Security Management Architecture for NFV/SDN-aware IoT Systems

Alejandro Molina Zarca*, Jorge Bernal Bernabe*, Ruben Trapero†, Diego Rivera‡,
Jesus Villalobos †, Antonio Skarmeta *, Stefano Bianchi §, Anastasios Zafeiropoulos¶ and Panagiotis Gouvas¶
*Department of Information and Communications Engineering,
University of Murcia, Spain
{alejandro.mzarca, jorgebernal, skarmeta}@um.es
†ATOS Research, Spain,
{ruben.trapero, jesus.villalobosnieto}@atos.net
‡Montimage, Paris, diego.rivera@montimage.com
§Softeco, Geneva, stefano.bianchi@softeco.it
¶Ubitech, {azafeiropoulos,pgouvas}@ubitech.eu

***Abstract*—The Internet of Things brings a multi-disciplinary revolution in several application areas. However, security and privacy concerns are undermining a reliable and resilient broad-scale deployment of IoT-enabled Critical Infrastructures (IoT-CIs). To fill this gap, this paper proposes a comprehensive architectural design that captures the main security and privacy challenges related to Cyber-physical Systems and IoT-CIs. The architecture is devised to empower IoT systems and networks to make autonomous security decisions through the usage of novel technologies such as Software Defined Networking (SDN) and Network Function Virtualization (NFV), as well as endowing them with intelligent and dynamic security reaction capabilities by relying on monitoring methodologies and cyber-situational tools. The architecture has been successfully implemented and evaluated in the scope of ANASTACIA H2020 EU research project.**

***Index Terms*—IoT, cybersecurity, SDN/NFV, architecture**

I. Introduction

The Internet of Things (IoT) [1] is changing the industrial landscape by leveraging network capabilities of heterogeneous, pervasive and autonomous smart-objects. IoT promotes a distributed global network with billions of devices interacting using Machine-to-Machine (M2M) communications, which facilitates the creation of innovative smart services and applications.

However, the constrained nature of IoT devices and networks as well as their distributed and pervasive conditions, makes the IoT subject to new kind of vulnerabilities and cyber-threats. Traditional network protocols and security solutions are not suitable for IoT, as they do not cope with unforeseen interoperability and adaptability problems and do not meet the dynamic needs, responsiveness and lightness required in IoT.

The lack of automatized software updates, vendor support as well as user's mis-configurations make the IoT prone to new kind of cyber-attacks. These problems and their consequences are aggravated in IoT-enabled Critical Infrastructures (IoT-CIs), like energy systems in smart buildings. In this context, there is a need of advanced and adaptive mechanisms able to ensure dynamically the proper security levels in the IoT systems and providing system resiliency through self-healing and self-repair capabilities, thereby countering cyber-attakcs and mitigate cyber-threats whenever occur in the managed IoT network.

In this sense, contextual and monitoring information obtained from the surrounded IoT environments can be used as baseline for data analysis detect anomalous behaviours, and in turn, infer smart control and management decisions through different actuators, agents and controllers deployed either at the edge or in the core of the IoT network. Indeed, this contextual and real-time monitoring can be also applicable to deal with diverse kind of cyber-threats and IoT attacks, thereby countering them by adapting the security policies and enforced configurations of the managed IoT system according to the context [2].

Software-defined networking (SDN) brings forward new network capabilities by decoupling the control and data planes, which can introduce novel security defense mechanisms in IoT such as managing malicious traffic or IoT devices isolation. Likewise, Network Function Virtualization (NFV) can exploit virtualization to provide on-demand, dynamic, flexible, scalable and elastic orchestration and deployment of virtual appliances. NFV can enable the autonomous management of virtual network security functions that can off-load the security of IoT networks to the network edge. In this sense, [3], introduces virtualized versions of security hardware, e.g. firewalls, DPIs, as well as support mobile IoT scenarios.

Some researches are starting to define architectures aimed to deal with Cyber-situational network and system management by exploiting SDN and NFV features in dedicated scenarios-environments, such as 5G network management [4] or cloud/-fog Computing [5] applications. However, unlike our proposed architecture, those ones have not been tailored for dealing with the security and privacy in SDN/NFV-enabled IoT and critical

CPS (Cyber Physical Systems) scenarios.

This paper presents an architecture aimed to deal with the security management of SDN/NFV-enabled IoT scenarios. The architecture has been already implemented and evaluated in Critical Infrastructures (CI) deployed in Smart Buildings, coming up with a policy-based cyber-situational security framework that has demonstrated its benefits to adapt dynamically the enforced security mechanisms of IoT networks. The automatic reaction against IoT threats and attacks is done through special purpose IoT agents, SDN and IoT controllers as well as NFV-based virtual security appliances, which enforce reaction countermeasures needed to mitigate cyber-attacks and dynamically adapt the system according to the context obtained and analyzed by the integrated monitoring tools and SIEM (Security Information and Event Management) system.

The rest of this paper is structured as follows. Section 2 studies the current state of the art in this research field. Section 3 delves into the main IoT attacks/threats as well as detection and reaction mechanisms proposed. Section 4 details the proposed Architecture whereas section 5 describes its main architectural processes. Section 6 is devoted to the implementation of the proposed architecture, which is evaluated in Section 7. Finally, section 8 concludes the paper.

II. Related work

SDN is starting to be exploited as mechanism to handle security threats [6], as it provides numerous advantages such as dynamic flow control, IoT traffic and devices isolation, network monitoring to identify attacks (e.g. botnets) and flexibility to support deployment of virtual network security functions.

In this sense, Rawat et al. [7] proposes diverse security threats and attacks and how they can be mitigated using proper countermeasures based on SDN. An SDN architecture to strengthen the IoT is presented by Chakrabarty et al. [8]. It aims to protect IoT communications by relying on the SDN centralized controller delivering secure routing and enhancing system management. Bull et al. [9] use IoT traffic monitored from SDN gateway, for detecting misbehaviour in the network and reacting accordingly, either by blocking or forwarding the traffic. Flauzac et al. [10] delivered an SDN security solution for IoT wireless networks, that is intended to avoid compromising a security domain by deploying security rules throughout different security controllers. Choi et al. [11] described software-defined security framework for IoT that allows delivery of security appliances e.g. access control or channel protection, however they do not exploit NFV.

On the other side, NFV can realize the security as a service paradigm [12], abstracting security software from hardware, to achieve performance improvement on virtual network security functions. Lightweight virtualization enables deployment of Virtual Network Functions (VNF) at the edge of IoT networks. For instance, [13] provides a solution based on SDN, NFV, and cloud computing that can enhance network management in 6LoWPAN IoT networks. Yu et al. [3] presented IoTSec, an IoT security architecture that allows delivering micro-middleboxes, instantiated on demand over lightweight IoT systems. NFV enables scaling up/down security VNFs at the edge of the network, such as, for instance, vFirewalls, vIDS, according to the network and system status. SDN can be used along with NFV to steering the traffic towards VNFs, optimizing the chaining of virtualized middleboxes and enhancing resource utilization.

Yang et al. [14] detail the challenges, opportunities as well as security issues in NFV. Similarly, other recent surveys [15] [16] recaps the main benefits of adopting SDN and NFV to increase security in IoT networks.

The aforementioned research works, unlike ours, do not provide a holistic cyber-security framework that relies on SDN and NFV in order to enhance IoT security. The framework is able to dynamically detect cyber-security incidents and react according to the actual status of IoT networks, systems and deployed security policies.

Regarding policy-based frameworks, Shankar et al. [17] propose a framework which relies on an extended model of Event-Condition-Action (ECA) policies in order to include post conditions to verify the successful completion of policy actions. Rensing et al. [18] provide a policy-based architecture and framework specific for AAA services. Hadjiantonis et al. [19] adopts a policy-based network management together with context awareness for Mobile Ad hoc Networks (MANETs) and Basile et al. [20] provides a policy-based framework management for securely deployment and configuring components which processes network traffic.

A first overview of Anastacia was presented at the beginning of the project in conference paper [21], which outlined the main objectives, challenges and foundations of the project. Similarly, a first insight about how SDN/NFV-based can be exploited to provide security features over IoT scenarios was presented in [22]. The Anastacia policy enforcement mechanism was recently detailed by Zarca et al. [23]. However, unlike in the research described herein, those previous publications did not present the whole final architecture and did not deal with the entire autonomic loop for self-protection and self-healing of the IoT managed system. Moreover, the monitoring, detection, and the reaction policies to counter cyber-attacks were not addressed neither evaluated.

III. Security handling in NFV/SDN-enabled IoT scenarios

This section gives an overview of the Main threats and vulnerabilities in IoT, enlightening how our proposed framework detects, and reacts against those threats/attacks by using NFV/SDN-based security enablers. Table I summarizes those concepts.

IoT attacks can be split in two main categories: *active* where malicious attacker alters or injects messages to exploit vulnerabilities and *passive* where the attacker interacts actively within the network.

Regarding Active attacks, it is work mentioning the Replay Attack (re-transmission of previous victim's packets) that aims to perform a disservice or enable other more advanced attacks such as Masquerading attacks or impersonation, which in turn, aims to get victims' data and/or replicate nodes. In a Tampering attack, the attacker node makes an unauthorized/malicious

action against the IoT device. In our framework, these kinds of attacks can be detected by the AAA system and analyzing the logs, the applicable countermeasure might perform device flashing or its network isolation. vAAA and special agents can be dynamically deployed in the edge of the IoT network to facilitate authentication, authorization and key management redistribution.

IoT Devices infected with Malware, e.g. worms, trojans, ransomware can be detected through vulnerability assessment and the countermeasures. When it comes to the protection of the rest of the IoT system/network, it lies on measures such as updating the firmware, isolate device, block connection attempts.

