Internet of Things 7 (2019) 100080

Contents lists available at ScienceDirect

Internet of Things

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

Research article

A security monitoring system for internet of things

Valentina Casola [a,*], Alessandra De Benedictis [a], Antonio Riccio [a], Diego Rivera [b], Wissam Mallouli [b], Edgardo Montes de Oca [b]

[a] Department of Electrical Engineering and Information Technologies, University of Naples Federico II, Naples, Italy
[b] Montimage R & D, Paris, France

ARTICLE INFO

Article history:
Received 30 January 2019
Revised 14 June 2019
Accepted 24 June 2019
Available online 2 July 2019

Keywords:
IoT monitoring
IoT security
IoT threats

ABSTRACT

The wide adoption of the Internet of Things (IoT) paradigm in several application domains has raised new security issues, which should be carefully taken into account to achieve a real benefit from the indisputable innovation potential of IoT. In fact, the heterogeneity of involved technologies, including the integration of different resource-constrained devices and networks, has led to the introduction of new threats affecting all architectural layers and urging for the design and enforcement of adequate security countermeasures, including effective monitoring capabilities. In this paper, we present a monitoring tool for IoT systems based on the extension of the Montimage network monitoring tools. The proposed solution, validated within the H2020 ANASTACIA project, proved to be well suited to monitor IoT-level networks thanks to the exploitation of protocol-specific plugins.

1. Introduction

Internet of things (IoT) is a concept that describes a network of interconnected devices which has advanced capabilities to interact with devices and also with human beings and its surrounding physical world to perform a variety of tasks [1]. In this context, the use of sensors on IoT devices ensures a seamless connection between the devices and the physical world. Indeed, modern IoT devices come with a wide range of sensors (e.g., accelerometer, gyroscope, microphone, light sensor, etc.) which enable more efficient and user-friendly applications [2]. By the adoption of these sensors, IoT devices can sense any changes in their surrounding and take necessary actions to improve any ongoing task efficiently [3]. The ability to sense changes in the physical world have made IoT devices able to make autonomous decisions, whereas efficient communication between the devices and the physical world have made the IoT devices very popular in various application areas: from personal healthcare to home appliances, from big industrial applications to smart cities.

IoT has produced many expectations due to its capacity of transforming physical objects into Internet hosts. However, attackers may take advantage of this great potentiality to threaten privacy and security of users [4]. Denial-of-Service (DDoS) attacks that leveraged Internet of Things (IoT) devices were favored heavily by criminals in the last years.

As a matter of fact, unlike traditional networks connecting embedded nodes or wireless sensor networks (WSNs), typical IoT devices are directly connected to the Internet, and an attacker can get access to the resource-constrained things from anywhere. This possibility makes the things vulnerable to intrusions from both the Internet and the other devices inside the IoT networks.

* Corresponding author.
E-mail address: casolav@unina.it (V. Casola).

https://doi.org/10.1016/j.iot.2019.100080
2542-6605/

According to NETSCOUT Arbor s 13th Annual Worldwide Infrastructure Security Report (WISR)[1], executives took more notice of the cost of cybercrime in the enterprise and struggled with staffing issues that impacted their ability to mitigate threats. Security must be considered of vital importance but, at the same time, ensuring security is mainly obstructed by device computation and power limitations and proper countermeasures cannot be applied as in the traditional ICT systems. Hence, monitoring security acquires even more importance within this critical context.

Intrusion detection systems (IDSs) are one of the most widespread solutions to support security monitoring. Unfortunately, most of the available solutions work on IP-based communications and cannot be easily adapted for IoT technologies and protocols. In fact, the development of an IDS in the IoT domain is still challenging, since the nodes are always accessible and resource-constrained, are connected through typically un-secure links, and use recent (not well established and hardened) IoT protocols such as CoAP or 6LoWPAN. Therefore, it is worth investigating and providing an IDS for the IoT exploiting these opportunities and threats.

In this paper, we present an IoT monitoring solution based on an extension of the *Montimage Monitoring tool* (MMT), a commercial network monitoring tool developed by the Montimage[2] company. The tool has been recently enhanced with a number of security plugins in the context of the MUSA European Project [5–7], and is currently being adapted to the IoT domain within the ANASTACIA European Project [8][3]. This paper discusses the design and development of the MMT IoT extension, named *MMT-IoT*, intended to support the monitoring of the communications over 6LoWPAN networks. The extension entailed the modification and adaptation of several modules and plugins to cope with specific IoT security issues and specific IoT protocols. In particular, suitable packet analysis modules were introduced to enable the correct parsing of the information contained in IoT protocol headers, and novel security rules were added to the detection engine to recognize IoT-specific attack patterns.

The implementation of the MMT-IoT extension has been validated within a real IoT infrastructure, provided by the ANASTACIA research project. We analyzed two different case studies, related respectively to the detection of a DDOS attack and of a SQL Injection attack, and demonstrated that the proposed tool is able to recognize such attacks and generate an alert with a good reliability and within a small time interval even under critical conditions.

The reminder of this paper is structured as follows. In Section 2, we discuss some relevant studies on IoT security issues and available solutions for monitoring. In Section 3, we present an overview of the ANASTACIA project, which was used to validate the proposed IoT extensions to the MMT monitoring tools. In Section 4, we illustrate the main components of the Montimage security monitoring suite, whose extension is presented in Section 5 with focus on the novel components introduced to manage IoT specific protocols. In Section 6, two case studies from the ANASTACIA projects are illustrated to analyze the performance of the proposed tool. Finally, in Section 7, some conclusions and future work are drawn.

2. Related work

The increasing popularity and utility of IoT devices in different application domains made the IoT industry to grow at a tremendous rate. According to a report by Business Insider[4], there will be 30 billion devices connected to the Internet by 2020 and more than 6 trillion dollars will be invested in manufacturing of IoT devices in the next five years.

The use of sensors in IoT devices inevitably increases the functionality of the devices; however, the sensors can also be used as vehicles to launch attacks on the devices or applications. For instance, recently, there have been several attempts to exploit the security of IoT devices via their sensors [9–11]. Attackers can use the sensors to transfer malicious code or trigger messages to activate a malware planted in an IoT device [12], capture sensitive personal information shared between devices (e.g., smartphone, smartwatch, etc.) [13], or even extract encrypted information by capturing encryption and decryption keys. These sensor-based threats can pose significant risk to the IoT systems and applications, as the manufacturers are not yet fully aware of these threats. Indeed, sensor-based threats are becoming more prevalent with time because of the easy access to the sensors and limited security measures that consider these threats [14]. Furthermore, attackers do not need complicated tools to access the sensors, which makes sensor-based threats easier to execute. Hence, trivial execution, easy access to the sensors, and lack of knowledge about the sensor-based threats constitute significant risks for the IoT devices and applications. Understanding these sensor-based threats is necessary for researchers to design reliable solutions to detect and prevent these threats efficiently [4].

Organizations that have a mature cybersecurity strategy work in a holistic way: they focus on all components with a clear shift to the endpoints (e.g. the 'user'), with the adoption of real-time security monitoring and cybercrime prevention approaches, even involving predictive analytics. But still 71% of IoT Enterprise Security Professionals do not monitor IoT devices in real time.

Monitoring IoT security is different from monitoring general IT security. In fact, it relies on other software and hardware components and communication protocols, and the security approach of course encompasses IoT devices.

[1] https://pages.arbornetworks.com/rs/082-KNA-087/images/13th_Worldwide_Infrastructure_Security_Report.pdf

[2] https://montimage.com/

[3] http://www.anastacia-h2020.eu/

[4] J. Greenough, How the internet of things will impact consumers, businesses, and governments in 2016 and beyond, April 2015. [Online]. Available: http://www.businessinsider.com/ how-the-internet-of-things-market-will-grow-2014-10.

Currently, there are no IDSs that meet the requirements of the IPv6-connected IoT, since the available approaches are either customized for Wireless Sensor Networks (WSN) or for the traditional Internet. Some monitoring solutions have been proposed in the literature [15,16], most of them based on WSN monitoring systems or traditional IT systems [17]. Indeed, the peculiar features of IoT as well as WSN systems pose new constraints to the IDS design. In [18], authors proposed a hybrid and distributed IDS for WSN, based on both misuse-based and anomaly-based detection techniques.

Traditional intrusion detection techniques usually rely on information gathered or inferred from network, such as, for instance, header's values or timing between events during message exchanging. However, it might be possible to rely on other sources of information, thus detecting threats by recognizing their side effects. Those effects can be related to anomalous physical parameters consumption or unusual activities on application logs. In [19], the authors propose a lightweight, distributed intrusion detection method based on evaluating, over time, the energy consumption of sensing devices. The proposed system classifies devices with irregular energy consumption as malicious nodes, compared with energy consumption models built for communications in a 6LoWPAN network.

Usually, intrusion detection techniques aim at discovering an ongoing attack. In [20], the authors propose an infrastructure to identify attackers by detecting changes in their neighborhood and reporting to a centralized module, which is deployed on the border router. The centralized module, in turn, is responsible for storing and analyzing this data to identify the possible attackers.

Two interesting projects to monitor IoT devices are worth mentioning: Foren6[5] and SVELTE [21]. Foren6 is a non-intrusive 6LoWPAN network analysis tool that leverages passive sniffer devices to mainly reconstruct a visual representation of network information and support IoT applications. It mainly focuses on visualizing the network topology and analyzing routing issues. SVELTE has been one of the first attempts to develop an IDS specifically designed for the IoT. In SVELTE, the detection techniques mostly target routing attacks (such as sink-hole, selective forwarding, and spoofed or altered routing information), and a hybrid approach to IDSs localization is used (leveraging both centralized and distributed IDSs nodes) to comply with the processing capabilities of the constrained IoT nodes.

MMT-IoT aims at integrating security monitoring capabilities into resource constrained IoT networks. It has been been developed to target the IoT technology, and enables to capture the IoT network traffic near to the IoT devices and to analyze it to detect potential attacks. To the best of our knowledge, no industrial tool is available today on the market that provides such a service.

3. The ANASTACIA project

ANASTACIA (Advanced Networked Agents for Security and Trust Assessment in *CPS*/IoT Architectures) is a European project funded under the *H2020* program and started in January 2017. The main objective of ANASTACIA is to provide a *trustworthy-by-design* security framework addressing all the phases of the system development lifecycle. It leverages the concepts of *Software Defined Networks* (SDN) and *Network Function Virtualization* (NFV) to guarantee security. In particular, it devises the definition of *SDN* forwarding rules specifically designed to provide isolation among IoT services, and proposes to leverage *NFV* by defining security policies that can be updated to dynamically reflect the changes ongoing in the network. Security is applied through a policy-based approach, consisting in the definition of a set of security constraints in an almost natural language and in the translation of these policies into something that can be applied to the physical network, according to the capabilities and configurations of devices. In other words, the objective is to match security and privacy requirements with the actual configuration of the network, making this process transparent to the user.

