




A flow-based intrusion detection framework for internet of things networks

Leonel Santos^{1,2}  · Ramiro Gonçalves^{3,4} · Carlos Rabadão^{1,2} · José Martins^{3,4,5}

Received: 28 October 2019 / Revised: 4 January 2021 / Accepted: 13 January 2021 / Published online: 10 February 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC part of Springer Nature 2021

Abstract

The application of the Internet of Things concept in domains such as industrial control, building automation, human health, and environmental monitoring, introduces new privacy and security challenges. Consequently, traditional implementation of monitoring and security mechanisms cannot always be presently feasible and adequate due to the number of IoT devices, their heterogeneity and the typical limitations of their technical specifications. In this paper, we propose an IP flow-based Intrusion Detection System (IDS) framework to monitor and protect IoT networks from external and internal threats in real-time. The proposed framework collects IP flows from an IoT network and analyses them in order to monitor and detect attacks, intrusions, and other types of anomalies at different IoT architecture layers based on some flow features instead of using packet headers fields and their payload. The proposed framework was designed to consider both the IoT network architecture and other IoT contextual characteristics such as scalability, heterogeneity, interoperability, and the minimization of the use of IoT networks resources. The proposed IDS framework is network-based and relies on a hybrid architecture, as it involves both centralized analysis and distributed data collection components. In terms of detection method, the framework uses a specification-based approach drawn on normal traffic specifications. The experimental results show that this framework can achieve $\approx 100\%$ success and 0% of false positives in detection of intrusions and anomalies. In terms of performance and scalability in the operation of the IDS components, we study and compare it with three different conventional IDS (Snort, Suricata, and Zeek) and the results demonstrate that the proposed solution can consume fewer computational resources (CPU, RAM, and persistent memory) when compared to those conventional IDS.

Keywords Internet of things · Network monitoring · Intrusion detection · Network security · Network attacks

✉ Leonel Santos
leonel.santos@ipleiria.pt

Ramiro Gonçalves
ramiro@utad.pt

Carlos Rabadão
carlos.rabadao@ipleiria.pt

José Martins
jose.l.martins@inesctec.pt

² School of Technology and Management, Polytechnic of Leiria, Leiria, Portugal

³ INESC TEC (Formerly INESC Porto), Porto, Portugal

⁴ AquaValor – Centro de Valorização e Transferência de Tecnologia da Água, Chaves, Portugal

⁵ Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal

¹ Computer Science and Communication Research Centre, Polytechnic of Leiria, Leiria, Portugal

1 Introduction

IoT is a fairly recent model that empowers novel applications in different domains such as industrial control, building automation, health, and environmental monitoring, thus making the security and privacy of IoT data a key point of concern.

With the increasing number of devices being connected to the Internet, the network administration activities such as devices management, traffic monitoring and security, are a subject that must be on everyone's mind [1]. According to Bradley et al. [2] and Lee and Lee [3], by 2020 there will be fifty billion devices connected to the Internet, and each individual will own around seven devices. Also, with the speedy progress of IoT that is penetrating the mainstream, more and more IoT data is being transmitted over networks and Internet, thus making it more exposed to attacks and increasing the risk of cyber security attacks [4].

Sha et al. [5] have stated that designing specific security mechanism for IoT systems is far from simple because the IoT landscape is heterogenous, fragmented and not supportive of interoperability. Some security solutions such as a lightweight version of Datagram Transport Layer Security (DTLS) [6], Internet Protocol Security (IPsec) [7] or even the IEEE 802.15.4 link-layer security [8], have been proposed to improve data confidentiality, authentication, access control, trust, and privacy within IoT networks [9–12]. However, even with these mechanisms, IoT networks are still exposed to attacks and intrusions [13].

Zarpelao et al. [13], Santos et al. [14], and Hajiheidari et al. [15] highlight the necessity for the development of more IoT-directed security tools and argue that systems like Intrusion Detection System (IDS) could be used to address that necessity.

Regardless of the use of IDS technologies in conventional networks, existing solutions are inadequate to be applied to IoT networks as their architecture is not flexible enough against the complex and heterogenous IoT ecosystem [16]. The characteristics of IoT components, such as resource constraints, large scale, heterogeneity, preference of functions over security, higher privacy requirements, low-cost design, and harder trust management, make it extremely difficult to use conventional IDS solutions. As argued by Sha et al. [5], network architecture, scalability, heterogeneous devices and communications, integration with the physical world, resource constraints, privacy, the large scale, trust management and lesser preparation for security, are aspects that both explain and enforce the need for development of IDS for IoT.

To understand the context behind the development of IDS solutions for IoT networks, the existing literature on the topic has been analysed. According to Zarpelao et al.

[13] and Santos et al. [14], just a few of the existing works on IDS development were specifically focused on IoT systems. The referred authors have also stated that it is necessary to develop solutions that are able to: (a) defend against a wide range of attacks; (b) provide variety in detection technics; (c) address more IoT technologies; and (d) secure IDS alert traffic and management.

In order to contribute to the mitigation of the problems identified above, we propose a flow-based Intrusion Detection System (IDS) framework to protect IoT networks from external and internal threats in real time. The proposed framework was designed to consider both IoT networks architecture and other IoT contextual characteristics such as scalability and heterogeneity. The framework presented follows a network-based approach and follows a hybrid architecture as it involves both centralized and distributed components. Regarding the detection method, the proposed framework uses a specification-based approach. The collection of network data is made using probes in all IoT layers which are responsible for collecting the IP flows. Collected IP flows are forwarded to the central IDS components that are responsible for analysing them to detect attacks, intrusions, and other types of anomalies. The IDS components apply specification-based methods based on normal IP flows traffic specifications. The central IDS components can be placed on the IoT border router, on a dedicated machine or on a cloud-based system. The communications between the probes and the central IDS components are made over secure channels to avoid security and privacy issues. Finally, in our proposed framework, no software modification of IoT devices and application is required. To the best of our knowledge, this paper represents a novel approach as it encompasses an IP flow-based IDS to IoT networks.

The contributions and findings of the proposed work are summarized as follows:

- Proposal for a framework for intrusion detection in IoT systems for detecting internal or external intrusions in a timely manner.
- Proposal for a pattern of characteristics and specifications of IoT communications made through application protocols such as CoAP and MQTT.
- Development of a prototype to test and evaluate the proposed framework, simulating an IoT environment.

The rest of the document is organized as follows. Section 2 introduces some relevant terms regarding IoT and IoT security. Section 3 details the basic notions of flow-based network traffic monitoring systems, providing a revision of relevant terms and applications. Section 4 introduces relevant terms regarding IDS and present the most important IDS solutions for IoT. In Sect. 5, we describe our proposed design, including the architecture

and its main components and, in Sect. 6, the framework is evaluated. Finally, in the 7th section, we present a brief set of conclusions complemented with future work considerations.

2 The internet of all things

As Internet-related technology evolves, the very own concept behind the Internet has also changed. With the advent of the Internet of (all) Things (IoT), the concept of internet connectivity has been spreading to all types of electronic devices [4]. IoT simply means the interconnection of vast heterogeneous network frameworks and systems in different patterns of communication, such as human-to-human, human-to-thing, or thing-to-thing [17].

Moreover, IoT is a realm where physical items are consistently integrated to form an information network with the specific goal of providing advanced and smart services to users [18].

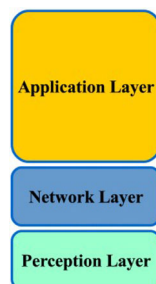
Although the application areas of IoT systems may have different goals, there is a set of characteristics and features that must be supported by all IoT systems [19], regardless of their application area, which are: (a) device heterogeneity; scalability; (b) ubiquitous data transmission; (c) energy optimization; (d) localization and tracking capability; (e) self-organizing ability; (f) data management and interoperability; (g) security; and (h) privacy.

2.1 IoT architecture

As the growing number of proposed IoT architectures fail to converge to a reference model [20], we can perceive the critical need for a flexible layered architecture.

Of the existing models, the basic model is a three-layer architecture like those proposed by Khan et al. [21], and Yang et al. [22], consisting on the layers of perception, network and application, as illustrated in Fig. 1. In recent literature, other models have been proposed, which tend to add more abstraction to the initial IoT architecture, such as Atzori et al. [4], Khan et al. [21], and Yang et al. [22].

Fig. 1 Three-layer architecture. Adapted from [17]



Considering, on the one hand, that in the existing literature the three-layer architecture is the most adopted [17] and, on the other hand, the simplicity of this proposal, the basis for our research was exactly this model. Conceptually, the three-layer model is composed of three layers which can be described as follows [23]:

- Perception layer: Also known as the sensory layer, its main objective is to collect data or act on the physical environment [4].
- Network layer: Also known as the transmission layer, its purpose is to transmit data between the perception layer and the application layer of IoT systems, that is, between sensors/actuators and services/users [24].
- Application layer: Also known as the business layer, it gives to applications the ability to process, manage and use data obtained at the perception layer about the physical environment [17].

At topology level Zegzhda and Stepanova [25] state that IoT systems could use the following three possibilities: a) point to point; b) star; or c) mesh.

2.2 IoT standards and challenges

Different alliances, consortiums, special interest groups, and standard development organizations have proposed a considerable amount of communication technologies for IoT, which generates a big challenge for end-to-end security in IoT applications [26].

Most popular technologies for IoT include infrastructure protocols like IEEE 802.11, IEEE 802.15.4, BLE, WirelessHART, Z-Wave, LoRaWAN, 6LoWPAN, DTLS and RPL, and application protocols like CoAP and MQTT.

Understanding IoT systems is not an easy task because there are many challenges that need to be addressed, such as availability, reliability, mobility, performance, scalability, interoperability, security, privacy, device management, and trust. Addressing these challenges allows service providers as well as application developers to deploy IoT services efficiently. Most of the identified challenges were addressed by the studies presented by Gubbi et al. [27], Gluhak et al. [28], Sheng et al. [29], Stankovic [30], Chen et al. [31] and Al-Fuqaha et al. [17]. In addition, there were some research and development projects such as IoT6, Ziegler et al. [20] which aimed to investigate IoT challenges and shortcomings and provide guidance for their solutions.

2.3 IoT security and threats

In cyber security, the Confidentiality—Integrity—Availability (CIA) triad is well known. Known, although only a small part of the existing literature relates CIA back to IoT.

Besides CIA, Lin et al. [23] adds more features to be addressed like Identification and Authentication, Privacy and Trust. Alaba et al. [16] outlined some security challenges in each layer of IoT architecture, therefore presenting the most common vulnerabilities and attacks.

In the perception layer, the main security threats are the forging of collected data and the destruction of perception devices by the following attacks: (a) node capture; (b) malicious code injection; (c) false data injection; (d) replay or freshness; (e) cryptanalysis and side channel; (f) eavesdropping and interference; and (g) sleep deprivation.

The network layer is faced with threats to the availability of network resources through multiple types of attacks, such as network scans, denial of service (DoS), jamming, spoofing, sinkholes, wormholes, man-in-the-middle (MITM), routing information, sybil and unauthorized access.

Finally, the main concerns of the application layer are software attacks like phishing attacks, malicious virus/worms, botnets, and malicious scripts.

In order to protect IoT systems from the security threads mentioned above, one can also use some conventional security countermeasures that have been implemented in conventional ICT systems such as applications, services, communications, cloud-based systems, among others [17]. Among these security countermeasures we can find solutions such as firewall, intrusion detection and prevention systems, authentication and authorization mechanisms, audit processes and data encryption application.

3 Flow-based monitoring

Network traffic monitoring and analyses represents a key component for network administration as it allows the development of several types of mechanisms, such as flow analysis, threats and anomalies detection, and performance monitoring.

3.1 Network flow monitoring

Network traffic monitoring approaches have been proposed and developed throughout the years. They can be classified as active or passive.