Zero-day vulnerabilities refer to exploitation of software vulnerabilities not seen before in the network. They are extremely difficult to detect and mitigate. Our proposed framework covers the detection and handling of this kind of attacks by means of Artificial Intelligence-based (AI) anomaly detection. For example, IoT devices malfunctioning and reporting unusual sensors values can be detected through machine learning of IoT data in context broker. The countermeasure is done through special-purpose IoT Controllers that act directly on the malicious IoT device, using protocols like CoAP or LWM2M.

Man in the middle attacks allow malicious entities to intercept communications and impersonate endpoints, to spoof and alter packets, e.g modifying IoT sensors values to produce damages. Like in the previous case, it can be detected through AI-based anomalous detection tools, and counter through IoT controllers actions over final devices and agents.

Distributed DoS attacks are perpetrated by infecting IoT devices with malware which, in turn, make them become part of botnets that launch massive and coordinated DoS attacks against external victims [24]. DoS can also target IoT networks, flooding and exhausting them, and causing service unavailability. In our proposed architecture this can be detected by a vIDS and special IoT agents. The countermeasures based on NFV/SDN can be manifold in this case, either deploying vLoadBalancer next to the victim entity, or stopping the network traffic in the source, that is, by deploying vFirewalls that filter traffic coming from infected devices or just stop the affected device if possible. Network traffic can be also filtered directly by the SDN controller in the virtual switch (that can be also deployed on demand in the edge as VNF). In addition, a virtual IoT honeynet deployed as VNF can emulate the real IoT network, and the attacker can be redirected to such a controlled honeynet, thereby countering the potential damage. In any of the cases, the traffic is diverted to the vFirewall, or vIoTHoneyNet adding new flow rules using SDN.

Malicious code injection in memory or in services, e.g. SQL Injection, can alter normal IoT and services behaviour, and it can be detected through monitoring tools that identify unwanted network accesses, or unexpected queries/accesses in databases/services. Our architecture aims to mitigate this kind of attack by isolating Attacker's traffic using SDN.

On the other hand, regarding passive attacks such as traffic analysis or sniffing/Eavesdropping private IoT communications, our framework detects those problems through deep packet inspectors deployed as security VNFs that detect encrypted traffic. The SDN controller can mirroring the traffic towards the vDPI for analysis. The countermeasure can be deploying vChannelProtection VNFs, to facilitate re-bootstrapping of encryption keys and establishing encrypted network tunnels (e.g. using DTLs), either, end-to-end or through vProxies.

As identified by Gao et al. [25] there are other specific attacks such as, for instance, routing attack, sink node attack, black hole attack, flooding attack, trapdoor, Sybil attack, but they can be handled in a similar way that the ones summarized above.

IV. Architecture

ANASTACIA is envisioned as a framework integrated on top of an IoT infrastructure where IoT devices, physical and virtual network elements interact in the *Data Plane*. On top of that, a *Control Plane* manages the computing, storage, and networking resources in the Data Plane by leveraging SDN controllers, NFV orchestration platforms, and IoT controllers.

The ANASTACIA architecture is shown in Figure 1. It is comprised of diverse planes that provides the intelligence and dynamic behavior to the system. Namely, the Security Orchestrator Plane, Monitoring and Reaction Plane, Security Enforcement Plane, User plane and Seal Manager. The following subsection details each Plane and its domains.

A. Security Enforcement Plane

The **Control and Management domain** oversees and controls the usage of resources and run-time operations of the security enablers deployed either over virtualized or physical environment, including IoT networks. It connects the Orchestration plane with the IoT Platform (Data and Control planes), managing the interactions among objects and components for the enforcement of the security policy defined at the User Plane. The SDN controllers are responsible of communicating though southbound APIs with the network elements to manage connectivity applying the networking flow rules. NFV ETSI MANO-compliant modules are responsible for management and orchestration of the virtual network and security virtual functions over the Virtual infrastructure, e.g. Openstack. Finally, IoT controllers provide different southbound interfaces in order to manage and control different kind of IoT devices depending on the IoT protocols. These IoT controllers are deployed at the network edge (e.g. gateways) to enforce security functions in heterogeneous IoT domains. Regarding the main interfaces in this domain it is important to highlight the following ones:

SDNI: SDN-oriented Security Enforcement Plane Interface: allows to manage the SDN networking configuration via the SDN controller(s). The Security Orchestrator can request the enforcement of the SDN traffic flow rules received as outcome of the policy refinement process.

NFVI: NFV-oriented Security Enforcement Plane Interface: allows to manage the security VNFs via the ETSI-oriented NFV MANO modules. This is, the Security Orchestrator maintains the knowledge about the VNF descriptors, the Network

IoT Attack/Threat	Detection (s)	Countermeasure(s)	Security Enabler(s)
Replay Attack, Masquerading, Tampering	AAA system detects illegitimate access attempts using forged credentials	Device flashing, Device isolation	vAAA deployment in the Edge (NFV) — IoT device keys re-Bootstrapping — Channel Protection (e.g. DTLs)
Devices infected with Malware	Vulnerability Assessment, logs analysis	device's firmware update block connection attempts	special purpose IoT controllers
Zero-day Vulnerabilities (e.g. IoT malfunctioning)	Anomaly behavior detection based on AI IoT data analysis using AI	Isolate/Stop IoT device	IoT Controller action on device (e.g. CoAP, LWM2M)
Main in the Middle	Anomaly-based IDS Anomaly behavior detection based on AI	Reset device	IoT controllers, IoTAgents
Distributed DoS (IoT botnets)	Combined IDS + AI traffic analysis	Virtual IoT HoneyNet — Load Balancing	Virtual Firewall (NFV) — SDN traffic filtering — SDN traffic divert
SQL Injection (SQLi)	Monitoring, SIEM	Attacker's traffic isolation	SDN traffic isolation
Sniffing/Eavesdropping	Deep Packet Inspector Traffic mirrored using SDN towards vIDS	M2M Network Channels encryption Channel keys redistribution encryption through vProxy	vChannelProtection (L2 Encryption, DTLs, data encryption...) vProxy

TABLE I
MAIN IOT THREATS/ATTACKS AND OUR SUGGESTED POTENTIAL DETECTION AND REACTION MECHANISMS BASED ON NFV-SDN

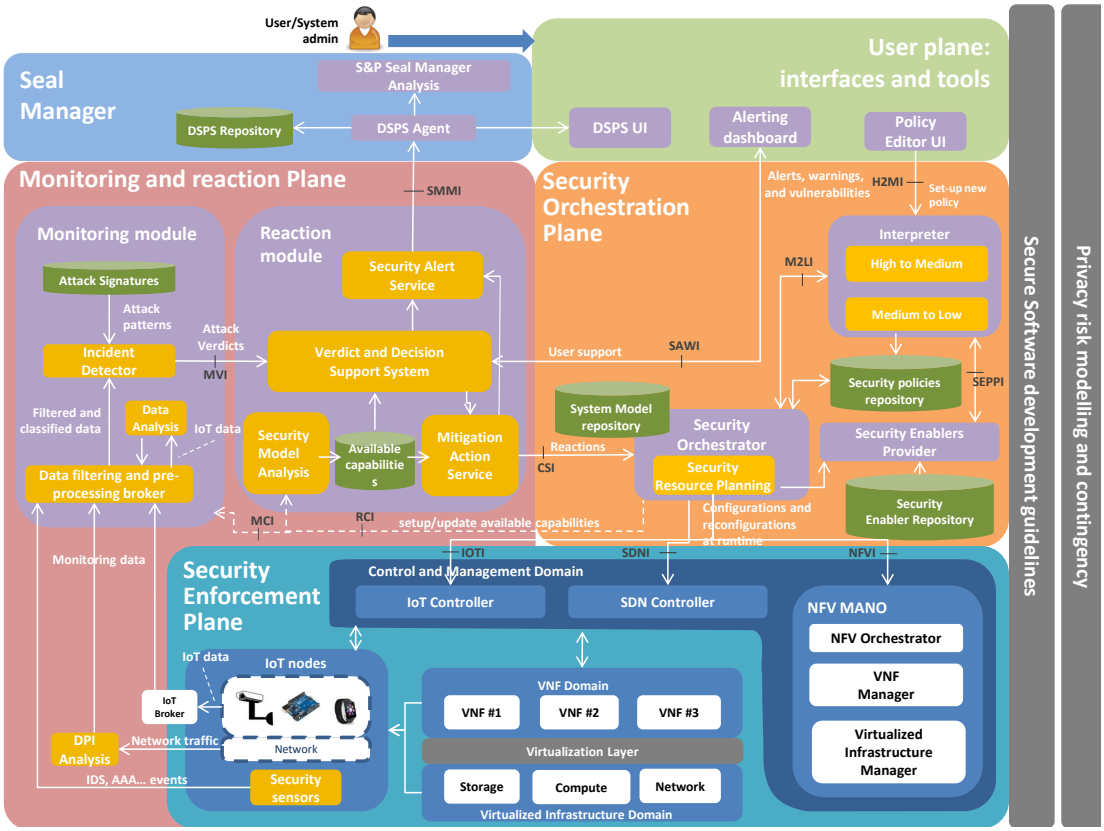

Fig. 1. ANASTACIA Reference Architecture

Services (NS) catalogue as well as the current deployment. When a VNF management is required, the Security Orchestrator requests it by sending the configuration to the NFV MANO through the NFVI. Then, the NFV MANO uses the Resource Orchestrator (RO) in order to provide the required resources to cope with the received configuration through a specific Virtualized Infrastructure Manager (VIM). Finally, the RO requests the VNF configuration enforcement to the VNF Configuration and Abstraction (VCA). More information on this topic can be found at [23].

