detected. Our goal was to discover the processing speed difference between the platforms.

As we can see, WSO2 was faster at processing simple events and complex events than Mule when there was a single broker topic, so this experiment confirms the results obtained in the previous section. It seems that, regardless of the packet delay, WSO2 is faster at processing simple events and complex events when there are no relationships between them.

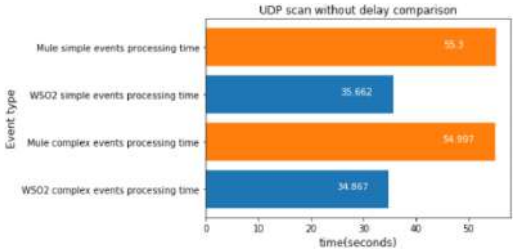


Figure 15 **UDP scan without delay comparison.** Full-size DOI: 10.7717/peerj-cs.787/fig-15

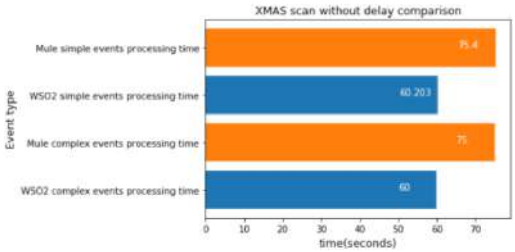


Figure 16 **XMAS Scan without delay comparison.** Full-size DOI: 10.7717/peerj-cs.787/fig-16

UDP scan without delay

In this case, the UDP scan was launched 100 times, which took about 37 s.

Figure 15 depicts a similar result to the TCP-SYN experiment: WSO2 was faster again. It is interesting that the differences in the experiments without a delay between WSO2 and Mule are far bigger than in those with a delay; this is because these experiments generate many more events than the experiments with a delay.

XMAS scan without delay

The XMAS scan was executed 100 times again, which took about 60 s.

As we can see in Fig. 16, the results are consistent with those we have observed above. In this experiment, WSO2 was faster again.

Mirai first stage without delay

The first stage of Mirai was executed 100 times, which took about 48 s.

Figure 17 shows that WSO2 was faster again at processing simple events.

DoS big message without delay

The DoS scenario does not use time windows, instead it compares each packet with its prediction. We executed the DoS experiment without delay once, which took about 20 s.

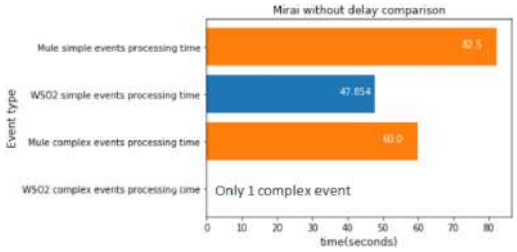


Figure 17 **Mirai first stage without delay comparison.** Full-size DOI: 10.7717/peerj-cs.787/fig-17

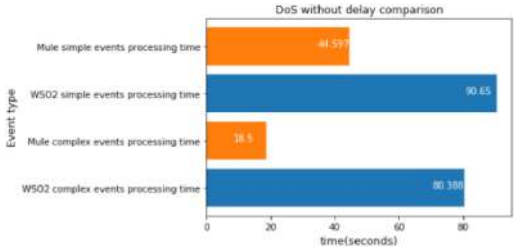


Figure 18 **DoS without delay comparison.** Full-size DOI: 10.7717/peerj-cs.787/fig-18

The results are very interesting, as illustrated in Fig. 18. They show that Mule was faster than WSO2 when there was an operation between 2 broker topics. The performance difference between implementations was even bigger than in the experiments with one type of simple event.

MQTT disconnect wave without delay

We executed the discwave attack for about 27 s, and used the *FeatureAnomaly* prediction pattern to detect it.

Fig. 19 shows that Mule was faster again when we compared 2 different events. This experiment had the highest workload, and therefore the difference between WSO2 and Mule was even bigger than before.

MQTT subscription fuzzing without delay

This experiment consisted in running the subfuzzing attack for approximately 47 s.

As we can see in Fig. 20, Mule was much faster again, so we can conclude that WSO2 is only faster than Mule when there are no comparison operations between different events.

In short, each CEP engine has different advantages. The Esper CEP engine integrated with the Mule ESB is better when there are comparisons between different events, so

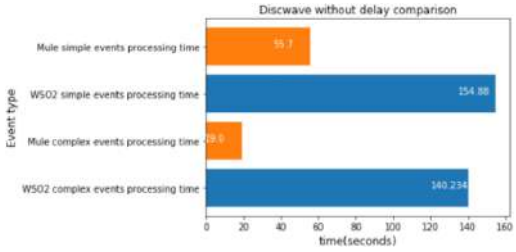


Figure 19 **Discwave without delay comparison.** Full-size DOI: 10.7717/peerj-cs.787/fig-19

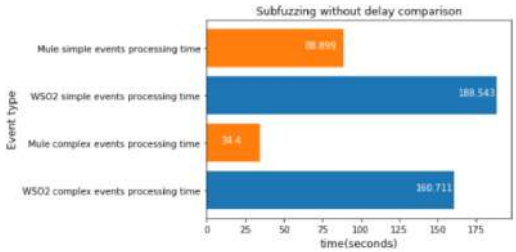


Figure 20 **Subscription fuzzing without delay comparison.** Full-size DOI: 10.7717/peerj-cs.787/fig-20

Esper/Mule performs better on patterns where different events are compared. As an example, we can see this behavior in the anomalous packet pattern. However, when there are no comparisons between different events, WSO2 is faster than Esper/Mule. We can conclude that WSO2 provides a better raw performance, in other words, WSO2 is able to process network packets faster than Esper/Mule but its performance is worse when there are comparisons between events.

DISCUSSION

With the obtained results, we can discuss and answer the four research questions posed in *Introduction* section:

Answers to the Research Questions

- **RQ1**: Can a real-time data stream processing architecture be implemented with the WSO2 ESB together with the Siddhi CEP engine and be integrated with ML techniques?
  - We can definitely affirm that it is possible to implement a streaming data processing architecture using WSO2 ESB together with the Siddhi CEP engine and integrate them with ML techniques. In fact, we have implemented an architecture equivalent to

the one presented in *Roldán et al. (2020)*, but using the said WSO2 technologies. We have also tested its functionality in a realistic environment consisting of security attacks in the field of the IoT.

- **RQ2**: Can a streaming data processing architecture based on the integration of ML techniques with the WSO2 CEP engine and ESB achieve or improve upon the performance of the previously proposed architecture (*Roldán et al., 2020*)?
  - We can undoubtedly say that WSO2 CEP and ESB can achieve a performance similar to that achieved by integrating Esper CEP and Mule ESB in an equivalent streaming data processing architecture for detecting security attacks in the IoT. We have carried out a series of tests with a number of typical attacks on communication protocols in the IoT environment, and we have seen that both architectures achieve an appropriate and similar performance, although we did detect that each of them can achieve a better performance with certain types of patterns, which allows us to answer our next research question.
- **RQ3**: What kind of event patterns are processed faster with WSO2/Siddhi and which ones with Mule/Esper, and which of the two architectures is more suitable for supporting high-stress situations?
  - On the one hand, we have observed that the WSO2-based architecture is faster at processing simple events when there are no pattern comparisons between different event types. This is because WSO2 has a higher performance when processing simple events. On the other hand, the Mule-based architecture has shown to be faster when comparing different types of events. The behavior of the architectures under stress will depend on the type of pattern conditions. If we are able to avoid patterns with comparisons between events of different types, WSO2 will be faster in a high-stress situation, since its ESB has a higher performance when processing simple events. Otherwise, Mule will be faster.
- **RQ4**: Which of these architecture implementations is the best to be deployed in an IoT security attack detection environment?
  - Both implementations are effective, but in this context we advocate the choice of WSO2 because it allows us to integrate the different types of events in a general unified event. This dramatically increases the performance of WSO2. Both ESBs can be deployed in an IoT environment, but WSO2 is faster when using this general event (as we can see from the stress experiments). Despite this, Mule can be deployed successfully too, but its performance is worse than that of WSO2.

CONCLUSIONS AND FUTURE WORK

This paper has presented and compared two implementations of an intelligent SOA 2.0-based architecture integrated with CEP technology and ML techniques that are designed to

detect security attacks against IoT systems. Each of the implementations incorporates a CEP engine and an ESB from prestigious vendors: Esper CEP and Mule ESB on the one hand, and WSO2 ESB and Siddhi CEP on the other.

The validation process, through which the behavior of both architectures was evaluated under the same conditions in a realistic scenario of security attacks on IoT protocols, allowed us to draw the following relevant conclusions:

- Both implementations of the architecture allow us to detect well-known attacks in the field of IoT protocols, with the corresponding event patterns of these attacks.
- Thanks to the use of ML techniques, the architecture can detect novel attacks that have not previously been defined through specific event patterns.
- Our architecture is able to work as a pure rule-based IDS with patterns defined by an expert, as well as allowing the addition of patterns for detecting non-modeled attacks in order to act as an anomaly detection architecture.
- Both architecture implementations present a suitable degree of efficiency for the field of security attacks in the IoT, but each one has its own advantages and drawbacks.
- The Mule-based architecture is faster when the architecture makes use of 2 message broker topics to compare the values of their features.
- The WSO2-based architecture is faster when there is a single topic and the system has a heavy workload.
- To mitigate the performance degradation, suffered by the system under heavy workloads, the operations between the topics can be modified by joining the prediction and network packet data in a general topic, thus mitigating this problem when comparing 2 topics in the WSO2-based architecture.
- In the Mule-based architecture it is more difficult to overcome this problem because our experiments have shown that the performance of Mule does not improve when there is a single type of topic.

Although our work achieved the proposed objectives, there are certain limitations in specific contexts. One is that although, the architecture makes it possible to define a threshold automatically, it is still necessary to perform a feature selection process. Another is that despite the fact that the architecture is capable of defining a threshold for one or more features, it is not able to fully generate the pattern.

As future work, we plan to test our architecture in a different network to validate our proposal with other protocols and conditions. We would like to point out that the performance of our proposal is subject to a correct ML process (data extraction, data preprocessing, algorithm selection, etc.). It would also be of interest to implement the architecture with additional ESBs and CEP engines to extend the comparison with the products of other vendors. Another interesting line of future work would be to automate the process of feature selection, as proposed in *Wajahat et al. (2020)*, thus providing useful information for the selection of the machine learning model with different

underlying structures in network traffic. These modifications should solve the current limitations of the architecture mentioned above.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported by the Spanish Ministry of Science, Innovation and Universities and the European Union FEDER Funds [grant numbers FPU 17/02007, RTI2018-093608-B-C33, RTI2018-098156-B-C52 and RED2018-102654-T]. This work was also supported by the JCCM [grant number SB-PLY/17/180501/ 000353] and the Research Plan from the University of Cadiz and Grupo Energetico de Puerto Real S.A. under project GANGES [grant number IRTP03' UCA]. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
Spanish Ministry of Science, Innovation and Universities and the European Union FEDER Funds: FPU 17/02007, RTI2018-093608-B-C33, RTI2018-098156-B-C52 and RED2018-102654-T.
JCCM: SB-PLY/17/180501/ 000353.
Research Plan from the University of Cadiz and Grupo Energetico de Puerto Real S.A. under project GANGES: IRTP03_UCA.

Competing Interests

The authors declare that they have no competing interests.

Author Contributions

- José Roldán-Gómez conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Juan Boubeta-Puig conceived and designed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Gabriela Pachacama-Castillo performed the computation work, authored or reviewed drafts of the paper, and approved the final draft.
- Guadalupe Ortiz conceived and designed the experiments, analyzed the data, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Jose Luis Martínez conceived and designed the experiments, analyzed the data, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The data is available at Mendeley: Roldán-Gómez, José; Boubeta-Puig, Juan; Pachacama-Castillo, Gabriela; Ortiz, Guadalupe; Martínez, José Luis (2021), "Dataset for Detecting Security Attacks in Cyber-Physical Systems: A Comparison of Mule and WSO2 Intelligent IoT Architectures", Mendeley Data, V1, doi: 10.17632/fvb9pp5xsh.1.

The code is available as a Supplemental File and the code and patterns are available at GitHub: https://github.com/josE4roldan/Detecting-security-attacks-in-cyber-physical-systems-a-comparison-of-mule-and-WSO2-intelligent-IoT.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.787#supplemental-information.

REFERENCES

Bamhdi A. 2021. Requirements capture and comparative analysis of open source versus proprietary service oriented architecture. *Computer Standards & Interfaces* **74**:103468 DOI 10.1016/j.csi.2020.103468.

Benito-Parejo M, Merayo MG, Núñez M. 2020. An evolutionary technique for supporting the consensus process of group decision making. In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2201–2206.

Bertino E, Choo KKR, Georgakopolous D, Nepal S. 2016. Internet of things (IoT): smart and secure service delivery. *ACM Transactions on Internet Technology* **16(4)**:1–7 DOI 10.1145/3013520.

Boubeta-Puig J, Ortiz G, Medina-Bulo I. 2015. MEdit4CEP: a model-driven solution for real-time decision making in SOA 2.0. *Knowledge-Based Systems* **89**:97–112 DOI 10.1016/j.knosys.2015.06.021.

Buczak AL, Guven E. 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials* **18(2)**:1153–1176 DOI 10.1109/COMST.2015.2494502.

Corral-Plaza D, Medina-Bulo I, Ortiz G, Boubeta-Puig J. 2020. A stream processing architecture for heterogeneous data sources in the Internet of Things. *Computer Standards & Interfaces* **70(C)**:103426 DOI 10.1016/j.csi.2020.103426.

Corral-Plaza D, Ortiz G, Medina-Bulo I, Boubeta-Puig J. 2021. MEdit4CEP-SP: a model-driven solution to improve decision-making through user-friendly management and real-time processing of heterogeneous data streams. *Knowledge-Based Systems* **213**:106682 DOI 10.1016/j.knosys.2020.106682.

Dayarathna M, Perera S. 2018. Recent advancements in event processing. *ACM Computing Surveys* **51(2)**:1–36 DOI 10.1145/3170432.

Demeter D, Preuss M, Shmelev Y. 2019. IoT: a malware story. Securelist. *Available at https://securelist.com/iot-a-malware-story/94451/* (accessed 9 May 2021).

EsperTech. 2019. 7+ Million events-per-second. EsperTech. *Available at https://www.espertech.com/2019/03/07/6-million-events-per-second/* (accessed 9 May 2021).

EsperTech. 2021. Esper. *Available at http://www.espertech.com/esper/* (accessed 9 May 2021).

Freire DL, Frantz RZ, Roos-Frantz F. 2019. Ranking enterprise application integration platforms from a performance perspective: an experience report. *Software: Practice and Experience* **49(5)**:921–941 DOI 10.1002/spe.2679.

Fremantle P. 2015. A reference architecture for the Internet of Things. *Available at https://www.researchgate.net/publication/308647314_A_Reference_Architecture_for_the_Internet_of_Things* (accessed 18 September 2021).

Geurts P, Ernst D, Wehenkel L. 2006. Extremely randomized trees. *Machine Learning* **63(1)**:3–42 DOI 10.1007/s10994-006-6226-1.

Giatrakos N, Alevizos E, Artikis A, Deligiannakis A, Garofalakis M. 2020. Complex event recognition in the big data era: a survey. *The VLDB Journal* **29(1)**:313–352 DOI 10.1007/s00778-019-00557-w.

Gutnikov A, Badovskaya E, Kupreev O, Shmelev Y. 2021. Analytical report on DDoS attacks in the second quarter of 2021. *Securelist. Available at https://securelist.com/ddos-attacks-in-q2-2021/103424/* (accessed 29 September 2021).

Górski T, Pietrasik K. 2017. Performance analysis of Enterprise Service Buses. *Journal of Theoretical and Applied Computer Science* **10**:16–32.

Kaspersky. 2021. Kaspersky Security Bulletin 2020-2021. EU statistics. *Available at https://securelist.com/kaspersky-security-bulletin-2020-2021-eu-statistics/102335/* (accessed 29 September 2021).

Luckham DC. 2012. *Event processing for business: organizing the real-time enterprise.* Hoboken, NJ: John Wiley & Sons.

Lueth KL. 2018. State of the IoT 2018: number of IoT devices now at 7B–market accelerating. *Available at https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/* (accessed 9 May 2021).

Montgomery DC, Peck EA, Vining GG. 2021. *Introduction to linear regression analysis.* Hoboken: John Wiley & Sons.

Moore S. 2018. Gartner says 25 percent of customer service operations will use virtual customer assistants by 2020. *Available at https://www.gartner.com/en/newsroom/press-releases 855 /2018-02-19-gartner-says-25-percent-of-customer-service-\operati 856 ons-will-use-virtual-customer-assistants-by-2020.*

Moss S. 2016. Major DDoS attack on Dyn disrupts AWS, Twitter, Spotify and more. *Available at https://www.datacenterdynamics.com/en/news/major-ddos-attack-on-dyn-disrupts-aws-twitter-spotify-and-more/#* (accessed 9 May 2021).

MuleSoft. 2021a. Enterprise hybrid integration platform | Anypoint platform. *Available at https://www.mulesoft.com/platform/enterprise-integration* (accessed 9 May 2021).

MuleSoft. 2021b. Mule ESB | Enterprise Service Bus | Open Source ESB. *Available at https://www.mulesoft.com/platform/soa/mule-esb-open-source-esb* (accessed 9 May 2021).