Fig. 1 depicts the overall ANASTACIA framework architecture, consisting in the *User Level*, the *Seal Management Level* and the *Autonomic Level*.

The *User Level* is the front-end of the platform and includes all the tools needed to manage the system, such as to set privacy policies or check system state. The *Seal Management Level* is responsible for measuring and providing the indicators of the overall security and privacy level. These indicators reflect criticalities that might harm security and privacy of the system. Finally, the *Autonomic Level* provides the core features, i.e. monitoring, orchestration and enforcing services. In particular, monitoring services are provided by the *Monitoring and Reaction Plane*, in charge for analyzing the network traffic to detect suspect malicious activities and to trigger a reaction strategy to mitigate existing threats. The *Security Orchestration Plane* is in charge for mapping security policies to the underlying architecture. In other words, it tries to enforce security policies according to the capabilities of the system and to the devices it is composed of. This mapping includes, for instance, changing device configurations or enforcing *SDN* forwarding rules. Finally, enforcing capabilities are offered by the *Security Enforcement Plane*. This module is in charge for preserving security on the target asset, which implies fulfilling security policies and applying mitigation countermeasures during ongoing attacks. It is further decomposed into the *Data Level*, containing the assets to protect (i.e., the actual IoT network and the network appliances needed for the external and internal communication with the protected asset), and the *Control Level*, including the appliances used to control the networking infrastructure along with the devices devoted to managing the IoT network, such as *SDN* and IoT controllers.

[5] http://cetic.github.io/foren6/

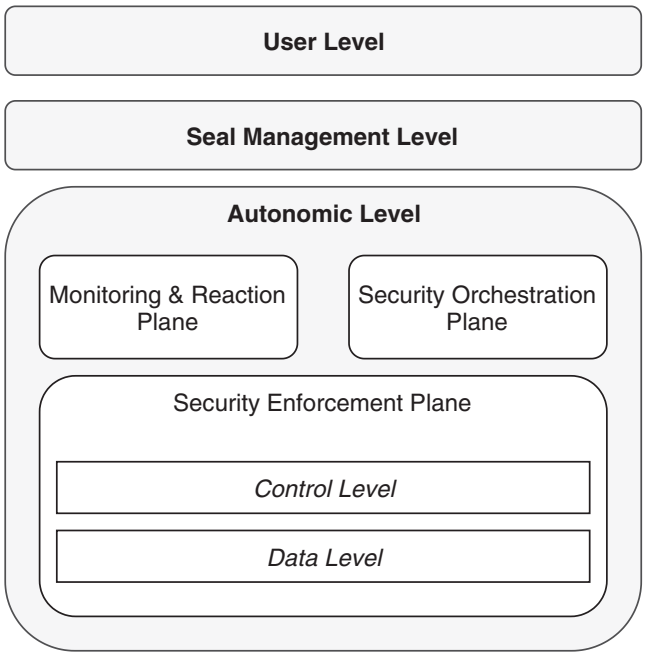

Fig. 1. ANASTACIA framework architecture.

3.1. ANASTACIA monitoring module

As already said, one of the main goals of the ANASTACIA project is to provide security monitoring services able to detect potential security breaches and attacks on cyber-physical networks. In particular, the project leverages both anomaly-based detection techniques relying upon artificial intelligence algorithms to ameliorate the likelihood of detection and specific technical extensions to cope with different IoT architectural layers and technologies.

Fig. 2 shows the main components of the ANASTACIA Monitoring Module, included in the Monitoring and Reaction Plane of the ANASTACIA architecture. The *Data Analysis* component contains an anomaly-based detection tool able to detect suspect changes in the behavior of sensors. The *Data Filtering and Pre-processing* component receives the data from the different available monitoring agents and provides an initial pre-processing and aggregation. The *Incident Detector* is the core component of the module. It collects and analyzes the processed data, raising alerts once a security incident has been detected. Finally, the *Attack Signatures* contains the repository of attacks patterns needed for the signature-based analysis.

In the following subsections, an overview of the technologies adopted by each of the above components will be provided. It is worth noting that, despite the availability of different powerful monitoring tools in the current landscape, several adaptations were required in order to correctly cope with the ANASTACIA use cases.

3.1.1. Data analysis component

The Data Analysis subsystem is represented by an anomaly-based intrusion detection system developed by the UTRC[6] company. The tool builds a profile, or a data-model, for what is considered the 'normal' behavior, using either statistical or unsupervised machine learning methods. Any deviations from this normal behavior are considered possible threats, and are therefore flagged as alarms.

A *profile* is composed of a set of relations, each learned by using constraint learning decision models. The output of the learning process is a series of thresholds, that are used to compare the actual behavior of the system with the nominal one. Consequently, once a value is above or under a certain threshold, the system is considered as potentially under attack. The learning process is dynamic, in the sense that thresholds are continuously updated in order to cope with changes in nominal behavior.

3.1.2. Data filtering and pre-processing component

The Data Filtering and Pre-processing subsystem is designed to collect monitoring information from multiple sources, and to filter and classify this information before making it available for analysis to the Incident Detector subsystem. Data coming from the monitoring agents is stored and collected using a Message Oriented Middleware (*MOM*), i.e., *Kafka*. This

[6] http://www.utrc.utc.com/

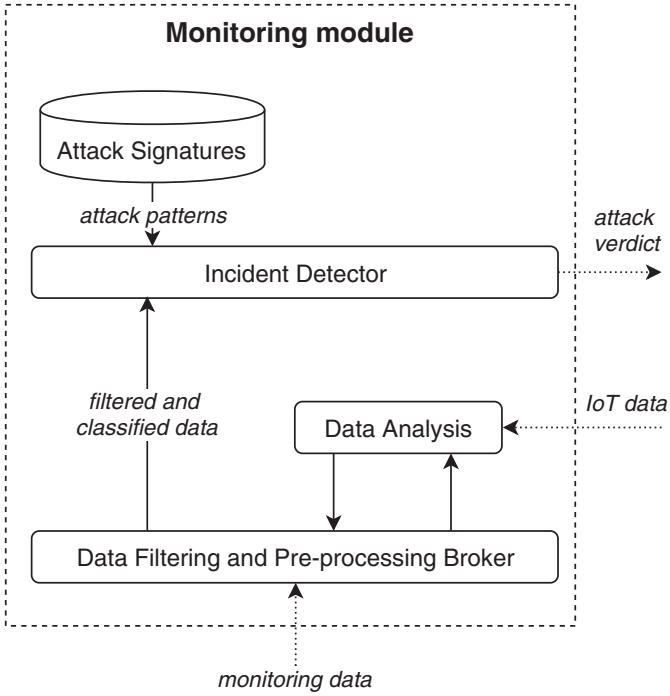

Fig. 2. Monitoring Module Architecture.

MOM provides good performance in terms of reliability and scalability, two important constraints since this broker has a central role, not only for the Monitoring Module, but also for the entire framework.

The Data Filtering module is the component responsible for filtering and pre-processing of information. In particular, pre-processing is needed to set a unique structure for the messages: this includes filling missing fields, setting-up a unique way of formatting data (for example for time or numbers) and, finally, translating the messages according to the *syslog* syntax, required by the Incident Detector. The pre-processing task is pursued through the Apache Storm[7] technology, a distributed stream processing computation framework.

3.1.3. Incident detector component

The Incident Detector subsystem is composed of two modules: the *Montimage Monitoring Tool* (MMT) and the *XL-SIEM*[8]. The latter is analyzed in this section, since the *MMT* tool with its IoT extensions will be described in next sections.

The *XL-SIEM* is a *Security Information and Event Management* (*SIEM*) system developed by the Atos[9] company. It combines the features of the *Security Event Management* (SEM) and the *Security Information Management* (SIM). The segment of security management that deals with real-time monitoring, correlation of events, notifications and console views is the Security Event Management (SEM). The services of long-term storage as well as analysis, manipulation and reporting of log data and security records collected by SEM software is the Security Information Management (SIM). A *SIEM* system gathers outputs from multiple sources, and uses alarm filtering techniques to distinguish malicious activity from false alarms. It differs from an *IDS*, since the latter retrieves information from a single source, either a network or a system, while it does not use correlation techniques in order to detect security threats.

The Atos *XL-SIEM* tool is composed of two parts: the Server and the Agent. The latter serves to gather information from the sources, normalize it and send it to the Server. On the other hand, the Server retrieves these data in order to continue with the actual analysis. An Agent can be instructed to retrieve data from a new source through development of plugins. Fig. 3 represents the internal of the XL-*SIEM* Agent and the plugins developed within *ANASTACIA*.

At the time of writing of this paper, three category of events are considered:

- *AAA* (Authentication, Authorization and Accounting) events.
- *IDS* (Intrusion Detection System) events.
- Anomaly-based analysis.

[7] http://storm.apache.org/
[8] https://atos.net/en/products/integrated-systems#siem-by-bullion
[9] https://atos.net

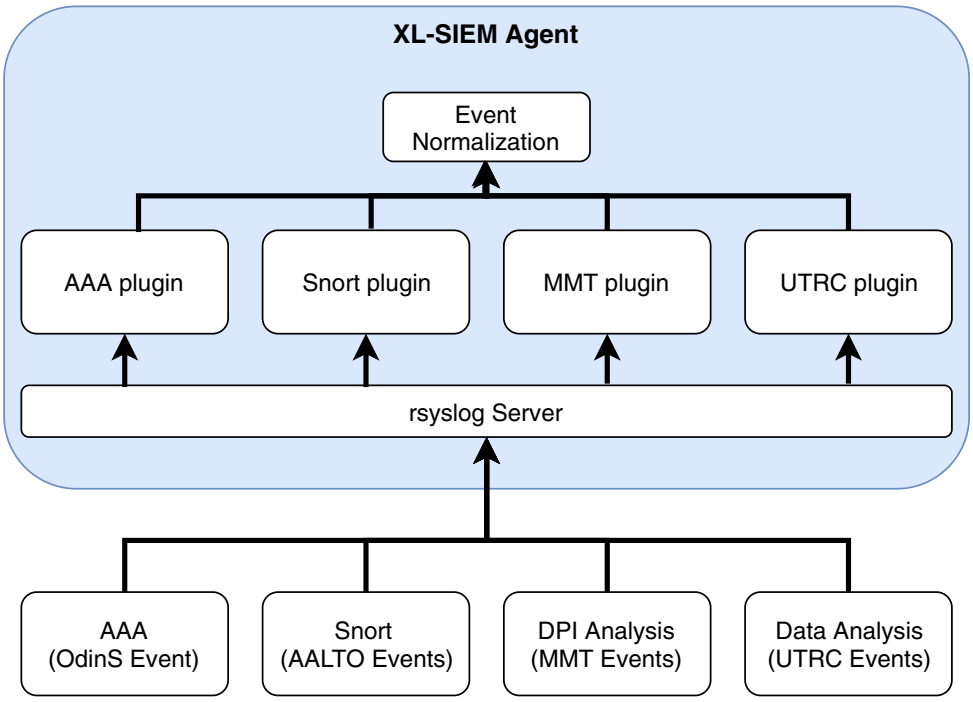

Fig. 3. Atos XL-SIEM Agent.