Whereas active approaches, such as the ones implemented by tools such as Ping, Traceroute, SNMP, and NETCONF, inject traffic into a network to perform different types of measurements and to perform analysis, passive approaches observe existing traffic as it passes by a measurement point and therefore observe and collect traffic generated by users and systems for being analysed [32].

Packet capture and flow export are common passive monitoring approaches. In the first, complete packets are captured providing deep insight into the traffic. This approach requires some hardware and infrastructure for storage and analysis.

Flow export aggregates packets into flows and exports them to a collector for storage and analysis. In our proposal, we follow the revised definition of flow proposed by Velan [33]. His definition is drawn on the Claise et al. [34] initial conceptualization.

If used in high-speed networks, this approach is more scalable and less costly than packet capture due to the integration of flow export protocols into network devices, such as routers, switches, and firewalls. Other advantages are the reduction of the amount of data stored, the possible use for forensic investigation, and the achievement of the privacy of the traffic data captured, since traditionally only packet headers are considered.

3.2 Flow monitoring architecture

The typical architecture of flow monitoring setups consists of several steps [35], each of which is explained below and are represented in Fig. 2.

Packet observation is the process of capturing packets from the line and pre-processing them for further use. This stage involves capturing packets from an observation point that is part of an observation domain.

The Flow Metering & Export stage is where packets are aggregated into flows and flow records are exported.

Flow records are defined in Claise et al. [34] as “*information about a specific flow that was observed at an observation point*”, which include flow keys, such as characteristic properties of a flow (e.g., IP addresses and port numbers), and measured properties (e.g., packet and byte counters).

In Data Collection, the flow collectors are an important step of flow monitoring setups, as they receive, store and pre-process flow data from one or more flow exporters.

The flow data storage format is an important characteristic of the flow data collecting stage because it defines the performance and functionality level of the flow collectors.

Data Analysis is the final stage in a flow monitoring setup. There are three main areas where analyses of flow data can be applied [36]: (a) Flow analysis & reporting; (b) Threat detection; and (c) Performance monitoring.

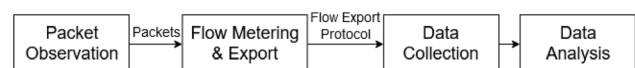


Fig. 2 Architecture of a flow monitoring setup. Adapted from [36]

The applicability statement of IPFIX issued by the IETF [37] and Li et al. [38] survey on network flow applications should also be considered for exploring more applications examples of flow data analysis.

4 IOT intrusion detection

4.1 Intrusion detection systems

According to Halme and Bauer [39], an Intrusion Detection System is “...an anti-intrusion approach that aims to discriminate intrusion attempts and intrusion preparation from normal system usage...”. A typical IDS is composed of sensors, an analysis engine and a reporting system. Sensors are positioned at different network places or hosts, and their main task is to collect data. The data collected is sent to the analysis engine, which is responsible for examining the collected data and detecting intrusions. If an intrusion is detected by the analysis engine, then the reporting system generates an alert to the network administrator.

Intrusion Detection Systems can be classified as Host-based IDS (HIDS) or Network-based IDS (NIDS). HIDS is attached to a device/host and monitors malicious activities occurring within the system. NIDS connects to one or more network segments and monitors network traffic for malicious activities [14].

IDS placement strategical approaches can be classified as distributed, centralized or hybrid [13]. In the centralised approach the entire IDS is placed in a central, either remote or host-based location. In the distributed strategy, the IDS nodes are placed in network nodes and the responsibility is divided amongst them. The hybrid placement strategy combines any strategy of the above and is often found in tandem with multiple detection types.

IDS detection methods approaches can be classified as signature-based, anomaly-based, specification based, or hybrid [13]. In signature-based approaches, IDS detect attacks when system or network behaviour matches an attack signature stored in the IDS internal databases. If a given system or network activity matches with stored patterns/signatures, then an alert will be triggered. This approach is accurate and very efficient at detecting known threats, and their mechanism is easy to understand. However, this approach is ineffective to detect new attacks and variations of known attacks, as a matching signature for these attacks is still unknown.

Anomaly-based IDSs compare the activities of a system at a given instant with a normal behaviour profile and generate the alert whenever a deviation from normal behaviour exceeds a threshold. This approach is efficient to detect new attacks, however, anything that does not match

to a normal behaviour is considered an intrusion and learning the entire scope of the normal behaviour is not a simple task.

Specification is a set of rules and thresholds that define the expected behaviour for network components such as nodes, protocols, and routing tables. Specification-based approaches detect intrusions when network behaviour deviates from specification definitions. Therefore, specification-based detection has the same purpose of anomaly-based detection: identifying deviations from normal behaviour. However, there is one important difference between these methods: in specification-based approaches, a human expert should manually define the rules of each specification. Manually defined specifications usually provide lower false positive rates in comparison with the anomaly-based detection.

Hybrid approaches will involve any combination of the above, whereby issues related to the efficacy of one technique is mitigated by the strengths of another.

Conventional intrusion detection systems use deep packet or state-full protocol inspection to detect intrusions or attacks. Deep packet inspection (DPI) techniques scan the packet header and examine its payload and filter the packet content searching for any attack traces [40]. Though, DPI is impractical for high-speed links [41] and inspection is not possible when the packet payload is encrypted.

In state-full protocol inspection (SPI), the semantics of the protocol are verified and any out of the range register is considered an intrusion or anomaly. On the downside, this technique is specific and cannot be used on unknown protocols. Finally, both techniques are computationally costly and could create a bottleneck [42, 43].

Considering both the limitations of the presented techniques, and considering the information presented above, an alternative solution for IoT networks against intrusions and attacks could be an IP flow-based IDS [1].

4.2 Flow-based IDS solutions

Flow-based intrusion detection is an active area of research and this type of systems are based on the generic IDS model proposed by Garcia-Teodoro et al. [44]. In recent years, some research has been published on this topic. Sperotto et al. [35] and Umer et al. [45] have published some of the most relevant research on this topic.

Sperotto et al. [35] made an overview of IP flow-based intrusion detection and their approach consists in summarizing each research made in this area according to the type of attack that could be detected. After presenting a list of the various categories of attacks the authors mentioned that since this type of intrusion detection only relies on the packet header information, it can only address a subset of

the attacks already identified. Despite their valuable effort, the authors have only reviewed approaches focused on detecting DoS, Scans, Worms, and Botnet attacks. Sperotto et al. [35] also point out some advantages and disadvantages of using this type of technique. The main advantages are that only flow records are analysed, the flow records contain aggregated information of packet headers, the traffic information is summarized in IP flows reducing the amount of data that needs to be processed, thus having a near real time response, low deployment cost, fewer privacy concerns and the fact that it can be used with encrypted protocols. As disadvantages the authors mention that since the IP flow records only contain generalized network information, it is therefore difficult to detect some attacks using just generalized information. As packet payload are not scanned, the detection of network attacks hidden in the packet payload is not so accurate as packet-based detection.

Umer et al. [45] provided a comprehensive analysis on the state of art of flow-based intrusion detection. Additionally, they proposed a taxonomy for flow-based IDS based on the techniques used for intrusion detection using flow records. The authors have also identified important challenges for future research in this area, such as the development of public flow-based datasets with a variety of attacks, the relationship between flow attributes and attack types, the evolving into Network Behaviour Analysis (NBA), among others. The same authors performed their research by revising all the identified literature according to the techniques that have been used. As a complement the authors have also reported some advantages and disadvantages for each main category. The main categories proposed were: (a) Statistical; (b) Machine Learning; and (c) Other Techniques.

Based on the works previously mentioned, several flow-based techniques using statistical and machine learning methods have been proposed with the aim of detecting malicious flows.

In Abuadlla et al. [46], the authors have proposed a two-stage neural network for intrusion detection using flow records. The first stage detects changes in the traffic that could be an attack. If an attack is detected, the flow data is forwarded to a second stage classifier which determines the type of attack. The technique has been assessed, and the first stage gives 94.2% detection rate and 3.4% false positive rate. For the second stage, best detection rate of 99.42% is also obtained with a false positive rate of 2.6%.

An improved nature-inspired technique for optimum-path forest clustering (OPFC) is proposed in Costa et al. [47]. The approach was evaluated, and results show that the optimum-path forest clustering outperforms k-means and SOM in flow-based detection.

A ward clustering approach to detect the dictionary attacks over SSH is presented by Satoh et al. [48]. The best results include a 99.90% detection rate for unsuccessful SSH attack attempts and 92.80% detection of successful SSH attempts.

Although there is extensive work in flow-based intrusion detection, our approach significantly differs from the existing work because the target is not a high-speed network and we do not want to use an in-line probe making use of a hardware-acceleration card with FPGA in order to reduce CPU load during packet capture and guarantee packet capture without loss under modest CPU [36]. Our goal is to design an IDS for IoT considering IoT features/characteristics such as architecture, scalability, heterogeneous devices and communications, integration with the physical world, resource constraints, privacy, the large scale, trust management, and less preparation for security.

4.3 IDS solutions for IoT

In recent years, several review articles have been published on IDSs for technologies related to IoT such as mobile ad hoc networks, wireless sensor networks, cloud computing, and cyber-physical systems. Although these articles primarily focus on the design of IDSs for several IoT related elements, only some of them (Zarpelao et al. [13], Santos et al. [14], and Hajiheidari et al. [15]) provide a study on IDS techniques specific for the IoT paradigm.

Nevertheless, there are some researchers that have already started to address IDS in the IoT context. In their research Liu et al. [49] proposed a signature-based IDS that employs Artificial Immune System mechanisms. Detectors with attack signatures were modelled as immune cells that can classify datagrams. The work does not indicate in which way this approach could be implemented in IoT networks and computational overhead needed to run learning algorithms might be a disadvantage.

On the other hand, Kasinathan et al. [50] proposed a centralized solution where their main objective is to detect DoS attacks in 6LoWPAN-based networks. In order to implement the IDS, the authors adapted to 6LoWPAN networks a known signature-based method, called Suricata. After considering their initial research, Kasinathan et al. [51] extended their work and reached a more complete set of assumptions.

Raza et al. [7] presented an IDS for IoT named SVELTE, whose objective is to detect sinkhole and selective forwarding attacks. This IDS had the participation of the border router and network nodes in the detection system. This work implements a hybrid approach to the detection method, trying to balance the computing cost of the anomaly-based method and the storage cost of the signature-based method. SVELTE was extended by

Shreenivas et al. [52] with an intrusion detection module that uses the ETX (Expected Transmissions) metric. Their experimental results show that compared with rank-only mechanisms the overall true positive rate increases.

A work presented by Jun and Chi [53] proposed the use of Complex Event-Processing (CEP) techniques for intrusion detection on IoT. The results achieved indicated that their approach was more CPU intensive, consumed less memory and took less processing time than traditional IDS.

Pongle and Chavan [54] proposed an IDS for IoT where network nodes must detect changes in their neighbourhood and must send information to centralized modules running in the border router. The results showed their solution is appropriate for IoT since its power and memory consumption are low.

Midi et al. [55] present an IDS for IoT called Knowledge-driven Adaptable Lightweight Intrusion Detection System (Kalis). The approach for detecting intrusions is based on the fact that Kalis is a self-adapting, knowledge-driven IDS for IoT systems running different communication protocols. Experimental tests have shown very good results on detection of DoS, routing and conventional attacks compared with traditional IDS.

Aloqaily et al. [56] introduced an intrusion detection mechanism to resist attack such as Denial of Service, Probe, Remote to user, etc. The authors proposed deep belief network for data dimension reduction, and decision tree using ID3 algorithm for intrusion classification. The detection accuracy of the systems is 99.92% which is quite high; however, the false negative rate is 1.53%.

Diro and Chilamkurti [57] proposed a deep learning-based and centralized method to detect the attacks in social IoT. Since fog nodes are closer to the smart infrastructure of social IoT, they are used to educate and maintain IDS at the edge of distributed fog networks. In the performance evaluation, the proposed method is evaluated over NSL-KDD dataset. High detection accuracy, online and low false positive rate are the upsides of the method and high training time and several resources usage in training are downsides of it.