OASIS. 2019. MQTT Version 5.0. *Available at http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html* (accessed 9 May 2021).

Papazoglou M. 2012. *Web services and SOA: principles and technology.* Second Edition. Essex, New York: Pearson Education.

Roldán J, Boubeta-Puig J, Martinez JL, Ortiz G. 2020. Integrating complex event processing and machine learning: an intelligent architecture for detecting IoT security attacks. *Expert Systems with Applications* **149**:113251 DOI 10.1016/j.eswa.2020.113251.

Roldán-Gómez J, Boubeta-Puig J, Ortiz G, Pachacama G, Martinez JL. 2021. Detecting-security-attacks-in-cyber-physical-systems-a-comparison-of-mule-and-WSO2-intelligent-IoT. *GitHub.*

Available at https://github.com/josE4roldan/Detecting-security-attacks-in-cyber-physical-systems-a-comparison-of-mule-and-WSO2-intelligent-IoT-.

Valero V, Diaz G, Boubeta-Puig J, Macia H, Brazalez E. 2021. A compositional approach for complex event pattern modeling and transformation to colored Petri nets with black sequencing transitions. *IEEE Transactions on Software Engineering* DOI 10.1109/TSE.2021.3065584.

Wajahat M, Yele A, Estro T, Gandhi A, Zadok E. 2020. Analyzing the distribution fit for storage workload and internet traffic traces. *Performance Evaluation* **142**:102121 DOI 10.1016/j.peva.2020.102121.

Warburton D. 2021. DDoS attack trends for 2020. *Available at https://www.f5.com/labs/articles/threat-intelligence/ddos-attack-trends-for-2020* (accessed 18 September 2021).

WSO2. 2020. Separating the worker and manager nodes. *Available at https://docs.wso2.com/display/ADMIN44x/Separating+the+Worker+and+Manager+Nodes* (accessed 9 May 2021).

WSO2. 2021a. PMML based predictive analytics extension. *Available at https://docs.wso2.com/display/DAS310/PMML+Based+Predictive+Analytics+Extension* (accessed 9 May 2021).

WSO2. 2021b. Siddhi. *Available at http://siddhi.io/* (accessed 9 May 2021).

WSO2. 2021c. WSO2 | The open source technology for digital business. *Available at https://wso2.com/* (accessed 9 May 2021).

WSO2. 2021d. WSO2 enterprise service bus. *Available at https://wso2.com/products/enterprise-service-bus/* (accessed 9 May 2021).

CHAPTER 5

Attack pattern recognition in the Internet of Things using Complex Event Processing and Machine Learning

- **Title**: Attack Pattern Recognition in the Internet of Things using Complex Event Processing and Machine Learning.
- **Authors**: José Roldán-Gómez, Juan Boubeta-Puig, Juan Manuel Castelo Gómez, Javier Carrillo-Mondéjar, José Luis Martínez Martínez
- **Type**: Conference paper.
- **Conference**: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
- **Publisher**: IEEE.
- **ISSN**: 2577-1655.
- **Status**: Published.
- **Publication date**: January 2022.
- **DOI**: 10.1109/SMC52423.2021.9658711
- **GGS ranking**: A-.

2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
17-20 October, 2021. Melbourne, Australia

Attack Pattern Recognition in the Internet of Things using Complex Event Processing and Machine Learning

José Roldán-Gómez[1], Juan Boubeta-Puig[2], Juan Manuel Castelo Gómez[1],
Javier Carrillo-Mondéjar[1] and José Luis Martínez Martínez[1]

2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 978-1-6654-4207-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/SMC52423.2021.9658711

***Abstract*— The Internet of Things (IoT) paradigm demands adapting traditional cybersecurity solutions to address the inherent limitations of IoT environments, in particular their low computational power and limited amount of memory and bandwidth. The Complex Event Processing (CEP) technology has proven to be useful in this context by deploying a CEP engine for detecting real-time attacks in an IoT network. However, CEP is only capable of detecting attacks that have been previously modeled as event patterns. This requires a domain expert who knows the conditions that must be satisfied so that certain attacks can be detected, thus identifying unmodeled ones is not possible. This paper aims to address this problem by proposing a machine learning algorithm that allows for the automatic creation of CEP patterns based on categorized data if the goal is to classify attacks, or even uncategorized data if the objective is to detect anomalies. An evaluation of the effectiveness of the automatically generated patterns for recognizing different attacks in IoT environments is also conducted in this paper.**

I. Introduction

The Internet of Things (IoT) is an increasingly common paradigm in our everyday lives. It has the potential to completely change the way in which we interact with machines and devices. The IoT can be analyzed from the point of view of a global network comprised of devices (also known as *things*) [1]. Undoubtedly, the IoT could become a key pillar of our lives in the medium and long term. As such, IoT-based approaches already exist in a wide range of applications, either from an industry and an academic perspective. In addition, the ability to interconnect all devices offers new possibilities in fields such as healthcare, economics, engineering, or resource management, among many other fields.

The IoT has a number of peculiarities; first, it is a very heterogeneous paradigm, from the point of view of both devices and protocols. It is possible to find very diverse devices with different computational capabilities, although most devices used in this context have very low processing power and limited amount of memory and bandwidth. In addition, it combines the use of both classic protocols and ones specifically for this new paradigm.

However, this set of features also has an effect on the cybersecurity paradigm. The impossibility of implementing traditional security measures due to this lack of resources means that it is necessary to adapt or generate new ones that can address current threats without consuming too many resources [2].

In addition to these problems associated with the limitations of memory, network and computational capacities, the trend of malware and threats must be taken into account. In fact, signature detection is becoming increasingly fragile, as there is is a large amount of polymorphic malware and ones belonging to the same family with slight differences can be found. Behavior-based threat detection is becoming useful as a supplement to signature-based detection [3].

A technology capable of operating in low capacity environments is Complex Event Processing (CEP) [4]. A CEP engine is the software for detecting situations of interest (event patterns) through the analysis and correlation of huge amounts of data. However, a CEP engine is not able to detect certain behaviors not previously modeled by an expert. This poses the challenge of, without having to design a new technology, finding a way to create CEP patterns automatically from the behavior of the occurred events.

To solve such fundamental problems of automatic CEP pattern creation, this paper proposes a Machine Learning (ML) algorithm that uses the Principal Component Analysis (PCA) [5]. Since PCA reduces the dimensionality of a given dataset, the proposed algorithm is appropriate for IoT environments where there are limited resources.

Therefore, the main research objective of this work is to develop a ML algorithm that integrates the CEP technology (1) to automatically generate patterns for abnormal events, detecting unmodeled attacks, (2) to greatly reduce the network usage required to detect attacks, and (3) being relatively lightweight, which is ideal for environments with low requirements. Furthermore, an evaluation of the effectiveness of the automatically generated patterns for recognizing different attacks in IoT environments is carried out in this paper.

This paper is organized as follows. Section II introduces the concepts necessary to understand the rest of the paper. Section III describes our proposal. Once the algorithm is presented, we introduce the test environment and perform an evaluation of the algorithm in Section IV. This is followed by

*This work has been supported by the Spanish Ministry of Science and Innovation and the European Regional Development Funds under projects RTI2018-098156-B-C52 and RTI2018-093608-B-C33, by the JCCM under project SB-PLY/17/180501/00035, by the Spanish Ministry of Science, Innovation and Universities under grants FPU 17/02007 and FPU 17/03105, and by the University of Castilla-La Mancha under grant 2018-PREDUCLM-7476.

[1]José Roldán-Gómez, Juan Manuel Castelo Gómez, Javier Carrillo-Mondéjar and José Luis Martínez Martínez are with the University of Castilla-La Mancha, Campus Universitario s/n, 02071, Albacete, Spain {jose.roldan, juanmanuel.castelo, javier.carrillo, joseluis.martinez}@uclm.es

[2]Juan Boubeta-Puig is with the Department of Computer Science and Engineering, University of Cadiz, Avda. de la Universidad de Cadiz 10, 11519 Puerto Real, Cádiz, Spain juan.boubeta@uca.es

Section V which discusses solutions related to our proposal. Finally, Section VI highlights the conclusions and future research lines.

II. Background

This section briefly introduces CEP technology and ML.

A. Complex Event Processing

CEP is a technology capable of capturing and analyzing a huge number of simple events to infer relevant situations in a particular domain [6], [7]. These simple events are produced from the devices to be monitored. It should be noted that they are called simple events because, as a general rule, they are the fundamental unit of information of the system, that is, raw data without any type of preprocessing or derived data.

The main feature of CEP is that conclusions can be drawn in real time by processing these simple events when there is prior knowledge of the correlation of these events. These conclusions are known as complex events, since they are directly derived from these simple events. The obtaining of these complex events, or situations of interest in the domain, occurs through the definition of patterns. Patterns are defined by domain experts, who specify the conditions to be met in order to generate such complex events. Each CEP engine provides its own *Event Processing Language* (EPL), for example, Esper EPL or SiddhiQL [8], [9].

One of the main advantages of CEP over other traditional solutions is the large amount of information it is able to process in real time, in addition to its quick response time when a complex event occurs. There is also evidence of successful integration of CEP with IoT environments [10].

B. Machine Learning

ML refers to the set of algorithms, techniques and systems capable of obtaining knowledge based on data previously gathered and directly related to the problem domain. Due to its high success rate, partly due to the increase in processing power on modern computers, ML is spreading widely and being used in many applications in various fields of knowledge [11]. These fields include, for example, medical diagnostics, object recognition, search engines or inferring a user's preferences among many others. One of the most interesting applications in this context is the use of ML to improve computer security. This ranges from fraud or malware detection systems to network attack or Denial of Service (DoS) detection systems [12], [13], [14].

Basically, there are two different types of algorithms depending on the information they need: supervised and unsupervised. On the one hand, supervised algorithms need the data from which they learn to be correctly labeled, what allows them to learn to categorize or predict values based on their characteristics. On the other hand, unsupervised algorithms do not need to train with labeled categories, the goal of these algorithms may be looking for patterns in the data, or finding different categories in it and group them, among many other applications [15].

In this work, we use PCA, which reduces the dimensionality of a dataset given a number of components in an unsupervised way. The algorithm finds linear combinations of the features to create the target components in such a way that these components are orthogonal and maximize the variance as much as possible. Thus, PCA reduces the dimension while minimizing the loss of information [5].

III. Proposed Algorithm

This section describes our algorithm for recognizing attack patterns. An explanatory diagram of the proposal can be found in Figure 1.

A. Dimensionality Reduction

The first stage of our algorithm consists in reducing the dimensionality of the dataset. This stage can be divided into three steps. This first stage consumes the input. The input is composed of training data, which are obtained from the system (in this case an IoT network). The elements belonging to the training dataset must be correctly labeled in order to identify the category of each element.

1) Preprocessing: Raw data can rarely be used directly for training the model, therefore it is necessary to carry out a preprocessing work in which the data are accommodated to the algorithm that is going to be used. Some of the treatments that can be given to the data are the following:

- Treatment of empty values: This involves filling in the empty fields of the records that do not have data. There are many ways to fill them either the mean, the mode, or an arbitrary value among many other options. Although with the latter one must be especially careful because it can affect the performance of subsequent algorithms.
- Normalization: This is to normalize the different values in magnitude and scale. This is usually done to avoid that features with very high values have more weight than they should have in the later stages of the algorithm.
- Binarization: It consists in giving numeric values to non-numeric features, it is necessary because PCA is not able to directly consume alphanumeric data.
- Others: In other scenarios it may be necessary to perform more preprocessing steps. This proposal is flexible and can be adapted to different preprocessing methods.

2) PCA: Once the data have been preprocessed, it is ready to be used for training the PCA model, which is capable of reducing the dimensionality of the elements. When the model has been trained, it is possible to reduce each new element into n components almost immediately. This reduction is ideal for IoT systems because the network bandwidth decreases as a consequence. In this case it is reduced by up to 85%, because events with 20 features are reduced to events with 3 features.

This will be an advantage over other solutions, because the algorithm can obtain functional patterns while considerably reducing the need for bandwidth and computational capacity. The number of components to be used will depend on the context, in this case study three components were used.

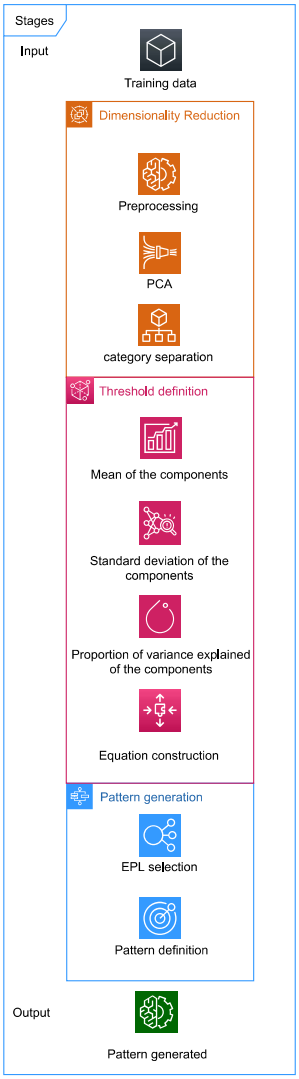


Fig. 1. Diagram with the stages of the proposal.

A larger number of components allows for more information to be collected, but requires more bandwidth and computational time.

3) Category Separation: The elements are divided into categories, this is simple since the dataset must be labeled.

B. Threshold Definition

The second stage is the fundamental novelty of this work. This phase allows finding patterns that make it possible to differentiate the elements from different categories. It consists of four steps.

1) Obtaining the mean of the components: This step consists in averaging each of the components with respect to its category, so that we obtain n averages, being n the number of components.

2) Obtaining the standard deviation of the component: Once the averages have been calculated, the standard deviation of each component with respect to the mean of its category is calculated. This allows defining a threshold for classifying an element in that category. A higher standard deviation indicates that the threshold must be higher.

3) Obtaining the proportion of variance explained of the components: The proportion of variance explained for each component is calculated with values ranging from 0 to 1 and representing the percentage of information that each component collects. This provides information on the importance of each component, which is crucial for assigning a weight to each component during the classification process.

4) Building the equation: Equation 1 defines in a formal way the classifier function, which will finally generate the corresponding pattern. Being x each one of the elements with reduced dimension that we want to check, in this way, x will have n components. The letters m and v represent the mean of each component with respect to its category and the proportion of variance explained of each component respectively. Finally, std is the standard deviation of each component with respect to the mean of its category.

To check whether an element corresponds to a family, the difference of each component with the mean is checked. This value must be positive, as the square root of the difference is squared. This value is multiplied by the proportion of variance explained of that component. In this way, the components with a higher weight are more decisive than those with a lower weight. The weighted differences are summed to obtain a single value that will be compared to the threshold for that category.

To generate the threshold, the standard deviations with respect to the mean of that category weighted by the proportion of variance explained of the component are used. The values obtained are summed to obtain a single value that defines the threshold. So a higher standard deviation results in a higher threshold. The same is true for the proportion of variance explained, a higher proportion of variance explained assigns more weight to the standard deviation of that component.

In certain categories there may be minority items that belong to the category but are quite different from the rest of the category. If there are only a few elements, they

$$f(x) = \begin{cases} f(x) = \sum_i^n \sqrt{(x_i - m_i)^2} \cdot v_i \leq (\sum_i^n std_i \cdot v_i) + \alpha & \text{if } x \text{ belongs to the category} \\ f(x) = \sum_i^n \sqrt{(x_i - m_i)^2} \cdot v_i > (\sum_i^n std_i \cdot v_i) + \alpha & \text{if } x \text{ does not belong to the category} \end{cases} \quad (1)$$

may not have sufficient influence on the standard deviation with respect to the mean of that category. To improve the performance in these situations, a correction element acting as bias is introduced, which allows to increase the threshold based on cross-validations or empirically.

C. Pattern Generation

At this time it is simply necessary to turn that equation into a CEP pattern. This stage is divided into two steps.

1) EPL selection: In this step you choose the EPL to be used, in this case the Siddhi engine is used to generate patterns that are deployed in WSO2 [9], [16], [17].

2) Pattern Definition: Once that the function which describes the pattern has been obtained, it is necessary to transform it into the syntax of the EPL to be used. This is achieved by defining the equation in the chosen syntax and filling the parameters (means, proportion of variance explained and threshold) with the values obtained in the previous stage.

An important detail to note is that the algorithm can generate patterns for two different purposes. In the first case the pattern detects anomalies, in the second case it detects known attacks. In the former, the model will be trained with data from the scenario in a normal state. In the latter, it is necessary to have data from those attacks labeled.

Listing 1. Pattern generated in Siddhi.

```
from mqtt_stream[
  math:abs(c1 - -1.0706718) * 0.44575194) +
  (math:abs(c2 - 1.20521006) * 0.23347243) +
  (math:abs(c3 - 0.03525247) * 0.08234665) >
  (0.2789)]
select c1, c2, c3
insert into AnomalyStream;
```

Listing 1 shows the pattern that is generated. In this case there is only one pattern that aims to find anomalies within the system, so events that exceed the given threshold are sent to the anomaly stream.

IV. Evaluation

This section describes the test environment and the operation of the patterns generated by the proposed algorithm.