Each source provides data to the Agent according to *syslog* format, then the Agent normalize those data in order to be sent them to the Server for the actual analysis. The Server receives normalized events, correlates them with information coming from other sources and generate alerts once security breaches have been detected. In order to do that, the Server uses several rules and directives. If a security breach occurs, an alert is forwarded to the Reaction Module, in charge of deciding mitigation actions to be enforced by the Security Orchestrator.

4. Montimage monitoring tool - base architecture

The MMT is a *Network Intrusion Detection System* (*NIDS*) designed and developed by the Montimage company. An NIDS is an intrusion detection system specialized in monitoring traffic to and from all devices on a network. It differs from a *HIDS*, which instead is capable of monitoring and analyzing the internals of a single computing system as well as the network packets passing through its network interfaces.

The software is characterized by a modular architecture that provides flexibility and adaptability according to domain's requirements. As shown in Fig. 4, the MMT architecture includes two main components, namely *MMT-Probe* and *MMT-Operator*. The *MMT-Probe* is in charge for data extraction, carried out by the *MMT-DPI* sub-component, and for security analysis, accomplished by the *MMT-Security* sub-component. The *MMT-Operator* is a web application serving as front-end for the platform, displaying the information collected from *MMT-Probe*.

The *MMT-Probe* is capable of periodically generating statistic reports about the protocols and flows, once the extracted data is aggregated and filtrated. Network parameters as well as the granularity of the statistics reports can be suitably configured. The tool is also able to extract data, either directly from the network interface or from sample files to perform an offline analysis.

The *MMT-DPI* is the core packet processing module, devoted to providing packet inspection capabilities, consisting in extracting a set of information of interest from the network traffic, including protocol field values, network and application parameters. The sub-component, implemented by a C library, supports several protocols and network stacks, each of them implemented in form of a plugin. A plugin contains all the necessary logic to define the structure of the protocol and the mechanisms of extraction. With plugins, it is possible to minimize modifications to the core and focus all the logic in a single place. Moreover, this modular approach allows expanding MMT monitoring capabilities with new protocols, including all the necessary logic in a plugin that would be later compiled along with the whole library.

Security analysis is performed by the *MMT-Security* module, which basically analyzes the data provided by the *MMT-DPI* library and correlates network and application events to detect security incidents. The detection leverages both signature-based and anomaly-based techniques. Signature-based detection techniques mainly look for specific patterns in communication, such as specific byte sequences in network traffic. Although signature-based IDS can easily detect known attacks, it is not possible to detect new attacks, for which no pattern is available [22]. Anomaly-based IDS were primarily introduced to

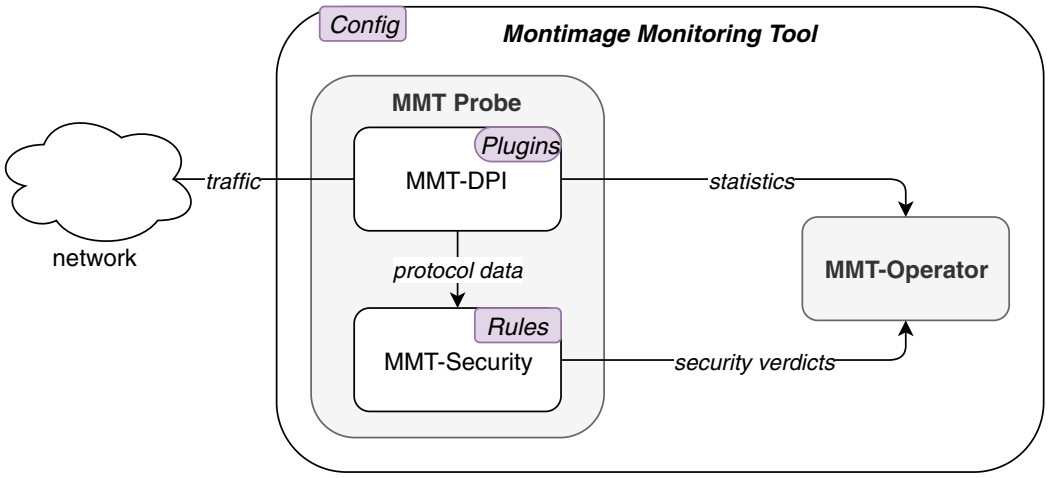

Fig. 4. MMT architecture.

detect unknown attacks. The basic approach is to use machine learning to create a model of trustworthy activity, and then compare new behavior against this model. Although this approach enables the detection of previously unknown attacks, it may suffer from false positives: previously unknown legitimate activity may also be classified as malicious. Signature-based detection in *MMT-Security* makes use of rules in order to describe the patterns of security vulnerabilities. These rules are expressed in XML language and can be loaded either statically, that is at compiling time, and dynamically, that is at execution time. In this way, new rules might be added whenever a new threat is discovered and without stopping the execution of the tool. With regard to anomaly-based detection, artificial intelligence techniques are used to enable the detection of new attack vectors exploiting for example 0-day vulnerabilities. On the other hand, anomaly-based detection is performed by the *MMT-Behaviour* module, included as part of *MMT-Security*. This module employs artificial intelligence techniques for detection of security vulnerabilities. The component can be included optionally in the *MMT-Security*, according to the configuration required. The advantages of using those techniques can be seen regarding detection of new attack vectors, that exploits 0-day vulnerabilities for instance.

4.1. Packet inspection

The *MMT-DPI* module contains all the logic for parsing protocols in form of plugins. Each plugin is self-contained, including all the necessary logic to describe header structure and extraction procedures. As explained in detail later, in order to add IoT-related packet inspection functionalities, we developed a plugin for the main IoT protocols. The first step toward the creation of a plugin consists in defining a structure containing the specification of all header fields and a link to the function in charge of the actual extraction for that field. Then, the next step requires the implementation of a classification function. Protocol classification, in fact, is the main task to complete when analyzing a header and consists in assessing a protocol's identity by comparing the arrangement of the header bytes with known patterns. Classifying a protocol is a task that differs significantly based on the considered protocol: it might range from a simple checking of the transport port to the adoption of heuristics aiming to guess an encrypted header.

A classification function may be further specified in order to define which is the lower layer protocol, in the cases it is fixed. Moreover, if some operations are needed before or after the classification for reasons related to, for instance, reducing the complexity of the function or improving the performance, the classification task can be exploded in more sub-tasks.

To summarize, in order to implement a plugin it is necessary to define at least three elements: the structure of the protocol header to be inspected, the classification function and, finally, all the required functions for the extraction.

4.2. Signature-based detection

As anticipated, the *MMT-Security* module implements signature-based detection techniques. Each security incident is characterized by a pattern, represented by a particular sequence of messages, commands or header values. In the *MMT-Security* module, an incident is described in form of a security rule, which specifies a sequence of events.

Conceptually, a security rule could be structured as a tree [23]. According to Fig. 5, the root node represents the type of the rule, which can be either *security* or *attack*. A *security rule* specifies the sequence of events that have to occur in order to enforce a security feature of the system. On the other hand, an *attack rule* specifies a sequence of not allowed events, which basically represents the signature of an attack (e.g., they could be related to some malformed header fields or to anomalies in messages payload). A security or attack rule is composed of two parts (the two children nodes on the first level of the

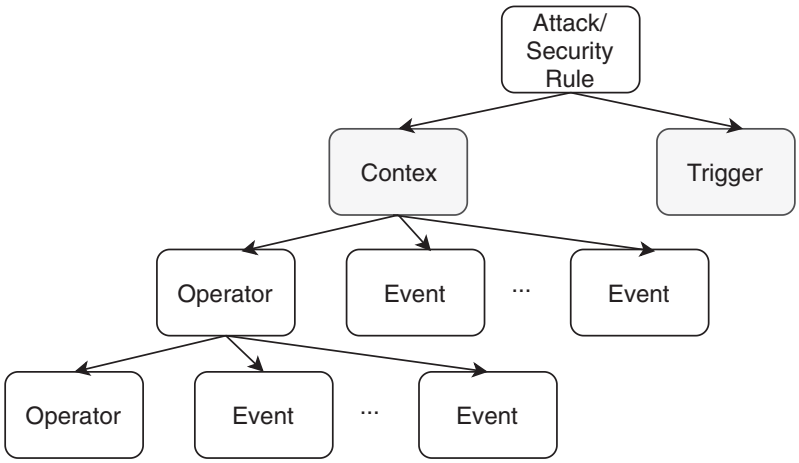

Fig. 5. Security rule structure.

tree), namely the *context* and the *trigger*, which represent the underlying conditions that must hold when a specific event or sequence of events (the trigger) occur, in order to have a rule verified.

Both the context and the trigger of a rule can be composed of *operators* and *events*. Events express a pattern identifying a rule: for a security rule for example, an event may express the use of encryption or of a particular hash technique, while for an attack rule it may express the detection of bad header values or anomalies in the payload. Operators are adopted when the context or the trigger are composed of more than one events, and specify how these events should be combined (for example, an operator may express a chronological or positional order among events).

The adoption of the concepts of context and trigger allows for a more precise evaluation of rules, helping reduce the chance of getting false positives. In the case of attack rules, having the context verified does not mean that the corresponding attack has occurred if the related trigger is not verified as well. Conversely, in the case of a security rule, having either the context or the trigger not verified would certainly lead to a security breach, since the related events must all happen to correctly enforce a specific security feature of the system. Furthermore, the adoption of this approach based on context and trigger allows a rule to be checked on multiple packets or on multiple sessions. For instance, with this approach it would be possible to rise an alert only after that a certain amount of packets, say ten, have passed through the network. In this case, the context would be valid for the previous nine packets while the trigger would be valid only on the final packet and then validate the rule, generating an alert. In conclusion, rules can be applied to a broad range of cases, which are the following:

- Single packet.
- Multiple packets, single session.
- Multiple packets, multiple sessions.

In the first case, context and trigger would be verified on the same packet while, in the second case, there would be several packets, all belonging to the same session, which contribute to the validity of the single rule. Finally, the last case involves information coming from several packets belonging to different sessions in order to verify the validity of a rule. It is important to note that a context can occur before, at the same time or after a trigger, depending on the temporal constraints that have been set.

5. MMT extensions to support IoT

The MMT does not provide native support for IoT, therefore it has been extended in order to also monitor specific IoT assets. In particular, the proposed extensions include novel capabilities in terms of IoT traffic capturing and analysis from a security perspective. The purpose of this section is to describe the design and implementation of the devised extension to the MMT for IoT environments, named *MMT-IoT*.