Li et al. [58] proposed an AI-based two-stage IDS which detects an anomaly in the network by capturing network flows. In the first phase, the Bat algorithm with Swarm Division and Binary Differential Mutation are used to extract features of the network. In the second phase, the Random Forest is used as a classifier to classify the network flows. For evaluation, KDD Cup 1999 dataset has been used. Evaluation results have validated the optimality of the proposed algorithms in achieving high accuracy and low overhead. The disadvantage of the method is that it is not implemented and evaluated in real-world scenarios.

Deng et al. [59] studied and presented the current challenges of IoT network intrusion detection and

discussed the IoT architecture. In-depth, they studied and evident that using data mining and machine learning technique to solve the problem of anomaly and intrusion IoT network traffic identification is an excellent topic and proposed a new solution. The paper proposes a new intrusion detection scheme for Internet of things, that is, lightweight intrusion detection method combined with FCM algorithm and PCA algorithm. Simulation results show that the proposed method can improve the detection efficiency and make the false positive rate lower.

Gajewskiet al. [60] proposed an IDS for smart home systems where the local resource monitoring and preliminary log analysis was made to the Home Gateway device, whereas the processing of the long-term anomaly analysis of the user's behaviour is done at the ISP premises.

Pajouh et al. [61] presented an anomaly IDS built with Two-layer Dimension Reduction and Two-tier Classification (TDTC) for IoT Backbone. They concentrated mainly on most common low-frequency attacks while their experiments were based on NSL-KDD dataset. They started with Naive Bayes (NB) for anomaly detection then results are refined with Certainty Factor version of K-Nearest Neighbour (CF-KNN). Their work proved computation reduction with faster detection and less resource requirements. They achieved a detection rate of about 84.86% for binary classification with 4.86% of false alarm.

Siddiqui and Boukerche [62] have proposed a network-based intrusion detection method which learns patterns of normal flows in a temporal codebook. Based on the temporally learnt codebook, they proposed a feature representation method to transform the flow-based statistical features into more discriminative representations, called TempoCode-IoT. They developed an ensemble of machine learning-based classifiers optimized to discriminate the malicious flows from the normal ones. The effectiveness of the proposed method was empirically evaluated on a realistic dataset (CICIDS2017) as well as on a real botnet infected IoT dataset (NBaIoT), achieving high accuracies and low false positive rates across a variety of intrusion attacks.

Eskandari et al. [63] proposed Passban IDS. The proposed IDS is able to apply a protection layer on IoT devices which are directly connected to it. The attacks targeted by the system are Port Scanning, HTTP and SSH brute force and SYNflood. The system does not require intensive calculations and can be deployed also on cheap edge devices and/or IoT gateways. While the IDS aim to protect devices against a relatively low number of attacks, the system shows a very low false positive rate and high accuracy.

Regardless of the considerable evolution in the development of IDS solutions specifically designed for IoT networks [64], existing solutions still have numerous

limitations. Some solutions require considerable computational overhead or modification of IoT devices software which in a resource constraint environment is a disadvantage. Existing solutions do not guarantee the protection against more than one or two types of attacks, as well as not being able to handle more than one technology IoT. In addition, just a few solutions concern the security of IDS alert traffic and management. In sum, a new improved solution is needed to protect IoT networks from intrusions and attacks.

5 Proposed framework

5.1 System requirements

The vast array of IoT applications in diverse domains will require a wide coverage of the security in IoT infrastructures. A solution for this challenge will certainly include the ability to, in a timely manner, detect attacks and intrusions in IoT devices, communications, and applications.

As presented in the previous sections, there is a lack of systems and mechanisms designed specifically and able to guarantee security of IoT networks. With this challenge in mind, an effort has been made to define and present the set of requirements inherent to the design of an IDS solution specific aimed at IoT networks:

- Cross-layer detection: The IDS should be able to detect attacks at all layers of IoT architecture.
- External and internal intrusion detection: ability to detect intrusions originated at external hosts and at internal devices.
- Near real time detection: ability to detect intrusions within a reasonable time frame.
- Scalability: capability to accommodate a growing amount of analysis that results from the expansion of IoT network in terms of size or traffic.
- Interoperability and extensibility: support different communication channels and protocols, intrusion detection mechanisms, among others. Must be extensible to new standards, technologies, and intrusion types as they emerge.
- Reconfigurability: supports different intrusion policies throughout the IoT system lifetime.
- No software changes: minimization of the IDS footprint in the use of resources and software modification in IoT devices.
- No performance overhead: the IDS should not impact the performance of the IoT devices' applications.
- Protection of IDS communications: ensure security of communications between probes and IDS components.

The flow-based intrusion detection framework we describe in the following sections was designed to respond to the previous set of requirements. We start by analysing the system architecture of the proposed framework in what concerns its intrusion detection on IoT networks and their components.

5.2 Architecture and components

The design of intrusion detection solutions for IoT networks must make minimal use of the resources accessible in constrained IoT devices while, on the other hand, it could benefit from the availability of more resources in other types of devices, such as border routers or cloud-based systems.

Considering the requirements presented before, a solution able to ensure the non-modification of software on IoT devices and a minimal use of resources should be based on a hybrid placement strategy. For this type of environments, we argued that the most adequate solution is the use of a combination of a distributed capture of data alongside a centralized analysis of the network traffic data.

In Fig. 3 one can perceive the architecture of the proposed intrusion detection framework to be applied to IoT networks.

In order to minimize issues related to the demanding requirements for efficient detection capabilities adaptable to the various types of (internal and external) threats and to the wide variety of IoT technologies, in the proposed artefact the monitoring and capture of IoT communications will be done in the three layers of the IoT architecture (perception,

network, and application), as illustrated in Fig. 3. This three-layer solution ensures the detection of intrusions that may occur in any layer of an IoT application. The capture of these communications will be done using probes that will be placed in all layers. These probes locations provide a holistic view of the IoT application and could be a positive point to the detection of intrusions or attacks.

The various stages of the proposed IDS are the following (Fig. 4): capturing communications, exporting IP flows, collecting IP flows, and analysis of IP flows. In the proposed flow-based IDS framework, the probes are responsible for capturing the communications and converting them into IP flow records of the network section where they are present.

In the perception layer, the probes will act as dedicated devices and will be capturing the internal communications of the IoT networks. The internal IoT communications (made by sensors, actuators, among others), could be done using multiple technologies and protocols (eg.: 802.15.4, BLE, 802.11, etc.). The probes used in the perception layer must support the IoT technologies and protocols used and

Fig. 3 Architecture of the proposed flow-based IDS for IoT

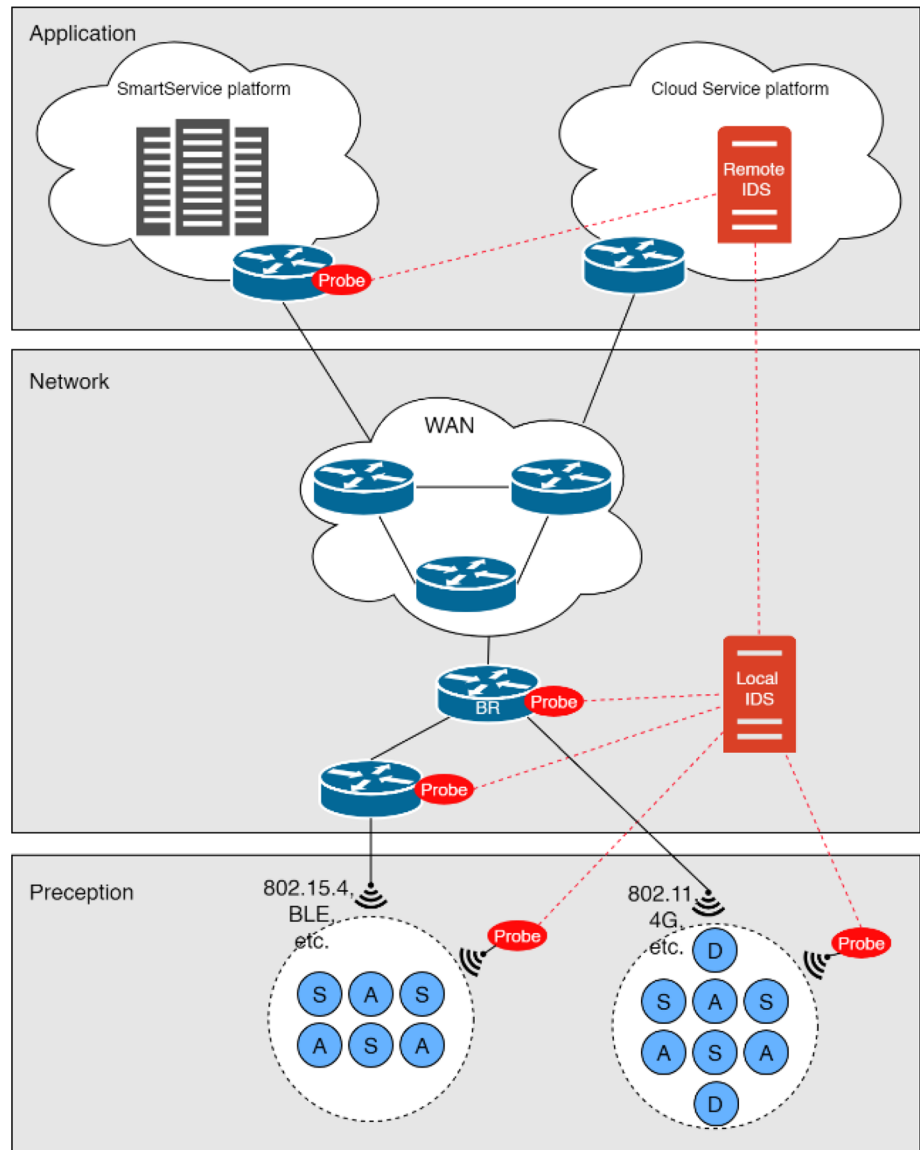
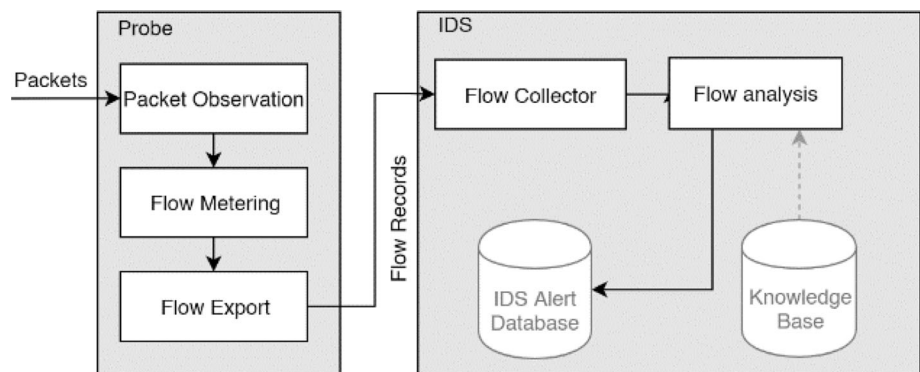


Fig. 4 Framework of proposed flow-based IDS for IoT



thereby ensure the technological coverage through minimal hardware adjustments.

The probes located in the network layer, will capture incoming and outgoing communications that pass through

the interfaces of routing devices. In these cases, probes functions may be supported by the firmware of the routing devices, thus avoiding the installation of any additional software. The probe responsible for capturing the

communications at the application layer will be installed in the routing device to which the IoT smart service will connect. The goal is capturing the communications made to this access point.

Once captured, all communications will be converted in IPFIX flow records and then exported to the IDS modules. The probes must be able to communicate near real time with the IDS modules enabling intrusion detection in useful time. The IPFIX flow records collected by the perception and network layers probes will be exported to a local IDS module. The IPFIX flow records collected by the application layer probes will be exported and forwarded to the remote IDS module.

All stored IPFIX flow records in the IDS modules will be analysed and if any intrusion is detected, an IDS alert message will be stored in the IDS alert database.