We use a dataset which is composed of packets extracted from a network using MQTT (Message Queue Telemetry Transport) [18]. MQTT is a lightweight IoT-oriented publish-subscribe protocol and has been selected as it is widely used in IoT systems. In this system, we launched various scans and attacks. This experiment is able to show how well the proposal works in environments with different kinds of attacks. The scanners and attacks used are the following ones:

- *Subscription fuzzing*: In this attack the malicious client tries to subscribe to different topics using brute force or dictionaries, it can be used when we have access to a MQTT/MQTT-SN system.
- *Disconnection wave*: This attack attempts to spoof the MQTT/MQTT-SN protocol *id* and launch the disconnect command. If the system configuration is not correct, then it is possible to steal the *id* of the legitimate device and eject it from the system. The goal of this attack is to disconnect all devices from the system.
- *TCP syn scan*: This is the classic scanner used to check which TCP ports are open. The attacker starts with a SYN packet, if they receive a SYN/ACK then they assume that the port is open, if they receive a RST then they assume that it is closed.
- *UDP scan*: This involves sending UDP packets to each port to be scanned. If a UDP response is received, then the port is considered open, if no response is received, the port is either open or filtered. A packet of ICMP type *port unreachable error* means that the port is closed and any other type of ICMP error means that the port is filtered.
- *Xmas scan*: This is a rather unusual scanner. It involves sending to each TCP port a packet with the FIN, PSH and URG flags set to 1. If no response is received, the port is considered open or filtered. If an RST is received, it is considered a closed port. If any ICMP packet *unreachable error* is received, it is considered a filtered port.
- *Telnet connection*: In this experiment, packets are sent trying to connect via Telnet with different users and passwords. The goal is to simulate the first stage of Mirai which is a very common malware [19].

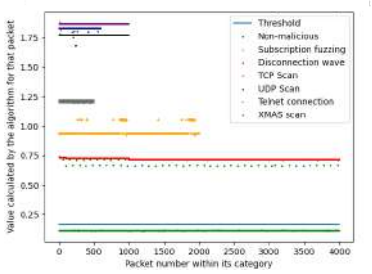


Fig. 2. Threshold and calculated values for the Anomaly detection pattern.

Several patterns have been deployed. First, an anomaly pattern that attempts to detect any packets that may be malicious. This pattern should detect all malicious packets regardless of the type of attack. An example of an anomaly

TABLE I
NUMERICAL REPRESENTATION OF THE PERFORMANCE OF THE GENERATED PATTERNS

Patterns	True Positive	True Negative	False positive	False negative	Recall	Precission	F1 score
Anomaly detection	9095	3918	82	0	1	0.991	0.996
Disconnection wave detection	4000	9095	0	0	1	1	1
Subscription fuzzing detection	1783	11095	0	217	0.891	1	0.942
Subscription fuzzing detection*	2000	11095	0	0	1	1	1
TCP SYN scan detection	999	12093	0	3	0.997	1	0.998
UDP port scan detection	580	12506	0	9	0.984	1	0.991
XMAS scan detection	999	12094	0	2	0.998	1	0.999
Telnet connection detection	503	12592	0	0	1	1	1

* means that a correction element is used.

detector pattern deployed would be as shown in Listing 1.

In addition to this pattern, six other patterns have been generated, one per attack. The purpose of each pattern is to detect the packets that correspond to that attack. Each one is evaluated based on the classic metrics of precision, recall and F1 score.

A. Anomaly Detection Pattern

The goal of this pattern is to detect any existing anomalies. That is, any packet that could be a malicious one.

Figure 2 shows a graphical representation of the results. It should be noted that the packets which trigger the pattern are those whose position is above the threshold. Since these packets are outside the non-malicious category, they should be classified as anomalies. The results are adequate since no anomaly is left out of pattern.

Only 82 packets out of 4000 non-malicious packets are detected as anomalies, this is because they are *ping request* packets sent to the broker. As they are rare events in a system in which packets of type *publish* are much more common, this means that these events are not detected as "*normal events*" by the model. In an environment in which it is necessary to avoid false positives, an additional pattern can be generated by making the corresponding categorical separation. However, since this is a common situation in anomaly detection, these packets are included in the results to see how they are affected.

Table I shows the results based on the current value and its prediction. This table shows false positives, false negatives, true positives and true negatives. It also provides the well-known metrics of precision, recall and F1-score.

Therefore, it can be concluded that the anomaly pattern works very well.

B. Disconnection Wave Detection Pattern

This pattern detects only attacks of type *disconnection wave*. This pattern does not look for anomalies but for a particular attack category.

Figure 3(a) shows good results, all packets of this type are detected. Moreover, no items outside this category are detected. It is important to note that the packets that trigger the pattern are those whose position is below the threshold. This occurs with all specific attack patterns.

Table I shows the numerical representation of the good scores. The results in this case are appropriate.

C. Subscription Fuzzing Detection Pattern

This pattern detects the subscription fuzzing attack. In this case, the use of the corrector element of the algorithm is very clear, because there are packets in this attack that are quite different from each other. Therefore, we use the correction element to improve the accuracy of the threshold. If the algorithm is used without the correction element, 217 packets would not be detected as attacks. As this attack generates many packets, it is not necessary to detect all of them in a real scenario. That is, even without a correction element, the system is able to detect this attack.

Figure 3(b) shows the good results of the pattern with correction element.

Table I shows the numerical results. The results would not be bad even without using a corrector element, but it shows the improvement of using it in the right way.

D. TCP SYN Scan Detection Pattern

In this case, it is a matter of detecting TCP SYN port scanners. In particular, three different scanners without regulating element have been used to check that the algorithm classifies well similar packets that are of different families.

These results are illustrated in Figure 3(c) and summarized in Table I. Only 3 packets are not detected and no other scanner is detected with this pattern. So, it can be concluded that the results are appropriate.

E. UDP Scan Detection Pattern

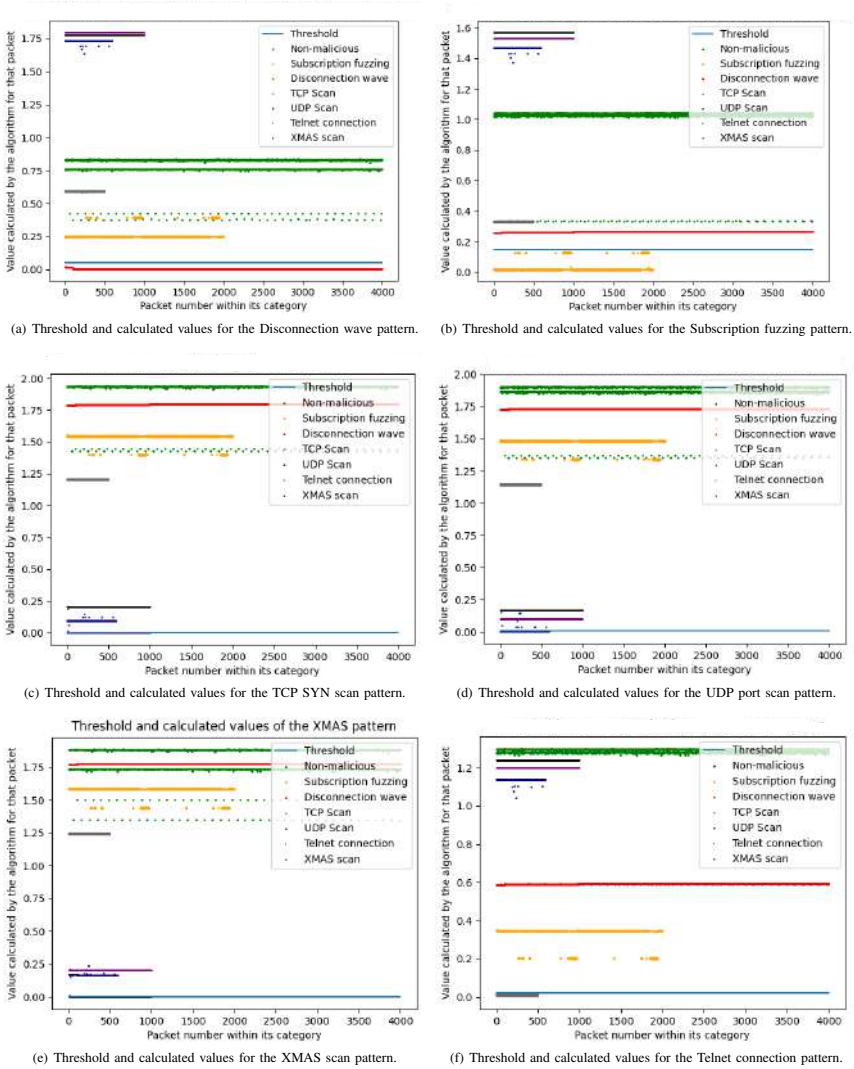
This pattern detects UDP scanners. Figure 3(d) shows the graphical representation of the results and Table I the numerical representation. In this case, 9 packets are above the threshold.

Even though 9 packets remain above the threshold, the results of this pattern are still good.

F. XMAS Scan Detection Pattern

This pattern detects the XMAS scan. As previously mentioned, it is not a very common scan nowadays. Since it is similar to TCP SYN, it is interesting to check if the proposal differentiates both scans.

Figure 3(e) shows the graphical representation of the results and Table I the numerical representation. It can be seen that only 2 packets from the category are not detected, and no packets from outside the category are detected by



(a) Threshold and calculated values for the Disconnection wave pattern.

(b) Threshold and calculated values for the Subscription fuzzing pattern.

(c) Threshold and calculated values for the TCP SYN scan pattern.

(d) Threshold and calculated values for the UDP port scan pattern.

(e) Threshold and calculated values for the XMAS scan pattern.

(f) Threshold and calculated values for the Telnet connection pattern.

Fig. 3. Results of patterns generated to detect specific attacks.

mistake, despite being quite similar at the feature level with respect to XMAS scan.

The results are very good again, as can be checked in Table I. Therefore, it can be concluded that the different scanners are classified correctly.

G. Telnet Connection Detection Pattern

This pattern detects Telnet connections. The idea is to test if the algorithm is effective against the first stage of Mirai.

Figure 3(f) shows the graphical representation and Table I shows the numerical representation of the good obtained results.

V. Related Work

There are some relevant pieces of research addressing this topic. Simsek et al. [20] offer an approach based on Deep Learning (DL) and data mining methods for pattern extraction. Although this work is interesting, it has two limitations. Firstly, the use of DL is only feasible when there is a substantially large amount of data. Secondly, it does not offer any form of network data compression, and DL usually works best with a high-dimensional dataset. Considering the limited computational, memory and network capacity of IoT devices, we consider that this mechanism may not be the best in some contexts.

Luong et al. [21] does not attempt to generate CEP patterns automatically, instead they choose to adapt the ML models to use the inputs from the streams used by a CEP engine. In this way, they achieve a hybrid model that can be adequate in certain applications. However, this model does not solve the automatic generation of patterns, and forces the devices (or *things*) receiving the data streams to implement the algorithm to be used or to store the already trained model.

Another related work is published by Sun et al. [22]. This work does not address the problem of generating new patterns, but rather the updating of existing patterns. It uses a Support Vector Machine (SVM) approach to update rules in which a loss function determines the quality of the rules, while an activation function determines whether the rule is good or not.

Petersen et al. [23] propose to use k-means to create clusters. In this way, their SVM-based proposal allows us to apply a rule to each category of events.

There are other different approaches to improve security in the IoT paradigm. Ding et al. [24] review different strategies to generate models capable of detecting insecure states in an IoT network.

Finally, our previous work [10] shows how it is possible to automatically define thresholds using regression models that best fit the distribution of the data for the particular use case. This work does not focus on the creation of a complete pattern, but on finding the values of the patterns defined by the domain experts and facilitating their work.

In contrast to the mentioned proposals, our proposed algorithm allows defining precise patterns automatically and operating without a high bandwidth cost.

VI. Conclusions and Future Work

This work proposes an algorithm, which integrates CEP and ML approaches for recognizing attack patterns in IoT scenarios. This algorithm makes it possible to automatically generate CEP patterns by reducing the network bandwidth usage, with datasets that do not need to be extremely large. Additionally, this proposal can be implemented with low-performance devices [25]. The results obtained from the conducted experiments are very promising.

Several research lines can be addressed to improve our proposal in different contexts. PCA is especially good when it is possible to find linear relationships in the features so that the components can collect enough information to reduce dimensionality without losing too much information. However, if an adequate characterization of the data is not achieved, or the distribution of the data does not allow it, PCA results may not be optimal. A possible improvement is to use PCA with kernels (KPCA) [26], this solution tries to map the distribution of the data so that a linear representation of the data can be made. However, it is necessary to return the data to its original dimension, so this enhancement is likely to slightly change the way in which patterns are created. To achieve this, the original idea can be used but these kernels, which represent the mappings between the different dimensions in the patterns, must be included. Moreover, it would be very interesting to test our algorithm in other contexts such as healthcare, home automation or precision agriculture.

References

[1] A. Thierer and A. Castillo, "Projecting the growth and economic impact of the Internet of things," *George Mason University, Mercatus Center, June*, vol. 15, 2015.

[2] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security Challenges in the IP-based Internet of Things," *Wireless Personal Communications*, vol. 61, no. 3, pp. 527–542, Dec. 2011. [Online]. Available: https://doi.org/10.1007/s11277-011-0385-5

[3] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, p. 102526, 2020.

[4] D. C. Luckham and B. Frasca, "Complex Event Processing in Distributed Systems," Stanford University, Tech. Rep., 1998.

[5] A. M. Martinez and A. C. Kak, "PCA versus LDA," *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

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

[7] J. Boubeta-Puig, G. Ortiz, and I. Medina-Bulo, "MEdit4CEP: A model-driven solution for real-time decision making in SOA 2.0," *Knowledge-Based Systems*, vol. 89, pp. 97–112, Nov. 2015. [Online]. Available: https://doi.org/10.1016/j.knosys.2015.06.021

[8] EsperTech, "Esper," http://www.espertech.com/esper/, 2021.

[9] WSO2, "Siddhi," http://siddhi.io/, 2021.

[10] J. Roldán, J. Boubeta-Puig, J. Luis Martínez, and G. Ortiz, "Integrating complex event processing and machine learning: An intelligent architecture for detecting IoT security attacks," *Expert Systems with Applications*, vol. 149, p. 113251, 2020. [Online]. Available: https://doi.org/10.1016/j.eswa.2020.113251

[11] K. Das and R. N. Behera, "A survey on machine learning: concept, algorithms and applications," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, no. 2, pp. 1301–1309, 2017.

[12] M. Ozay, I. Esnaola, F. T. Yarman Vural, S. R. Kulkarni, and H. V. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 8, pp. 1773–1786, Aug 2016.
[13] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, vol. 20, no. 1, pp. 343–357, Jan 2016.
[14] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "Denial-of-service attack detection based on multivariate correlation analysis," in *Neural Information Processing*, B.-L. Lu, L. Zhang, and J. Kwok, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, Jan. 2011, pp. 756–765.
[15] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, and M. Guizani, "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security," *arXiv preprint arXiv:1807.11023*, 2018.
[16] "Complex event processor," 2021. [Online]. Available: https://wso2.com/products/complex-event-processor/
[17] "Streaming Integrator," 2020. [Online]. Available: https://wso2.com/integration/streaming-integrator/
[18] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S–A publish/subscribe protocol for Wireless Sensor Networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE, 2008, pp. 791–798.
[19] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai Botnet," 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis
[20] M. U. Simsek, F. Yildirim Okay, and S. Ozdemir, "A deep learning-based CEP rule extraction framework for IoT data," *The Journal of Supercomputing*, Jan 2021. [Online]. Available: https://doi.org/10.1007/s11227-020-03603-5
[21] N. N. T. Luong, Z. Milosevic, A. Berry, and F. Rabhi, "An open architecture for complex event processing with machine learning," in *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, 2020, pp. 51–56.
[22] Y. Sun, G. Li, and B. Ning, "Automatic Rule Updating based on Machine Learning in Complex Event Processing," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, Nov. 2020, pp. 1338–1343.
[23] E. Petersen, M. Antonio To, S. Maag, and T. Yamga, "An Unsupervised Rule Generation Approach for Online Complex Event Processing," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, Nov. 2018, pp. 1–8.
[24] D. Ding, Q.-L. Han, X. Ge, and J. Wang, "Secure State Estimation and Control of Cyber-Physical Systems: A Survey," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 176–190, Jan. 2021.
[25] D. H. Hoang and H. D. Nguyen, "A PCA-based method for IoT network traffic anomaly detection," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, 2018, pp. 381–386.
[26] H. Hoffmann, "Kernel PCA for novelty detection," *Pattern Recognition*, vol. 40, no. 3, pp. 863–874, 2007. [Online]. Available: https://doi.org/10.1016/j.patcog.2006.07.009

CHAPTER 6

An automatic Complex Event Processing rules generation system for the recognition of real-Time IoT attack patterns

- **Title**: An Automatic Complex Event Processing Rules Generation System for the Recognition of Real-Time IoT Attack Patterns.
- **Authors**: José Roldán-Gómez, Juan Boubeta-Puig, Javier Carrillo-Mondéjar, Juan Manuel Castelo Gómez and Jesús Martínez del Rincon
- **Type**: Journal paper.
- **Journal**: Engineering Applications of Artificial Intelligence
- **Publisher**: Elsevier.
- **ISSN**: 0952-1976
- **Status**: Under review
- **JCR IF/ranking**: 7.802/Q1 (JCR2021).

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 Rincon[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*

Abstract

In recent years, the Internet of Things (IoT) has rapidly grown, crucially becoming the core of many application areas, and allowing the integration of sensors, smartphones, wearables together with IoT devices. However, the number of cyber 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 classical 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 anomalies. This combination makes it possible to achieve efficient CEP rules at computational level, with the results showing that the CEP rules obtained using our approach substantially improve upon the performance of the classical CEP rules that can be independently defined by a domain expert.