The design of the MMT-IoT architecture faced several challenges, mainly related to traffic capturing and analysis activities. Fig. 6 illustrates the high-level architecture of the MMT-IoT tool: as shown, the *IoT Analyser* module whose added to the base MMT architecture. This module, whose primary goal is to provide isolation with all platform-dependent details, is composed of two sub-modules: *MMT-IoT Sniffer* and *MMT-IoT Bridge*. Beside the introduction of the *IoT Analyser*, we also extended the *MMT-DPI* module with additional plugins to support IoT specific protocols.

As illustrated in Fig. 7, before the IoT traffic is analyzed by the*MMT-IoT-Probe*, it passes through the *IoT Analyser*. In particular, the traffic is captured by the *MMT-IoT Sniffer* and encapsulated by the *MMT-IoT Bridge*, before being mirrored to the MMT.

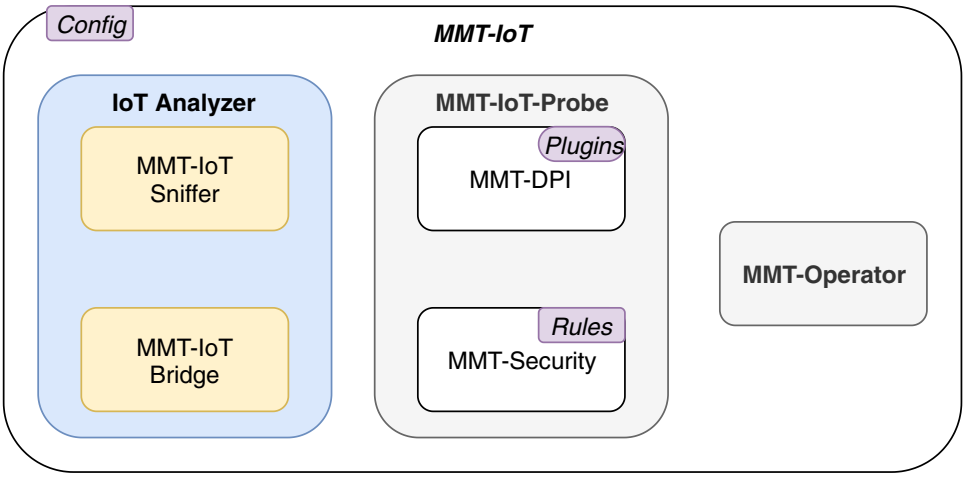

Fig. 6. MMT-IoT high-level architecture.

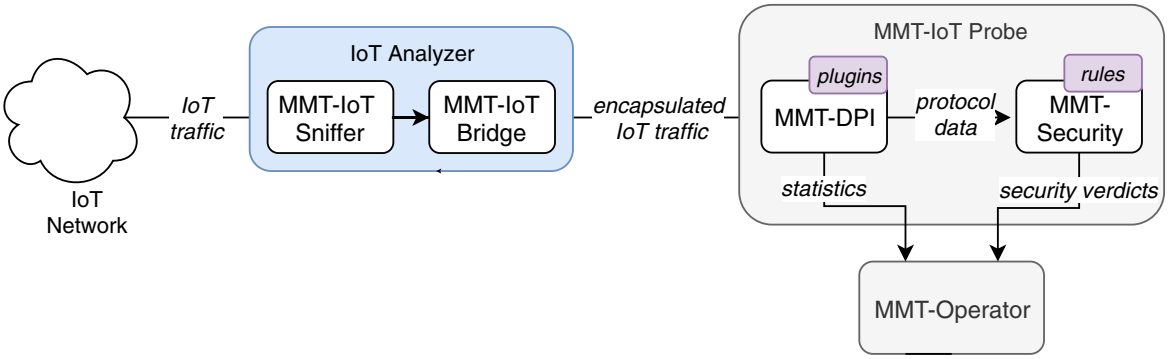

Fig. 7. MMT-IoT main interactions.

The implementation of the MMT-IoT modules has been partially influenced by some technological constraints introduced by the ANASTACIA Project in terms of the need for supporting different and heterogeneous operating systems for IoT devices. In fact, MMT provides native support for *Linux* and *Windows* OSs, while sensors belonging to the ANASTACIA infrastructure run firmware based on *Contiki* OS. The *Contiki* OS is among the most diffused operating systems for the IoT, with large support from the community.

To face this problem, two solutions were initially considered: either to develop a porting of MMT to *Contiki* or to create a native *Contiki* firmware that mirrors the traffic to MMT, which would still be running on a *Linux/Windows* machine. From the performance perspective, porting MMT to *Contiki* would minimize latency, since the analysis would be directly done on devices, although computational power demand would be significant for a device with limited capabilities. Indeed, a higher computational load would narrow the set of devices supporting the software, since the demand would be too high for an average constrained device. Moreover, there would also be problems in power consumption. On the other hand, developing a native *Contiki* firmware would sacrifice latency, because traffic is not analyzed locally at the IoT network, but mirrored to another place for remote processing. Conversely, this solution would improve the efficiency in terms of power consumption, since the workload for the IoT devices would be lighter.

The development effort, along with the heterogeneity of involved IoT operating systems, are the main reasons that have led us to the adoption of the second solution, that is, the development of a *Contiki* firmware that mirrors the traffic to the MMT. Next subsections will illustrate further details.

5.1. MMT-IoT sniffer

The *MMT-IoT Sniffer*, sketched in Fig. 8, implements the traffic capturing capability of the *MMT-IoT*. As anticipated, we developed a firmware for IoT devices devoted to capturing IoT traffic and forwarding it to the *MMT-IoT Probe* for further processing. In particular, the firmware listens to the wireless channel by setting the network interface in promiscuous mode: once a 6LOWPAN packet is received from the radio interface, the network driver notifies to a handler process that a packet is

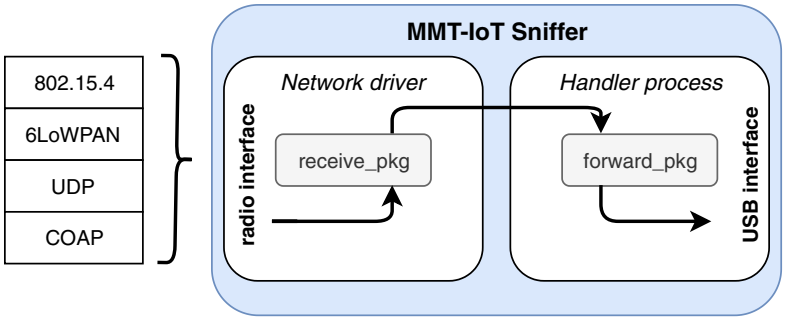

Fig. 8. MMT-IoT Sniffer Architecture.

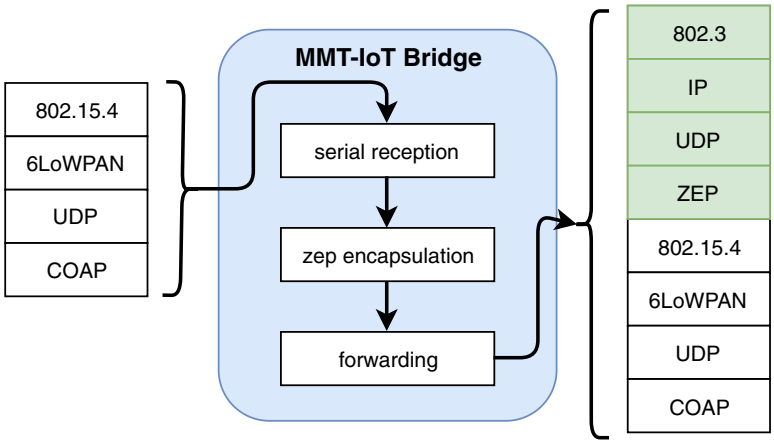

Fig. 9. MMT-IoT Bridge architecture.

ready to be processed through a *callback* mechanism. Then, the handler process sends the traffic out through a USB interface. In particular, the SLIP library is used to fulfill this task.

5.2. MMT-IoT bridge

Since *MMT-IoT* must be capable of analyzing both the IoT traffic and the traditional IP traffic, the heterogeneity of involved technologies must be properly handled. In particular, our solution devises the adaptation of 6LOWPAN packets to IP packets, performed by the *MMT-IoT Bridge*, depicted in Fig. 9, which covers the mirroring capabilities of the *Packet Analyser* component.

The *MMT-IoT Bridge* module uses the SLIP library to capture a packet from the serial interface, applies an IP encapsulation and forwards the encapsulated packet to the *MMT-IoT-Probe* through a network interface. The encapsulation employs ZEP, one of the most adopted protocols to send 802.15.4 packets over traditional 802.11/802.3 networks [24]. Encapsulated traffic is transmitted through *raw sockets*, i.e., network sockets that enable direct sending and receiving of IP packets without any specific transport-layer formatting [25]. While traditional sockets automatically encapsulate the payload according to the chosen transport-layer protocol, raw sockets are unaware of the existence of protocol headers, which are simply broadcast with the payload. This enables to craft packet headers during transmission, and turned out to be useful to add an IP header to the encapsulated packets coming from the IoT network.

5.3. MMT-DPI

The extensions provided at the *MMT-DPI* component mainly regard the development of new plugins needed to correctly parse IoT network packets and to extract meaningful data for the analysis.

In particular, suitable plugins have been written for IEEE 802.15.4, 6LoWPAN and COAP protocols (other protocols, such as UDP and IPv6, are already supported by the MMT base architecture). The development process faced several challenges, mainly related to compression mechanisms adopted in the considered protocols. In fact, being the 802.15.4 frame length limited to only 127 bytes, compression techniques have to be employed in order to reduce the header length of those protocols which have not been initially designed for IoT networks, such as IPv6, TCP and UDP and, at the same time, to guarantee that the deleted information could be reconstructed once the packet arrives at destination. These techniques

compress the 40-byte IPv6 and 8-byte UDP headers by assuming the usage of common fields, or elide some fields if they can be derived from the link layer header. Compression mechanisms for TCP are not defined so far [26], but the protocol can still be used in a 6LoWPAN network. The deletion of compression of some header fields makes the length of headers variable. This brings to several challenges when defining the dissection logic. In fact, the first step of the development phase has been devoted to an exhaustive analysis of all the possible combinations of header fields and lengths. Once all the patterns have been enumerated, it has been possible to compute the right header length, required for identifying the beginning of the next protocol header.

The following subsections describe the main development challenges associated with each protocol plugin.

5.3.1. Plugin for IEEE 802.15.4

IEEE 802.15.4 is a technical standard which defines the operation of *low-rate wireless personal area networks* (LR-WPANs). It specifies the physical layer and media access control for LR-WPANs, and is maintained by the IEEE 802.15 working group. It can be used by 6LoWPAN, the technology used to deliver the IPv6 version of the Internet Protocol (IP) over WPANs. The IEEE 802.15.4 standard has undergone several revision stages since its first release in 2003. The latest release, dated in 2015 and identified as IEEE Std 802.15.4-2015 [27], introduced the presence of PAN addresses in the frame. Different combinations of several header fields indicate the presence of these address fields. Moreover, the latest release was specified with the goal of maintaining backward compatibility with the previous releases. In order to do that, the header was enriched with a value, the *Frame Version* field, that is a 2-bit value to indicate which version of the standard the frame is compliant with.