In terms of communications exchanged between the components of the proposed IDS framework, they must be preferably be made using hard-wired links, and be transmitted using an encrypted link (IPFIX over TLS over TCP), in order to ensure the security and privacy of IPFIX messages. In addition, communications between perception and network layers probes and local IDS module must use a dedicated VLAN for this type of communications, adding another layer of security to internal communications of the proposed IDS framework.

5.3 Detection methodology

With the purpose of detecting intrusions, our proposal to analyse IoT communications is a solution that will use a detection method based on detailed specifications. The analysis of the IPFIX flow records made by IDS modules will be done in a centralized manner, making use of external devices relatively to the IoT application. These external devices have the computational resources that this analysis technique requires, releasing IoT devices from these tasks and assuring scalability to the proposed solution.

Thus, to detect intrusions, it is proposed that the analysis of the IPFIX flow records should be done in a local device and in a remote device installed in a cloud-based system. The process of analysing individual IP flow records should comply with the schema illustrated in Fig. 5.

Regarding the analysis of the collected IP flow records, both local and remote IDS modules have responsibilities in the analysis process as the result of their expected computational capabilities.

Given that the specification-based detection method is not so computationally demanding, this detection method can be applied to the analysis made on both IDS modules. As mentioned, in the cases where the border router has some computational capacity available, it can host the local

IDS module as an integrated module of the router software. Otherwise, if the border router does not have the computational capabilities needed, the local IDS module may be installed on a dedicated device. To perform the analysis of the communications collected by the probes, the IDS modules will query a knowledge database with specifications of the traffic expected and considered normal for the IoT application within the operation scope. This knowledge database can be updated or reviewed whenever changes are made in the IoT application or whenever new specifications of IoT standards or protocols arise. When a threat or attack is detected, an IDS alert message will be generated and registered in the IDS alert database available in the IDS modules.

5.4 System analysis

As mentioned before, it was indicated that doing the analysis of the IP flow records collected by probes located in the three layers of the IoT architecture, significantly increases the capacity of intrusion detection in an IoT application. In addition, since traffic flow records are exported in a near real-time manner, this allows for the detection of intrusions to happen within a reasonable time frame.

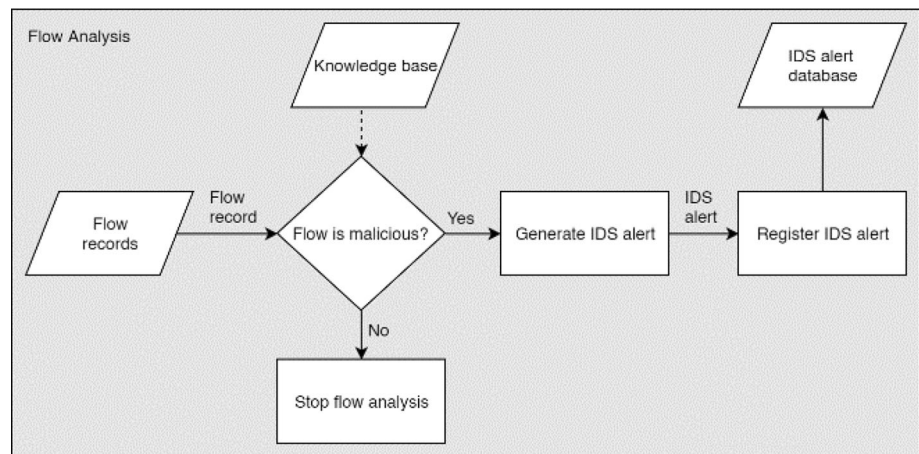
In terms of scalability, it can be ensured in different manners. The use of an IDS based on IP flow analysis ensures that the size of the information collected on the IoT communications is considerably smaller (0.1%), than systems based on network packet analysis. Thus, if there is a large growth in the number of devices and communications to be analysed, the impact will not be immediate, either in terms of storage capacity or in terms of increased bandwidth usage.

Due to the use of IP-based communications, interoperability between systems can be ensured by the fact that the vast majority of IoT technologies support or are about to support the IP protocol (examples: BLE, 802.15.4 and RFID). Another important aspect is that the flow records are exported using the IPFIX standard instead of the proprietary NetFlow solution. Lastly, intrusion alert messages use the standard syslog, thus allowing their use and integration with other security monitoring mechanisms and tools.

The reconfiguration of the proposed IDS is possible due to the possibility that the specification database may be updated or modified whenever changes are made to the IoT system.

Due to the use of probes and IDS modules implemented in IoT devices with available computational resources and integrated support for collecting and exporting IP traffic flows through the IPFIX protocol, there is no need to change the software or firmware of the IoT devices or

Fig. 5 Processing scheme for analysis of flows



services. Nevertheless, in these situations, there is a consumption of the computational resources associated with the probe and the IDS module operation.

In order to ensure the security and privacy of the communications exchanged between the various components of the proposed IDS, these are transmitted using encryption methods. In the case of IPFIX communications between the probes and IDS modules, they must be made using the TLS over TCP protocol. For intrusion alert messages (that are sent via syslog to other non-IDS security mechanisms), these should also be sent using the TLS over TCP protocol.

6 Framework evaluation

In this section, we introduce a performance evaluation of the proposed intrusion detection framework. We present the application of a prototype in a real environment that incorporates sensors, actuators and IoT services that will allow for the generation of diverse IoT-based communications. With this test we aimed at validating and testing the artefact in terms of its data collection, storage and communications analysis capacities.

6.1 Experimental testbed

The developed prototype (represented in Fig. 6), aimed at simulating a real environment of an IoT system and contemplates an application scenario, with the use of several devices that use CoAP and MQTT application protocols, in order to perform data exchange between sensors, actuators, and IoT services.

It was included the use of CoAP clients and servers, as well as MQTT publishers, brokers, and subscribers. In addition to these, two more devices were available. In this context a Raspberry PI was used to act as border router where the IDS probe was deployed, while on the other

device (Ubuntu Server 1) was the IDS module. Finally, there was also a device that would work as a generator of attacks and intrusions to the IoT system and inherent services.

The attacks were executed using net scan tools such as nmap and hping, performing network analysis, flood, and DoS attacks, as well as through invalid or abnormal MQTT and CoAP actions. The IDS probe was implemented in the border router and was responsible for network packet observation, aggregation of packet information into traffic streams, and exporting (through the IPFIX protocol) IP traffic flow records that were transmitted between the border router network interfaces. Thus, IoT communications between the internal network and the Internet, i.e. between IoT service clients and IoT service servers, were also monitored.

The probe used was an open source solution developed by Carnegie Mellon University's CERT group, YAF (Yet another Flowmeter). YAF supports all IEs selected for use in the framework.

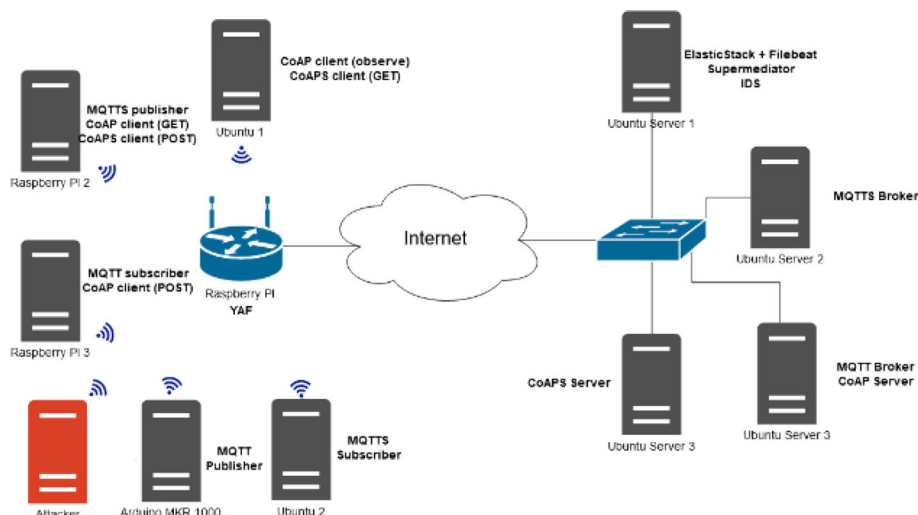
YAF was configured with the following characteristics:

- Packet capture on the WAN interface.
- Real-time capture with PF_RING library.
- Exporting using IPFIX over TLS over TCP.

The IDS module was implemented in the host (Ubuntu Server 1) and was responsible for collecting traffic flow records sent over the IPFIX protocol, analysing the stored IP traffic flow records based on the specification database, and generating intrusion alert messages.

For collection, decoding and storing the IP traffic flow records exported by the probe we used an open source solution developed by Carnegie Mellon University's CERT group, the *super_mediator*. *Super_mediator* is an IPFIX protocol message receiver and decoder and can be used simultaneously with YAF.

Fig. 6 Prototype developed for framework evaluation



The *super_mediator* was configured with the following functions:

- To listen to the Fast Ethernet network interface.
- To receive flow records through the IPFIX over TLS over TCP.
- To store flow records in JSON files.

During the process of receiving and storing traffic flow records, JSON files were created and added in a host directory where *super_mediator* was running. Figure 7 shows the structure of only one IPFIX flow record.

In order to analyse the logs of IP flow records that were stored in JSON files, a Python IDS application was developed considering the workflow illustrated in Fig. 8.

The developed IDS application reads the database of specifications about the normal behaviour of IoT communications based on the CoAP and MQTT protocols. These parameters were obtained by analysing the specifications of the CoAP and MQTT protocols and, in addition, considering an analysis of .

CoAP and MQTT traffic in a real environment made in previous works [65–67]. In their works, authors analysed IP flow records obtained from a IoT environment and concluded that a normal flow related with a confirmable piggybacked CoAP GET or POST message must have the following features: flowendreason is active/idle; octettotalcount and reverseoctettotalcount are > 32 bytes; packettotalcount and reversepackettotalcount is 1:1; databytecount and reversedatabytecount are > 4 bytes; protocolidentifier is 17.

The most common CoAP messages were GET, POST, and GET Observe with piggybacked acknowledgment. MQTT messages were published and subscribed with QoS level 0. Thus, the specifications that allows to determine if a CoAP and MQTT traffic flow record is normal or abnormal were obtained.

Each new flow record was compared with the previously loaded specifications. In cases where the flow record was classified as normal, its verification process ended immediately, returning to the process of checking for new records. In case a flow record was classified as abnormal, the IDS would generate an intrusion alert message via the syslog protocol. This message would be stored in the IDS system log file and was sent via the same protocol to integrate other monitoring tools such as a SIEM.

6.2 Validation results analysis

The test plan used the prototype presented in the previous section and was implemented with the goal of evaluating the proposed framework regarding its functionality, performance, scalability, and security. The test plan was also defined to demonstrate and validate the requirements and characteristics defined for the framework.

To perform all the tests, all the IoT CoAP and MQTT devices inherent to the prototype were activated to initiate normal IoT traffic through data exchange between CoAP clients and servers, as well as between MQTT publishers, brokers, and subscribers. Next, the attacker device was used to initiate the generation of abnormal and malicious IoT traffic by exchanging data only between that device and the CoAP servers. In addition, this attacking device would also perform the role of MQTT publisher and topic subscriber to the MQTT brokers operating on the prototype. In addition to normal and abnormal IoT communications, there were also a DNS and NTP service on the internal network that also generated traffic between the internal network devices and the Internet.