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

Preprint submitted to Engineering Applications of Artificial Intelligence August 31, 2022

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 [1, 2, 3]. The growth of this paradigm brings with it a series of new problems and challenges in many fields [4], such as the one in 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 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 classic systems to the IoT without prior adaptation, as they consume a higher number of resources than IoT devices can provide [5]. These characteristics, coupled with the ever-increasing use of these devices [6, 7], are leading to cybercriminals targeting this paradigm, with recent reports highlighting the high number of threats against it [8].

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) [9].

CEP is a technology that is able to work perfectly on devices with limited resources [10], which makes it ideal to be used in IoT devices. A CEP engine is a software component designed to detect specific situations. This is achieved through the analysis and correlation of a large number of previously defined simple events. 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 that can recognize attack CEP rules 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) [11]. PCA has two objectives: the first one is reducing the size of single events in the CEP engine; and the second is facilitating the charac-

terization 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. 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 [12].

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

- **(RQ1)** Are the CEP rules generated capable of systematically detecting attacks?
- **(RQ2)** Will these rules perform adequately for IoT environments?
- **(RQ3)** Are the rules generated efficient at the network traffic level?
- **(RQ4)** Can CEP rules detect unknown attacks?
- **(RQ5)** Is it possible to add an attack classification mechanism by proximity?

The proposed architecture is based on our previous work [13], in which we designed a preliminary algorithm capable of generating CEP rules. The novelties of the present work are the following:

- Simplified attack detection CEP rules have been implemented in order to generate a scenario comparable to the CEP rules obtained by our algorithm.
- The throughput of the CEP rules generated is measured against the productivity of CEP rules that are not generated by our architecture.
- A comparison is made of network usage when using our architecture and when not using it.
- A discussion of the advantages of the CEP rules obtained with our algorithm compared with the classical CEP rules is presented.

- A new mechanism based on the Mahalanobis distance is proposed for creating pure classification rules that determine which attack family is the closest. This function is useful when an anomaly is detected, as unknown attacks can be linked with 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.

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. Finally, Section 7 presents the conclusions drawn from the experiments.

2. Background

This section describes the background to security in the IoT, ML 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 [14], 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 [15]. 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 [8].

But not only conventional protocols are the target of cybercriminals, the newly IoT developed ones are also of interest for them. A good example of this is the Message Queuing Telemetry Transport (MQTT), which is a very popular network protocol used in the IoT [16] that is 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 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 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 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 TCP and UDP flooding, DNS reflection or LDAP reflection, among many others [17].

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.

2.2. Complex Event Processing

CEP is a technology capable of capturing, analyzing and correlating a large number of events in order to detect specific and relevant situations in a given domain [18]. Simple events can be obtained from different sources and have different attributes depending on the problem domain. The processing of these simple events allows CEP to infer information with a higher level of

semantic knowledge in real time. To achieve this, CEP rules are defined in such a way that if these rules are fulfilled, a complex event that represents a situation of interest is generated [19]. It is important to understand the key concepts of CEP technology. The first one is the CEP engine, which is the software element in charge of performing the processing of complex events. It receives the simple events, correlates them with the CEP rules and, when necessary, generates complex events. In our specific case, we use WSO2 CEP. CEP rules are the elements that allow the detection of events. In other words, CEP rules are patterns described and implemented by a domain expert that describe situations of interest to be identified and are written in an Event Processing Language (EPL) that may vary depending on the CEP engine used. In our case each CEP rule can identify a family of attacks. In the case of WSO2 CEP, the EPL used is SiddhiQL [20]. Another key concept are simple events, which are the raw data that are received by the CEP engine. In this case, where real-time network attack detection is pursued, these simple events will be the network packets. However, this may change depending on the context and the problem under study. The last important concept is that of a complex event. Complex events are equivalent to a situation of interest and are generated by CEP rules. When one of these rules is fulfilled, a complex event is generated. In our case a complex event means that an attack of a particular family has been detected.

The main advantage of CEP when compared with other traditional data analysis software lies mainly in its performance. CEP is able to process a large amount of data and report it in real time, thus offering a reduction in decision-making process times

2.3. Principal Component Analysis

PCA is a statistical method that tries to reduce the complexity of a sample space by reducing its dimensions. 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 $m<<n$. 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 by the vector of variables $n = \{n_1, n_2, ..., n_n\}$, the new variables of the vector $m = \{m_1, m_2, ..., m_m\}$ will have the representation that we can observe in Equation 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.

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

Each component collects an amount of information called 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.

2.4. 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.

$$d(x-\mu) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)} \tag{2}$$

The Equation 2 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 weight in the distance function.

In this case we will apply the Mahalanobis distance to simple events reduced with PCA, 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 Equation 3. Equation 4 shows the Mahalanobis distance weighted with the explained variance ratios.

$$VE = diag(rv_1, rv_2, ..., rv_m) \tag{3}$$

$$d(x - mu) = \sqrt{(x-\mu)^T (\Sigma^{-1} \times VE)(x-\mu))} \tag{4}$$

Thus Equation 4 allows to know the closest family 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 family of an event that may be an anomaly. This mechanism can be useful to act against that anomaly based on the closest known family.

3. Related work

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

There are several works that deal with threat detection using machine learning algorithms(i.e [21, 22]). However, our work address the open issue of using CEP engines to perform the detection by generating CEP rules automatically, due to their high data processing capacity. Therefore, this section focuses exclusively on papers that attempt to generate CEP rules.

There are some relevant works that try to solve the problem of automatic rule generation in CEP. It is possible to classify the proposals according to the need for prior rules.

3.1. Proposals that require previous rules

Yunhao Sun et al. [23] focuses their work on generating new rules, based on previous CEP rules that already exist. The requirements to use this proposal is to 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 rules set. With that new set of rules, clusters containing all the elements of each family are created and a rule is generated to represent those families. Although this work manages to update previous rules in a satisfactory way, it is not able to create them from scratch.

Other approach where the authors employ CEP in a completely different way is in the work proposed by Nathan Tri Luong et al. [24]. In this work, the authors use CEP to perform the pre-processing of the data and Tensor Flow, as 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, Haoyu Ren et al. [25] use a completely different approach, which is focused on optimizing performance in IoT environments. The advantage of this work is that 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 does not require previous rules

In this group we find proposals that do not require prior rules, but label complex events based on historical data. Ralf Bruns and Jürgen Dunkel. [26] 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, however, it needs a context in which complex events are known as a function of simple events. This requirement is not always easy to reach without prior rules.

Another work that manages to extract rules automatically without prior rules is the one proposed by José Roldán-Gómez et al. [27]. 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.

Reference	Performance focused	Need for prior rules	Novelty / Highlight
[23]	Yes	Yes	Pre-filtering rules before training improves performance.
[24]	No	Yes	It uses CEP for performing data preprocessing.
[26]	No	No	It uses Bat algorithm for generating new rules.
[27]	No	No	It is based on comparing the prediction of key features with their actual values.
[25]	Yes	Yes	Manually defined CEP rules and pretrained neural networks.
[28]	No	No	It uses GRU and Furia for generating CEP rules in an unsupervised manner
This work	Yes	No	PCA enables the generation of high-performance CEP rules

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

Finally, Mehmet Ulvi Simsek et al. [28] compare different algorithms to label simple events, then uses the most common algorithms for rule extraction. The paper concludes that GRU (used for labeling) together with the FURIA algorithm (used for rule extraction) obtain the best results in their experiments. The comparison made in this 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 analyzed papers focusing on the need for prior rules, and the focus on the performance of the papers. As we can see this proposal focused on performance without the need for prior rules is an improvement over other work in the state of the art. Although the use of PCA is very common in classical machine learning applications, it is not commonly used in CEP environments for rule generation. This novelty offers very promising results that improve the computational performance of manually generated CEP rules.

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. Firstly, we explain the system architecture with the rule generator deployed, as well as the context in which the proposal works. Then we will analyze in depth the different phases of the rule generator to understand how it works.

4.1. Proposed architecture

Figure 1 shows our proposed architecture deployed on an IoT network. The first step for generating CEP rules is to obtain training data for generating the model. To achieve this, the different packets are extracted from

the broker. These packets must be correctly labeled to train the model to later feed the CEP rule generator, which provides three outputs. The first output is a pre-trained PCA model that will reduce the dimensionality of simple events and generate lighter rules that offer better computational performance. This PCA model is used in rule generation, but it is also sent to MQTT clients for generating reduced simple events to enable real-time attack detection with the CEP rules generated by our proposal. The second output is the CEP rules, which feed the CEP engine and are in charge of detecting the different attacks suffered by the system. The third and last output is the covariances and averages of the different families. These are sent to the clients and allow using the Mahalanobis distance to add an additional mechanism for identifying the closest family in simple events that do not match any generated CEP rule.

Once the CEP rule generator has sent its three outputs to the different components that use them, the operation is as follows: Firstly, MQTT clients send the reduced simple events to the broker. To achieve this, the network packets, which are the simple events in our context, are reduced with the PCA pre-trained model. In addition, in these reduced events, the closest known family according to the Mahalanobis distance is also sent, using the means and covariances that the clients also have thanks to the CEP rule generator. Then, the broker, by means of an attack detection topic, sends these reduced single events to the CEP engine. Then, the CEP Engine will compare that single event with its different CEP rules, which were generated with the CEP rule generator. If the simple event falls within the threshold of any CEP rule, a complex event is generated that associates that simple event with the attack that defines that rule. If the simple event fits within the anomaly pattern and does not fit with any known attack, a complex event is generated indicating that it is an anomaly together with the closest family of that anomaly. This is possible thanks to the Mahalanobis distance. If a simple event is a legitimate packet it will not fit into any rule. Finally, when these complex events are generated, they are sent to an endpoint. This endpoint can be in charge of taking actions against the different attacks, these actions do not fall within the scope of this work.

4.2. CEP rules generator stages

Once the architecture as a whole has been detailed, we focus on the operation of the CEP rule generator. To achieve this goal, this pipeline is composed of 3 stages, which are as follows:

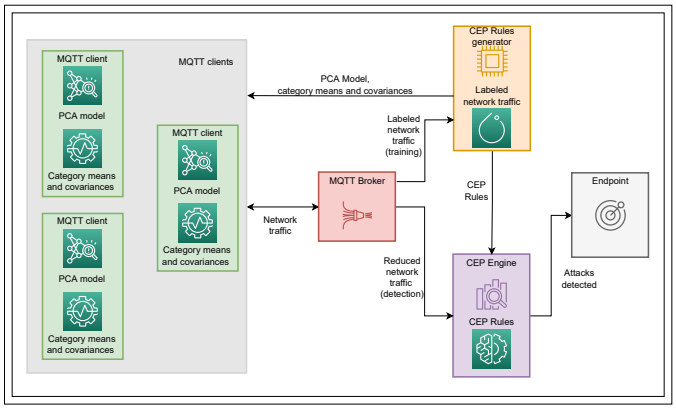


Figure 1: Diagram of the complete system with rule generator deployed.

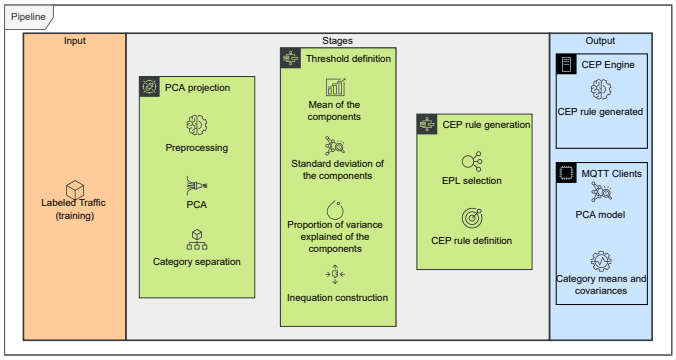


Figure 2: Proposed architecture pipeline for generating CEP rules.

- PCA projection. This stage generates a PCA model that is able to reduce the dimensionality of the different events of the system by pro-

jecting the input data into a space of reduced dimensionality. In order to obtain a successful projection, preprocessing of the training dataset is first required. This step requires an adaptation according to the training dataset. In this case it consists of 3 phases.

  - 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 characteristics 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 modelled 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 single 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 $x^t = [x_1, x_2, \dots x_i, \dots x_n]$.

For each category $j \in [1, c]$ being c 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, and then the standard deviation, defined as std_i^j of each component with respect to its category, is calculated as follows:

$m_i^j = 1/R \sum x_i^r \forall r \in R : y^r = j$

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

In addition, 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. By using these elements, an inequation similar to the one shown in Equation 5 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.

- 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.

  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 componet})
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 line 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*).

- Proximity classification mechanism. As discussed above during 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 single events with respect to the means of the different families. The closest family is then sent to the CEP engine together with the reduced single event. This is useful when there is no specific attack whose rule detects that single event, but it falls within the anomaly CEP rule. This way we can know that it is an anomalous attack together with the closest known class, which can be vital to take countermeasures quickly. Although such countermeasures are not within the focus of this paper.

Algorithm 1 CEP rules generation algorithm

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

```
Preprocess_dataset
Train_pca_model
Transform_preprocessed_dataset_with_pca
Category_separation
for each_component_of_pca do
    Extract_proportion_of_variance_explained
    for each_category do
        Extract_mean
        Extract_std
    end for
end for
Inequation_construction
Select_EPL
for each_category do
    Generate_Rule
    if proximity_classification_mechanism then
        Generate_proximity_classification_pattern
        Extract_covariance_matrix;
    end if
end for
```

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

16

$$f(x,j) = \begin{cases} 1 \text{ if } f(x) = \sum_i^n \sqrt{(x_i - m_i)^2} \cdot rv_i \leq (\sum_i^n std_i \cdot rv_i) + \alpha \\ 0 \text{ if } f(x) = \sum_i^n \sqrt{(x_i - m_i)^2} \cdot rv_i > (\sum_i^n std_i \cdot rv_i) + \alpha \end{cases} \quad (5)$$

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 setting. In addition, different CEP rules generated with our proposal are shown.

5.1. Dataset

The dataset is obtained from an MQTT network composed of 3 clients and a broker. In this network a 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.

- 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.
- Disconnection wave. This is an attack that attempts to disconnect all the clients from the broker. This attack works in configurations where the broker disconnects old clients when a new client tries to 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.
- 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 an *SYN/ACK* packet, and if it is closed, it will send an *RST* packet.

- UDP scan. The UDP scanner is used to search for open UDP ports on a computer, the broker in this case. If the broker sends an 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.
- 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, in this case the attack is against the broker.
- Xmas scan. This is a scan targeting TCP ports. To perform this attack, TCP packets are sent with the PSH, FIN and 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. If the port does not respond, the port may be filtered or open.

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

5.2. 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 CSV file.

For the feature selection, we follow the selection made in KDD99 [29], 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 is adequate to characterize our dataset we have employed extremely randomized trees [30]. This allows us to affirm that our selection of features is correct. Table 2 shows a ranking with different features.

5.3. 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 3.
As we can see in Table 3, not all categories have the same percentage in

Table 2: 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

Table 3: 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%)

the net training set. This is because there are not as many scan and telnet events as the previous ones. If we train the PCA model with so few samples of the latter, the resulting components may not correctly characterize these attacks.

5.4. 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 Equation 5.

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. Classical CEP rules against generated CEP rules

To test the performance of the CEP rules generated by our architecture, it is necessary to define classical CEP rules. In this case we define basicrules that look at a single feature to determine whether there is an attack. In a real environment these CEP rules would not be sufficient because they would yield many false positives. However, the objective of these rules in our experiments is to compare the CEP rules generated by our architecture with the simplest possible rules. This allows the experiments to be more representative, because if the CEP rules of our architecture offer a better computational performance than these rules, it means that the reduced CEP rules offer a better performance than the best case of the non-reduced CEP rules. Some classical attack CEP rules are shown in Listings 5, 6 and 7. These rules are built manually without using our proposal.

Listing 5: Classical 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: Classical 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: Classical 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.

6.2. Metrics used

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 seconds. This means that it is calculated by dividing all the events processed during this interval by 10.
- 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.
- Precision. This metric measures the ability of the CEP engine to avoid false positives. It is defined as follows:

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

- Recall. This metric measures the ability of the CEP rules to avoid false negatives, defined as follows:

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

- F-score. This metric averages precision and recall in order to to provide a more global view of CEP rules performance. It is defined as follows:

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

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.

Table 4: Metrics obtained by the generated CEP rules.

Metrics of generated rules without Mahalnobis	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 Mahalnobis	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

6.3. Accuracy 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 4 shows the results obtained by the proposal for classifying each attack. The results are very good, as in the worst case scenario (subfuzzing without corrector) an F1-score of 0.94 is obtained. We can conclude that the architecture is able to classify attacks correctly without problems.

Table 4 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 4 shows the results obtained by the proximity classification mechanism. We see that it is able to approximate any single event to its closest family. This is useful in environments with many anomalies where it is necessary to implement countermeasures quickly.

Table 5: 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
Discconection 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

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 seconds. The objective is to send a high workload to the system and observe how it behaves. It 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.