IoTI: IoT-oriented Security Enforcement Plane Interface: manages the configuration of IoT nodes through the IoT controllers. The Security Orchestrator can request the enforcement of the security controls within the IoT nodes according to the configurations generated by the policy refinement process.

The **Infrastructure and Virtualization domain** comprises all the physical machines capable of providing computing, storage, and networking capabilities, as well as the virtualization technologies, to provide an Infrastructures as a Service (IaaS) layer. This domain also includes the network elements responsible for traffic forwarding, following the SDN controller's rules, and a distributed set of security probes for data collection to support the monitoring services.

VNF domain represents the Virtual Network Functions enforcing the security within network services. Anastasia currently considers diverse kind of network security functions

deployed as VNFs in order to provide the defense mechanisms and threat countermeasures, including vFirewall, vIDS/IPS, vAAA, vChannelProtection, vIoTHoneyNet.

IoT domain refers both, to the final IoT devices as well as the the security enablers and actuators required to apply the security directives within the IoT devices. To this aim, the orchestration plane (either as proactive policy enforcement by the admin, or as reaction), will apply, over the IoT Controller the appropriate security service, which, in turn, will contact the final IoT device or actuator by using specific IoT protocols.

B. Security Orchestration Plane

The **Security Orchestrator Plane** organizes the resources that support the Enforcement Plane, carrying out activities such as the transformation of security properties to configuration rules (through a *Policy Interpreter*) and aligning the security policies defined by the Policy Interpreter with the provisioning of relevant security mechanisms (through a *Security Enabler Providers*). It has the whole vision of the underlying infrastructure in order to orchestrate (through the *Security Orchestrator*) resources and interfaces available at the Security Enforcement Plane.

The **Security Enabler Provider** is able to identify the list of security enablers which can provide specific security capabilities to meet the security policies requirements. Besides, this component will be endowed with an interface for delivering Medium-to-Low translation plugins (M2Lplugins) for technology dependant policy translations into low-level configurations. In this way any kind of software or hardware could be supported as security enabler by implementing the corresponding translator plugin. Currently, the framework provides plugins focused on the current experimentation phase, encompassing network filtering and forwarding through SDN and virtual router, XACML authorization, IoT management through the IoT controller, IoT honeynet configuration and DTLS configuration.

A **Security orchestrator** is responsible for selecting the security enablers to be used in the policy refinement process. To choose the proper security enabler to apply in a particular situation (e.g. after a reaction), it oversees the security capabilities, the available resources in the underlying infrastructure, and the policy requirements. Once received the configuration of the security enablers by the Policy Interpreter, the Security Orchestrator interacts with relevant SDN/NFV/IoT control and management components, so to enforce the required features in the IoT devices and in the physical/virtual network elements of the underlying infrastructure.

The *SMI* interface (Security Orchestrator System Model) allows to request a system model of the underlying architecture. This information will be used by the Security Orchestrator in order to make decisions like what could be the best security enabler for a specific policy enforcement.

The **Policy Interpreter** transforms the Security policy (closer to a human readable policy) to a machine-readable policy that is able to represent lower configurations parameters. Namely, performs the refinement processes from High-level Security Policy Language (HSPL) to Medium-level Security Policy Language (MSPL). These security policy models have been extended from [26]. Finally, the MSPLs are translated to Enablers/VNFs configurations or tasks.

The *H2MI* (High to Medium) interface allows to request a policy refinement from a High-level Security Policy Language (HSPL) to a Medium-level Security Policy Language (MSPL). The *M2LI* Medium to Lower interface allows to request a policy refinement from a Medium level Security Policy Language (MSPL) to a specific enabler configuration/task. The *SEPPI*: Security Enabler Provider Plugin Interface allows to request for a plugin which implements the MSPL to Enabler translation.

Currently, the framework provides security policy models for authentication, authorization, channel protection (validated in [27]), filtering, traffic divert, monitoring, IoT management, IoT honeynet, Anonymity, Privacy (Identity-based, Attribute-based and PKI based), QoS, Data Aggregation and policy for orchestration. In order to prevent conflicts between the security policies, the framework provides a policy conflict detection engine which verifies that the security policy will not generate conflicts like redundancy, priorities, duties (e.g. packet inspection vs channel protection), dependencies or contradictions. To this aim, the security policy is processed against the rule engine which extracts context information from the policy repository and the system model in order to perform the verification.

Regarding the policy for orchestration, the framework allows the security administrator to define multiple security policies related between themselves and with the already deployed ones by establishing priorities and dependencies. This is, the security administrator can establish different level of priorities depending on the magnitude or relevance of the policy. On the other hand, since the enforcement of a security policy can be conditioned by the deployment of another policy or even by the triggering of any kind of event, the security administrator is able to specify different kind of dependencies depending on the deployment requirements (e.g. an authorization policy could depend on a success authentication event).

C. Monitoring and reaction plane

The *Monitoring and Reaction Plane* connects to the IoT Platform through the Security Enforcement Plane in order to collect, through *monitoring agents*, security-focused information related to the system behavior. This plane also evaluates the fulfilment of the security policy by checking security models and threats signatures, performing *Filtering* activities and *data analysis* for the detection of anomalies. At this plane, and based on the anomalies detected, reactions are designed to mitigate such anomalies through the *Mitigation Action Service* that is directly connected to the Service Orchestration plane. Reactions are designed based on the *security model analysis* of the IoT/CPS infrastructure – which determine the set of possible actions to enforce, and the capabilities of the network that is being monitored. The specific reaction chosen is based on a *decision support system* that evaluate the convenience of the reaction depending on the available resources. Additionally, alerts and other reports are available for system administrators through a *security alert service*.

The Monitoring and Reaction plane acts the security-enabling member of the architecture, providing monitoring and self-reaction capabilities to the platform. To this end, the plane needs to communicate with other modules, using different interfaces:

Monitoring Configuration Interface (MCI): allows configuring the Monitoring Module from the Security Orchestrator. It is intended to provide the required parameters to refine the detection of potential threats on the network. *Reaction Security Configuration Interface (RCI)*: enables the Security Orchestrator to provide the Security Model-related data to the Reaction Module. In general terms, this information will be composed by the Capabilities of the Security Policy and the applied countermeasures on the network as a reaction to a detected security issue. *Monitoring Verdicts Interface (MVI)*: This interface is intended to provide the required monitoring information from the Monitoring to the Reaction Module. The transferred data is mainly composed of the verdicts of the security properties tested on the network. *Countermeasures Suggestions Interface (CSI)*: it was conceived to transmit the set of suggested countermeasures from the Reaction module to the Security Orchestrator in form of MSPL files.

D. User plane

The **Policy Editor Tool** allows administrators to model security policies with a high/medium-level of abstraction.

SAWI: Security Alerts and Warnings Interface: This interface will transfer the alerts and warnings from the Reaction Module to the end-user interfaces. It is designed as a communication channel between the Reaction Module and the ANASTACIA User Plane.

E. Seal Manager Domain

Using security and privacy standards, the Dynamic Security and Privacy Seal monitors in real time the security and privacy, Verdict and decision support system. It provides a graphical representation of the system status to the end-user, through different GUIs for user and data controller, and data bases to hold privacy policies and logs.

SMMI: Seal Manager Metadata Interface: provides the requested information to evaluate the security and the privacy in a real-time fashion. The security and privacy policies defined by the user are stored inside the policies repository and an interface is available to retrieve and set them from the seal manager.

V. Main Architectural flows

A. Proactive policy deployment

The framework features two different policy enforcement approaches, depending on the entity initiating the enforcement process. Concretely, the proactive (or set-up) approach and the reactive one. In the first approach the security policy is enforced as part of a preventive or by-default security plan. On the other hand, the reactive approach rises the policy enforcement as part of an automated security countermeasure.

Our previous work [23] already showed the main flow for the policy-based security enforcement in the proactive approach. In this case, the security administrator defines an HSPL through a user-friendly interface using the Policy Editor Tool and once the policy has been properly defined, the security administrator requests the policy enforcement. The Policy Editor Tool then sends the HSPL policy to the Policy Interpreter which identifies the main capabilities of the security policy and obtains a list of security enablers which could enforce the security policy. When there is at least one security enabler capable of enforcing the security policy (otherwise the administrator is notified), the Policy Interpreter performs a high-to-medium level policy refinement, generating a MSPL policy, still independent of the underlying technology, but filling some required context information like real IP addresses or endpoints. This refinement is registered in the Policy Repository, allowing traceability among policy abstraction levels. Once the MSPL has been generated, the Policy Interpreter requests the MSPL policy enforcement to the Security Orchestrator, providing, not only the policy, but also the candidate's security enablers. The Security Orchestrator receives the request and its Resource Planner makes the decision concerning the most suitable security enabler to enforce, according to its knowledge about the architecture. Then, the Security Orchestrator requests a security policy translation to the Policy Interpreter in order to get a security enabler specific configuration from the MSPL security policy. The Policy Interpreter downloads the proper security enabler translator plugin from the Security Enabler Provider and the Policy Interpreter performs the policy translation, generating the specific security enabler configuration and registering the relationship among the MSPL and the enabler configuration in the Policy Repository. Finally, the configuration is returned to the Security Orchestrator, who enforces it through the specific technology, deploying if required new resources, and updating the new status of the security policy in the Policy Repository.

B. Monitoring and Attack Detection

The architecture has been conceived as a security-enabling framework that allows an autonomic detection of the security incidents and computation of the countermeasures. To enable these features, the architecture comprises a monitoring module that will actively observe the network and ensure its security.