Then, on the IDS probe, YAF was simultaneously performed to listen and capture the network packets exchanged between the internal network IoT devices and the IoT service servers that were found on the Internet. At the same

```

{
  "flows": {
    "flowStartMilliseconds": "2019-06-06 17:06:06.162",
    "flowEndMilliseconds": "2019-06-06 17:06:28.118",
    "flowDurationMilliseconds": 21.956,
    "reverseFlowDeltaMilliseconds": 0.001,
    "protocolIdentifier": 6,
    "sourceIPv4Address": "10.42.0.94",
    "sourceTransportPort": 50936,
    "packetTotalCount": 6,
    "octetTotalCount": 284,
    "flowAttributes": "00",
    "sourceMacAddress": "b8:27:eb:fc:b4:95:",
    "destinationIPv4Address": "192.168.111.22",
    "destinationTransportPort": 1883,
    "reversePacketTotalCount": 6,
    "reverseOctetTotalCount": 248,
    "reverseFlowAttributes": "00",
    "destinationMacAddress": "08:00:27:6f:48:8a:",
    "initialTCPFlags": "S",
    "unionTCPFlags": "APF",
    "reverseInitialTCPFlags": "AS",
    "reverseUnionTCPFlags": "APF",
    "tcpSequenceNumber": "0xa968782b",
    "reverseTcpSequenceNumber": "0x4f9c13cf",
    "ingressInterface": 0,
    "egressInterface": 0,
    "vlanId": "0x000",
    "silkAppLabel": 0,
    "ipClassOfService": "0x00",
    "flowEndReason": "",
    "collectorName": "C1",
    "tcpUrgTotalCount": 0,
    "smallPacketCount": 2,
    "nonEmptyPacketCount": 2,
    "dataByteCount": 40,
    "averageInterarrivalTime": 4391,
    "firstNonEmptyPacketSize": 27,
    "largePacketCount": 0,
    "maxPacketSize": 27,
    "firstEightNonEmptyPacketDirections": "02",
    "standardDeviationPayloadLength": 7,
    "standardDeviationInterarrivalTime": 8771,
    "bytesPerPacket": 20,
    "reverseTcpUrgTotalCount": 0,
    "reverseSmallPacketCount": 1,
    "reverseNonEmptyPacketCount": 1,
    "reverseDataByteCount": 4,
    "reverseAverageInterarrivalTime": 4391,
    "reverseFirstNonEmptyPacketSize": 4,
    "reverseLargePacketCount": 0,
    "reverseMaxPacketSize": 4,
    "reverseStandardDeviationPayloadLength": 0,
    "reverseStandardDeviationInterarrivalTime": 8746,
    "reverseBytesPerPacket": 4
  }
}

```

Fig. 7 Structure of one IPFIX flow record

time, the network traffic was captured using the *tcpdump* software on the border router so that we could record a copy of the communications made between the probe and the IDS module. After that, in the IDS module, the *super_mediator* was executed to receive, decode, and store the records of IP traffic flows exported by the probe through IPFIX. After a few minutes, the YAF and *tcpdump* processes were terminated on the IDS probe and the same

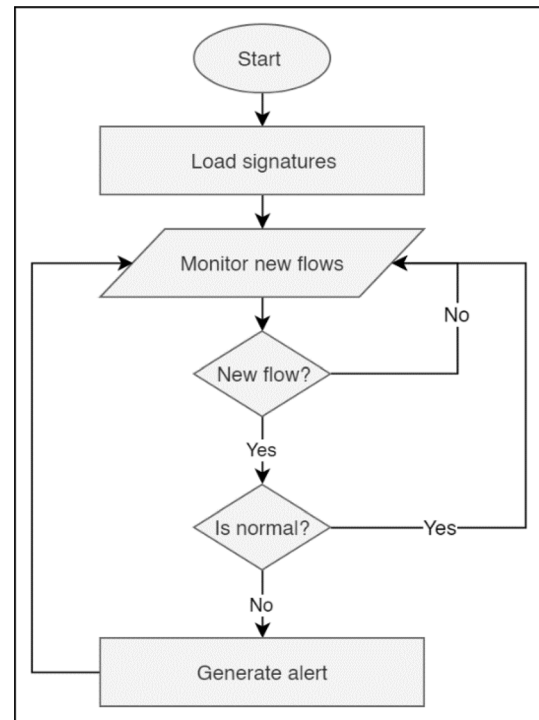


Fig. 8 Workflow of IDS traffic flow analysis process

was made on the *super_mediator* process of the IDS module.

Finally, in the IDS module, the IDS application was executed. After loading the information from the specification database, the application was able to read and analyse the IP traffic flow records of normal and abnormal IoT communications that were stored in JSON format in the IDS module, which resulted in an IDS report.

6.2.1 Functionality tests

These tests were defined in order to test the functionality of the framework regarding: (a) network packet capture and consequently aggregating and exporting securely the corresponding IP traffic flow records by the IDS probe; (b) reception, decoding and storage by the IDS module of IP traffic flow records transmitted via the IPFIX protocol; (c) analysis of IP traffic flow records stored in the IDS module to detect IoT communications anomalies in the test scenario; and (d) generation and storage of intrusion alert messages.

In order to validate the functionality of the framework, the results of the tests performed were demonstrated by presenting and analysing the JSON files stored in the IDS module containing the traffic flow records exported by the IDS probe. These files showed the network packet capture made by the IDS probe which, after aggregating them into traffic flow records, exported them to the IDS module.

In addition to those files, the network packets belonging to the traffic exchanged between the internal network and the Internet are also stored in the border router utilizing PCAP files. The communications between the probe and the IDS module were also presented and analysed.

Additionally, the IDS application reports, which were stored in the IDS module and contained the results of the analysis of traffic flow records indicating whether they were classified as normal or abnormal, were considered.

Finally, the IDS application log files, which were stored in the IDS module device, would contain the intrusion alert messages generated and stored by the IDS application via the syslog protocol.

In order to validate and test the IDS proposed in terms of functionality of the probe and the IDS module, specifically network packet capture and consequently aggregating and exporting securely the corresponding IP traffic flow records by the IDS probe, the PCAP files extracts presented in Fig. 9 and Fig. 10 demonstrated the existence of CoAP traffic exchanged between server and client as well as the IPFIX traffic exchanged between the probe and the IDS module (made via TLS over TCP).

The reception, decoding and storage by the IDS module of IP traffic flow records transmitted via the IPFIX protocol can be demonstrated by observing Fig. 11. It shows the list of files created in the local device folder by the *super-mediator* to store JSON files containing the IPFIX records.

Figure 12 presents the content of a JSON file containing diverse IPFIX flow records stored in the IDS module.

In terms of validation and test of the IDS proposed for the detection and identification of normal and abnormal IoT communications through the analysis of IPFIX flow records stored in the IDS module, we are able to detail that the tests were divided into two different sets of IP flow records. One test was done using only IP flow records from normal communications and the other test was done only with IP flow records from abnormal communications. Both were compared and verified through the IDS application that used the specifications stored in the specifications database.

Figures 13, 14, 15, 16 show the results for each test of the IoT traffic flow log analysis. The information presented in each table indicates the type(s) of message(s) analysed

No.	Time	Source	Destination	Protocol	Length	Info
126	18.948069	192.168.111.22	192.168.111.22	CoAP	58	CON, MID:54487, GET, /temperature
130	18.954590	192.168.111.22	192.168.111.22	CoAP	60	ACK, MID:54487, 2.05 Content
337	48.981492	192.168.111.22	192.168.111.22	CoAP	58	CON, MID:54488, GET, /temperature
341	48.987556	192.168.111.22	192.168.111.22	CoAP	60	ACK, MID:54488, 2.05 Content
535	79.014947	192.168.111.22	192.168.111.22	CoAP	58	CON, MID:54489, GET, /temperature
539	79.021093	192.168.111.22	192.168.111.22	CoAP	60	ACK, MID:54489, 2.05 Content
762	109.088...	192.168.111.22	192.168.111.22	CoAP	58	CON, MID:54490, GET, /temperature
764	109.084...	192.168.111.22	192.168.111.22	CoAP	60	ACK, MID:54490, 2.05 Content
10...	139.053...	192.168.111.22	192.168.111.22	CoAP	58	CON, MID:54491, GET, /temperature

```

Frame 126: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
> Ethernet II, Src: Raspberri.fc:b4:95 (08:27:eb:fc:b4:95), Dst: PcsCompu_6f:48:8a (08:00:27:6f:48:8a)
> Internet Protocol Version 4, Src: 192.168.0.46, Dst: 192.168.111.22
> User Datagram Protocol, Src Port: 45661, Dst Port: 5683
> Constrained Application Protocol, Confirmable, GET, MID:54487
    
```

Fig. 9 Network traffic capture of GET CoAP messages

Time	Source	Destination	Protocol	Length	Info
8090	263.772531	192.168.95.129	TCP	74	4740 → 33446 [SYN, ACK] Seq=0 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=81928215
9116	263.782000	192.168.95.129	TCP	66	33446 → 4740 [ACK] Seq=1 Ack=182 Min=29320 Len=0 TSval=1946611 TSecr=81928215
9117	263.782000	192.168.95.129	TCP	66	33446 → 4740 [ACK] Seq=1 Ack=182 Min=29320 Len=0 TSval=1946611 TSecr=81928215
9148	263.801987	192.168.95.129	TCP	66	33446 → 4740 [ACK] Seq=182 Ack=1449 Min=32128 Len=0 TSval=1946617 TSecr=81928215
9150	263.808993	192.168.95.129	TCP	66	33446 → 4740 [ACK] Seq=182 Ack=1829 Min=39372 Len=0 TSval=1946617 TSecr=81928215
9167	263.820504	192.168.95.129	TLSv1	2370	Server Hello, Certificate, Certificate Request, Server Hello Done
9168	263.821818	192.168.95.129	TLSv1	66	4740 → 33446 [ACK] Seq=1829 Ack=2486 Min=34488 Len=0 TSval=81928215 TSecr=1946655
9192	263.842138	192.168.95.129	TLSv1	1164	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
9953	280.646973	192.168.95.129	TLSv1	2942	Application Data, Application Data
9955	280.647076	192.168.95.129	TCP	66	4740 → 33446 [ACK] Seq=2927 Ack=3068 Min=41392 Len=0 TSval=8209097 TSecr=1963483
9956	280.647084	192.168.95.129	TLSv1	342	Application Data, Application Data
9958	280.648089	192.168.95.129	TCP	66	4740 → 33446 [ACK] Seq=2927 Ack=6644 Min=64336 Len=0 TSval=8209098 TSecr=1963484

Fig. 10 Network traffic capture of CoAP over DTLS and IPFIX over TLS over TCP

```

ids4tot@kserver:~/Flows$ ls
ids4tot@kserver:~/Flows$ ls
yaf.20190623181353.json
ids4tot@kserver:~/Flows$ ls
yaf.20190623181353.json yaf.20190623191353.json
ids4tot@kserver:~/Flows$ ls
yaf.20190623181353.json yaf.20190623191353.json yaf.20190623201353.json
ids4tot@kserver:~/Flows$
    
```