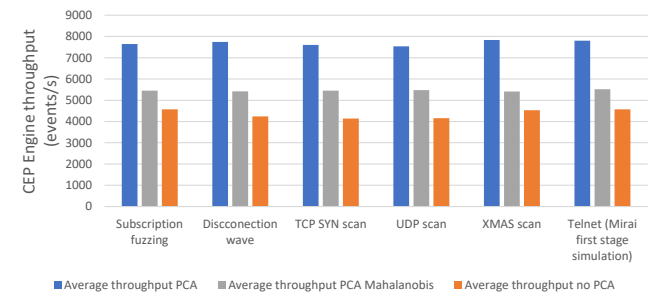


Figure 3: Average throughput for PCA vs no PCA

As we can see in Figure 3 and Table 5, 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 ruless generated by our proposal are capable of processing a higher number

Table 6: 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
Discconection 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

of events per second, which provides the system with a greater capacity for high workload situations. When the nearest category clasification mechanism is used, the performance improvement is limited to 24 percent over common CEP rules.

The last part of the results focus on the size of the events. A smaller event size means less saturation of the network.

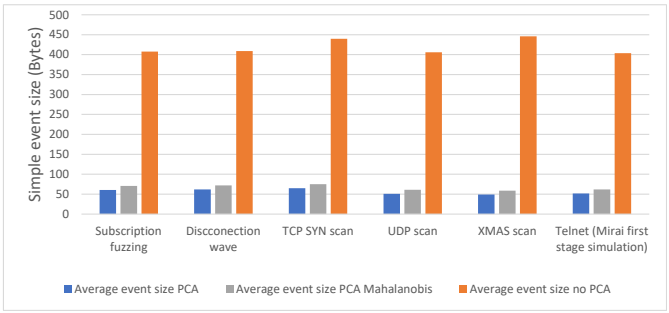


Figure 4: Average event size for PCA vs no PCA

As we can see in Figure 4 and Table 6, 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 family of that single event. 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** Are the CEP rules generated capable of systematically detecting attacks?
  - We can state that it is possible to generate CEP rules automatically. The results shown in Table 4 demonstrate that it is possible to generate functional CEP rules and that they work correctly.
- **RQ2** Will these rules perform adequately for IoT environments?
  - 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.
- **RQ3** Are the rules generated efficient at the network traffic level?
  - The simple events that are sent with the CEP rules generated are much fewer thanks to the use of PCA. In our experiments the average event size was reduced by 86%. This drastically reduces network usage by improving the effectiveness of the attack detection and facilitates real-time detection.
- **RQ4** Can CEP rules detect unknown attacks?
  - The results obtained from the experiments performed show that anomaly detection CEP rules can be created. These ruless 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. 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, being able to detect attacks perfectly. In addition, 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 that we 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. Moreover, 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 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.

Funding

This work was supported by the Spanish Ministry of Science, Innovation and Universities and the European Union FEDER Funds [grant numbers FPU 17/02007 and FPU 17/03105, RTI2018-093608-B-C33, RTI2018-098156-B-C52 and RED2018-102654-T], by the University of Castilla La Mancha [grant number DO20184364]. This work was also supported by the JCCM [grant number SB-PLY/17/180501/ 000353], and the Research Plan from the University of Cadiz and Grupo Energético de Puerto Real S.A. under project GANGES [grant number IRTP03_UCA].

References

[1] I. Calvo, M. G. Merayo, M. Núñez, A methodology to analyze heart data using fuzzy automata, Journal of Intelligent & Fuzzy Systems 37 (6) (2019) 7389–7399. doi:10.3233/JIFS-179348.

[2] P. Asghari, A. M. Rahmani, H. H. S. Javadi, Internet of Things applications: A systematic review, Computer Networks 148 (2019) 241–261. doi:10.1016/j.comnet.2018.12.008.

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

[4] M. M. Sadeeq, N. M. Abdulkareem, S. R. Zeebaree, D. M. Ahmed, A. S. Sami, R. R. Zebari, IoT and cloud computing issues, challenges and opportunities: A review, Qubahan Academic Journal 1 (2) (2021) 1–7.

[5] S. A. R. Shah, B. Issac, Performance comparison of intrusion detection systems and application of machine learning to Snort system, Future Generation Computer Systems 80 (2018) 157–170. doi:10.1016/j.future.2017.10.016.
URL https://www.sciencedirect.com/science/article/pii/S0167739X17323178

[6] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, E. K. Markakis, A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues, IEEE Communications Surveys Tutorials 22 (2) (2020) 1191–1221. doi:10.1109/COMST.2019.2962586.

[7] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, B. Sikdar, A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures, IEEE Access 7 (2019) 82721–82743. doi:10.1109/ACCESS.2019.2924045.

[8] Kaspersky, Kaspersky Security Bulletin 2020-2021. EU statistics, (accessed 23 December 2021) (2021).
URL https://securelist.com/kaspersky-security-bulletin-2020-2021-eu-statistics/102335/

[9] D. Corral-Plaza, I. Medina-Bulo, G. Ortiz, J. Boubeta-Puig, A stream processing architecture for heterogeneous data sources in the Internet of Things, Computer Standards & Interfaces 70 (2020) 103426. doi:10.1016/j.csi.2020.103426.

[10] G. Ortiz, J. Boubeta-Puig, J. Criado, D. Corral-Plaza, A. Garcia-de Prado, I. Medina-Bulo, L. Iribarne, A microservice architecture for real-time IoT data processing: A reusable Web of things approach for smart ports, Computer Standards & Interfaces 81 (2022) 103604. doi:10.1016/j.csi.2021.103604.

[11] A. Martinez, A. Kak, PCA versus LDA, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (2) (2001) 228–233. doi:10.1109/34.908974.

[12] R. De Maesschalck, D. Jouan-Rimbaud, D. L. Massart, The Mahalanobis distance, Chemometrics and Intelligent Laboratory Systems 50 (1) (2000) 1–18. doi:10.1016/S0169-7439(99)00047-7.

[13] J. Roldán-Gómez, J. Boubeta-Puig, J. M. Castelo Gómez, J. Carrillo-Mondéjar, J. L. Martínez Martínez, 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), 2021, pp. 1919–1926, iSSN: 2577-1655. doi:10.1109/SMC52423.2021.9658711.

[14] Kaspersky, IoT under fire: Kaspersky detects more than 100 million attacks on smart devices in H1 2019, (accessed 23 December 2021) (May 2019).
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

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

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

[17] D. Warburton, DDoS Attack Trends for 2020, (accessed 23 December 2021) (May 2021).
URL https://www.f5.com/labs/articles/threat-intelligence/ddos-attack-trends-for-2020

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

[19] V. Valero, G. Díaz, J. Boubeta-Puig, H. Macià, E. Brazález, A Compositional Approach for Complex Event Pattern Modeling and Transformation to Colored Petri Nets with Black Sequencing Transitions, IEEE Transactions on Software Engineering (2021). doi:10.1109/TSE.2021.3065584.

[20] Query Guide - Siddhi, accessed 8/01/2022.
URL https://siddhi.io/en/v5.1/docs/query-guide/

[21] I. Martins, J. S. Resende, P. R. Sousa, S. Silva, L. Antunes, J. Gama, Host-based IDS: A review and open issues of an anomaly detection system in IoT, Future Generation Computer Systems 133 (2022) 95–113. doi:10.1016/j.future.2022.03.001.
URL https://www.sciencedirect.com/science/article/pii/S0167739X22000760

[22] Y. Zhang, Q. Liu, On IoT intrusion detection based on data augmentation for enhancing learning on unbalanced samples, Future Generation Computer Systems 133 (2022) 213–227. doi:10.1016/j.future.2022.03.007.
URL https://www.sciencedirect.com/science/article/pii/S0167739X22000826

[23] Y. Sun, G. Li, B. Ning, Automatic Rule Updating based on Machine Learning in Complex Event Processing, in: 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), 2020, pp. 1338–1343. doi:10.1109/ICDCS47774.2020.00176.

[24] N. N. T. Luong, Z. Milosevic, A. Berry, F. Rabhi, An open architecture for complex event processing with machine learning, in: 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC), 2020, pp. 51–56, iSSN: 2325-6362. doi:10.1109/EDOC49727.2020.00016.

[25] H. Ren, D. Anicic, T. A. Runkler, 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, 2021, pp. 126–135. doi:10.1145/3465480.3466928.
URL https://doi.org/10.1145/3465480.3466928

[26] R. Bruns, J. Dunkel, Bat4CEP: a bat algorithm for mining of complex event processing rules, Applied Intelligence (Mar. 2022). doi:10.1007/s10489-022-03256-2.
URL doi.org/10.1007/s10489-022-03256-2

[27] J. Roldán, J. Boubeta-Puig, J. Luis Martínez, G. Ortiz, Integrating complex event processing and machine learning: An intelligent architecture for detecting IoT security attacks, Expert Systems with Applications 149 (2020) 113251. doi:10.1016/j.eswa.2020.113251.
URL https://www.sciencedirect.com/science/article/pii/S0957417420300762

[28] M. U. Simsek, F. Yildirim Okay, S. Ozdemir, A deep learning-based CEP rule extraction framework for IoT data, The Journal of Supercomputing 77 (8) (2021) 8563–8592. doi:10.1007/s11227-020-03603-5.

[29] H. G. Kayacik, A. N. Zincir-Heywood, M. I. Heywood, 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, 2005, pp. 1723–1722.

[30] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Machine Learning 63 (1) (2006) 3–42. doi:10.1007/s10994-006-6226-1.

CHAPTER 7

An automatic unsupervised Complex Event Processing rules generation architecture for real-time IoT attacks detection

- **Title**: An Automatic Unsupervised Complex Event Processing Rules Generation Architecture for Real-Time IoT Attacks Detection.
- **Authors**: José Roldán-Gómez, Jesús Martínez del Rincón,Juan Boubeta-Puig and José Luis Martínez
- **Type**: Journal paper.
- **Journal**: Wireless Networks.
- **Publisher**: Springer.
- **ISSN**: 1572-8196
- **Status**: Published.
- **Publication date**: January 2023.
- **DOI**: 10.1007/s11276-022-03219-y
- **JCR IF/ranking**: 2.701/Q2 (JCR2021).

An Automatic Unsupervised Complex Event Processing Rules Generation Architecture for Real-Time IoT Attacks Detection

José Roldán-Gómez[1*], Jesús Martínez del Rincon[2], Juan Boubeta-Puig[3] and José Luis Martínez[1]

[1*]Albacete Research Institute of Informatics (I3A), University of Castilla-La Mancha, Campus Universitario s/n, Albacete, 02006, Spain.
[2]Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Belfast, BT3 9DT, UK.
[3]Department of Computer Science and Engineering, University of Cadiz, Avda. de la Universidad de Cadiz 10, Puerto Real, Cadiz, 11519, Spain.

*Corresponding author(s). E-mail(s): jose.roldan@uclm.es;
Contributing authors: j.martinez-del-rincon@qub.ac.uk;
juan.boubeta@uca.es; JoseLuis.Martinez@uclm.es;

Abstract

In recent years, the Internet of Things (IoT) has grown rapidly, accompanied by a vast number of attacks against it. Certain limitations of the paradigm, such as reduced processing capacity and limited main and secondary memory, make it necessary to study new methods to detect attacks in real time due to the difficulty of adapting techniques used in other paradigms. In this paper, we propose an architecture capable of generating Complex Event Processing (CEP) rules for real-time attack detection in an automatic and completely unsupervised manner. To this end, the CEP technology, which allows for analyzing and correlating a large amount of data in real time and can be deployed in IoT environments, is integrated with Principal Component Analysis (PCA), Gaussian Mixture Models (GMM) and Mahalanobis distance. This architecture has been tested through two different experiments to simulate real attack scenarios in an

IoT network. The results obtained show that the generated rules obtained an F1 score of 0.9890 in detecting six different IoT attacks in real time.

Keywords: Attack detection, Cybersecurity, Internet of Things, CEP, Machine learning

1 Introduction

The Internet of Everything (IoE) has grown rapidly in the last decade and it does not seem that this growth will stop or slow down any time soon, due to the obvious potential offered by this new paradigm. Proof of this is the increasing number of interactions with certain applications oriented to this paradigm through devices such as smartphones or wearables. IoE can be considered an extension of the Internet of Things (IoT) [1]. While the two key elements of IoT are things and networks, in IoE there are five key concepts, these are things, networks, people, data and process [2]. IoT and IoE have been shown useful in a myriad of contexts and applications, such as healthcare applications, home automation, resource management and many more [3–6].

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

It is essential to understand that solutions from other paradigms cannot always be directly applied in IoT environments, mainly due to the limitations of IoT devices. These limitations include: low computational capacity, limited bandwidth, low-cost sensors, low memory and battery usage. If we add to this the increase in the use of this type of devices [8, 9], which has led to an increase in cybercriminals who focus on this paradigm, this has resulted in researchers having to adapt or design new solutions in different areas of security, such as cryptography [10] or reliability models [11]. Within the different areas of IoT cybersecurity, this work focuses on real-time IoT attack detection because early detection of an attack can be vital to protect the system. This is important to improve data protection in IoT and IoE environments.

To detect network attacks in real time within IoT environments we need to meet two non-negotiable requirements. The first is that the system can be deployed in IoT environments with the limitations mentioned above, the second is that the system is able to process a large amount of data, this allows the system to be scalable and work in networks of different sizes. Complex Event Processing (CEP) [12] is a technology that perfectly meets these requirements. CEP allows a large amount of data to be collected in the form of simple events.

By means of rules defined by an expert, situations of interest can be extracted from these simple events, thus forming complex events. This functionality is ideal, for example, for detecting network attacks in real time. To this aim, network packets are defined as simple events and the detected attacks are the resulting complex events. The successful deployment of CEP engines in IoT environments has been widely demonstrated [13–15]. Although CEP is very advantageous for real-time attack detection, it has a limitation, namely the need for a domain expert who is able to define the rules that must be followed to carry out such detection.

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

To evaluate this proposal, a baseline scenario is deployed on MQTT and attacked with six different attacks. Using these attacks, two different experiments based on an evolving scenarios were attacks are added iteratively are performed. In the first one a new attack is presented for the first time in each iteration which, if detected, is then used to retrain and improve the model to be tested in the next iteration. In this way we can observe how the system can detect anomalies and generate rules to detect them by adding new families. In the second experiment the attacks are distributed uniformly over the iterations, so that in each iteration both new attacks and new packets of already known attacks appear. This experiment is useful to check how existing rules evolve while generating new attack families in the same iteration.

The main contributions of this paper are the following:

- The integration of PCA with GMM and Mahalanobis distance for the first time in a CEP engine, which allows us to generate CEP rules in an unsupervised manner.
- The generated CEP rules allow the CEP engine to be able to detect attacks in IoT environments in real time.
- The use of dynamic tables makes it possible to generate new rules very easily without the need to modify the CEP application in real time.
- Our proposed framework is able to, from an initial state where it has only been exposed to normal traffic, detect unseen attacks as anomalies and progressively and incrementally incorporate them in the rule set.
- The architecture has been successfully evaluated using a MQTT network use case using two different experimental scenarios and achieving an average F1 score $> 0.9890\%$.

The remainder of the article is structured as follows. Section 2 describes the concepts necessary to understand the whole article. Section 3 discusses related works on generating CEP patterns automatically. Subsequently, Section 4

describes the design and implementation of the proposal. The results are described and discussed in Section 5. Lastly, conclusions and future work are presented in Section 6.

2 Background

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

2.1 MQTT protocol

MQTT is a protocol that operates at the application layer and is supported by TCP/IP. It is oriented to network communication through a publisher/-subscriber scheme using topics. In this way, devices (clients) that require information subscribe to the corresponding topic. The clients that generate this information publish in that topic. There is a central node called broker that is in charge of orchestrating the behavior of the network, receiving the packets and forwarding them to the corresponding nodes. This protocol is especially useful in IoT networks because it is especially lightweight, which has made it very popular within the IoT paradigm [16].

2.2 Complex event processing

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

More specifically, a CEP engine is a specific software used to perform this type of data processing in real time. In our case, Siddhi CEP [17] is used.

The language used to define the rules in a CEP engine is called Event Processing Language (EPL). There is a plenty of EPLs, specifically SiddhiQL is the one provided by Siddhi CEP.

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

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

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

3 Related work

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

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

3.1 Supervised with prior rules

In this group we find proposals that require labeled training datasets and prior rules and aim to update existing patterns. This makes it possible to generate new rules that offer better results than the original ones. A work that fits in this category is the one proposed by Yunhao Sun et al. [18]. In this work, a historical set of training data and CEP rules are used. First, a loss function is used, which is obtained from the error of the previous rule measurements with respect to the actual labelled results. A loss function and an activation function are used to filter out rules that are considered bad based on a manually defined threshold. Using the remaining rules, a given set of support vectors is determined to build a coverage region for each class. Finally, updated rules are created by the projection of regional boundary This work is interesting although it addresses the problem of updating rather than generating new rules.

In the work proposed by Nathan Tri Luong et al. [19], CEP is used to preprocess the data , while Tensor Flow is used to implement and additional component that performs the training and classifications of the different events. In this type of approach, CEP rules only perform the processes prior to training and classification. The limitation of this architecture is that the bottleneck can be transferred to the component in charge of performing the classifications. This results in not taking full advantage of the capacity of CEP engines to process a large amount of data.

3.2 Supervised without prior rules

In this group we find proposals that do not require prior rules, but label complex events based on historical data. A paper in this category is that of Bruns et al. [20]. This paper succeeds in adapting the bat algorithm to the CEP rule search by structuring the different CEP operators, attribute values and time windows in the form of a tree. In this way the algorithm determines these values in the rule they represent. The results they obtain average an F1 score of 0.9923, and are achieved using an unusual algorithm in the CEP context. The only limitation of the proposal, in addition to required labeling for training,

is that it needs a definition of the complex events as a function of the simple events. This is not always easy without prior rules.