Figure 2 shows the design of the Monitoring Component that is composed of four principal components: (1) *Data Pre-Processing and Filtering* is an intermediate layer between the Incident Detector and the Network Agents. It is intended to perform an initial filtering and reformatting of the information captured by the Network Agents and feed it in a normalized format to the Incident Detector; (2) *Incident Detector* is the core component of the monitoring module. This unit analyses the processed data form the network agents and executes the security analysis, searching for security issues and attacks; (3) *Attack signatures* is a database containing the set of attacks that are being monitored in the network. Despite this component is shown as module from the incident detector, it is usually embedded in the latter. *Data Analysis* is an AI-based module

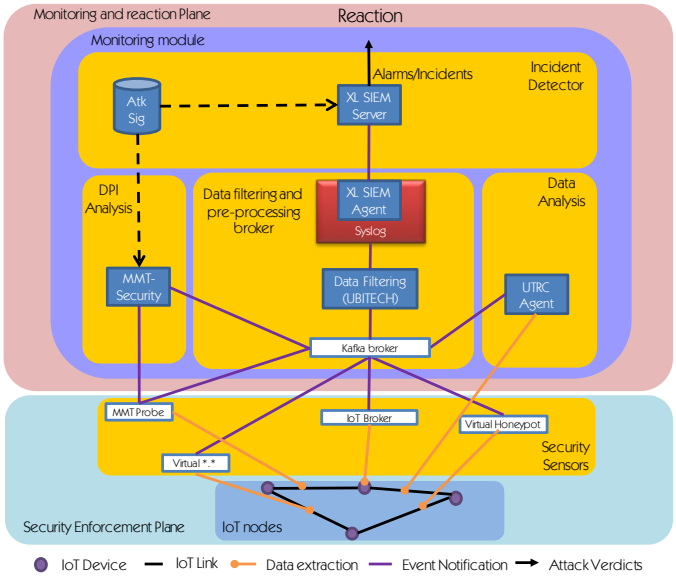

Fig. 2. Design of the Monitoring Module (From ANASTACIA Architecture)

that applies machine-learning techniques on the extracted data to detect behaviour anomalies.

The aforementioned components rely on the data extracted by the *Monitoring Agents* of our architecture, which can be seen at the bottom of Figure 2. Considering their position in the whole architecture, the monitoring agents take the role of directly interacting with the monitored network, continuously extracting information from the data plane that will be used by the components mentioned above to perform the security analysis.

Monitoring IoT-based CPS networks introduced a particular constraint on the monitoring agents: the challenge arose when using traditional monitoring techniques in state-of-the-art networks. The proposed framework tackles this challenge by adapting the traditional tools to the particular requirements of the IoT networks: (1) Low energy consumption monitoring agents: since IoT devices are restrained in energy, the implemented monitoring agents efficiently use the available energy, relegating any complex task to more capable devices; (2) Restricted computation capabilities: As a consequence of the restricted energy, the devices that act as monitoring agents for IoT network have limited computation capability. In this sense, any complex task (such as the analysis of the data) has to be delegated to devices with more capacity.

Figure 2 shows how the IoT monitoring agents interact with the monitoring module. This design allows to use modified IoT devices to extract information directly from the IoT network (e.g. capture digital packets from IoT networks or extract analogue data from sensors) and send it into the Data Pre-processing and Filtering Module (DPF). This design also allows to connect the architecture to any IoT system by deploying the monitoring agents on the network to be protected. Moreover, since the whole architecture has been conceived in a distributed way, the analysis of the extracted data can be done in external premises, minimizing the impact on the monitored network.

a) Security Monitoring Process: The design of the module allows actively monitoring IoT networks by following the process depicted in Figure 3.

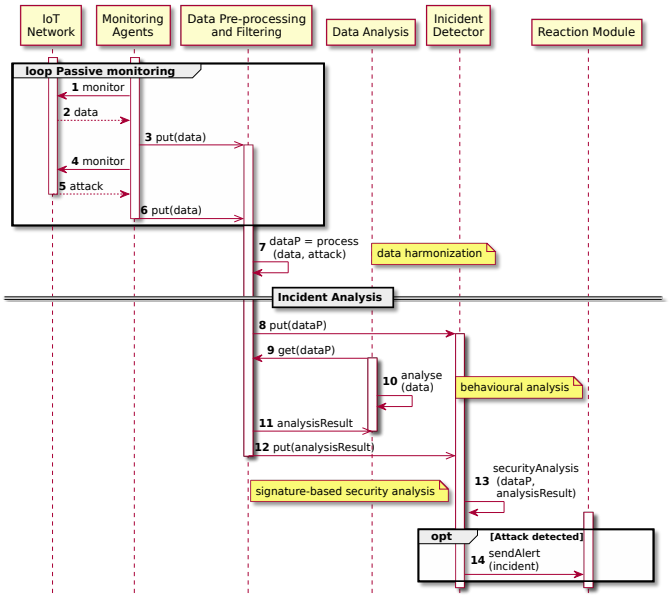

Fig. 3. ANASTACIA Monitoring phase workflow

The architecture is devised to employ passive monitoring techniques to continuously extract information form the network and then put it in the data pre-processing and filtering component. This module performs an initial processing of the extracted data in order to harmonize it in a common format. Once the information has been normalized, it is sent to the Data Analysis module and the Incident Detector modules. The functionality of these modules is intentionally different to guarantee a deep security analysis. The former, on one hand, is in charge of performing a behavioural analysis using machine learning techniques to detect any abnormal behaviour. The latter, on the other hand, performs a signature-based security analysis, ensuring lower detection times on known attacks.

Once the Data Analysis component has performed the behavioural analysis, the result is returned to the Pre-processing component, which sends the normalized results to the Incident Detection module. Finally, this module uses all the available information (raw data extracted from the network and the behavioural analysis results) to perform the security analysis. Afterwards, the incident detector generates the security alerts that are sent to the reaction component. This process concludes the monitoring phase and triggers the reaction on the platform, according to the security issues and attacks detected by the Incident Detector.

b) Monitoring Potential: The monitoring process has been conceived to be agnostic of the underlying attack to keep the generality of the security analysis. All the connected monitoring agents deliver the extracted information in the DPF component, which makes it available for both the data analysis and the incident detector components. This design decision delegates the attack detection functionalities on the latter components allowing the detection of any attack on the

monitored network.

Figure 3 shows that, despite the generic workflow of the monitoring process is complex, it has been designed to have a low latency in whole detection process, as shown in section VII.

The implemented instance of our architecture takes advantage of this feature by using multiple monitoring agents such as MMT-Probe, IoT Brokers, IDS instances (e.g. Snort), multiple Incident Detectors (like MMT-Security) and AI Data Analysis modules, such as [28]. The AI Data Analysis based on machine learning techniques allows dealing with anomaly-based intrusion detection, by analyzing deviations from the historical a normal data reported by devices. The detection can be done using statistical methods, such as correlation of time series, or supervised machine learning techniques such as, for instance, One Class Support Vector Machine (SVM).

This features empower the architecture to detect a wide variety of attacks detailed in Table I. Additionally, the monitoring agents also take advantage of the NFV and SDN techniques, allowing a dynamic deployment of them on the monitored network. Using these techniques, the Monitoring and Reaction Plane can determine to deploy new instances of monitoring agents to harden the security levels. These instructions (expressed in MSPL language) will be translated by Policy Interpreter upon demands from the Security Orchestrator, which will deploy the new monitoring agents – as virtual security functions – and use SDN to perform the modifications on the network and deploy the new instances.

Despite the recommended countermeasures are the general guidelines to mitigate the detected security issues, the framework is designed to dynamically analyze the enforced security policy and the security capabilities deployed in the network to compute the most appropriate countermeasure in the detected case. This empower the framework with highly reactive and dynamic security capabilities to cope with a wide variety of attacks.

C. Self-healing and self-protection processes

The reaction process uses the verdicts on ongoing incidents to infer the possible countermeasures to mitigate them. The decision depends on the reactions that are possible to be enforced in the infrastructure and on the resources needed to be executed. The available reactions depend on the security model of the infrastructure, which also depends on the security capabilities available at the platform.

During the reaction set-up, the security model of the IoT infrastructure is received by the Verdict and Decision Support System (VDSS) of the Reaction module from the Security Model Analysis component, which receives this information from the Security Orchestrator. This security model is directly related to the security policy being enforced, as it contains information about the security capabilities available, the potential countermeasures that are possible to be enforced and information about the cost associated to their enforcement. The cost is understood here as the amount of resources required to enforce certain countermeasure, either in terms of time, human, computation or monetary resources. This set-up process is to be triggered mainly at the starting point,

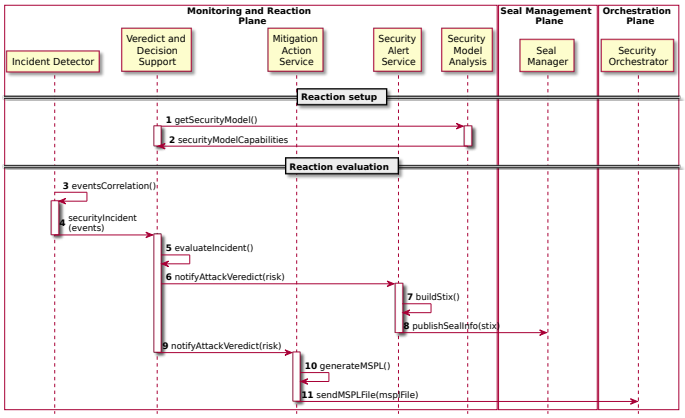

Fig. 4. ANASTACIA reaction process

although updates of the security model can be pushed into the VDSS in case of updates of the security policy that entail any modification of the security capabilities.