The 802.15.4 plugin developed within the *MMT-DPI* module was designed to enable the correct management of the information included in the packet headers based on the Frame Version field, including the management of all the possible combination of fields related to the presence of PAN addresses in the frames. Furthermore, an accurate analysis of 802.15.4 packets led us to discover that, instead of using the traditional *network byte order* convention based on *big endian* encoding, data inside the 802.15.4 headers are codified using *little endian* scheme [28]. This peculiar feature, not mentioned in the standard, was properly handled by our plugin.

5.3.2. Plugin for 6LoWPAN

6LoWPAN stands for *IPv6 over Low-Power Wireless Personal Area Networks* and is the name associated with a concluded IETF working group. This working group has defined encapsulation and header compression mechanisms that allow IPv6 packets to be sent and received over 802.15.4 networks. As already said, compression mechanisms are used in order to adapt protocols not specifically designed for IoT to constrained networks. Those mechanisms compress or elide several header fields in order to reduce the number of bytes required to encode protocol headers. IPv6 source and destination addresses are among those fields. In the original protocol, these addresses are 127 bit or 16 bytes long. Therefore, including their full length would occupy a significant part of the available MTU, without counting other header fields supported by 802.15.4, that admits a maximum 127 bytes-long MAC frame. As a consequence of these compression mechanisms, the network address could be present in variable length or, in some cases, it may be completely absent, provided that it can be obtained through shared-context information.

Although these compression techniques allow for an efficient transmission of IPv6 packets over 802.15.4 networks, they pose several challenges regarding the address reconstruction logic. Due to this reason, most of the effort has been spent on network address reconstruction, especially in the case of a completely elided addresses, where the information about the context have to be retrieved. In fact, although RFC 6282 [29] illustrates where to find context data in the header structure, it does not specify how to use this context in the reconstruction process. Consequently, even in this case, analysis of the raw data, byte per byte, provided needed information to cope with the problem.

Finally, compression mechanisms can use link-layer addresses to compose the network identifier. This feature required additional effort to develop the dissection logic, since it was necessary to share information between plugins, partly losing their modularity features.

5.3.3. Plugin for CoAP

COAP is an application protocol tailored for constrained devices and networks with critical bandwidth and high error rate such as 6LoWPAN [30]. COAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specific requirements such as multicast support, low overhead, and simplicity. It is specified by the IETF standard RFC 7252 [30] and encapsulated using UDP, even if other channels like TCP or DTLS might be used as well.

Regarding this protocol, the development effort focused mainly on two aspects: protocol classification and options parsing. The latter has been challenging in relation to defining an efficient approach, in terms of execution time, to the parsing. The task has been supported by the RFC 7252 [30] that explains, in a clear way, the structure of the options.

Protocol classification is the first step when analyzing a header. It consists in assessing a protocol's identity by confronting the arrangement of bytes in the packet with known patterns. Classifying a protocol is a task that differs significantly among protocols. It might range from a simple check of the transport port to the adoption of heuristics in presence of encrypted headers. The latter is the case in which the classification of COAP falls. At this stage of implementation, a simple classification rule has been adopted, assuming that COAP packets could be found only on 802.15.4 networks. This choice enables to classify a header as COAP if, besides checking other COAP header fields, the link layer protocol would correspond to IEEE 802.15.4.

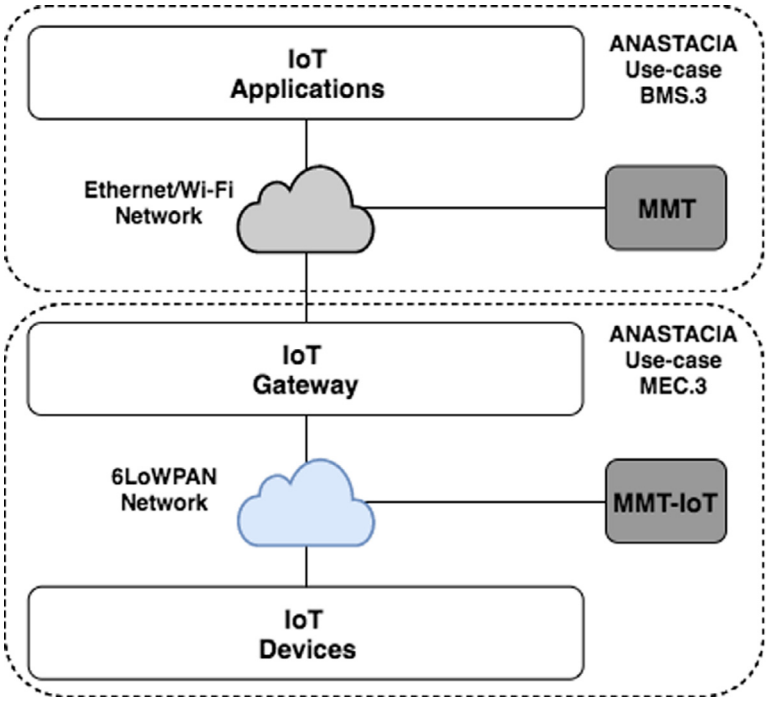

Fig. 10. Test-bed architecture.

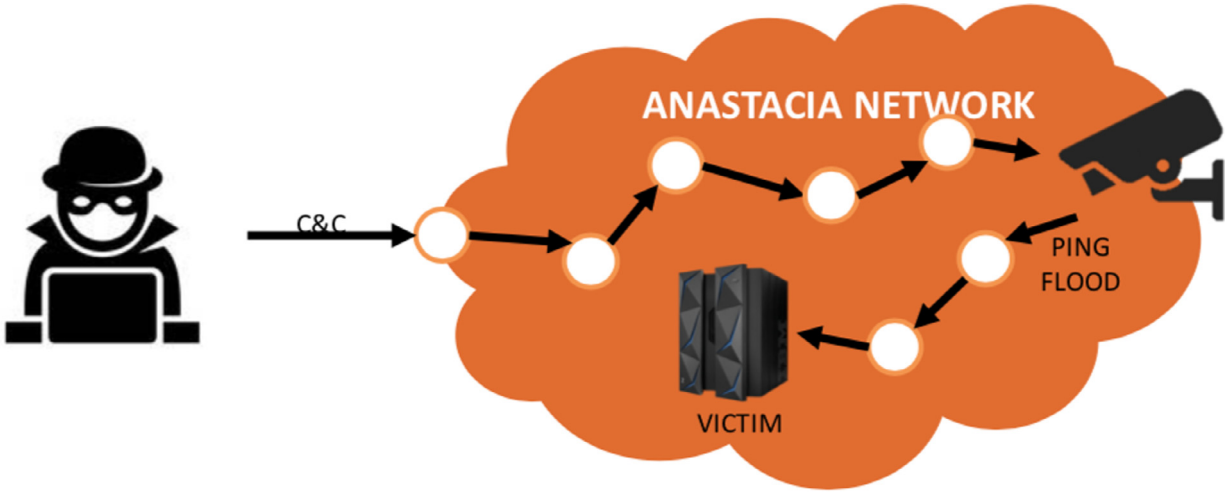

Fig. 11. DDOS Scenario.

6. The ANASTASIA case study

In order to validate the proposed MMT-IoT extensions, we exploited two different use-cases and the deployment infrastructure of the ANASTACIA project. In the developed testbed shown in Fig. 10, the MMT and MMT-IoT tools have been respectively placed in the back-end network, to monitor Ethernet/Wi-Fi networks and in the front-end network, to also monitor IEEE 802.15.4 network.

6.1. Use Case 1: DDoS attack to smart cameras

The First scenario, depicted in Fig. 11, consists of a DDOS attack aiming to disrupt services provided by an edge server, which is in charge of elaborating video signals coming from a series of smart-cameras. The attack starts with the attacker taking possession of a group of smart-cameras, thus forming a botnet. Then, the attacker instructs the smart-cameras in order to flood an edge server with an abnormal number of ICMP echo requests. The aim of the attacker, in this case, is to

```
<operator value="THEN" delay_units="s" delay_min="0" delay_max="1"
>
    <!--Context-->
    <event value="COMPUTE" event_id="1"
           description="Context: ICMP ping"
           boolean_expression="((icmpv6.type == 128) &amp;amp;&amp;amp; (
lowpan.iphc_dst == lowpan.iphc_dst))"/>
    <event value="COMPUTE" event_id="2"
           description="Context: 2nd ICMP ping"
           boolean_expression="((icmpv6.type == 128) &amp;amp;&amp;amp; (
lowpan.iphc_dst == lowpan.iphc_dst.1))"/>
  <!--Trigger-->
  <event value="COMPUTE" event_id="3"
      description="Trigger: 3rd consecutive ICMP ping packet"
      boolean_expression="((icmpv6.type == 128) &amp;amp;&amp;amp; (lowpan.
  iphc_dst == lowpan.iphc_dst.1))"/>
```

```
<property value="THEN" delay_units="s" delay_min="0" delay_max="1"
  property_id="60" type_property="ATTACK"
  description="3 consecutive ICMPv6 echo packets in a second. Possibly
  ICMP echo flood.">
  <!-- Context -->
  <!-- Trigger -->
</property>
```

Fig. 12. ICMP Flood Attack: Security Rule, Context and Trigger.

create disservice on the network and make the surveillance service unavailable to its intended users. It is assumed smart-cameras are already part of the botnet.

6.1.1. Security configuration

The detection strategy depends on some assumptions we do on the attacker model and capabilities. The strategy should not rely on the source of the attack since the attacker might spoof the IP source of the packets. An attacker might also change source IP address either periodically, e.g. each 500 ms, or for each packet. Those techniques add further difficulties when detecting the attack. Indeed, in the first case, the detection technique would work until the attacker changes the IP address, while in the second one no attack would be detected. For this reason, it is reasonable to rely on the IP destination address, instead of the IP source address of the packets.

Legitimate traffic is another aspect that has to be considered as well: ICMP *echo* requests are usually used for checking the reachability of a remote node, therefore this traffic must be allowed somehow. One approach is allowing this traffic but limiting its available bandwidth. In particular, we proposed to allow a maximum of *ten* requests every five seconds, permitting only two requests per second to the same destination. Such trade-off still lets legit traffic flow through the network and, at the same time, detects DOS or DDOS on the victim, since the maximum allowed bandwidth is extremely low, compared to the bandwidth needed to successfully lead a DOS.

In conclusion, the detection strategy might be summarized as follows: *no more than two ICMP echo requests are allowed towards the same destination in a second*. In other words, observing at least three consecutive ICMP *echo* requests to the same destination should raise an alert if all requests fall in less than a second. In contrast, there should not be an alert when the same requests are spread over more than one second.