Another work that manages to extract rules automatically without prior rules is the one proposed by Roldán-Gómez et al. [21]. In this case, the rules are constructed from the prediction of the value of the most important feature for a category. If the difference between the actual value and the prediction exceeds a threshold, this simple event does not correspond to a category. This article is able to detect all attacks, although the main limitation of this work is the difficulty that may exist in generating certain rules based only on a key variable and an expected value.

A natural evolution of the previous work is seen in the work of Roldán-Gómez et al. [22]. In this work authors reduce the dimensions of individual events using Principal Component Analysis (PCA), thereby achieving two goals. The first is to simply characterize the individual events, the second is to drastically improve the performance of the CEP engine and the system network by reducing the dimension of the individual events. From the labels of the individual events, the averages of the reduced events are calculated. The rule consists of a Euclidean distance weighted by the weights of each component of the reduced event. This difference is compared with the sum of errors of each component weighted again with the weights of each component and with the standard deviation of each component. The results obtained from this study show an average F1 score of 0.9878, in addition to the reduction of the event size and the consequent improvement of the performance of the network and the CEP engine. A small limitation of this work is that it is a supervised way to calculate the rule for each category.

3.3 Unsupervised with prior rules

This group is the least common as it requires unsupervised training and the existence of rules capable of detecting items of interest. However, we can find works such as the work of Ren et al. [14]. This one focuses on optimizing performance in IoT environmentsas main differentiation factor from other proposals. To achieve this goal, a micro CEP engine and a model based on Tensorflow Lite Micro with pre-trained neural networks are used. These neural networks (either supervised or unsupervised such as autoencoders) can be updated to adapt to the changing behavior of a real system. The main change of this proposal with respect to the others analyzed is that the output of these neural networks feeds the CEP engine, which has manually defined rules. It may seem that this proposal does not fall within the scope of automatic CEP rule generation. However, it is possible to generate simple rules that detect the output of neural networks. The main limitation of this proposal is that the CEP rules are defined manually, unlike our proposal in which they are generated automatically.

Reference	Unsupervised	Need for prior rules	Novelty / Highlight
[18]	No	Yes	Pre-filtering rules before training improves performance.
[19]	No	Yes	It uses CEP to perform data preprocessing.
[20]	No	No	It uses Bat algorithm to generate new rules.
[21]	No	No	It compares the prediction of key features with their actual values.
[22]	No	No	PCA allows rules to be generated with high performance.
[14]	Yes	Yes	Defined CEP rules and pretrained neural networks will generate efficient CEP rules.
[23]	Yes	No	It uses GRU and Furia to generate CEP rules in an unsupervised manner
This work	Yes	No	PCA and GMM enable unsupervised generation of high-performance CEP rules

Table 1 Comparison of the works analyzed.

3.4 Unsupervised without prior rules

In this group we find proposals that do not require prior rules or labels on the data. Some works mainly focus on labeling simple events and then use known rule extraction algorithms. The work by Simsek et al. [23] performs a study using different classifiers to label simple events, then uses the most common algorithms for rule extraction. Their conclusions show that Gated Recurrent Unit (GRU) together with the FURIA algorithm obtain the best results in their experiments. The value of this work lies in the comparison made with different algorithms. The fundamental disadvantages of this approach lie in the large amount of data required for deep learning models and the computational cost involved.

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

4 Proposed architecture

This section describes the architecture for recognizing real-time IoT attack patterns. Figure 1 illustrates a graphical scheme of the architecture. Our proposal focuses on the automatic CEP rule generator and the training data is obtained from the IoT network. As discussed earlier these packets are not labeled and feed the CEP rule generator.

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

These phases are discussed in detail below.

4.1 Preprocessing

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

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

4.2 PCA phase

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

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

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

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

To implement our proposal we use incremental PCA [24], this version allows to recalculate the model, i.e. the new eigenvectors W_{n+1} if new data are added, using the existing eigenvectors W_n with their corresponding eigenvalues and covariance matrix of the current PCA model, plus the new samples X_{n+1}. In this way it is not necessary to generate a new model from scratch, and/or to store the previous samples X_n in memory, if new training data arrives. Instead, it is possible to obtain an estimate of the $n+1$ iteration using the eigenvectors and eigenvalues of n. This makes it possible to obtain new PCA models incrementally from already trained PCA models. The advantage of this is that we do not have to train from scratch the PCA model in each iteration, thus achieving a lighter training in new iterations.

Once the trained PCA model is obtained, it is sent to the IoT network broker, this model is also used to reduce the input traffic. We also extract the variance ratios explained in each component, and we obtain the diagonal matrix of them that will be further used for thresholding purposes in Section 4.4. This reduction is necessary for the following stages.

4.3 GMM phase

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

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

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

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

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

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

On the other hand $\mathcal{N}(x|\mu_k, \Sigma_k)$ is called the mixture model component, it models and describes each of the normal distributions, μ_k is the mean and Σ_k is the covariance.

The main advantage of GMM is that it allows some flexibility in each category, so that 2 normal distributions can be very different, and it does not have a bias for circular groups and works well even in certain non-linear distributions [25].

In this case a variational version of the algorithm is used [26], which allows to infer an optimal number of normal distributions. The objective of using this

version is not to have to indicate the number of K families a priori, this allows the process to be completely unsupervised, since we do not need to know a priori how many families or grouping are composing the normal traffic and how many different types of attacks we may be exposed to.

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

GMM has to be recalculated with training data from previous iterations on the new PCA model [26]. This is because each iteration modifies the PCA model, this causes the original distributions to be useless in the new model.

4.4 Threshold phase

At this stage, the threshold is calculated for each family k using the Mahalanobis distance. The Mahalanobis distance is a distance function that takes into account the covariance matrix to weight it [27]. The fundamental advantage of the Mahalanobis distance is that it takes into account the scale differences that may exist between the different variables and families as well as the correlation that may exist between variables.

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

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

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

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

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

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

$$VE = diag(rv_1, rv_2, ..., rv_m) \tag{6}$$

Using the Eq. 5, each element is compared to the mean of each family. Once we have all the distances we can calculate the threshold for that family, using

the farthest element of the family with respect to the mean and the closest non-family element with respect to the family mean. With these distances we calculate the midpoint, which defines the threshold for that category k.

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

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

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

Equation 7 and 8 define how to obtain the element farthest from the mean of a family k and the closest outside the family k respectively. With these elements, obtaining the threshold is simple as we can see in equation 9.

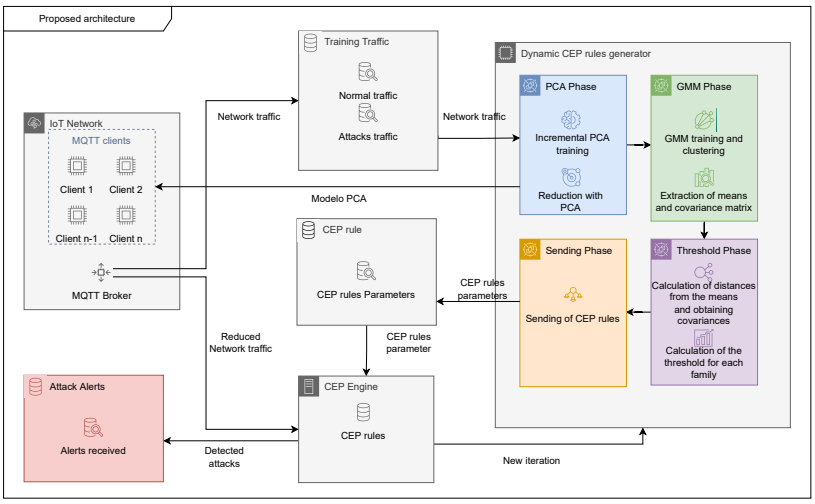


Fig. 1 Diagram of the proposed architecture to detect IoT attacks in real time.

4.5 Sending phase

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

Regarding the operation of dynamic CEP rules, when the CEP rule generator generates new rules, it is not necessary to generate a new Siddhi file,

which is used to generate an application in the Siddhi engine, instead it makes use of dynamic tables containing the parameters of the current rules. This implementation advantage reduces the network data transfer when updating or generating new rules, and greatly facilitates the implementation, creation and update.

Once in operation, the broker reduces the packets with PCA and sends them to the CEP engine. With these reduced packets, the distance of the same packet with respect to the average of each family is calculated with Eq. 5, if this distance is less than the threshold of that family, the packet is considered to belong to that family. In case a packet does not fall within the threshold of any family, that packet is considered to be an anomaly.

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

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

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

5 Experiments and results

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

The scenario we propose is an MQTT network with three legitimate clients and a broker. The clients generate numerical data and send it to the broker, this allows to simulate a temperature sending scenario. To demonstrate that unknown attacks are correctly detected and , different attacks have been implemented to demonstrate the correct operation of our proposal. The attacks are the following:

- *Subscription fuzzing*: This attack consists of trying to subscribe to different topics, it can be used when we have access to an MQTT system.

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

Table 2 Distribution of the dataset used.

- *Disconnection wave*: It consists of spoofing the *id* of the MQTT protocol and launching the disconnect command, if not configured correctly it is possible to steal the *id* of the legitimate device and expel it from the system. The goal of this attack is to disconnect all devices from the system.
- *TCP syn scan*: This is the classic scanner used to check which TCP ports are open. The attacker starts with a SYN packet. If he receives a SYN/ACK he assumes the port is open, if he receives an RST he assumes it is closed.
- *UDP scan*: This involves sending UDP packets to each port to be scanned, if a UDP response is received the port is considered open, if no response is received the position is open or filtered, a packet of type ICMP *port unreachable error* means that the port is closed and any other type of ICMP error means that the port is filtered.
- *Xmas scan*: This is a rather unusual scanner nowadays, however we use it in the scenario because it is different from the UDP and TCP SYN scanner. It involves sending to each TCP port a packet with the FIN, PSH and URG flags set to 1. If no response is received the port is considered open or filtered, if an RST is received it is considered a closed port, if any ICMP packet is received *unreachable error* it is considered a filtered port.
- *Telnet connection*: These are packages that try to connect via Telnet with different users and passwords, to simulate the first stage of Mirai. The idea is to test the proposal against a very usual scenario [28].

The training and testing datasets are generated by collecting the normal packets and the attacks. The dataset is accessible from the following repository https://data.mendeley.com/datasets/pzhm3jnw6w/draft?a=1565272f-bc8b-4eac-a566-11ec45124a44 [29]. The distribution of the dataset can be seen in Table 2. Each event is considered a separate attack in our experiments so that we can more accurately measure the effectiveness of the CEP rules.

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

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

Table 3 Data input at each iteration in experiment 1.

5.1 Experimental Scenario 1: Detecting new attacks

The first scenario seeks to demonstrate that the proposed architecture is capable of detecting new attacks in an unsupervised and incremental manner. The experiments of this first scenario are performed in several iterations. In the first iteration we train only with normal packets, since this would be the usual case when the architecture is deployed for the first time. However, please note that it is possible for our framework to do the first training with attack packets without problem. From the first iteration onwards, new attacks are introduced in each iteration and the model is retrained with the packets that have not been classified as belonging to any of the existing GMM families (i.e. their distance to all families if larger than the learned thresholds). by any previous rule. Please note that the predictions given by our system are used in the following iterations for retraining and not the real groups, in order to preserve the unsupervised setting and not to require annotated groundtruth by human experts. This means that high misclassification could potentially lead to contamination of the exiting family models or creation of incorrect rules if the performance of the system was poor. This is applicable to both experiments.

An important detail to take into account is that the first time an attack is detected it will not be included in any CEP rule because it is an anomaly. With the subsequent training of the model with the new data, the new CEP rule will be generated. Table 3 shows the input of the different attacks in each iteration. Each row represents one type of traffic and each column one iteration of the experiment. The character X represents the testing dataset, the character A represents the training dataset, which is an anomaly in that particular iteration.The following conventional metrics are used to evaluate the results of these experiments:

- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1Score = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$

Where TP are the true positives, FP are the false positives and FN are the false negatives.

Thus a high *Recall* score means that a CEP rule detects many events that actually belong to that family, a high *Precision* score means that that CEP

	Iteration number								
Traffic type	1	2	3	4	5	6	7	8	9
Normal traffic	X_1	X_2	X_3	X_3	X_3	X_3	X_3	X_3	X_3
Subscription Fuzzing	A	X_1	X_2	X_3	X_3	X_3	X_3	X_3	X_3
Disconnection Wave		A	X_1	X_2	X_3	X_3	X_3	X_3	X_3
TCP Syn Scan			A	X_1	X_2	X_3	X_3	X_3	X_3
UDP Port Scan				A	X_1	X_2	X_3	X_3	X_3
Telnet					A	X_1	X_2	X_3	X_3
Xmas Scan						A	X_1	X_2	X_3

Table 4 Data input at each iteration in experiment 2.

rule does not detect many false positives. Finally *F1 Score* makes use of the two scores to obtain a balanced metric between the two.

Table 5 shows the results of the first experiment scenario. We can observe how the proposal behaves when new attacks appear and how it converges to a system capable of detecting and correctly classifying the packets of the different attacks. An important detail that we can verify in the fifth iteration is that a single rule is generated to detect UDP and TCP scans, this is because they present a very similar behavior and characterization. This unification of both attacks demonstrates the capacity of the generated rules to classify attacks based on their behavior. An important detail, which highlights the system's ability to detect anomalies, is that the first time a new attack is sent, there are no CEP rules for that attack.

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

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

In the second experiment the testing dataset X is divided into 3 datasets X_1, X_2 and X_3. These datasets have the same size, that is one third of the size of the testing dataset X as shown in Table 2. The training dataset A of each type of traffic is similar to the previous experiment. Figure 4 shows the datasets entering each iteration. The training dataset A of each type of traffic is detected as an anomaly, as in experiment 1. The novelty of this scenario is that this training now continues with the first and second testing datasets (X_1 and X_2) that go on to train the model once they have been detected by the CEP rules. The third testing set of each dataset X_3 never trains the model.

Iteration Number	Traffic type	TP	FP	TN	FN	Precision	Recall	F1 score
1	Normal Traffic	7780	0	3277	156	1	0.9893	0.9900
1	Subscription Fuzzing (Anomaly)	3277	156	7780	0	0.9545	1	0.9767
2	Normal Traffic	7936	0	3820	0	1	1	1
2	Subscription Fuzzing	819	0	10936	1	1	0.9987	0.9993
2	Disconnection Wave (Anomaly)	3000	1	8755	0	0.9996	1	0.9998
3	Normal Traffic	7780	0	18721	156	1	0.9803	0.9900
3	Subscription Fuzzing	819	0	25837	1	1	0.9987	0.9993
3	Disconnection Wave	16999	0	9657	1	1	0.9999	0.9999
3	TCP Syn Scan (Anomaly)	901	158	25598	0	0.8508	1	0.9193
4	Normal Traffic	7780	0	18451	156	1	0.9803	0.9900
4	Subscription Fuzzing	819	0	25567	1	1	0.9987	0.9993
4	Disconnection Wave	16999	0	9387	1	1	0.9999	0.9999
4	TCP Syn Scan	101	1	26285	0	0.9901	1	0.9950
4	UDP Port Scan (Anomaly)	530	157	25700	0	0.7714	1	0.8709
5	Normal Traffic	7780	0	18432	156	1	0.9803	0.9900
5	Subscription Fuzzing	819	0	25548	1	1	0.9987	0.9993
5	Disconnection Wave	16999	0	9368	1	1	0.9999	0.9999
5	TCP Syn + UDP Port Scan	160	2	26206	0	0.9876	1	0.9937
5	Telnet (Anomaly)	451	157	25759	1	0.7417	0.9977	0.8509
6	Normal Traffic	7780	0	18931	156	1	0.9803	0.9900
6	Subscription Fuzzing	819	0	26047	1	1	0.9987	0.9993
6	Disconnection Wave	16999	0	9867	1	1	0.9999	0.9999
6	TCP Syn + UDP Port Scan	160	1	26706	0	0.9937	1	0.9968
6	Telnet	51	0	26816	0	1	1	1
6	Xmas Scan (Anomaly)	900	157	25810	0	0.8514	1	0.9197
7	Normal Traffic	7780	0	18131	156	1	0.9803	0.9900
7	Subscription Fuzzing	819	0	25247	1	1	0.9987	0.9993
7	Disconnection Wave	16999	0	9067	1	1	0.9999	0.9999
7	TCP Syn + UDP Port Scan	160	1	25906	0	0.9937	1	0.9968
7	Telnet	51	0	26016	0	1	1	1
7	Xmas Scan	100	0	25967	0	1	1	1

Table 5 Results of the first experiment scenario.

This is done in order to be able to correctly evaluate the CEP rules at each iteration.

Table 6 shows the results of the second set of experiments. An average F1 score of 0.9938 was obtained, even slightly better than those obtained in the first scenario. The first detection of each attack is the most improved in this new scenario. These results seem to indicate that a training reinforcement for previously learned rules can improve the classification of CEP rules,while keeping the ability to add unseen attacks to the rule base.