Fig. 11 Folder content with stored flow records

```

[~/yaf.2019061220756.json]
1 [{"Flow": {"FlowStartMilliseconds": "2019-06-12 12:07:34.425", "FlowEndMilliseconds": "2019-06-12 12:07:56.030", "FlowID": "1", "FlowType": "FlowStartMilliseconds", "FlowStartMilliseconds": "2019-06-12 12:07:19.484", "FlowEndMilliseconds": "2019-06-12 12:07:19.484", "FlowID": "2", "FlowType": "FlowEndMilliseconds", "FlowStartMilliseconds": "2019-06-12 12:07:19.484", "FlowEndMilliseconds": "2019-06-12 12:07:19.501", "FlowID": "3", "FlowType": "FlowStartMilliseconds", "FlowStartMilliseconds": "2019-06-12 12:07:34.425", "FlowEndMilliseconds": "2019-06-12 12:07:34.451", "FlowID": "4", "FlowType": "FlowEndMilliseconds", "FlowStartMilliseconds": "2019-06-12 12:07:34.425", "FlowEndMilliseconds": "2019-06-12 12:07:34.451", "FlowID": "5", "FlowType": "FlowStartMilliseconds", "FlowStartMilliseconds": "2019-06-12 12:07:34.542", "FlowEndMilliseconds": "2019-06-12 12:07:34.542", "FlowID": "6", "FlowType": "FlowEndMilliseconds", "FlowStartMilliseconds": "2019-06-12 12:07:38.074", "FlowEndMilliseconds": "2019-06-12 12:07:38.085", "FlowID": "7", "FlowType": "FlowStartMilliseconds", "FlowStartMilliseconds": "2019-06-12 12:07:38.085", "FlowEndMilliseconds": "2019-06-12 12:07:38.085", "FlowID": "8", "FlowType": "FlowEndMilliseconds", "FlowStartMilliseconds": "2019-06-12 12:07:12.639", "FlowEndMilliseconds": "2019-06-12 12:07:12.639", "FlowID": "9", "FlowType": "FlowStartMilliseconds", "FlowStartMilliseconds": "2019-06-12 12:07:12.649", "FlowEndMilliseconds": "2019-06-12 12:07:12.649", "FlowID": "10", "FlowType": "FlowEndMilliseconds", "FlowStartMilliseconds": "2019-06-12 12:07:19.485", "FlowEndMilliseconds": "2019-06-12 12:07:19.485", "FlowID": "11", "FlowType": "FlowStartMilliseconds", "FlowStartMilliseconds": "2019-06-12 12:07:34.443", "FlowEndMilliseconds": "2019-06-12 12:07:45.471", "FlowID": "12", "FlowType": "FlowEndMilliseconds"}]}
    
```

Fig. 12 JSON file content with received flow records

CoAP message type	Number of flows in the dataset	Number of flows CoAP	NFR	AFR	DR	FP	TP
GET	100	100	100	0	100%	0%	100%
POST	100	100	100	0	100%	0%	100%
GET Observe	100	100	100	0	100%	0%	100%
GET/POST/GET Obs.	17260	1153	1153	0	100%	0%	100%

Fig. 13 Tests results for normal CoAP flow records

MQTT message type	Number of flows in the dataset	Number of flows MQTT	NFR	AFR	DR	FP	TP
PUBLISH	100	100	100	0	100%	0%	100%
SUBSCRIBE	100	100	100	0	100%	0%	100%
PUBLISH/SUBSCRIBE	17260	2321	2321	0	100%	0%	100%

Fig. 14 Tests results for normal MQTT flow records

CoAP anomaly type	Number of flows in the dataset	Number of flows CoAP	NFR	AFR	DR	FP	TP
net scan	100	100	0	100	100%	0%	100%
CoAP individual invalid request	100	100	0	100	100%	0%	100%
CoAP flood of invalid requests	100	100	0	100	100%	0%	100%
CoAP flood of valid requests	100	100	10	90	90%	0%	100%
Net scan / invalid request / flood of valid and invalid requests	98334	15847	2352	13495	99%	0%	100%

Fig. 15 Tests results for abnormal CoAP flow records

MQTT anomaly type	Number of flows in the dataset	Number of flows MQTT	NFR	AFR	DR	FP	TP
net scan	100	100	0	100	100%	0%	100%
MQTT individual invalid request	100	100	0	100	100%	0%	100%
MQTT flood of invalid requests	100	100	0	100	100%	0%	100%
MQTT flood of valid requests	100	100	21	79	79%	0%	100%
Net scan / invalid request / flood of valid and invalid requests	97435	28317	3398	24919	93%	0%	100%

Fig. 16 Tests results for abnormal MQTT flow records

(normal or abnormal), the number of general IP flow records in the dataset, the number of CoAP/MQTT IP flow records in the dataset, the number of flow records that were classified as normal (NFR), the number of flow records that were classified as abnormal (AFR), the detection rate (DR), the false positive rate (FP) and the rate of true positives (TP).

The first test just used IP flow records from normal communications generated by different types of messages from CoAP and MQTT protocols. The results obtained indicates that the specifications were well defined, and the IDS application can classify as normal, with 100% of DR, 100% of TP and 0% of FP, all the IP flow records from normal CoAP and MQTT communications that were analysed.

The second test just used IP flow records from abnormal communications generated by the attacker device using different types of messages from CoAP and MQTT protocols. The results allowed to perceive that the specifications were well defined for most of the abnormal traffic. The only exception were the IP flow records concerning the flooding (1000) of CoAP and MQTT normal requests, due to their similarity with the normal CoAP and MQTT requests.

The IDS application was able to classify as abnormal, with 100% of DR, 100% of TP and 0% of FP, all the IP flow records from abnormal (net scan, invalid requests, flood of invalid requests) CoAP and MQTT communications that were analysed. In terms of the analyses of IP flow records from a flood of valid requests CoAP and MQTT, the IDS application could only classify as abnormal, with 90% of DR, 100% of TP and 0% of FP, the flood of valid requests CoAP. In relation to the flood of valid requests MQTT, the IDS application could only classify as abnormal, with 79% of DR, 100% of TP and 0% of FP.

To validate and test the proposed IDS in terms of the functionality of the generation and storage of intrusion alert messages by the IDS module, the IDS application prints the results of the classification of the flow records as normal or

```

$ python3 main.py
IS ATTACK
IS ATTACK
IS ATTACK
...
IS ATTACK
IS ATTACK
IS ATTACK
IS ATTACK
IS ATTACK
IS ATTACK
FLOW COUNTERS - 2212 attacks ----- 356 normal
    
```

Fig. 17 Output of the IDS application execution

abnormal and also a summary of the tests that were done (Fig. 17).

In the cases that the flow records were classified as abnormal, the IDS application generated and stored a log record in the syslog file as presented in Fig. 18.

6.2.2 Performance and scalability tests

These tests were defined to test the performance and scalability of the framework regarding: (a) computational resource consumption by IDS components; and (b) network traffic overhead generated by the IDS proposed.

In order to validate the performance and scalability of the framework, an analysis was made to the PCAP files stored on the border router that contained the network packet captures inherent to the traffic exchanged between the internal network and the Internet, as well as the communications between the probe and the IDS module. In addition to those files we also used results of the analysis made to the data related to the consumption of computational resources that was collected from the devices where the probe and IDS module were running during the tests.

The computational resources analysed were related with the CPU usage, RAM memory usage, disk or ROM memory used by the probe and the module of the IDS. The presentation of results related to the consumption of computational resources is divided into two parts: probe and IDS module.

Regarding the consumption of computational resources that result from the operation of the IDS probe in the border router, several measurements were performed to quantify, in a more accurate manner, the resources that were consumed. At the same time, in addition to measuring the router in normal operation, the same measurements were made for the operation of three different conventional IDS: *Snort*, *Suricata*, and *Zeek*. Therefore, measurements were performed considering the following features: CPU usage

```

Aug 4 01:13:45 ubuntu main.py: ATTACK - {"flowStartMilliseconds": "2019-08-04 00:13:41.246", "fl
Aug 4 01:13:45 ubuntu main.py: ATTACK - {"flowStartMilliseconds": "2019-08-04 00:13:41.246", "fl
Aug 4 01:13:45 ubuntu main.py: ATTACK - {"flowStartMilliseconds": "2019-08-04 00:13:41.246", "fl
Aug 4 01:13:45 ubuntu main.py: ATTACK - {"flowStartMilliseconds": "2019-08-04 00:13:41.246", "fl
Aug 4 01:13:45 ubuntu main.py: ATTACK - {"flowStartMilliseconds": "2019-08-04 00:13:41.246", "fl
Aug 4 01:13:45 ubuntu main.py: ATTACK - {"flowStartMilliseconds": "2019-08-04 00:13:41.246", "fl
Aug 4 01:13:45 ubuntu main.py: ATTACK - {"flowStartMilliseconds": "2019-08-04 00:13:41.246", "fl
    
```

Fig. 18 Syslog file content with alert messages

(in percent and number of processes); RAM usage; persistent memory usage (disk). Figure 19 presents the results of a simulated network operation scenario with measurements made with 50 Mbps of data bandwidth on the network interface.

As one can perceive by analysing Fig. 19, it can be seen that the YAF, used as a probe in the proposed prototype, is the solution that consumes the least computational resources when compared to the conventional IDS tested.

Particularizing CPU consumption, YAF measurements showed that it does not add significant overhead compared to normal operation, either at the used CPU percentage level or in number of process. This can be explained by the fact that YAF's function is focused on packet capture and flow records aggregation and exportation and does not perform any analysis of packet information or content. Regarding RAM consumption, YAF also did not cause a large increase in the consumption of this type of resource either, since it frequently exports the records present in the cache memory. Finally, in terms of disk storage consumption, YAF also did not create a large fingerprint on the need to store information locally, since data was always exported to another device.

About the consumption of computational resources that result from the operation of the module of the IDS, several measurements were made in order to quantify with accuracy the resources consumed, either by the *super_mediator*, responsible for receiving, decoding and storing the flow records, or by the IDS application, which is responsible for analysing all new incoming flow records. Measurements were performed for the following features: CPU usage (in percent and number of processes); RAM usage; use of disk

storage. In an attempt to simulate network operation in different scenarios, the measurements are based on three data bandwidth situations on the IDS probe: 10Mbps, 25Mbps, and 50 Mbps. These results are described in Fig. 20.

By analysing the graphics presented for each different situation, it is possible to acknowledge that running the *super_mediator* and an IDS application can be considered as an IDS solution that consumes few computational resources. Given the level of CPU consumption, the IDS module measurements showed that it did not add significant overhead compared to normal operation, either at the CPU percentage level used or at the number of process increases (two). This can be explained by the fact that the functionality of the IDS module's applications is simple and focused on receiving, decoding, and storing traffic flow and further analysis, without the need for processing more complex tasks. As for RAM consumption, the IDS module also did not cause a large increase in the consumption of this type of resource. Finally, when it comes to disk storage space consumption, the central module might have some impact here if there is no policy of managing the files where the records of traffic flows received are stored.

In terms of test and validation of the network traffic overhead generated by the proposed IDS, this test aimed to verify the amount of data used by the internal IDS messages, i.e. the IPFIX traffic flow records sent between the probe and the IDS module.

In Fig. 10 the extracts of the PCAP files resulting from the traffic capture made in the border router can be verified. These extracts refer to IPFIX traffic, namely the connection between the probe and the IDS module.