In terms of *MMT-Security*, this situation corresponds to a security rule composed by three events, being each of them a detection of an ICMP *echo* request. However, to raise an alert all these events must happen in less than one second. A security rule is composed of a context and a trigger. In this case, the *context* of the rule is related to reception of the first two requests, where the IP destination address of the second echo request must match the destination of the first one. As it was mentioned before, the attacker might spoof the source address (in the case of DDOS attack), so the rule does not introduce any constraint regarding the source IP address of the echo requests.

Fig. 12 specifies the *context* of the security rule. Note that an ICMP *echo* request is identified by a value of 128 within the *type* field of the ICMP header. While the *trigger* of the rule corresponds to receiving a third ICMP *echo* of the same characteristics of the message related to the second event.

Nevertheless, in order for the *trigger* to be valid, this event must happen in less than one second related to previous events. Once the *trigger* becomes true an alert is generated. In order to verify this scenario, the whole environment has been tested under different types of traffic, either legitimate or malicious, in order to quantify false positive and negative rates. The results are presented later in this section.

6.2. Use Case 2: SQL injection attack

In the second scenario illustrated in Fig. 13, an attacker exploits a web page vulnerability to inject malicious SQL code in order to access or manipulate a SCADA database that is used to manage an energy micro-grid. Such exploitation leverages a security vulnerability in an application's software running on a controlled IoT device: namely the lack of incorrect user input filtering for string literal escape characters, thus allowing maliciously crafted queries to be executed on databases and letting an attacker obtain complete control on stored data. It is assumed the attacker has already taken control of the requesting device.

Fig. 13. SQL Injection Scenario.

```
<!--Context-->
<event value="COMPUTE" event_id="1"
        description="Context: a CoAP request"
        boolean_expression="((coap.class == 0) &amp;amp;&amp;amp; (ipv6.dst
== ipv6.dst))"/>
 <!--Trigger-->
 <event value="COMPUTE" event_id="2"
        description="Trigger: SQL statements in the payload"
        boolean_expression="(#em_check_sql_injection_on_coap(coap.
 payload) == true)"/>
```

```
<property vvalue="THEN" delay_units="ms" delay_min="0" delay_max="0"
  property_id="61" type_property="ATTACK"
  description="SQL Injection detected in CoAP payload">
  <!-- Context -->
  <!-- Trigger -->
</property>
```

Fig. 14. SQLInjection Attack: Security Rule, Context and trigger.

6.2.1. Security configuration

Here, the main goal of the detection strategy is to check the presence of those statements in packets payload in order to detect a possible attack. It is important to remark that only requests might contain this type of threats since they are issued by clients, therefore it would be more efficient to check only the payload of requests instead of responses if any. Remember that the application protocol adopted by ANASTASIA testing sensors is the COAP.

A security rule for this kind of threat would be composed by two events that must occur at the same time. In other words, they must be both valid in the scope of a single packet. The first event, whose code is shown in Fig. 14, is the context of the rule and it is related with the detection of a COAP request. A COAP request is a message of class 0. Hence this is the first parameter to be checked. The second condition has been added for performance reasons related to MMT internals.

The second event, the trigger of the rule, is verified by checking requests payload looking for the SQLstatements described previously. Fulfilling this task requires more than just checking header values, therefore an *embedded function* has been developed. In particular, this function scans the payload byte per byte searching for one of those SQL statements that have been previously explained. In order to be more general, those statements are expressed in a general form in order to cope with slightly variations. The embedded function returns a boolean value indicating the response of the search. If at least one of those patterns is found in the payload, the *trigger* is valid and an alert is raised, indicating an ongoing attack.

Finally, both the context and the trigger are combined inside the *property* tag to create the MMT-Security Rule shown in Fig. 14. As for the previous attack scenario, the whole environment has been tested under different types of traffic, either legitimate or malicious, in order to quantify false positive and negative rates and the results are presented in the following section. Notice that more attacks (Command and Control based attacks, Data exfiltration attack etc.) are also studied but not reported in this paper.

6.3. Evaluating the performance of the MMT-IoT

We set up a benchmark to simulate the performance and response time. The simulation uses the Cooja Simulator environment, which is part of the Contiki OS[10]. This emulated environment allowed us to deploy virtual IoT devices – based on Contiki – with modified firmwares that implemented both the attackers and the MMT-IoT Sniffer solution. Moreover, the Cooja emulation environment is also capable of emulating the spatial disposition of the devices, making it suitable to select the position of the sniffer to maximize the extracted packets.

In order to evaluate the performance and the effectiveness of the MMT-IoT engine, we have defined a methodology that aims to stress the solution and to determine the detection times using the architecture described here. As mentioned before, these experiments were conducted using the Cooja emulator software, allowing an easy instantiation of the developed tools.

[10] http://www.contiki-os.org/

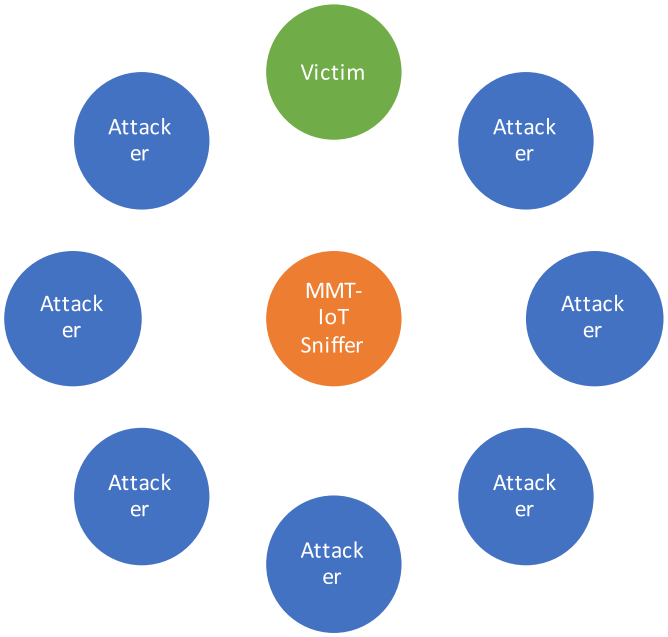

Fig. 15. Spatial disposition of the emulated nodes.

6.3.1. Testbed configuration

When working with IoT networks, the physical topology – i.e. the spatial disposition – plays a fundamental role, since the radial range of each device is limited. In this sense, a random disposition of the devices does not guarantee that all devices are reachable in a single hop. We cope with this issue manually deploying the devices in a star disposition, as shown in Fig. 15. With this topology, we ensure that each attacker (blue nodes) can reach the victim (green node) since both nodes are in range, ensuring that the malicious packet will not go through a third node using the IoT routing protocol (RPL).

This disposition is also helpful to maximize the traffic extracted by the MMT-IoT Sniffer and, therefore analyzed by MMT-Security. The center of a star disposition is a point that is in range of all other devices in the topology. This feature makes the center an ideal point to place the MMT-IoT Sniffer, which will be capable of extracting all the packets that are generated by the attackers. Considering that one of the objectives of the experiment is to perform scalability tests on MMT-IoT Sniffer, we placed this element in the center of the disposition, in order to maximize the traffic extracted from the network.

Finally, the sparsity of the disposition can also play a fundamental role in the general performance of the network. In order to avoid relying on routing protocols, all the devices have been positioned close enough to be in range, including the victim device. In this sense, the malicious packets can arrive from attackers to victim with a direct wireless communication, avoiding the delivery of the packet via other nodes. All the spatial considerations can be summarized in the disposition shown in Fig. 15.

One of the principal objectives of this study is to perform a benchmarking about the extraction tool MMT-IoT. In this sense, it is important to establish a methodology to steadily increase the amount of traffic in the network and the number of attacks triggered. By using the Cooja emulator we created different scenarios each one containing a different amount of attackers, thus emulating different amounts of network traffic.

To this end, we created 10 scenarios each one containing a fixed amount of attackers varying from 1 to 10. Each test case also contains an instance of the MMT-IoT Sniffer and a RPL router device, serving both as victim of the attack and as router to the outside network (which in this case, is an internal network emulated on the Cooja host machine).

Finally, the spatial considerations explained before were also integrated in each one of the use cases. In each one of the emulated scenarios we distributed the attackers and the victims as depicted in the star disposition. In addition, the MMT-IoT sniffer was placed in the center of the topology, in order to extract and analyze the traffic from the most of the devices possible. In Fig. 15 a disposition with 7 attackers is shown, however, all the other scenarios follow the same spatial organization but varying the amount of attackers present in the emulation.

6.3.2. Execution of the tests

For each use-case, the traffic generator has been configured to generate both nominal conditions traffic (to verify the occurrence of false positives) and attack scenarios (to verify the behavior in terms of false negatives). In this sense, the nominal traffic comprised the common packets interchanged due to the RPL routing protocol and their respective ACK answers.

To help the execution of the use cases, the emulations were compiled and packaged with Cooja in order to generate a self-contained, automatic and executable file of each scenario. The entire simulation was scheduled to last 37 s, which

```
*REF*       ::212:7402:2:202          acac::1                   ICMPv6          70 Echo (ping) request
0.021000    ::212:7402:2:202          acac::1                   ICMPv6          70 Echo (ping) request
0.043000    ::212:7402:2:202          acac::1                   ICMPv6          70 Echo (ping) request
0.044000    ::212:7402:2:202          acac::1                   ICMPv6          70 Echo (ping) request
0.045000    ::212:7402:2:202          acac::1                   ICMPv6          70 Echo (ping) request
0.064000    ::212:7402:2:202          acac::1                   ICMPv6          70 Echo (ping) request
```

Fig. 16. DDOS: Malicious Traffic Generation.


```
"attack","3 consecutive ICMPv6 ping packets in a second. Possibly ICMP ping flood.",
    "event_1":{"counter":8,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"acac::1"}]},
    "event_2":{"counter":10,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"acac::1"}]},
    "event_3":{"counter":12,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"acac::1"}]}
"attack","3 consecutive ICMPv6 ping packets in a second. Possibly ICMP ping flood.",
    "event_1":{"counter":10,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"acac::1"}]},
    "event_2":{"counter":12,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"acac::1"}]},
    "event_3":{"counter":14,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"acac::1"}]}
"attack","3 consecutive ICMPv6 ping packets in a second. Possibly ICMP ping flood.",
    "event_1":{"counter":12,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"acac::1"}]},
    "event_2":{"counter":14,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"acac::1"}]},
    "event_3":{"counter":16,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"acac::1"}]}
```


Fig. 17. DDOS: Attack Detection.

leaves time to trigger the attack 3 times. In particular, the script waits 20 s to setup the emulation and let the network converge (due to the RPL routing protocol), then it starts the extraction of the packets (by emulating pressing the button on the MMT-IoT sniffer device), waits 2 more seconds and then triggers the attack (by pressing the buttons on all the attackers) each 5 s of the emulation.