Iteration Number	Traffic type	TP	FP	TN	FN	Precision	Recall	F1 score
1	Normal Traffic (Test 1)	2593	0	3277	52	1	0.9803	0.9900
1	Subscription Fuzzing (Training)	3277	52	2593	0	0.9843	1	0.9921
2	Normal Traffic (Test 2)	2592	0	3273	53	1	0.9799	0.9898
2	Subscription Fuzzing (Test 1)	273	0	5645	0	1	1	1
2	Disconnection Wave (Training)	3000	53	2865	0	0.9826	1	0.9912
3	Normal Traffic (Test 3)	2594	0	6840	51	1	0.9807	0.9902
3	Subscription Fuzzing (Test 2)	273	0	9212	0	1	1	1
3	Disconnection Wave (Test 1)	5665	0	3819	1	1	0.9998	0.9999
3	TCP SYN Scan (Training)	901	52	8532	0	0.9454	1	0.9719
4	Normal Traffic (Test 3)	2594	0	6502	51	1	0.9807	0.9902
4	Subscription Fuzzing (Test 3)	273	0	8874	0	1	1	1
4	Disconnection Wave (Test 2)	5665	0	3481	1	1	0.9998	0.9999
4	TCP SYN Scan (Test 1)	33	0	9114	0	1	1	1
4	UDP Port Scan (Training)	530	52	8565	0	0.9106	1	0.9532
5	Normal Traffic (Test 3)	2594	0	6443	51	1	0.9807	0.9902
5	Subscription Fuzzing (Test 3)	273	0	8815	0	1	1	1
5	Disconnection Wave (Test 3)	5665	0	3422	1	1	0.9998	0.9999
5	TCP SYN Scan (Test 2)+UDP Port Scan (Test 1)	52	0	9036	0	1	1	1
5	Telnet (Training)	452	52	8584	0	0.8968	1	0.9456
6	Normal Traffic (Test 3)	2594	0	6908	51	1	0.9807	0.9902
6	Subscription Fuzzing (Test 3)	273	0	9280	0	1	1	1
6	Disconnection Wave (Test 3)	5665	0	3887	1	1	0.9998	0.9999
6	TCP SYN Scan (Test 3)+UDP Port Scan (Test 2)	52	0	9501	0	1	1	1
6	Telnet (Test 1)	17	0	9536	0	1	1	1
6	Xmas Scan (Training)	900	52	8601	0	0.9453	1	0.9719
7	Normal Traffic (Test 3)	2594	0	6041	51	1	0.9807	0.9902
7	Subscription Fuzzing (Test 3)	273	0	8413	0	1	1	1
7	Disconnection Wave (Test 3)	5665	0	3020	1	1	0.9998	0.9999
7	TCP SYN Scan (Test 3)+UDP Port Scan (Test 3)	52	0	8634	0	1	1	1
7	Telnet (Test 2)	17	0	8669	0	1	1	1
7	Xmas Scan (Test1)	33	0	8653	0	1	1	1
8	Normal Traffic (Test 3)	2594	0	6041	51	1	0.9807	0.9902
8	Subscription Fuzzing (Test 3)	273	0	8413	0	1	1	1
8	Disconnection Wave (Test 3)	5665	0	3020	1	1	0.9998	0.9999
8	TCP SYN Scan (Test 3)+UDP Port Scan (Test 3)	52	0	8634	0	1	1	1
8	Telnet (Test 3)	17	0	8669	0	1	1	1
8	Xmas Scan (Test2)	33	0	8653	0	1	1	1
9	Normal Traffic (Test 3)	2594	0	6041	51	1	0.9807	0.9902
9	Subscription Fuzzing (Test 3)	273	0	8413	0	1	1	1
9	Disconnection Wave (Test 3)	5665	0	3020	1	1	0.9998	0.9999
9	TCP SYN Scan (Test 3)+UDP Port Scan (Test 3)	52	0	8634	0	1	1	1
9	Telnet (Test 3)	17	0	8669	0	1	1	1
9	Xmas Scan (Test3)	33	0	8653	0	1	1	1

Table 6 Results of the second experiment scenario.