For every incident detected at the Monitoring module, a risk evaluation is carried out. The risk analysis is performed at the VDSS, which evaluates the severity of the incident with the criticality of the assets affected by the incident (which is received from the Security Model Analysis) and evaluating the relevance of the threat in terms of potential propagation, number of devices exposed to the threat or vulnerabilities exploited. The risk assessment uses statistical models for its analysis, which feeds from several sources of input:

1) Current incident (type o threat, severity, etc.).
2) Characteristic of the system affected (number of assets, whether they are exposed to vulnerability exploits, how critical these devices are).
3) Information about past incidents (past effects of similar incidents, costs associated to their mitigation or about damages caused).

Once the level of risk associated to the incident has been evaluated, the VDSS evaluates the available counter measures that the IoT infrastructure supports. This evaluation uses the information received from the Security Model Analysis, which also notifies about the cost associated to every mitigation. A potential human intervention is also considered here as the system admin can fine tune values related to costs and mitigations or evaluate the severity of the incident and criticality of the asset affected, providing with additional input to the DSS when proposing the most appropriate countermeasure to mitigate the ongoing incident. The selected countermeasure is notified to the Mitigation Action Service, which is in charge of notifying to the Security Orchestrator by adding the information about the selected countermeasure to a MSPL policy.

Listing 1. Except of MSPL file for representing a reaction

```
<configurationRule>
    <configurationRuleAction xsi:type='FilteringAction'>
      <FilteringActionType>DENY</FilteringActionType>
    </configurationRuleAction>
    <configurationCondition xsi:type='FilteringConfigurationCondition'>
        <isCNF>false</isCNF>
        <packetFilterCondition>
            <SourceAddress>2001:720:1710:4:1::7</SourceAddress>
            <DestinationAddress>abab::6/128</DestinationAddress>
            <ProtocolType>ICMP</ProtocolType>
        </packetFilterCondition>
    </configurationCondition>
```

```
    <externalData xsi:type='Priority'>
        <value>60000</value>
    </externalData>
    <Name>Rule0</Name>
    <isCNF>false</isCNF>
</configurationRule>
```

In parallel to the exchange of reaction information between the Reaction and the Orchestrator modules, additional information is exchanged with the Dynamic Security and Privacy Seal module through the Security Alert Service. A real time push mechanism is used to notify the Security Alert Service about the incidents detected, the alerts generated, and the mitigations chosen. Such information is transformed by the Security Alert Service into STIX (Structured Threat Information Expression) messages, whose format was designed by OASIS [1] for the description of cyber-security information in a standard way. It is mainly conceived for the threat intelligence exchange and it is used here to formalize the information received by the Seal Manager from the Monitoring and Reaction modules.

It is worth noticing that, although this process is designed to be executed in an automatic way, it is also considered the possibility of requiring explicit consent from the system administrator, which might also entail the modification of the security policy. The enforcement of the selected reaction is enforced at the orchestrator, using virtual network interfaces and SDN/IoT controllers available at the enforcement plane.

D. Reactive policy deployment

Figure 5 shows the main flow for a security policy enforcement as part of an automatic countermeasure. In this case the policy enforcement process is initiated by the Reaction module which is able to generate reactions depending on the threats (Fig. 5-step 1). As part of the reaction, it can be included one or several MSPL policies. These security policies are sent to the Security Orchestrator who identifies the main capabilities required by the policy (Fig. 5-step 3) and requests information to the System Model in order to get the available resources, as well as requesting a list of security enablers able to cope with the aforementioned capabilities to the Security Enabler Provider (Fig. 5-steps 4,5). Once the Security Orchestrator has gathered relevant information about the underlying architecture, the Resource Planner (in the Security Orchestrator) decides who is the best security enabler according to its knowledge of the architecture (Fig. 5-step 6). Then, it requests a policy translation to the Policy Interpreter, providing the MSPL policies and the selected security enablers for each one. When the Policy Interpreter receives the request, it obtains the specific plugin from the Security Enabler Provider for each security enabler and it performs the policy translation by executing the plugin for each received MSPL (Fig. 5-steps 8-12). Afterwards, the Policy Interpreter sends the result to both, the Policy Repository for traceability purposes, and the Security Orchestrator, in order to trigger the policy enforcement (Fig. 5-steps 13,14). Finally, the Security Orchestrator enforces the enabler configuration through different components of the framework depending on the nature of the countermeasure. In case the countermeasure requires a VNF which is not already deployed, the Security Orchestrator deploys it through the NFV MANO (Fig. 5-step 15). Otherwise it can use an existent VNF and enforce the configuration without a previous deployment. At this point, the configuration can be enforced through the NFV MANO, the SDN Controller or the IoT Controller depending on whether the countermeasure is a networking or IoT related configuration (Fig. 5-steps 16-18). Finally, once the policies have been enforced, the Security Orchestrator notifies the new policies status to the Policy Repository (Fig. 5-step 19).

[1]https://oasis-open.github.io/cti-documentation/stix/intro

VI. Architecture instantiation and deployment

This section describes the implementation of the architecture explained above as well as its deployment for validation in two main scenarios: Mobile edge Computing and Building Management system.

Regarding the implementation, the monitoring information is obtained from three main sources:

- Sensors attached to the IoT network: we include here sensors such as IDS (such as Suricata/Snort sensors), honeypots, access control logs, etc.
- Montimage Monitoring Tool (MMT) [29]: carrying out deep packet inspection for the detection of incidents, instantiating the monitoring module of the architecture.
- Operational data extracted from IoT devices: it is analyzed by Data analysis tool, which applies machine learning techniques for identifying anomalous behavior of IoT devices (for example, very high temperature detected by climate sensors).

A Kafka broker[2] is provided to gather the information from the different sources, normalizing it and getting it ready to be sent to the monitoring components.
The normalized data is sent, by using rsyslog, to the anomaly detection reasoner based on the Atos' XL-SIEM engine implementing the VDSS functionality of the architecture. The XL-SIEM provides a correlation engine that is able to detect incidents by analyzing events from different sources. The design of the XL-SIEM allows to attach different sources just by implementing new plugins that adapt and interpret new sources. Furthermore, the format used for the plugins uses a compatible format with open source SIEM solutions such as OSSIM[3], which increases the compatibility and flexibility of the model. The data retrieved by the Kafka broker, once normalized in rsyslog, is processed by the plugins of the XL-SIEM which prepares the received data for its correlation at the core of the XL-SIEM. The architecture of the XL-SIEM permits to decouple plugins and the XL-SIEM core. Since correlation activities can require a lot of computational resources, the XL-SIEM core allows to distribute the correlation activities among different instances, which increases not only efficiency but also robustness.

The correlation activities carried out by the XL-SIEM core generates alarms for the security incidents detected. The alarms are labelled with a certain level of criticality, which are

[2]https://kafka.apache.org/
[3]OSSIM https://www.alienvault.com/products/ossim

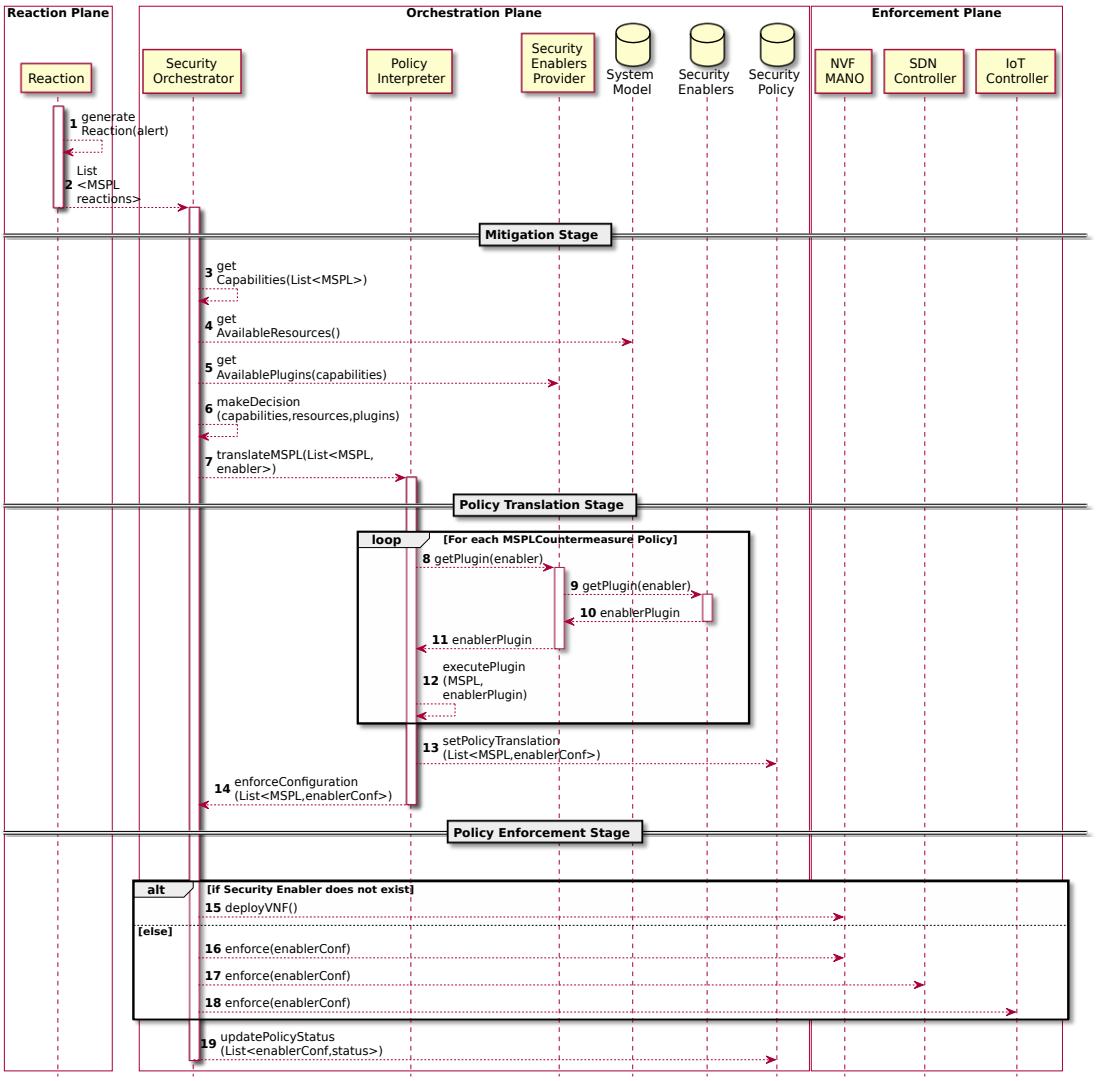

Fig. 5. ANASTACIA reactive policy enforcement process

used by the Mitigation Action Service to decide on the most suitable counter measure to react to the security incident. The orchestration engine is notified about the proposed reaction, along with specific information about the devices to receive the countermeasure (IPs of the devices affected, incidents to mitigate, etc.).