Fig. 19 Probe resources consumption with 50 Mbps

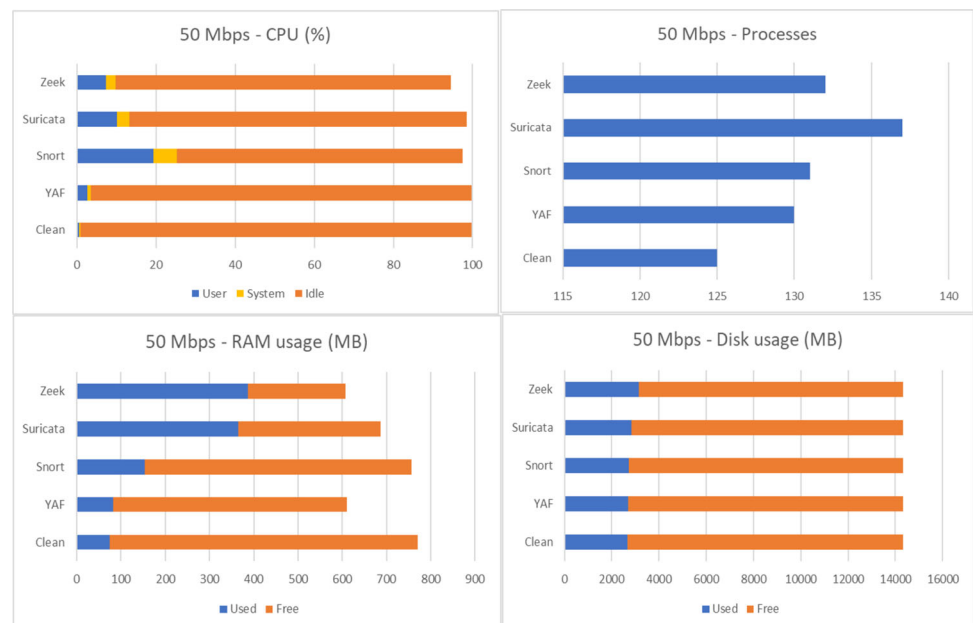
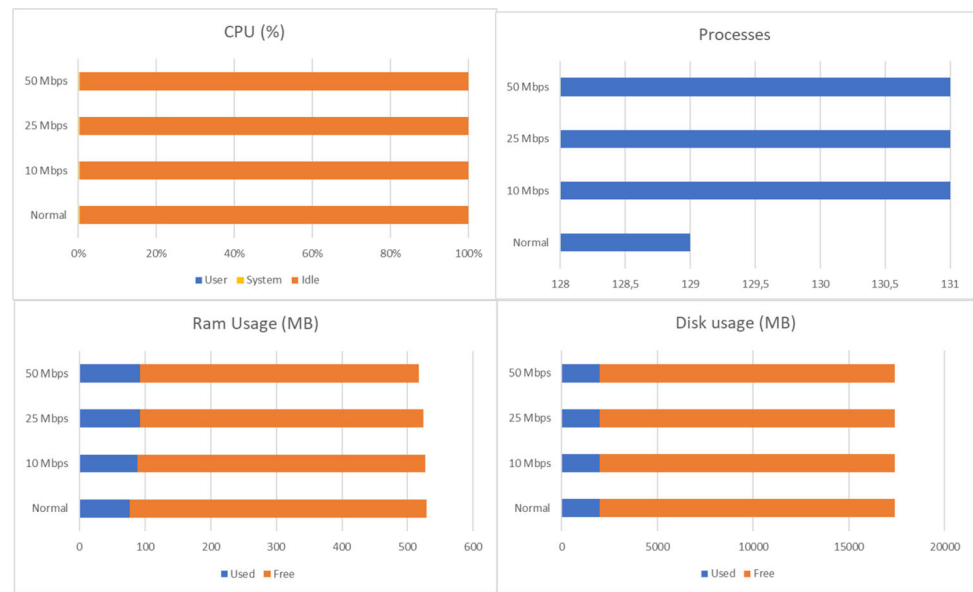


Fig. 20 IDS module resources consumption

Note that the size, in bytes, of the IPFIX data exchanged between the probe and the IDS module is around 8 KB in a 180 s time period, which, in addition to the overhead caused by using the TLS assumes a reduced overhead on the level of communication channel used.

6.2.3 Security tests

Security tests were defined to test the security of the framework regarding the confidentiality and the integrity of the internal communications and messages exchanged between IDS components (probe and IDS module).

In order to confirm the security assurance features of the proposed framework, an analysis was made to the PCAP files stored on the border router that contained the network packet captures inherent to the traffic exchanged between the internal network and the Internet, as well as the communications between the probe and the IDS module.

The extracts that are presented in Fig. 10 refer to the IPFIX traffic of the connection between the probe and the IDS module. They refer to the process of exporting the traffic flow records created by the probe through the IPFIX over TLS over TCP. After some flow records stored in the YAF cache reach any of the timeout timers (idle or active), they are exported to the IDS module using the IPFIX over TLS over TCP.

Given the test results, as expected, messages exchanged between the probe and the IDS module were protected by using the IPFIX over TLS over TCP, thus ensuring their confidentiality and integrity.

6.3 Discussion

The framework validity was assessed by analysing and evaluating the results of the performed tests. This evaluation was focused on the functionality of the IDS components that capture, export, collect, store, and analyse IoT communications based on traffic flow records. In addition to the functionality of the framework, the performance and scalability of the solution were also evaluated, considering the computational resources and bandwidth consumption. Finally, the results of the internal communications security tests of the proposed IDS system were validated.

The proposed solution showed the full functionality of the various IDS components of the framework. As it was described in the previous sections, the defined IDS components have the capacity to: (a) observe network traffic; (b) aggregate information about communications into traffic flow records; (c) export flow records via IPFIX secure connections; (d) collect, decode and store flow records in databases; (e) analyse flow records to detect anomalies in IoT communications; and (f) send intrusion alert messages using a standard protocol such as syslog, thus ensuring interoperability with other security and monitoring mechanisms.

In terms of detection, with normal and abnormal traffic tests the results achieved 100% on detection rate and true positives, and 0% on false positive rates. In the specific tests with anomalies based on flooding of normal application requests, the detection rate was not 100% because it is necessary to introduce extra request number validation mechanisms over a given period.

In terms of the level of performance, the results demonstrated and validated that the IDS probe and central

module do not require many computational resources (processing, storage and communication) to operate and, in some systems, they can be implemented on some IoT devices with limited resources. Finally, the security tests on the proposed solution have shown that the internal IDS components of this framework can communicate with confidentiality and integrity using an encrypted channel for transfer traffic flow records between the probe and the central IDS module.

7 Conclusions

As the amount of IoT application deployments increases across a variety of scenarios such as health, industry, etc., these applications and projects will use and transmit more sensitive data. As such, ensuring the IoT data privacy and security is mandatory. With preventive actions hard to be applied due to architectural limitations, security solutions must turn to second line methods of defence. We considered IDS as one such defence method and determined that despite the diversity of IDS solutions for IoT available in the market, none can protect against all categories of threats and attacks (from the perception layer up), due to their architectural application.

In this paper, we proposed a framework for an IDS specifically focused on IoT networks based on the use of IP flow records that will be captured by probes located in the three layers of the IoT architecture. We presented a list of system requirements, the design of a flow-based IDS for IoT framework architecture, and described its main components, features, and functions. In what concerns placement strategy, a hybrid architecture is proposed that includes both distributed data collection and centralized intrusion detection analysis. This placement strategy proposed allows the detection of threats or attacks originated from external networks as well as from internal (compromised) nodes in a near real-time manner. The proposed distributed data collection is going to involve the use of several probes that collect IP flow records and securely send them to the local and remote IDS components.

To take benefits from the use of a lightweight and efficient detection technique, our proposed framework uses a specific detection method based on the normal behaviour of IoT communications.

The proposed framework presented very good results in terms of security, since it uses IPFIX and Syslog over TLS over TCP to protect all communications done by the IDS probe and module.

In terms of functionality, the tests showed very promising results in terms of detection of normal IoT communications since it had 100% of detection rate, 100% of true positive rate, and 0% of False Positive rate in the

classification of normal flow records. Regardless of the detection of abnormal IoT communications, the proposed solution had also very good results in most of the tests with 100% of detection rate, 100% of true positive rate, and 0% of False Positive rate.

Regardless of performance and scalability, this solution ensures that there is a very low performance overhead and no software changes on the IoT devices and application. Interoperability and extensibility are also ensured due to the use of standards for the collect and storage of IoT communications data, and due to the use of a database with the specifications stored that can be updated.

In our future work we plan to test and validate the proposed framework in environments where communication technologies could be more diverse, such as 802.15.4, BLE, and LoRaWAN, and using probes in all layers. In addition, this framework will be tested using a widely used dataset for intrusion detection in IoT systems to allow a more effective comparison with other solutions in this field. The support for intrusion detection of more threats (worms and botnets) and improve the results of those already supported, such as (D)DoS are also planned for future work. Finally, we plan to use the correlation of data collected in different IoT layers to improve the intrusion detection mechanism.

Acknowledgements This work was supported by Portuguese national funds through the FCT—Foundation for Science and Technology, I.P., under the project UID/CEC/04524/2019.

References

1. Santos, L., Rabadão, C., Gonçalves, R.: Flow monitoring system for IoT networks. In: Rocha, Á., Adeli, H., Reis, L., Costanzo, S. (eds.) *WorldCIST'19 - 7th World Conference on Information Systems and Technologies*, Galicia, Spain 2019. *Advances in intelligent systems and computing*, New Knowledge in information systems and technologies, pp. 420–430. Springer, Cham (2019)
2. Bradley, J., Barbier, J., Handler, D.: Embracing the Internet of everything to capture your share of \$14.4 trillion. White Paper, Cisco (2013)
3. Lee, I., Lee, K.: The Internet of things (IoT): applications, investments, and challenges for enterprises. *Bus. Horiz.* **58**(4), 431–440 (2015)
4. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. *Comput. Netw.* **54**(15), 2787–2805 (2010)
5. Sha, K., Wei, W., Yang, T., Wang, Z., Shi, W.: On security challenges and open issues in Internet of things. *Future Gener. Comput. Syst.* **83**, 326–337 (2018)
6. Kothmayr, T., Schmitt, C., Hu, W., Brünig, M., Carle, G.: DTLS based security and two-way authentication for the Internet of things. *Ad Hoc Netw.* **11**(8), 2710–2723 (2013). <https://doi.org/10.1016/j.adhoc.2013.05.003>
7. Raza, S., Wallgren, L., Voigt, T.: SVELTE: real-time intrusion detection in the Internet of things. *Ad Hoc Netw.* **11**(8), 2661–2674 (2013)