Listing 1 Siddhi application for real-time IoT attack detection

```
App:name("DynamicPCAIncremental")

@App:description("Dynamic PCA Test")

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

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

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

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

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

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

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

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

from ThresholdParameters
select *
insert into ParametersTable;

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

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

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

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

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

6 Conclusions and future work

This paper proposed an architecture focused on the IoT paradigm that is capable of generating and updating CEP rules in an unsupervised manner to detect and classify network IoT attacks in real time without the need of a domain expert. The integration of CEP and PCA to reduce packet size makes the architecture optimal for IoT environments.

The rules generated by the proposed architecture work very well. The results obtained are very good (F1 score of 0.9890) generated in an unsupervised and incremental way, it can be deployed in all types of environments and learn constantly.

The architecture allows to detect unseen attacks and anomalies successfully, then these detected anomalies may be used to retrain the model, so the new attacks are progressively better defined, generated new CEP rules automatically and incrementally.

All these obtained conclusions demonstrate that our proposal can be successfully deployed in IoT environments with reduced constraints and generate dynamic unsupervised CEP rules that are able to detect network attacks in real time.

As future work, we plan to increase the number of protocols and attacks in experiments to test performance in other contexts. In addition, it is also interesting to create an ontology to classify new unknown attacks in predetermined families. Finally, it would be interesting to check if this architecture can be made robust against model poisoning attacks.

Funding

This work was supported by the Spanish Ministry of Science and Innovation and the European Union FEDER Funds [grant numbers FPU 17/02007 RTI2018-093608-B-C33, RTI2018-098156-B-C52 and PID2021-122215NB-C33]. This work was also supported by JCCM [grant numbers SB-PLY/17/180501/000353 SBPLY/21/180501/000195], and the Research Plan from the University of Cadiz and Grupo Energético de Puerto Real S.A. under project GANGES [grant number IRTP03_UCA].

References

[1] Langley, D.J., van Doorn, J., Ng, I.C.L., Stieglitz, S., Lazovik, A., Boonstra, A.: The Internet of Everything: Smart things and their impact on business models. Journal of Business Research **122**, 853–863 (2021). https://doi.org/10.1016/j.jbusres.2019.12.035. Accessed 2022-07-07

[2] Shilpa, A., Muneeswaran, V., Rathinam, D.D.K., Santhiya, G.A., Sherin, J.: Exploring the Benefits of Sensors in Internet of Everything (IoE). In:

2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 510–514 (2019). https://doi.org/10.1109/ICACCS.2019.8728530. ISSN: 2575-7288

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

[4] Asghari, P., Rahmani, A.M., Javadi, H.H.S.: Internet of Things applications: A systematic review. Computer Networks **148**, 241–261 (2019). https://doi.org/10.1016/j.comnet.2018.12.008

[5] Calvo, I., Merayo, M.G., Núñez, M.: A methodology to analyze heart data using fuzzy automata. Journal of Intelligent & Fuzzy Systems **37**(6), 7389–7399 (2019). https://doi.org/10.3233/JIFS-179348

[6] Sajid, M., Harris, A., Habib, S.: Internet of Everything: Applications, and Security Challenges. In: 2021 International Conference on Innovative Computing (ICIC), pp. 1–9 (2021). https://doi.org/10.1109/ICIC53490.2021.9691507

[7] Sadeeq, M.M., Abdulkareem, N.M., Zeebaree, S.R., Ahmed, D.M., Sami, A.S., Zebari, R.R.: IoT and cloud computing issues, challenges and opportunities: A review. Qubahan Academic Journal **1**(2), 1–7 (2021)

[8] Stoyanova, M., Nikoloudakis, Y., Panagiotakis, S., Pallis, E., Markakis, E.K.: A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues. IEEE Communications Surveys Tutorials **22**(2), 1191–1221 (2020). https://doi.org/10.1109/COMST.2019.2962586

[9] Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., Sikdar, B.: A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. IEEE Access **7**, 82721–82743 (2019). https://doi.org/10.1109/ACCESS.2019.2924045

[10] Mousavi, S.K., Ghaffari, A., Besharat, S., Afshari, H.: Security of internet of things based on cryptographic algorithms: a survey. Wireless Networks **27**(2), 1515–1555 (2021). https://doi.org/10.1007/s11276-020-02535-5

[11] Ferraz Junior, N., Silva, A., Guelfi, A., Kofuji, S.T.: IoT6Sec: reliability model for Internet of Things security focused on anomalous measurements identification with energy analysis. Wireless Networks **25**(4), 1533–1556 (2019). https://doi.org/10.1007/s11276-017-1610-2

[12] Corral-Plaza, D., Medina-Bulo, I., Ortiz, G., Boubeta-Puig, J.: A stream processing architecture for heterogeneous data sources in the Internet of

Things. Computer Standards & Interfaces **70**, 103426 (2020). https://doi.org/10.1016/j.csi.2020.103426

[13] Ortiz, G., Boubeta-Puig, J., Criado, J., Corral-Plaza, D., Garcia-de-Prado, A., Medina-Bulo, I., Iribarne, L.: A microservice architecture for real-time IoT data processing: A reusable Web of things approach for smart ports. Computer Standards & Interfaces **81**, 103604 (2022). https://doi.org/10.1016/j.csi.2021.103604

[14] Ren, H., Anicic, D., Runkler, T.A.: The synergy of complex event processing and tiny machine learning in industrial IoT. In: Proceedings of the 15th ACM International Conference on Distributed and Event-based Systems. DEBS '21, pp. 126–135. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3465480.3466928

[15] Roldán-Gómez, J., Boubeta-Puig, J., Pachacama-Castillo, G., Ortiz, G., Martínez, J.L.: Detecting security attacks in cyber-physical systems: a comparison of Mule and WSO2 intelligent IoT architectures. PeerJ Computer Science **7**, 787 (2021). https://doi.org/10.7717/peerj-cs.787

[16] Soni, D., Makwana, A.: A survey on mqtt: A protocol of internet of things(iot). (2017)

[17] Query Guide - Siddhi. https://siddhi.io/en/v5.1/docs/query-guide/ Accessed 2022-07-05

[18] Sun, Y., Li, G., Ning, B.: 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 (2020). https://doi.org/10.1109/ICDCS47774.2020.00176

[19] Luong, N.N.T., Milosevic, Z., Berry, A., Rabhi, F.: An open architecture for complex event processing with machine learning. In: 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC), pp. 51–56 (2020). https://doi.org/10.1109/EDOC49727.2020.00016

[20] Bruns, R., Dunkel, J.: Bat4CEP: a bat algorithm for mining of complex event processing rules. Applied Intelligence (2022). https://doi.org/10.1007/s10489-022-03256-2

[21] Roldán, J., Boubeta-Puig, J., Luis Martínez, J., Ortiz, G.: Integrating complex event processing and machine learning: An intelligent architecture for detecting IoT security attacks. Expert Systems with Applications **149**, 113251 (2020). https://doi.org/10.1016/j.eswa.2020.113251

[22] Roldán-Gómez, J., Boubeta-Puig, J., Castelo-Gómez, J.M., Carrillo-Mondéjar, J., Martínez, J.L.: Attack Pattern Recognition in the Internet of Things using Complex Event Processing and Machine Learning. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1919–1926 (2021). https://doi.org/10.1109/SMC52423.2021.9658711

[23] Simsek, M.U., Yildirim Okay, F., Ozdemir, S.: A deep learning-based CEP rule extraction framework for IoT data. The Journal of Supercomputing **77**(8), 8563–8592 (2021). https://doi.org/10.1007/s11227-020-03603-5

[24] Ross, D.A., Lim, J., Lin, R.-S., Yang, M.-H.: Incremental Learning for Robust Visual Tracking. International Journal of Computer Vision **77**(1), 125–141 (2008). https://doi.org/10.1007/s11263-007-0075-7

[25] Patel, E., Kushwaha, D.S.: Clustering Cloud Workloads: K-Means vs Gaussian Mixture Model. Procedia Computer Science **171**, 158–167 (2020). https://doi.org/10.1016/j.procs.2020.04.017

[26] Blei, D.M., Jordan, M.I.: Variational inference for Dirichlet process mixtures. Bayesian Analysis **1**(1), 121–143 (2006). https://doi.org/10.1214/06-BA104. Publisher: International Society for Bayesian Analysis

[27] De Maesschalck, R., Jouan-Rimbaud, D., Massart, D.L.: The Mahalanobis distance. Chemometrics and Intelligent Laboratory Systems **50**(1), 1–18 (2000). https://doi.org/10.1016/S0169-7439(99)00047-7

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

[29] Roldán-Gómez, J.: Dataset for An Automatic Unsupervised Complex Event Processing Rules Generation Architecture for Real-Time IoT Attacks Detection. Mendeley data (2022)

CHAPTER 8

Conclusions and Future Work

This chapter summarizes the conclusions drawn from this doctoral thesis and the lines for future research.

8.1 Conclusions

The potential of the IoT paradigm and its rapid growth has caught the attention of cybercriminals. This has led to a huge growth in both the number of connected IoT devices and threats targeting IoT devices. Moreover, this growth is not expected to slow down in the coming years, which inevitably means that this paradigm will require research and improvements in many aspects. For this reason, threat detection is the main focus of this thesis.

It is not easy to adapt classical solutions for addressing these challenges in this paradigm due to the limitations of the devices. It is therefore necessary to develop new solutions that can detect known and unknown threats and that can be deployed taking into account the limitations of the paradigm. To this end, different architectures have been designed, implemented and evaluated in this thesis, and different experiments have been carried out to ensure that they can operate effectively and efficiently in IoT environments.

Different goals have been established, and in order to make the conclusions more schematic, we will list the main conclusions we can draw from each goal. The first objective was to carry out a study of the state of the art, this being transversal to the realization of the Thesis. This is necessary to understand what solutions are adopted by other researchers. The conclusions drawn from this objective are the following:

- The technology and techniques used by this paradigm are constantly evolving, so it is necessary to carry out the process of analyzing the state of the art on an ongoing basis.

- The current trend is to use ML and DL techniques because signature-based techniques are not sufficient in a scenario with such constantly-changing and heterogeneous threats.
- It is essential to take into account the limitations of IoT devices, such as limited computing and memory capacity or low network bandwidth, when implementing any solution.

The second objective focuses on the creation of a test scenario for the validation of the different architectures that are generated. The following conclusions can be drawn:

- Since there are no public repositories that contain the attacks we describe, a dataset has been generated with a base scenario and different attacks.
- These attacks include attacks common to other paradigms (scanners), attacks that simulate known malware, and attacks designed specifically for the MQTT protocol.
- The dataset generated allows us to validate the different architectures in an IoT scenario.

Objective 3 consists of designing, deploying and validating an architecture to detect threats in IoT environments on the basis of one or more key features. This objective is achieved by applying a predictor based on linear regression to detect the expected value of these features. If these features do not fall within a threshold, that event does not belong to the family in question. In this way, with CEP rules we are able to detect anomalies.

The key points to highlight are as follows:

- The architecture works correctly and generates effective rules.
- This architecture requires a domain expert to choose the key features.
- The rules generated are capable of detecting events belonging to a category, but also anomalies when those events do not fall within a normal behavior or a known category.

The goal of Objective 4 is to check that the initial proposed architecture is compatible with another CEP engine to demonstrate that the generation of CEP rules is not limited to a specific engine, but also to check that the computational performance of these rules is adequate and which CEP engine is more efficient. This objective is achieved by comparing an architecture based on Mule ESB and the Esper CEP engine and another implementation based on WSO2 and the Siddhi CEP engine. Tests with realistic delays and tests in situations of stress are performed to check in which situations each CEP engine performs better.

The conclusions that can be drawn are as follows:

- Both engines have good computational performance.
- The Mule and Esper architecture is best when comparing different simple events, as is the case in the initial proposed architecture.

- The WSO2-based implementation is best when there is only a single event type, and it is much better in stressful situations and under high workloads.
- Avoiding the use of two different simple event types in the design of the following architecture can improve computational performance significantly.

Objective 5 is to for generate a new architecture that defines complete CEP rules for detecting attacks in real time. In this way, it is not necessary for a domain expert to make a selection of important features. In addition, PCA is used to reduce the size of single events and improve network usage and computational performance while facilitating the creation of CEP rules.

The main conclusions drawn from this objective are as follows:

- The CEP rules generated work well and a domain expert is not required to choose key features.
- The use of PCA in the creation of CEP rules greatly improves computational performance and decreases network usage.
- These rules also allow the detection of anomalies.
- Despite the automation offered by this architecture, it is still necessary to label the training datasets, i.e., this is supervised learning.

Objective 6 focuses on overcoming the limitation of the previous architecture, i.e., upgrading the previous architecture to be able to learn with unsupervised training. This allows rule generation without the need for a domain expert to be involved at any time.

The conclusions obtained from this objective are as follows:

- The rules generated work efficiently.
- Using GMM to generate anonymous attack families causes it to generate subfamilies of this attack that are not obvious to domain experts, and improves detection in most cases.
- These rules also allow the detection of anomalies.
- The use of PCA ensures that the performance improvements obtained in the previous implementation are maintained.

Therefore, all the objectives have been achieved in this work.

8.2 Conclusiones

El potencial del paradigma IoT y su rápido crecimiento han captado la atención de los ciberdelincuentes. Esto ha provocado un enorme crecimiento tanto del número de dispositivos IoT conectados como de las amenazas dirigidas a dispositivos IoT. Además, no se

espera que este crecimiento se ralentice en los próximos años, lo que inevitablemente significa que este paradigma requerirá investigación y mejoras en muchos aspectos. Por este motivo, la detección de amenazas es el tema principal de esta tesis.

No es fácil adaptar las soluciones clásicas para abordar estos retos en este paradigma debido a las limitaciones de los dispositivos. Por lo tanto, es necesario desarrollar nuevas soluciones que puedan detectar amenazas conocidas y desconocidas y que puedan desplegarse teniendo en cuenta las limitaciones del paradigma. Para ello, en esta tesis se han diseñado, implementado y evaluado diferentes arquitecturas, y se han llevado a cabo diferentes experimentos para asegurar que pueden operar de forma efectiva y eficiente en entornos IoT.

Se han establecido diferentes objetivos, y para que las conclusiones sean más esquemáticas, enumeraremos las principales conclusiones que podemos extraer de cada objetivo. El primer objetivo ha sido realizar un estudio del estado del arte, siendo este transversal a la realización de la Tesis. Esto es necesario para conocer las soluciones adoptadas por otros investigadores. Las conclusiones extraídas de este objetivo son las siguientes:

- La tecnología y las técnicas utilizadas por este paradigma están en constante evolución, por lo que es necesario llevar a cabo el proceso de análisis del estado del arte de forma continua.
- La tendencia actual es utilizar técnicas de ML y DL porque las técnicas basadas en firmas no son suficientes en un escenario con amenazas tan cambiantes y heterogéneas.
- Es fundamental tener en cuenta las limitaciones de los dispositivos IoT, como la limitada capacidad de computación y memoria o el escaso ancho de banda de la red, a la hora de implementar cualquier solución.

El segundo objetivo se centra en la creación de un escenario de pruebas para la validación de las diferentes arquitecturas que se generen. Se pueden extraer las siguientes conclusiones:

- Dado que no existen repositorios públicos que contengan los ataques que describimos, se ha generado un conjunto de datos con un escenario base y diferentes ataques.
- Estos ataques incluyen ataques comunes a otros paradigmas (scanners), ataques que simulan malware conocido y ataques diseñados específicamente para el protocolo MQTT.
- El conjunto de datos generado nos permite validar las diferentes arquitecturas en un escenario IoT.

El objetivo 3 consiste en diseñar, desplegar y validar una arquitectura de detección de amenazas en entornos IoT en base a una o varias características clave. Este objetivo se consigue aplicando un predictor basado en regresión lineal para detectar el valor esperado de estas características. Si estas características no caen dentro de un umbral, ese evento

no pertenece a la familia en cuestión. De este modo, con las reglas CEP podemos detectar anomalías.

Los puntos clave a destacar son los siguientes:

- La arquitectura funciona correctamente y genera reglas eficaces.
- Esta arquitectura requiere un experto en el dominio para elegir las características clave.
- Las reglas generadas son capaces de detectar eventos pertenecientes a una categoría, pero también anomalías cuando dichos eventos no entran dentro de un comportamiento normal o de una categoría conocida.

El objetivo 4 es comprobar que la arquitectura inicial propuesta es compatible con otro motor CEP para demostrar que la generación de reglas CEP no está limitada a un motor concreto, pero también comprobar que el rendimiento computacional de estas reglas es adecuado y qué motor CEP es más eficiente. Este objetivo se consigue comparando una arquitectura basada en Mule ESB y el motor CEP Esper y otra implementación basada en WSO2 y el motor CEP Siddhi. Se realizan pruebas con retardos realistas y pruebas en situaciones de estrés para comprobar en qué situaciones rinde mejor cada motor CEP.

Las conclusiones que se pueden extraer son las siguientes:

- Ambos motores tienen un buen rendimiento computacional.
- La arquitectura Mule y Esper es mejor cuando se comparan diferentes eventos simples, como es el caso de la arquitectura inicial propuesta.
- La implementación basada en WSO2 es mejor cuando sólo hay un único tipo de evento, y es mucho mejor en situaciones de estrés y bajo altas cargas de trabajo.
- Evitar el uso de dos tipos de eventos simples diferentes en el diseño de la siguiente arquitectura puede mejorar significativamente el rendimiento computacional.

El objetivo 5 es generar una nueva arquitectura que defina reglas CEP completas para la detección de ataques en tiempo real. De este modo, no es necesario que un experto en el dominio haga una selección de las características importantes. Además, se utiliza PCA para reducir el tamaño de los eventos individuales y mejorar el uso de la red y el rendimiento computacional, facilitando al mismo tiempo la creación de reglas CEP.

Las principales conclusiones extraídas de este objetivo son las siguientes:

- Las reglas CEP generadas funcionan bien y no se requiere un experto en el dominio para elegir las características clave.
- El uso de PCA en la creación de reglas CEP mejora en gran medida el rendimiento computacional y disminuye el uso de la red.
- Estas reglas también permiten detectar anomalías.

- A pesar de la automatización que ofrece esta arquitectura, sigue siendo necesario etiquetar los conjuntos de datos de entrenamiento, es decir, se trata de aprendizaje supervisado.

El objetivo 6 se centra en superar la limitación de la arquitectura anterior, es decir, mejorar la arquitectura anterior para que sea capaz de aprender con un entrenamiento no supervisado. Esto permite la generación de reglas sin necesidad de que intervenga en ningún momento un experto en el dominio.

Las conclusiones obtenidas de este objetivo son las siguientes:

- Las reglas generadas funcionan eficientemente.
- El uso de GMM para generar familias de ataques anónimos hace que genere subfamilias de este ataque que no son obvias para los expertos del dominio, y mejora la detección en la mayoría de los casos.
- Estas reglas también permiten detectar anomalías.
- El uso de PCA garantiza el mantenimiento de las mejoras de rendimiento obtenidas en la implementación anterior.

Por tanto, en este trabajo se han alcanzado todos los objetivos.

8.3 Future Work

Although we have addressed many problems during the development of this doctoral dissertation, there are still several factors that can be improved. We did not address these here as they are beyond the scope of the main work.

Firstly, the number of protocols and attacks could be increased in order to test the performance of the different architectures. This would make it possible to test a larger number of CEP rules and to have a more heterogeneous scenario.

The creation of heterogeneous scenarios could be a limitation for PCA. This algorithm tends to perform worse when it fails to find linear combinations between features to generate components. For this reason it would be interesting to make use of Kernel Principal Component Analysis (KPCA) [48]. This algorithm allows the generation of projections so that nonlinear distributions approach linearity. Checking whether this algorithm is able to work with our architecture would be very interesting. In addition, it would also be interesting to test other dimensionality reduction algorithms with the proposal.

Another interesting project would be the creation of an ontology that allows us to register the events detected with rules generated in an unsupervised way within general categories. In this way we could detect unknown attacks and also know what type of attack is involved.

The retraining process that the final proposed architecture is capable of performing could be targeted by model poisoning attacks if a malicious IoT device manages to enter the network. A useful improvement would be to design, implement and evaluate a mechanism that avoids this type of attacks in our architecture as far as possible.

Finally, another future line of work is to apply our architectures to other domains such as Industry 4.0. and smart cities.

Bibliography

[1] A. A. AlZubi, M. Al-Maitah, and A. Alarifi, "Cyber-attack detection in healthcare using cyber-physical system and machine learning techniques," *Soft Computing*, vol. 25, no. 18, pp. 12 319–12 332, Sep. 2021.

[2] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Internet of Things applications: A systematic review," *Computer Networks*, vol. 148, pp. 241–261, Jan. 2019.

[3] I. Calvo, M. G. Merayo, and M. Núñez, "A methodology to analyze heart data using fuzzy automata," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 6, pp. 7389–7399, Jan. 2019.

[4] M. Sajid, A. Harris, and S. Habib, "Internet of Everything: Applications, and Security Challenges," in *2021 International Conference on Innovative Computing (ICIC)*, Nov. 2021, pp. 1–9.

[5] "State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time," Nov. 2020. [Online]. Available: https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/

[6] "State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally," May 2022. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/

[7] R. Kollolu, "A Review on Wide Variety and Heterogeneity of IoT Platforms," Rochester, NY, Jan. 2020. [Online]. Available: https://papers.ssrn.com/abstract=3912454

[8] S. S. Dhanda, B. Singh, and P. Jindal, "Lightweight Cryptography: A Solution to Secure IoT," *Wireless Personal Communications*, vol. 112, no. 3, pp. 1947–1980, Jun. 2020. [Online]. Available: https://doi.org/10.1007/s11277-020-07134-3

[9] S. S. I. Samuel, "A review of connectivity challenges in IoT-smart home," in *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, Mar. 2016, pp. 1–4.

[10] A. Pop-Vadean, P. P. Pop, T. Latinovic, C. Barz, and C. Lung, "Harvesting energy an sustainable power source, replace batteries for powering WSN and devices on the IoT," *IOP Conference Series: Materials Science and Engineering*, vol.

200, no. 1, p. 012043, May 2017, publisher: IOP Publishing. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/200/1/012043

[11] F. Foresti and G. Varvakis, "Ubiquity and Industry 4.0," in *Knowledge Management in Digital Change: New Findings and Practical Cases*, ser. Progress in IS, K. North, R. Maier, and O. Haas, Eds. Cham: Springer International Publishing, 2018, pp. 343–358. [Online]. Available: https://doi.org/10.1007/978-3-319-73546-7_21

[12] A.-T.-T. I. I.-S. Institute, "AV-ATLAS." [Online]. Available: https://portal.av-atlas.org/

[13] B. Mahesh, *Machine Learning Algorithms -A Review*, Jan. 2019.

[14] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security?" *IEEE Signal Processing Magazine*, vol. 35, no. 5, pp. 41–49, Sep. 2018.

[15] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, and M. Guizani, "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security," *arXiv preprint arXiv:1807.11023*, 2018.

[16] G. Xiaoyan, "Research on Power IoT Intrusion Detection Method Based on Federated Learning," *Smart Innovation, Systems and Technologies*, vol. 299, pp. 183–190, 2023, iSBN: 9789811922541.

[17] B. Kumar, G. Rampalli, P. Kamakshi, and T. Senthil Murugan, "DDoS Botnet Attack Detection in IoT Devices," *Lecture Notes in Networks and Systems*, vol. 396, pp. 21–27, 2023, iSBN: 9789811699665.

[18] S. Kaura and D. Bhardwaj, "A Comprehensive Review on Intrusion Detection in Edge-Based IoT Using Machine Learning," *Lecture Notes on Data Engineering and Communications Technologies*, vol. 131, pp. 615–624, 2023.

[19] R. Almarshdi, L. Nassef, E. Fadel, and N. Alowidi, "Hybrid Deep Learning Based Attack Detection for Imbalanced Data Classification," *Intelligent Automation and Soft Computing*, vol. 35, no. 1, pp. 297–320, 2023.

[20] C. Catalano, L. Paiano, F. Calabrese, M. Cataldo, L. Mancarella, and F. Tommasi, "Anomaly detection in smart agriculture systems," *Computers in Industry*, vol. 143, 2022.

[21] P. Anand, Y. Singh, H. Singh, M. Alshehri, and S. Tanwar, "SALT: transfer learning-based threat model for attack detection in smart home," *Scientific Reports*, vol. 12, no. 1, 2022.

[22] Y. Kayode Saheed, A. Idris Abiodun, S. Misra, M. Kristiansen Holone, and R. Colomo-Palacios, "A machine learning-based intrusion detection for detecting internet of things network attacks," *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 9395–9409, 2022.

[23] H. Ren, D. Anicic, and T. A. Runkler, "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*, ser. DEBS '21. New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 126–135.

[24] Y. Sun, G. Li, and B. Ning, "Automatic Rule Updating based on Machine Learning in Complex Event Processing," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, Nov. 2020, pp. 1338–1343.

[25] N. N. T. Luong, Z. Milosevic, A. Berry, and F. Rabhi, "An open architecture for complex event processing with machine learning," in *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, Oct. 2020, pp. 51–56.

[26] R. Bruns and J. Dunkel, "Bat4CEP: a bat algorithm for mining of complex event processing rules," *Applied Intelligence*, Mar. 2022. [Online]. Available: doi.org/10.1007/s10489-022-03256-2

[27] J. Boubeta-Puig, G. Ortiz, and I. Medina-Bulo, "MEdit4CEP: A model-driven solution for real-time decision making in SOA 2.0," *Knowledge-Based Systems*, vol. 89, pp. 97–112, Nov. 2015.

[28] J. Rosa-Bilbao and J. Boubeta-Puig, "Model-Driven Engineering for Complex Event Processing: A Survey," *The Journal of Object Technology*, vol. 21, no. 4, pp. 1–13, 2022.

[29] EsperTech, "Esper," http://www.espertech.com/esper/, 2021, avaliable at: http://www.espertech.com/esper/ (accessed 9 May 2021).

[30] WSO2, "WSO2 | The Open Source Technology for Digital Business," https://wso2.com/, 2021, avaliable at: https://wso2.com/ (accessed 9 May 2021).

[31] "Chapter 5. EPL Reference: Clauses," avaliable at: http://siddhi.io/(accessed 12 Nov 2022). [Online]. Available: https://esper.espertech.com/release-5.2.0/esper-reference/html/epl_clauses.html

[32] WSO2, "Siddhi," http://siddhi.io/, 2021, avaliable at: http://siddhi.io/(accessed 9 May 2021).

[33] A. M. Martinez and A. C. Kak, "PCA versus LDA," *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[34] E. Patel and D. S. Kushwaha, "Clustering Cloud Workloads: K-Means vs Gaussian Mixture Model," *Procedia Computer Science*, vol. 171, pp. 158–167, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920309820

[35] M. Artac, M. Jogan, and A. Leonardis, "Incremental pca for on-line visual learning and recognition," in *2002 International Conference on Pattern Recognition*, vol. 3. IEEE, 2002, pp. 781–784.

[36] J. Roldán-Gómez, J. Boubeta-Puig, G. Pachacama-Castillo, G. Ortiz, and J. L. Martínez, "Dataset for detecting security attacks in cyber-physical systems: A comparison of mule and wso2 intelligent iot architectures," https://data.mendeley.com/datasets/fvb9pp5xsh/draft?a=a465700a-55b5-48a7-96fa-c8d27c0c772d, 2021.

[37] J. Roldán, J. Boubeta-Puig, J. Luis Martínez, and G. Ortiz, "Integrating complex event processing and machine learning: An intelligent architecture for detecting IoT security attacks," *Expert Systems with Applications*, vol. 149, p. 113251, Jul. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420300762

[38] J. Roldán-Gómez, J. Boubeta-Puig, G. Pachacama-Castillo, G. Ortiz, and J. L. Martínez, "Detecting security attacks in cyber-physical systems: a comparison of mule and wso2 intelligent iot architectures," *PeerJ Computer Science*, vol. 7, p. e787, 2021.

[39] J. Roldán-Gómez, J. Boubeta-Puig, J. M. Castelo Gómez, J. Carrillo-Mondéjar, and J. L. Martínez Martínez, "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)*, Oct. 2021, pp. 1919–1926, iSSN: 2577-1655.

[40] J. Roldán-Gómez, J. M. del Rincon, J. Boubeta-Puig, and J. L. Martınez, "Hacia la creacion de reglas cep no supervisadas para la deteccion en tiempo real de ataques en entornos iot," in *2022 Jornadas Nacionales de Investigación en Ciberseguridad (JNIC)*. JNIC, Jun. 2022, pp. 147–154, iSSN: 978-84-88734-13-6.

[41] D. M. Blei and M. I. Jordan, "Variational inference for Dirichlet process mixtures," *Bayesian Analysis*, vol. 1, no. 1, pp. 121–143, Mar. 2006, publisher: International Society for Bayesian Analysis. [Online]. Available: https://projecteuclid.org/journals/bayesian-analysis/volume-1/issue-1/Variational-inference-for-Dirichlet-process-mixtures/10.1214/06-BA104.full

[42] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The Mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, pp. 1–18, Jan. 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743999000477

[43] J. M. Castelo Gómez, J. Roldán Gómez, J. Carrillo Mondéjar, and J. L. Martínez Martínez, "Non-volatile memory forensic analysis in windows 10 iot core," *Entropy*, vol. 21, no. 12, 2019. [Online]. Available: https://www.mdpi.com/1099-4300/21/12/1141

[44] J. M. Castelo Gómez, J. Carrillo Mondéjar, J. Roldán Gómez, and J. L. Martínez Martínez, "A context-centered methodology for IoT forensic investigations," *International Journal of Information Security*, Nov. 2020. [Online]. Available: https://doi.org/10.1007/s10207-020-00523-6

[45] J. M. Castelo Gómez, J. Carrillo Mondéjar, J. Roldán Gómez, and J. Martínez Martínez, "Developing an iot forensic methodology. a concept proposal," *Forensic Science International: Digital Investigation*, vol. 36, p. 301114, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666281721000081

[46] J. Carrillo-Mondéjar, J. M. Castelo-Gómez, J. Roldán-Gómez, and J. L. Martínez, "An instrumentation based algorithm for stack overflow detection," *Journal of Computer Virology and Hacking Techniques*, vol. 16, no. 3, pp. 245–256, Sep 2020. [Online]. Available: https://doi.org/10.1007/s11416-020-00359-7

[47] J. Carrillo-Mondejar, J. M. Castelo Gomez, C. Núñez-Gómez, J. Roldán Gómez, and J. L. Martínez, "Automatic Analysis Architecture of IoT Malware Samples," *Security and Communication Networks*, vol. 2020, p. 8810708, Oct. 2020, publisher: Hindawi. [Online]. Available: https://doi.org/10.1155/2020/8810708

[48] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Artificial Neural Networks — ICANN'97*, ser. Lecture Notes in Computer Science, W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, Eds. Berlin, Heidelberg: Springer, 1997, pp. 583–588.