A. Mobile Edge Computing (MEC) scenario

Smart security cameras send recorded video to nearby MEC servers which can realize operations before sending the data to the cloud. These kind of scenarios are attractive to the hackers at different levels such as gain access to some security camera in order to get real time video, capture the data between the camera and the MEC server, compromise the MEC server in order to get or manipulate the data or the standard behavior, or even just take control of the IoT cameras in order to use them as an army with the aim of performing some kind of DDoS attack. In this scenario a ping flooding attack is considered.

1) MEC testbed: The testbed emulates several IoT devices inside a Cooja environment, which will attack with ICMP ping packets a victim outside the simulation – a server specifically deployed to act as the victim.

To detect the ongoing attack, an instance of the MMT monitoring tool is deployed inside the IoT network, including a specially-developed sniffer that allows extracting packets from an IoT network. The extracted flows will be analyzed using the DPI-enabled MMT-Probe tool, which is loaded with specific rules to detect ICMP ping bursts on IoT traffic. This detector will send events to the Kafka Broker – containing the IP of the affected device and the detected attack among other data – once the threat has been identified.

In the Kafka Broker, an Apache Storm[4] class will process these events and reformat the information using rsyslog, making them ready to be sent to the XL-SIEM tool for further analysis. Once the events have been read by the SIEM, the latter processes the information and produces an attack alert

[4]Apache Storm: http://storm.apache.org/

that is sent to the Mitigation Action Service by using a RabbitMQ channel.

To compute the countermeasures, our Mitigation Action Service implementation receives the SIEM alert and extracts the information contained within. This information is used to create the countermeasures defined in *MSPL* policies. The Mitigation Action Service uses a generates a MSPL with the the countermeasures for an ICMP flooding attack. The MSPL defines a filtering rule for the ICMP protocol, instantiated with context such as the IP addresses contained in the SIEM alert. The MSPL is sent to the Security Orchestrator that starts the deployment process of the countermeasures, retrieving the list of available security enforcers capable of enforcing the specified countermeasures.

To enforce the filtering policies, OSM (Open Source MANO)[5] together with ONOS[6] are suitable security enforcers, and the Security Enablers Provider returns this as a single security enforcer. Using this output, the Orchestrator selects a set of security enforcers and transforms the countermeasures, expressed in MSPL, into a lower-level configuration; the set of ONOS/OSM specific configurations that will be used to deploy the countermeasure. With this final translation, the Orchestrator deploys the countermeasures, creating a new Virtual Firewall using OMS and ONOS and deploying specific rules to redirect the traffic of the affected devices and filter the ICMP traffic.

The detection part of the instance described above is composed of 3 machines: a Cooja machine (for emulating an IoT network), an MMT-IoT-Probe machine (to run the incident detector), a Mitigation Action Service (MAS) machine (for executing the mitigation service). These instances were virtualized using VMs with the following features: 2 vCPUs @ 2.40GHz, 2 GB RAM and 20 GB of HDD.

B. Building Management System (BMS)

Nowadays it is very common to find a lot of sensors scattered in buildings. In fact, there are usually several sensors by room or dependencies measuring temperature, humidity, presence and luminosity among others. These measurements can be sent to a SCADA management system which usually implements different behaviors depending on the received measures (e.g., If the system receives a measurement beyond the configured threshold it could generate an alarm). Sometimes, this kind of reactions are really expensive to deploy, or just alter the standard behavior. On the other hand, if there is a real emergency but the sensors have been manipulated, the absence of reactions could be fatal. With this in mind, a malicious attacker could try to exploit the system in order to take some kind of advantage, to waste the resources of the target or harm in somehow to the target. This scenario permits to evaluate the benefits provided by the architecture to deal with the security challenges in IoT-enabled Critical Infrastructure (IoT-CIs) that can be affected by the sensor's manipulation. Instantiating this scenario, the architectural framework has been deployed in a real building where several IoT devices, including temperature, humidity and presence sensors deployed.

In our testbed, one of these IoT devices is compromised by an attacker, who gain access to it in order to manipulate the temperature sensor value to trigger the fire alarm of the building. As explained in [28] the monitoring modules are able to discern a regular situation of a abnormal one based on a previous training, so once the monitoring modules receive the tramped temperature, these analyze and correlate the information against its knowledge and them generates an alert to the reaction module (Mitigation Action Service). This reaction module computes the countermeasures based on the received information and determines that it is necessary to turn off the IoT device until a physical verification has been performed.

The countermeasure is modelled by a new MSPL policy, sent to the Security Orchestrator, that analyzes the security policy and obtains the required capability for the countermeasure. Since it is an IoT-related management operation, the Security Orchestrator decides to use the IoT controller and requests the policy translation to obtain the IoT configuration from the MSPL. Then, it gathers information from the system model in order to decide through which component it will enforce the configuration, that is sent to IoT controller. Finally, the IoT device receives the IoT message by the IoT controller to be turned off.

1) BMS testbed: The Policy Interpreter, Policy Repository, Security Enabler provider, Security Orchestrator and IoT controller have been developed from scratch in python using Django[7] rest framework and Falcon for the rest APIs and Mysql as main database engine. The MSPL policy models are an extended version of [26], that has been extended to consider additional requirements imposed by IoT. The IoT domain security enabler plugin also has been developed from scratch in order to translate the extended MSPL into IoT controller configuration. The Policy interpreter implements four different HTTP APIs to only translate and enforce the two different security policy levels. The policy Repository implements two different APIs to store the policy translations as well as the policy enforcement status. The Security enabler provider implements two different APIs to get the security enabler candidates as well as to get the selected plugin code. the Security Orchestrator implements three APIs to receive the reactions, to perform the policy translations and to gather information from the system model as well as the APIs for the infrastructure management. Finally, the IoT controller implements two APIs to receive the IoT devices configuration and to send the configuration using the specific IoT protocol implementation. In this case the IoT controller uses CoAP as IoT protocol.

Regarding the equipment used in this scenario, the monitoring and reaction modules has been deployed in the same premises as the previous scenario. The Policy Interpreter, Policy Repository, Security Enabler Provider and Security Orchestrator have been virtualized and dockerized in a Intel(R) Core(TM) i7-2600 CPU at 3.4GHz, using 3 vCores, 3.5GB

[5]OSM: https://osm.etsi.org/
[6]ONOS: https://onosproject.org/
[7]Django https://www.djangoproject.com/

of RAM and 30GB of HDD. The IoT Controller has been virtualized and dockerized in a Intel Core Processor (Haswell) at 1.5GHz using 2vCores, 2GB of RAM and 15GB of HDD. IoT devices: They are MSP430F5419A-EP at 25Mhz, 128 KB ROM and 16 KB RAM, running a customized version of Contiki OS 2.7 and erbium CoAP server. The 6lowPAN bridge: It is a MSP430F5419A-EP at 25Mhz, 128 KB ROM and 16 KB RAM, running a customized version of Contiki OS 2.7 in order to allow the communication between 802.15.4 and 802.3.

VII. Performance Evaluation

1) Monitoring performance: To measure the performance of the monitoring phase we have conducted different experiments in order to determine the detection time of a ping flooding attack in an IoT scenario. We have setup an emulated IoT network (using Cooja) that contains a compromised device which will perform a ping DoS attack towards another device. In addition, we have setup the MMT-IoT solution to extract the IoT packets and detect the ICMP flooding using MMT-Probe. The test script analyzed the logs of two tools: the Cooja logs to identify the timestamp when the attack was triggered and the logs of MMT-Probe in order to identify the timestamp when the attack was detected by the Incident Detector. The test script triggered the attack in the (emulated) compromised device. The verdicts generated by MMT are then sent to the Monitoring Module, composed by two components of the XL-SIEM tool: the XL-SIEM Agent and the XL-SIEM Server, measuring the processing time on both components. The former will receive, process and filter the events from the MMT-Probe sensor, while the latter will perform the risk analysis and generate a detailed attack report. Finally, this attack report is sent to the MAS which will generate the respective MSPL containing the reaction. The processing time is also measured in this last step. This experiment was run 100 times, in order to compute the average detection and reaction times for the ICMP flooding attack (MEC) at each component of the platform.

Figure 6 depicts the measured times in each component. For each iteration, the processing times have been stacked to show the total processing time of the ANASTACIA platform. The principal contributor of the platform's reaction time are the analysis and detection components – MMT in this case – with an average of 445 ms. The MMT needs to test a security property: observe different ICMP packets per second, raising the alert as soon as this condition is met. The second contributor to the total time is the XL-SIEM Server, with an average of 57 ms. This includes the risk analysis performed, aimed to enrich the data received from the detection engines and determine the best set of countermeasures to the ongoing attack. On the contrary, the XL-SIEM Agent and the MAS add constant times to the whole process (3.4 and 8.4 ms respectively) due to their high parallelism. The average time for the four analyzed components is 514 ms, counting from the moment the attack is triggered until the reaction is computed.