In the case of the DDoS, the attack triggered was the ping flooding attack. Each time the attack was triggered, an attacker was commanded to send 1 ping packets each 0.1 s for a total of 10 packets, as depicted in Fig. 16. According to the definition of legit traffic given in Section 6.1.1, this should be considered an attack, since the MMT technology will extract and observe the 10 packets within a second, violating the security rule. Moreover, considering that the security rule has been programmed to detect 3 successive ICMP packets. In this sense, it is possible to detect the attack at most 3 times in the same trigger. This fact will be used later when analysing the effectiveness of the MMT-IoT solution.

In order to automatize the execution of the software, we used a script that took in charge the execution of the precompiled emulations. To do so, the script was configured to execute the emulation, connect the required software (the IoT router software and the MMT-IoT bridge) and start the MMT-Probe. After this, the script will extract the benchmark information from the logs generated by each tool.

Finally, each emulated scenario was executed 10, in order to obtain the averages of the number of attacks detected and the detection times.

6.4. Analysis of results

As mentioned before, to verify the correct detection of attacks and extract the benchmark information, we analysed both the Cooja emulation and the *MMT-Security* logs. Whenever an attack is triggered, we generate a message in the Cooja log which contains a timestamp w.r.t to the start of the emulation, allowing the computation of the exact timestamp when the attack was triggered. On the other hand, when the attack has been detected, MMT Security generates a report in the *csv* format, containing information about the alert: timestamp of the report, network parameters, content of the report itself, among other.

Fig. 17 shows an extract of the MMT-Security reports illustrating the correct detection of the flooding attack, in relation to the DDOS use-case. As can be noted, each row describes the events which compose the context and trigger of the security rule. In particular, each event represents an ICMP packet observed in the network, showing its respective destination IP address. It is important to remark that according to the security rule explained in Section 6.1.1, the alert of the attack is risen when the third packet is observed, so the reported timestamp in the *csv* line will correspond to the detection timestamp of the attack.

The detection times extracted are given in Table 1, together with other statistics, including minimum, maximum and average detection time. These data are also graphically displayed in Fig. 18.

With the automatic script, it was also possible to detect the number of detections and compare them with the potential detections according to the behavior explained in Section 6.3.2. The results are given in Table 2, data being plotted in Fig. 19.

Table 1
ICMP flooding: detection times (in ms).

Iteration	Detection times									
	1 client	2 clients	3 clients	4 clients	5 clients	6 clients	7 clients	8 clients	9 clients	10 clients
1	325,00	432,00	813,00	748,00	1206,00	1033,00	1509,00	1753,00	1712,00	1403,00
2	440,00	586,00	479,00	947,00	863,00	1202,00	1554,00	1214,00	1571,00	1437,00
3	447,00	665,00	539,00	762,00	815,00	967,00	1349,00	1332,00	1410,00	1497,00
4	651,00	595,00	530,00	695,00	852,00	812,00	1363,00	1328,00	1801,00	1550,00
5	419,00	439,00	507,00	1069,00	813,00	1238,00	1383,00	1447,00	1746,00	1495,00
6	424,00	595,00	618,00	771,00	1893,00	1127,00	1589,00	1404,00	1436,00	1121,00
7	552,00	943,00	590,00	677,00	1444,00	1095,00	1581,00	1220,00	1287,00	1407,00
8	487,00	942,00	971,00	995,00	1329,00	1141,00	1551,00	1241,00	1716,00	1646,00
9	545,00	455,00	561,00	1474,00	1601,00	1116,00	1394,00	1497,00	1607,00	1497,00
10	438,00	541,00	533,00	733,00	778,00	1042,00	1386,00	1529,00	1983,00	1384,00
Min	325,00	432,00	479,00	677,00	778,00	812,00	1349,00	1214,00	1287,00	1121,00
Q1	427,50	476,50	530,75	736,75	824,25	1035,25	1383,75	1262,75	1469,75	1404,00
Average	472,80	619,30	614,10	887,10	1159,40	1077,30	1465,90	1396,50	1626,90	1443,70
Q3	545,00	665,00	590,00	995,00	1444,00	1127,00	1551,00	1447,00	1746,00	1497,00
Max	552,00	943,00	971,00	1474,00	1893,00	1238,00	1589,00	1529,00	1983,00	1646,00

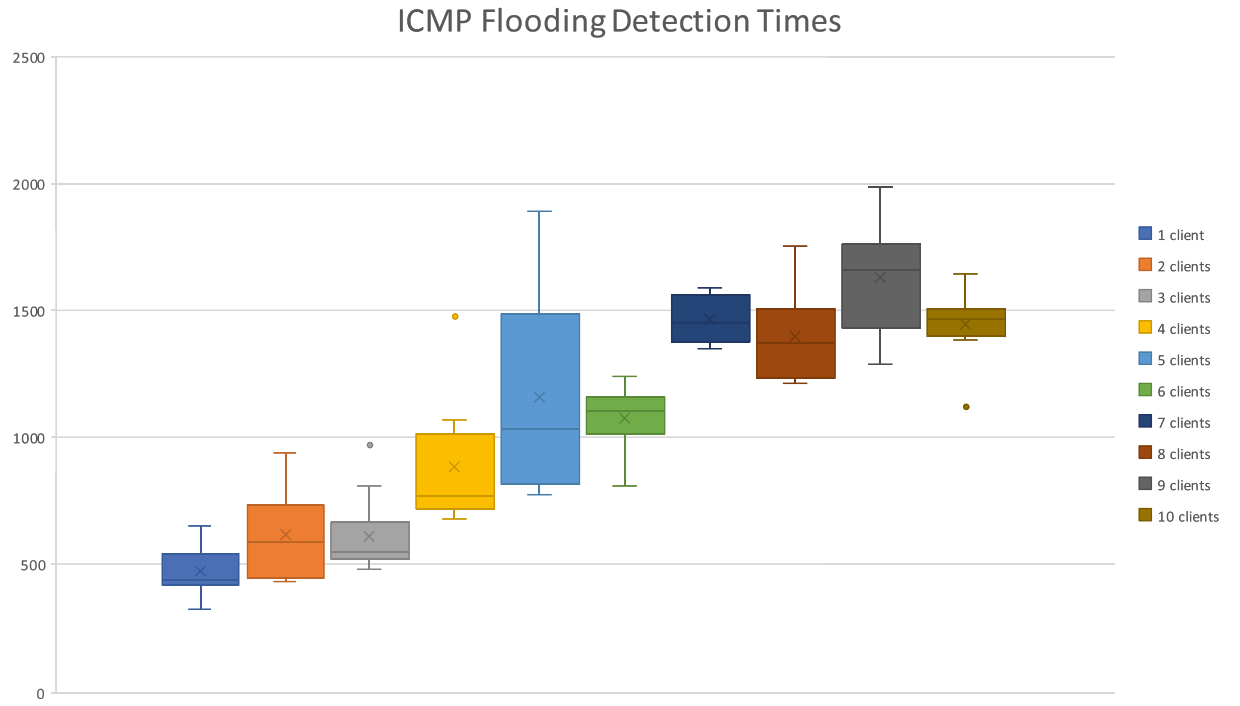

Fig. 18. ICMP Flooding: Box plot of the detection times.

6.4.1. Discussion

The data of Table 1 and Fig. 18 show that, in general, the average detection times grow as more attackers are added to the simulation. This can be easily explained by the fact that the MMT-IoT sniffer needs to handle more traffic, therefore, packets are enqueued for later processing in the sniffer. Considering this, the MMT-Probe will analise the same packets later, which adds a delay in the detection of the attack. In other words, since the sniffer can transfer only one packet at a time, it artificially adds some delay on packets that are sent simultaneously or really close in time one from another. This can impose a limit on the security analysis, since an attacker might take advantage of this feature to avoid being detected on attacks that require sending multiple packets within a close range of time. In this sense the detection will rely on the time constraints as observed as the MMT-Probe.

In addition to this, we also observe that the detection times start to stabilize around 7 clients. This might be due to the same processing limitations of the MMT-IoT sniffer; since the sniffer has been conceived to run on a restricted device, it has a limited processing power. In this sense, it is capable to handle a limited amount of packets per second, which limits the throughput that can be sniffed with the solution.

Table 2
ICMP flooding: number of detections per emulated scenario.

Iteration	Number of detections									
	1 Client	2 Clients	3 Clients	4 Clients	5 Clients	6 Clients	7 Clients	8 Clients	9 Clients	10 Clients
1	4	8	16	22	21	31	49	52	63	75
2	4	8	16	22	29	32	40	55	73	73
3	8	8	16	31	25	36	56	48	54	77
4	4	8	16	22	21	32	45	54	68	72
5	4	8	16	22	25	32	48	47	64	70
6	4	8	12	22	28	36	44	52	63	89
7	8	8	12	22	25	28	49	55	76	83
8	4	8	16	22	25	32	50	54	65	73
9	4	8	12	22	36	28	37	54	54	71
10	4	8	19	22	21	34	55	50	59	70
Average	4,80	8,00	15,10	22,90	25,60	32,10	47,30	52,10	63,90	75,30
Possible detections	9,00	12,00	27,00	36,00	45,00	54,00	63,00	72,00	81,00	90,00
Percentage detected	53,33%	66,67%	55,93%	63,61%	56,89%	59,44%	75,08%	72,36%	78,89%	83,67%

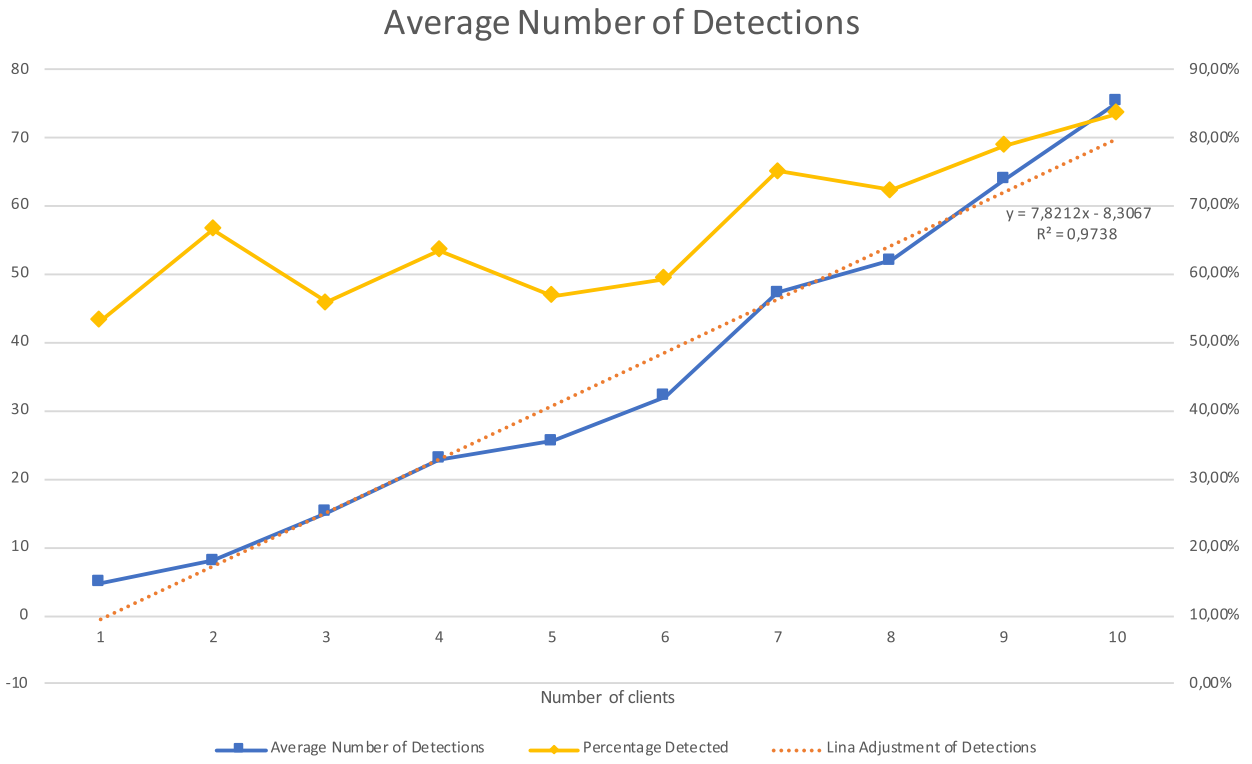

Fig. 19. ICMP Flooding: Number and Percentage of Attack Detections.