8. Raza, S., Duquenois, S., Höglund, J., Roedig, U., Voigt, T.: Secure communication for the Internet of things—a comparison of link-layer security and IPsec for 6LoWPAN. *Secur. Commun. Netw.* **7**(12), 2654–2668 (2014)
9. Granjal, J., Monteiro, E., Silva, J.: Security for the internet of things: a survey of existing protocols and open research issues. *IEEE Commun. Surv. Tutor.* **17**(3), 1294–1312 (2015)
10. Khan, M., Salah, K.: IoT security: review, blockchain solutions, and open challenges. *Future Gener. Comput. Syst.* **82**, 395–411 (2018)
11. AlRidhawi, I., Otoum, S., Aloqaily, M., Jararweh, Y., Baker, T.: Providing secure and reliable communication for next generation networks in smart cities. *Sustain. Cities Soc.* **56**, 102080 (2020)
12. Otoum, S., Kantarci, B., Mouftah, H.: Empowering reinforcement learning on big sensed data for intrusion detection. In *Icc 2019–2019 IEEE international conference on communications (ICC)*. IEEE, 2019
13. Zarpelao, B., Miani, R., Kawakani, C., de Alvarenga, S.: A survey of intrusion detection in Internet of things. *J. Netw. Comput. Appl.* **84**, 25–37 (2017)
14. Santos, L., Rabadão, C., Gonçalves, R.: Intrusion detection systems in Internet of things: a literature review. In: *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, Cáceres, Spain, pp. 1–7. IEEE, 2018
15. Hajiheidari, S., Wakil, K., Badri, M., Navimipour, N.J.: Intrusion detection systems in the Internet of things: a comprehensive investigation. *Comput. Netw.* **160**, 165–191 (2019)
16. Alaba, F., Othman, M., Hashem, I., Alotaibi, F.: Internet of things security: a survey. *J. Netw. Comput. Appl.* **88**, 10–28 (2017)
17. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of things: a survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutor.* **17**(4), 2347–2376 (2015)
18. Botta, A., De Donato, W., Persico, V., Pescapé, A.: Integration of cloud computing and internet of things: a survey. *Future Gener. Comput. Syst.* **56**, 684–700 (2016)
19. Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I.: Internet of things: vision, applications and research challenges. *Ad Hoc Netw.* **10**(7), 1497–1516 (2012)
20. Ziegler, S., Crettaz, C., Ladid, L., Krco, S., Pokric, B., Skarmeta, A., Jara, A., Kastner, W., Jung, M.: Iot6—moving to an ipv6-based future iot. In: Alex, G., Anastasius, G. (eds.) *The future internet assembly*, Dublin, Ireland 2013. Lecture notes in computer science, pp. 161–172. Springer, Berlin, Heidelberg (2013)
21. Khan, R., Khan, S., Zaheer, R., Khan, S.: Future internet: the internet of things architecture, possible applications and key challenges. In: *2012 10th international conference on frontiers of information technology*, Islamabad, Pakistan, pp. 257–260. IEEE, 2012
22. Yang, Z., Yue, Y., Yang, Y., Peng, Y., Wang, X., Liu, W.: Study and application on the architecture and key technologies for IOT. In: *2011 International Conference on Multimedia Technology*, Hangzhou, China, pp. 747–751. IEEE, 2011
23. Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., Zhao, W.: A survey on internet of things: architecture, enabling technologies, security and privacy, and applications. *IEEE Internet Things J.* **4**(5), 1125–1142 (2017)
24. Leo, M., Battisti, F., Carli, M., Neri, A.: A federated architecture approach for Internet of Things security. In: *2014 Euro Med Telco Conference (EMTC)*, pp. 1–5. IEEE, 2014
25. Zegzhda, D., Stepanova, T.: Achieving Internet of things security via providing topological sustainability. In: *2015 Science and Information Conference (SAI)*, pp. 269–276. IEEE, 2015
26. Meddeb, A.: Internet of things standards: who stands out from the crowd? *IEEE Commun. Mag.* **54**(7), 40–47 (2016)
27. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **29**(7), 1645–1660 (2013)
28. Gluhak, A., Krco, S., Nati, M., Pfisterer, D., Mitton, N., Razafindralambo, T.: A survey on facilities for experimental internet of things research. *IEEE Commun. Mag.* **49**(11), 58–67 (2011)
29. Sheng, Z., Yang, S., Yu, Y., Vasilakos, A., McCann, J., Leung, K.: A survey on the ietf protocol suite for the internet of things: standards, challenges, and opportunities. *IEEE Wirel. Commun.* **20**(6), 91–98 (2013)
30. Stankovic, J.: Research directions for the internet of things. *IEEE Internet Things J.* **1**(1), 3–9 (2014). <https://doi.org/10.1109/JIOT.2014.2312291>
31. Chen, S., Xu, H., Liu, D., Hu, B., Wang, H.: A vision of IoT: applications, challenges, and opportunities with china perspective. *IEEE Internet Things J.* **1**(4), 349–359 (2014)
32. Lee, S., Levanti, K., Kim, H.: Network monitoring: present and future. *Comput. Netw.* **65**, 84–98 (2014)
33. Velan, P.: Improving network flow definition: formalization and applicability. In: *NOMS 2018–2018 IEEE/IFIP Network Operations and Management Symposium*, Taipei, Taiwan, pp. 1–5. IEEE, 2018
34. Claise, B., Trammell, B., Aitken, P.: Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information. In: *RFC 7011 (INTERNET STANDARD)*, Internet Engineering Task Force. (2013)
35. Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., Stiller, B.: An overview of IP flow-based intrusion detection. *IEEE Commun. Surv. Tutor.* **12**(3), 343–356 (2010)
36. Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., Pras, A.: Flow monitoring explained: from packet capture to data analysis with netflow and ipfix. *IEEE Commun. Surv. Tutor.* **16**(4), 2037–2064 (2014)
37. Zseby, T., Boschi, E., Brownlee, N., Claise, B.: IP flow information export (IPFIX) applicability. In, vol. *RFC 5472*. Internet Engineering Task Force (IETF), (2009)
38. Li, B., Springer, J., Bebis, G., Gunes, M.: A survey of network flow applications. *J. Netw. Comput. Appl.* **36**(2), 567–581 (2013)
39. Halme, L., Bauer, R.: Aint misbehaving - A taxonomy of anti-intrusion techniques. In: Wakil, S., Davis, J. (eds.) *National information systems security 95*, pp. 163–172. DIANE Publishing, Baltimore, EUA (1996)
40. AbuHmed, T., Mohaisen, A., Nyang, D.: A survey on deep packet inspection for intrusion detection systems. *Magazine Korea Telecommun. Soc.* **24**(11), 25–36 (2008)
41. Husák, M., Velan, P., Vykopal, J.: Security monitoring of http traffic using extended flows. In: *2015 10th International Conference on Availability, Reliability and Security*, Toulouse, France, pp. 258–265. IEEE, 2015
42. Liao, H., Lin, C., Lin, Y., Tung, K.: Intrusion detection system: a comprehensive review. *J. Netw. Comput. Appl.* **36**(1), 16–24 (2013)
43. Koch, R.: Towards next-generation intrusion detection. In: *2011 3rd International Conference on Cyber Conflict*, Tallinn, Estonia, pp. 1–18. IEEE, 2011
44. Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E.: Anomaly-based network intrusion detection: techniques systems and challenges. *Comput. Secur.* **28**, 18–28 (2009)
45. Umer, M., Sher, M., Bi, Y.: Flow-based intrusion detection: techniques and challenges. *Comput. Secur.* **70**, 238–254 (2017)
46. Abuadlla, Y., Kvascev, G., Gajin, S., Jovanovic, Z.: Flow-based anomaly intrusion detection system using two neural network stages. *Comput. Sci. Inf. Syst.* **11**(2), 601–622 (2014)
47. Costa, K., Pereira, L., Nakamura, R., Pereira, C., Papa, J., Falcão, A.: A nature-inspired approach to speed up optimum-path forest

- clustering and its application to intrusion detection in computer networks. *Inf. Sci.* **294**, 95–108 (2015)
48. Satoh, A., Nakamura, Y., Ikenaga, T.: A flow-based detection method for stealthy dictionary attacks against secure shell. *J. Inf. Secur. Appl.* **21**, 31–41 (2015)
 49. Liu, C., Yang, J., Chen, R., Zhang, Y., Zeng, J.: Research on immunity-based intrusion detection technology for the internet of things. In: 2011 Seventh International Conference on Natural Computation, Shanghai, China, pp. 212–216. IEEE, 2011
 50. Kasinathan, P., Costamagna, G., Khaleel, H., Pastrone, C., Spirito, M.: An IDS framework for internet of things empowered by 6LoWPAN. In: Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security, Berlin, Germany, pp. 1337–1340. ACM, 2013
 51. Kasinathan, P., Pastrone, C., Spirito, M., Vinkovits, M.: Denial-of-service detection in 6LoWPAN based Internet of things. In: 2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob), Lyon, France, pp. 600–607. IEEE, 2013
 52. Shreenivas, D., Raza, S., Voigt, T.: Intrusion detection in the RPL-connected 6LoWPAN networks. In: Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security, Abu Dhabi, United Arab Emirates, pp. 31–38. ACM, 2017
 53. Jun, C., Chi, C.: Design of complex event-processing IDS in internet of things. In: 2014 Sixth International Conference on Measuring Technology and Mechatronics Automation, Zhangjiajie, China, pp. 226–229. IEEE, 2014
 54. Pongle, P., Chavan, G.: Real time intrusion and wormhole attack detection in internet of things. *Int. J. Comput. Appl.* **121**(9), 1–9 (2015)
 55. Midi, D., Rullo, A., Mudgerikar, A., Bertino, E.: Kalis—A system for knowledge-driven adaptable intrusion detection for the Internet of things. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 656–666. IEEE, 2017
 56. Aloqaily, M., Otoum, S., Al Ridhawi, I., Jararweh, Y.: An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Netw.* **90**, 101842 (2019)
 57. Diro, A.A., Chilamkurti, N.: Distributed attack detection scheme using deep learning approach for Internet of things. *Future Gener. Comput. Syst.* **82**, 761–768 (2018)
 58. Li, J., Zhao, Z., Li, R., Zhang, H.: Ai-based two-stage intrusion detection for software defined IoT networks. *IEEE Internet Things J.* **6**(2), 2093–2102 (2018)
 59. Deng, L., Li, D., Yao, X., Cox, D., Wang, H.: Mobile network intrusion detection for IoT system based on transfer learning algorithm. *Clust. Comput.* **22**(4), 9889–9904 (2019)
 60. Gajewski, M., Batalla, J.M., Mastorakis, G., Mavroumoustakis, C.X.: A distributed IDS architecture model for smart home systems. *Clust. Comput.* **22**, 1–11 (2019)
 61. Pajouh, H.H., Javidan, R., Khayami, R., Dehghantanha, A., Choo, K.R.: A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks. *IEEE Ann. Hist. Comput.* **02**, 314–323 (2019)
 62. Siddiqui, A.J., Boukerche, A.: TempoCode-IoT: temporal codebook-based encoding of flow features for intrusion detection in Internet of things. *Clust. Comput.* **1**, 1–19 (2020)
 63. Eskandari, M., Janjua, Z.H., Vecchio, M., Antonelli, F.: Passban IDS: an intelligent anomaly based intrusion detection system for IoT edge devices. *IEEE Internet Things J.* **7**(8), 6882–6897 (2020)
 64. Santos, L., Gonçalves, R., Rabadão, C.: A novel intrusion detection system architecture for internet of things networks. In: ECCWS 2019 18th European Conference on Cyber Warfare and Security, Coimbra, Portugal, p. 428. Academic Conferences and publishing limited, 2019
 65. Canuto, L., Santos, L., Vieira, L., Gonçalves, R., Rabadão, C.: CoAP flow signatures for the internet of things. In 2019 14th Iberian Conference on Information Systems and Technologies (CISTI). IEEE, 2019
 66. Leal, R., Santos, L., Vieira, L., Gonçalves, R., Rabadão, C.: MQTT flow signatures for the Internet of things. In 2019 14th Iberian Conference on Information Systems and Technologies (CISTI). IEEE, 2019
 67. Vieira, L., Santos, L., Gonçalves, R., Rabadão, C.: Identifying attack signatures for the internet of things: an IP flow based approach. In 2019 14th Iberian Conference on Information Systems and Technologies (CISTI). IEEE, 2019.

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



Leonel Santos is currently an Assistant Professor at Computer Engineering Department at Superior School of Technology and Management, Polytechnic Institute of Leiria (Portugal). He is a researcher on the Computer Science and Communications Research Center at Polytechnic Institute of Leiria and is a PhD student in University of Trás-os-Montes e Alto Douro (Department of Engineering). He has published 7 papers in conference proceedings and his major

research interests include Cybersecurity, Information and Networks Security, Internet of Things, Intrusion Detection Systems and Computer Forensics.



Ramiro Gonçalves is an Associate Professor with Habilitation at the University of Trás-os-Montes e Alto Douro (Department of Engineering), and Senior Researcher at INESC TEC research center. He has published over 200 articles in indexed journals and event proceedings focusing the Information Systems, Management Information Systems, Software Engineering and Human-Computer Interaction topics. Currently he is supervisor for

several master's degree dissertations and PhD thesis. During his research career, he has participated in several research projects and is currently a member of various research projects aimed at merging information systems and technologies with other fields of study.



Carlos Rabadão is Coordinator Professor at Computer Engineering Department at Superior School of Technology and Management, Polytechnic Institute of Leiria (Portugal). He is the Head of Computer Science and Communications Research Center at Polytechnic Institute of Leiria. He received his PhD degree in Computer Engineering from University of Coimbra (Portugal) in 2007 and his MSc degree in Electronics and Telecommunications Engineering,

from University of Aveiro (Portugal) in 1996. He has published more than 40 papers in conference proceedings and refereed journals, in the areas of Computer Engineering and Communications. His major research interests include Information and Networks Security, Information Security Management Systems, Security Incident Response Systems for Industry 4.0, Next Generation Networks and Services and Wireless Networks.



José Martins is currently an Invited Assistant Professor at the University of Trás-os-Montes e Alto Douro (Department of Engineering), Invited Assistant at the Polytechnic Institute of Bragança and Senior Researcher at INESC TEC research center. He has published over 90 articles in indexed journals and event proceedings focusing the Information Systems, Management Information Systems, Software Engineering and Human–Computer Interaction

topics. Currently he is supervisor for several master's degree

dissertations and PhD thesis. During his research career, he has participated in several research projects and is currently a member of various research projects aimed at merging information systems and technologies with other fields of study. Throughout his professional career José has also worked as an information systems and technologies senior consultant where he directly participated in several international projects. At the present time José Martins dedicates most of his time to his lectures and to his research activities where he tries to understand the variables and (in)direct impacts of ICT adoption at individual and firm levels, and how IS solutions can be idealized, specified and developed in order to fully address their audience needs and requirements.