2) Incident handling Performance: To evaluate the performance of the Incident handling in both MEC and BMS scenarios we have increasing number of events that are sent to the Incident Detector at different pace. The Incident Detector (XL-SIEM), is an Apache Storm based incident detector that uses an complex event processing (CEP) engine, namely Esper[8], to correlate events. Events received by the Incident Detector are filtered, processed and correlated to generate security alarms, which are then sent to the Reaction Module via the Kafka broker. The MAS collects these alarms and generates a MSPL file which is sent to the Security Orchestrator.

To perform the tests, we have sent a burst of 100 events at different frequencies: 20Hz, 10Hz and 4Hz. In order to test the performance of the components involved in the monitoring, detection and reaction, we have forced the generation of a security alert per event sent. Therefore, the Incident Detector has generated 100 alarms at the pace indicated above. It is worth noticing that, in a normal operation environment, events from the same source IP will trigger just one alarm every 60 seconds or several minutes (depending on the rule configured at the Esper engine). This would allow to filter large amounts of events and to avoid unnecessary overload to the system. In this performance evaluation we have simulated one of the worst cases in terms of workload, both for the Incident Detector, VDSS, MAS and Orchestrator.

The results of the tests are represented in the following chart (Figure 7). The y-axis represents the time taken by the Incident Detector to process each event, from the time it is received to the time when it is sent to the messaging queue to be consumed by the MAS. As it can be seen, the response time of Incident Detector is stable, although some peaks are present in all tests. The higher peak is present at 10Hz (in event #79), with a response time of 307ms. Despite those anomalies the Incident Detector recovers the normality quickly. It must be observed, however, that at 20Hz it takes longer to recover the stability (e.g. from event #38 to #45).

To understand those delays we have to consider the XL-SIEM architecture. It is implemented in Apache Storm which uses multiple threads in order to process several processes simultaneously. Each thread (or bolt, as Apache Storm names a singular thread or process) is in charge of a singular task, like storing events in database, correlate events, send alarms to RabbitMQ, store alarms in database, etc. As long as there are more threads than cores, the cores are shared by different threads. The execution order of the different tasks can be different in some cases, consequently the measured processing time may fluctuate. Furthermore, some tasks use shared resources (like databases), hence mutual exclusion situations may occur and increase the processing time. Those variations are appreciated in the charts as peaks.

Alarms generated by the Incident Detector are sent to a RabbitMQ message queue, where the MAS is attached to receive these alarms. There's a transfer time of approximately 40-50ms. This time may vary depending on the network conditions and on the sync difference between the machines where the time-stamps are measured.

In addition, some extra tests have been done to evaluate the XL-SIEM Server performance in a different stressing scenario.

[8]Esper http://www.espertech.com/esper

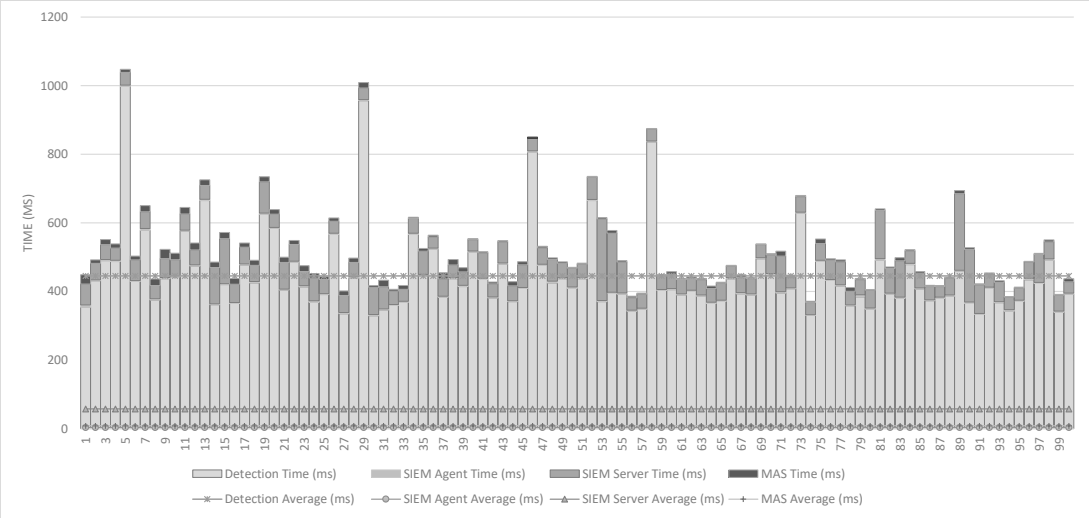

Fig. 6. DDoS attack detection performance in MEC scenario

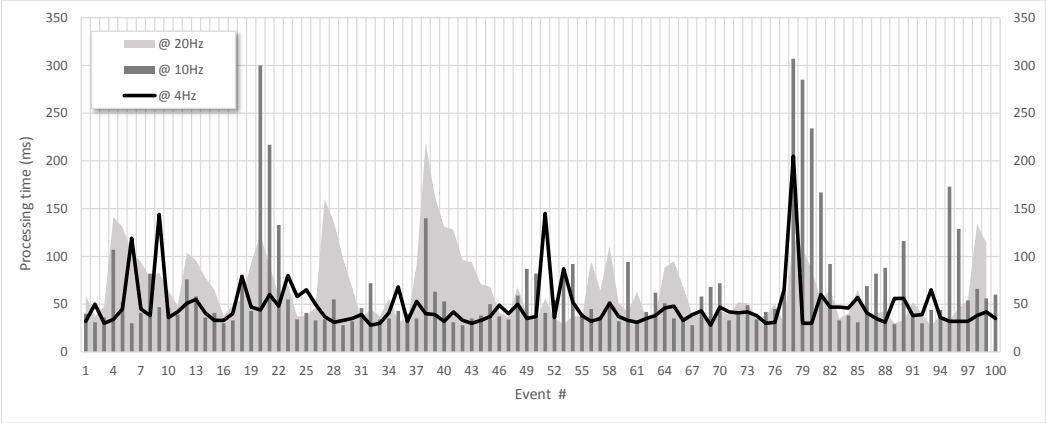

Fig. 7. Incident handling performance, bursts of 100 events at different frequencies

In this case, the alerts are triggered when detecting different combined attacks, mixing up to 11 events to generate an alarm. Once again, the attacks have been generated at different frequencies: 4Hz, 10Hz and 20Hz. This scenario stresses the correlation engine, creating a bottleneck in this component at higher frequencies. The tests results are represented in Figure 8, where the processing time increases constantly at 20Hz and the bottleneck effect can be appreciated. These tests demonstrate how the complexity of the alarm rules can impact the XL-SIEM performance.

3) Reaction enforcement Performance: Figure 9 shows the performance for the reactive policy translation. It measures along 100 iterations the time required by the Policy Interpreter to translate the reactive MSPL into both, filtering SDN and power management IoT configurations. As it can be seen, the measurements obtained for the policy translation are quite similar, with an average time close to 36 ms taking into account the variations that the system may suffer during execution. This is due to the extension and complexity of the filtering policies for both scenarios are similar.

Figure 10 shows the performance for the reactive policy enforcement, which measures along 100 iterations the time taken by the Security Orchestrator to enforce the configurations. It covers since the Security Orchestrator receives the MSPL policy until it receives the enforcement confirmation from the security enabler. In this case it is not taking into account the policy translation since it has been shown previously. In the results we can see that the policy enforcement though the SDN controller takes approximately half the time of the enforcement through the IoT controller. It can be explained as the enforcement against the IoT devices is sent through a 6LowPAN network while the SDN enforcement is performed through Ethernet.

If we look at the full life-cycle of the framework detection and reaction processes in terms of average times, taking as example the first scenario, we can see that the detection part takes around 460 ms, the incident handling takes around 58 ms, the policy translation is close to 36 ms, and finally, the policy enforcement shows results around 370 ms. If we consider the whole cycle, the framework is detecting and

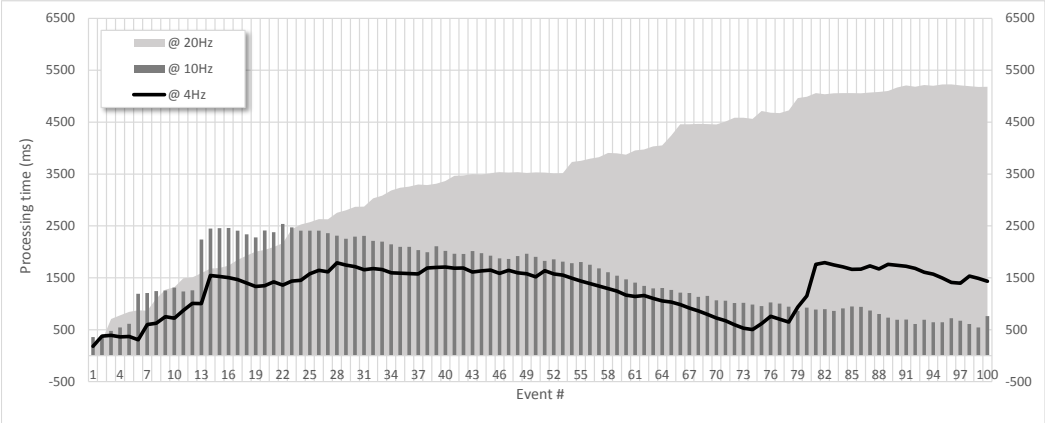

Fig. 8. Incident handling performance, bursts of 100 combined attacks (1100 events) at different frequencies