All these observations are complemented by the data of Table 2 and Fig. 19. First, we can clearly observe that a higher amount of clients generate a higher amount of detections, being its performance almost linear (adjusted by the orange dotted line in Fig. 19, $R^2 = 0.9738$). This behavior is in line with our initial thoughts that more attacks triggered should raise more alerts in the security analysis engine. However, more conclusions can be drawn when analysing the percentage of detected attacks.

When observing the percentage of detection (yellow line in Fig. 19), we need to remember that each trigger of the attack the tool might detect it 3 times since only 3 packets are required to verify the rule. Yellow line in Fig. 19 shows the number of detected attack versus the possible total detections (3 times for each trigger, times the number of triggers, times the number of clients). Being this said, we observe that when we add more devices to the simulation, the percentage of detected attacks slightly rises, meaning that a higher number of the possible groups of 3 ICMP packets are detected as attacks. Despite it seems contradicting the fact that more devices raise the detection times, it can be explained by the fact that the MMT-IoT sniffer is reaching its limits, meaning that it enqueues a higher amount of packets (simply because more clients means more packets) before transferring them to the security analysis module. This queuing process explains both the higher detection times (as explained above) and the higher number of detected attacks.

7. Conclusions and future works

Ensuring security in IoT is very challenging, since the enforcement of proper countermeasures is obstructed by many limitations, in terms of computational resources and power. In this scenario, security monitoring acquires more importance to guarantee security. Traditional IDSs are not suitable to fully monitor IoT infrastructure since the involved technologies and networks layers are different. In this paper we proposed the extensions of the Montimage Monitoring tools to support the monitoring of proper IoT network levels, namely the 6lowPAN protocol. Different plugins have been implemented to support specific deep packet inspection functionalities and specific probes for traffic capturing and analysis have been designed and implemented in order to let it be independent from sensor technologies.

The extended tool has been validated within the ANASTASIA EU project and two use cases have been discussed.

The results have been sufficiently positive to encourage future improvements. Those improvements impact not only design and implementation phases but enlarging the suite of protocols to be analysed and the pervasiveness of the solution that should me as much technology independent as possible. Furthermore, in relation to security detection, not only we need to enlarge the number of supported plugins (to include, for example, ZigBee) but also the adoption of anomaly-based techniques, provided by the MMT-Behavior module, should be exploited in order to detect new specific IoT threats (e.g. Slow DoS).

Acknowledgement

This article has been written in the context of the ANASTACIA project (www.anastacia-h2020.eu), which has received funding from the European Union Horizon 2020 Research and Innovation Programme under the Grant Agreement N. 731558 and from the Swiss State Secretariat for Education, Research and Innovation. Authors would like to tank all project beneficiaries for the support to the preparation of this article.

References

[1] N. Bari, G. Mani, S.Y. Berkovich, Internet of things as a methodological concept, in: Fourth International Conference on Computing for Geospatial Research and Application, COM.Geo '13, San Jose, CA, USA, July 22–24, 2013, IEEE, 2013, pp. 48–55, doi:10.1109/COMGEO.2013.8.
[2] A. Triantafyllidis, C. Velardo, D. Salvi, S.A. Shah, V.G. Koutkias, L. Tarassenko, A survey of mobile phone sensing, self-reporting, and social sharing for pervasive healthcare, IEEE J. Biomed. Health Inform. 21 (1) (2017) 218–227, doi:10.1109/JBHI.2015.2483902.
[3] T. Usländer, T. Batz, Agile service engineering in the industrial internet of things, Future Internet 10 (10) (2018) 100, doi:10.3390/fi10100100.
[4] V. Casola, A. De Benedictis, M. Rak, U. Villano, Toward the automation of threat modeling and risk assessment in iot systems, Internet Things 7 (2019) 100056, doi:10.1016/j.iot.2019.100056.
[5] E. Rios, E. Iturbe, L. Orue-Echevarria, M. Rak, V. Casola, Towards self-protective multi-cloud applications: musa-a holistic framework to support the security-intelligent lifecycle management of multi-cloud applications, in: CLOSER 2015 - 5th International Conference on Cloud Computing and Services Science, Proceedings, 2015, pp. 551–558.
[6] E. Rios, W. Mallouli, M. Rak, V. Casola, A. Ortiz, Sla-driven monitoring of multi-cloud application components using the musa framework, in: Proceedings - 2016 IEEE 36th International Conference on Distributed Computing Systems Workshops, ICDCSW 2016, 2016, pp. 55–60.
[7] V. Casola, A. De Benedictis, M. Rak, Security monitoring in the cloud: an sla-based approach, in: Proceedings - 10th International Conference on Availability, Reliability and Security, ARES 2015, 2015, pp. 749–755.
[8] S. Ziegler, A. Skarmeta, J. Bernal, E.E. Kim, S. Bianchi, Anastacia: advanced networked agents for security and trust assessment in cps iot architectures, in: 2017 Global Internet of Things Summit (GIoTS), 2017, pp. 1–6, doi:10.1109/GIOTS.2017.8016285.
[9] A.K. Sikder, H. Aksu, A.S. Uluagac, 6thsense: a context-aware sensor-based attack detector for smart devices, in: 26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16–18, 2017., 2017, pp. 397–414.
[10] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, Y. Kim, Rocking drones with intentional sound noise on gyroscopic sensors, in: 24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12–14, 2015., 2015, pp. 881–896.
[11] A. Nahapetian, Side-channel attacks on mobile and wearable systems, in: 13th IEEE Annual Consumer Communications & Networking Conference, CCNC 2016, Las Vegas, NV, USA, January 9–12, 2016, 2016, pp. 243–247, doi:10.1109/CCNC.2016.7444763.
[12] V. Subramanian, A.S. Uluagac, H. Cam, R.A. Beyah, Examining the characteristics and implications of sensor side channels, in: Proceedings of IEEE International Conference on Communications, ICC 2013, Budapest, Hungary, June 9–13, 2013, IEEE, 2013, pp. 2205–2210, doi:10.1109/ICC.2013.6654855.
[13] A. Maiti, M. Jadliwala, J. He, I. Bilogrevic, (smart)watch your taps: side-channel keystroke inference attacks using smartwatches, in: K. Mase, M. Langheinrich, D. Gatica-Perez, K.V. Laerhoven, T. Terada (Eds.), Proceedings of the 2015 ACM International Symposium on Wearable Computers, ISWC 2015, Osaka, Japan, September 7–11, 2015, ACM, 2015, pp. 27–30, doi:10.1145/2802083.2808397.
[14] G. Petracca, A. Atamli-Reineh, Y. Sun, J. Grosskiags, T. Jaeger, Aware: preventing abuse of privacy-sensitive sensors via operation bindings, in: 26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16–18, 2017., 2017, pp. 379–396.
[15] C. Stergiou, K.E. Psannis, A.P. Plageras, G. Kokkonis, Y. Ishibashi, Architecture for security monitoring in iot environments, in: 26th IEEE International Symposium on Industrial Electronics, ISIE 2017, Edinburgh, United Kingdom, June 19–21, 2017, IEEE, 2017, pp. 1382–1385, doi:10.1109/ISIE.2017.8001447.
[16] G. Cerullo, G. Mazzeo, G. Papale, B. Ragucci, L. Sgaglione, Iot and sensor networks security, in: Security and Resilience in Intelligent Data-Centric Systems and Communication Networks., 2018, pp. 77–101.
[17] L. Coppolino, S. D'Antonio, V. Formicola, L. Romano, Enhancing siem technology to protect critical infrastructures, Critical Information Infrastructures Security, vol 7722, Springer, 2013a, pp. 10–21. Lecture Notes in Computer Science
[18] L. Coppolino, S. D'Antonio, A. Garofalo, L. Romano, Applying data mining techniques to intrusion detection in wireless sensor networks, in: Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2013, Compiegne, France, October 28–30, 2013, 2013b, pp. 247–254, doi:10.1109/3PGCIC.2013.43.
[19] T.-H. Lee, Chih-HaoWen, L.-H. Chang, H.-S. Chiang, M.-C. Hsieh, A lightweight intrusion detection scheme based on energy consumption analysis in 6lowpan, in: Advanced Technologies, Embedded and Multimedia for Human-centric Computing, Springer, 2014, pp. 1205–1213.
[20] P. Pongle, G. Chavan, Real time intrusion and wormhole attack detection in internet of things., Int. J. Comput. Appl. 121 (2015) (9).
[21] S. Raza, L. Wallgren, T. Voigt, Svelte: real-time intrusion detection in the internet of things, Ad Hoc Netw. 11 (2013) 26612674 (8).
[22] P. Abhishek, B. Harsha, S. Vaibhav, M. Nalini, Classification of intrusion detection system, Int. J. Comput. Appl. 118 (7) (2015), doi:10.5120/20758-3163.
[23] A. Ortiz, D. Rivera, E. Montes de Oca, MS12a: Monitoring Component Services Specified and Agreed by the Board.
[24] W. Project, IEEE 802.15.4.
[25] T.F.E. Wikipedia, Network Socket.

[26] J. Olsson, in: 6LoWPAN Demystified, 2014, p. 13.
[27] IEEE Standard for Low-Rate Wireless Networks, 2015.
[28] O. Kirby, IEEE 802.15.4 dissector and libpcap support, 2007.
[29] J.W. Hui, P. Thubert, Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks, Technical Report, Internet Engineering Task Force, 2011, doi:10.17487/RFC6282.
[30] Z. Shelby, K. Hartke, C. Bormann, The constrained application protocol (CoAP), Technical Report, Internet Engineering Task Force, 2014.