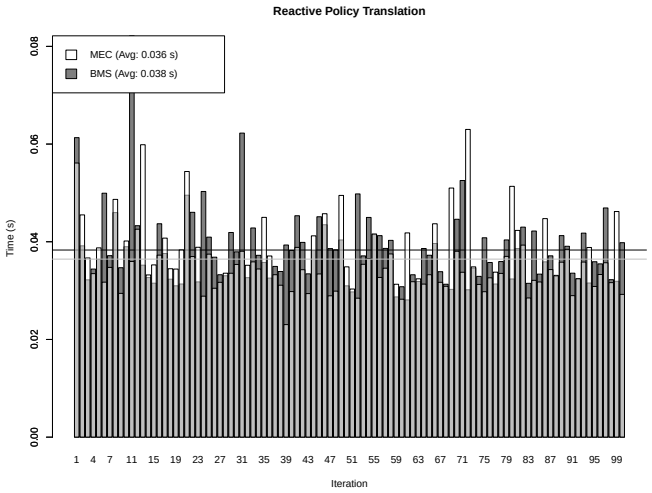

Fig. 9. Reactive Policy Translation

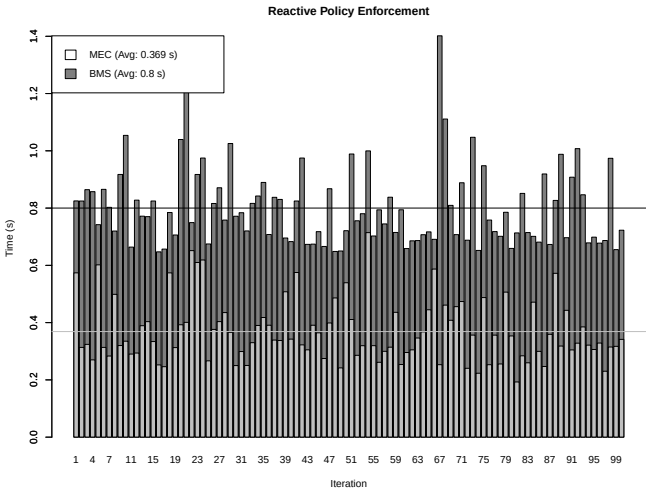

Fig. 10. Reactive Policy Enforcement

enforcing the countermeasures in less than one second. It is important to highlight that this result depends largely on the complexity of the countermeasure as well as the kind of attack. i.e. in general, the more complex countermeasure, the longer it will take its deployment.

VIII. Conclusions

This paper has presented Anastacia security management architecture aimed to deal with the security and privacy in NFV/SDN-enabled IoT scenarios, detailing the different planes of the architecture as well as the main architectural flows. In addition, the main IoT thread/attacks and their suggested potential detection and reaction mechanisms based on NFV-SDN has been presented. The architecture has been successfully instantiated and tested in two different scenarios; The Mobile Edge Computing and IoT-enabled Critical Infrastructure in Building Management Systems. In these scenarios we tested DDos and IoT malware attacks respectively detailing the autonomic reaction processes to mitigate them. The accomplished performance evaluation demonstrated the feasibility of the solution to automatically monitor, detect, react and mitigate IoT cyber-attacks, enforcing proper security policies, in reasonable times (depending on kind of attack and reaction countermeasure), accounting the latency and delays incurred in IoT networks.

As future work, we envisage to extend the supported cyber-threats detection and mitigation possibilities by addressing reactive VNF orchestration and deployment at the edge of IoT constrained systems and networks.

Acknowledgment

The research has been supported by the H2020 EU project ANASTACIA, Grant Agreement N 731558. The research has been also supported by a postdoctoral INCIBE grant within the ”Ayudas para la Excelencia de los Equipos de Investigación Avanzada en Ciberseguridad” Program, with code INCIBEI-2015-27363.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128610001568

[2] J. L. H. Ramos, J. B. Bernabe, and A. F. Skarmeta, "Managing context information for adaptive security in iot environments," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, March 2015, pp. 676–681.

[3] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIV. New York, NY, USA: ACM, 2015, pp. 5:1–5:7. [Online]. Available: http://doi.acm.org/10.1145/2834050.2834095

[4] J. P. Santos, R. Alheiro, L. Andrade, n. L. Valdivieso Caraguay, L. I. Barona López, M. A. Sotelo Monge, L. J. Garcia Villalba, W. Jiang, H. Schotten, J. M. Alcaraz-Calero, Q. Wang, and M. J. Barros, "Selfnet framework self-healing capabilities for 5g mobile networks," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1225–1232. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3049

[5] P. Östberg, J. Byrne, P. Casari, P. Eardley, A. F. Anta, J. Forsman, J. Kennedy, T. L. Duc, M. N. Mariño, R. Loomba, M. . L. Peña, J. L. Veiga, T. Lynn, V. Mancuso, S. Svorobej, A. Torneus, S. Wesner, P. Willis, and J. Domaschka, "Reliable capacity provisioning for distributed cloud/edge/fog computing applications," in *2017 European Conference on Networks and Communications (EuCNC)*, June 2017, pp. 1–6.

[6] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE transactions on reliability*, vol. 64, no. 3, pp. 1086–1097, 2015.

[7] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 325–346, Firstquarter 2017.

[8] S. Chakrabarty, D. W. Engels, and S. Thathapudi, "Black sdn for the internet of things," in *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, Oct 2015, pp. 190–198.

[9] P. Bull, R. Austin, E. Popov, M. Sharma, and R. Watson, "Flow based security for iot devices using an sdn gateway," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, Aug 2016, pp. 157–163.

[10] O. Flauzac, C. González, A. Hachani, and F. Nolot, "Sdn based architecture for iot and improvement of the security," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, March 2015, pp. 688–693.

[11] S. Choi and J. Kwak, "Enhanced sdiot security framework models," *International Journal of Distributed Sensor Networks*, vol. 12, no. 5, 2016.

[12] V. Varadharajan and U. Tupakula, "Security as a service model for cloud environment," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 60–75, 2014.

[13] B. R. Al-Kaseem and H. S. Al-Raweshidyhamed, "Sd-nfv as an energy efficient approach for m2m networks using cloud-based 6lowpan testbed," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1787–1797, Oct 2017.

[14] W. Yang and C. Fung, "A survey on security in network functions virtualization," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, June 2016, pp. 15–19.

[15] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10 – 28, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804517301455

[16] I. Farris, T. Taleb, Y. Khettab, and J. S. Song, "A survey on emerging sdn and nfv security mechanisms for iot systems," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2018.

[17] C. Shiva Shankar, A. Ranganathan, and R. Campbell, "An eca-p policy-based framework for managing ubiquitous computing environments," 08 2005, pp. 33– 42.

[18] C. Rensing and M. Karsten, "Aaa: a survey and a policy-based architecture and framework," 2002.

[19] A. M. Hadjiantonis, A. Malatras, and G. Pavlou, "A context-aware, policy-based framework for the management of manets," *Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, pp. 10 pp.–34, 2006.

[20] A. L. Shaw, L. Jacquin, A. Lioy, C. Pitscheider, C. Basile, F. Risso, R. Bonafiglia, F. Ciacca, M. Nemirovsky, J. Kuusijärvi, D. Montero, R. Serral-Gracià, M. Yannuzzi, and F. Bosco, "Specification of the secured architecture (alpha version)," Tech. Rep.

[21] S. Ziegler, A. Skarmeta, J. Bernal, E. Kim, and S. Bianchi, "Anastacia: Advanced networked agents for security and trust assessment in cps iot architectures," in *2017 Global Internet of Things Summit (GIoTS)*, June 2017, pp. 1–6.

[22] I. Farris, J. Bernabe, N. Toumi, D. Garcia-Carrillo, T. Taleb, A. Skarmeta, and B. Sahlin., "Towards Provisioning of SDN/NFV-based Security Enablers for Integrated Protection of IoT Systems," in *IEEE Conference on Standards for Communications and Networking (CSCN-2017)*, 2017.

[23] A. Molina Zarca, J. Bernal Bernabe, I. Farris, Y. Khettab, T. Taleb, and A. Skarmeta, "Enhancing iot security through network softwarization and virtual security appliances," *International Journal of Network Management*, vol. 28, no. 5, p. e2038, e2038 nem.2038. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.2038

[24] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, vol. 50, no. 2, pp. 76–79, Feb. 2017. [Online]. Available: doi.ieeecomputersociety.org/10.1109/MC.2017.62

[25] Y. Gao, Y. Peng, F. Xie, W. Zhao, D. Wang, X. Han, T. Lu, and Z. Li, "Analysis of security threats and vulnerability for cyber-physical systems," in *Proceedings of 2013 3rd International Conference on Computer Science and Network Technology*, Oct 2013, pp. 50–55.

[26] C. Basile, A. Lioy, C. Pitscheider, F. Valenza, and M. Vallini, "A novel approach for integrating security policy enforcement with dynamic network virtualization," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–5.

[27] A. Molina Zarca, D. Garcia-Carrillo, J. Bernal Bernabe, J. Ortiz, R. Marin-Perez, and A. Skarmeta, "Enabling virtual aaa management in sdn-based iot networks †," *Sensors*, vol. 19, no. 2, 2019. [Online]. Available: http://www.mdpi.com/1424-8220/19/2/295

[28] D. Mehta, A. E.-D. Mady, M. Boubekeur, and D. M. Shila, "Anomaly-based intrusion detection system for embedded devices on internet," in *The Tenth International Conference on Advances in Circuits, Electronics and Micro-electronics*, 2018.

[29] B. Wehbi, E. M. de Oca, and M. Bourdelles, "Events-based security monitoring using mmt tool," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, April 2012, pp. 860–863.