Received: 2 December 2017 | Revised: 24 May 2018 | Accepted: 24 May 2018
DOI: 10.1002/nem.2038

SPECIAL ISSUE PAPER

WILEY

Enhancing IoT security through network softwarization and virtual security appliances

Alejandro Molina Zarca[1] | Jorge Bernal Bernabe[1] | Ivan Farris[2] | Yacine Khettab[2] | Tarik Taleb[2] | Antonio Skarmeta[1]

[1]Department of Information and Communication Engineering (DIIC), University of Murcia, Murcia, Spain
[2]Department of Communications and Networking, Aalto University, Espoo, Finland

Correspondence
Jorge Bernal Bernabe, Facultad de Informatica, Universidad de Murcia, S/N, Espinardo, Murcia 30100, Spain.
Email: jorgebernal@um.es

Funding information
Horizon 2020 Research and Innovation Programme H2020 ANASTACIA, Grant/Award Number: 731558; Ayudas para la Excelencia de los Equipos de Investigacion Avanzada en Ciberseguridad, Grant/Award Number: INCIBEI-2015-27363

Summary

Billions of Internet of Things (IoT) devices are expected to populate our environments and provide novel pervasive services by interconnecting the physical and digital world. However, the increased connectivity of everyday objects can open manifold security vectors for cybercriminals to perform malicious attacks. These threats are even augmented by the resource constraints and heterogeneity of low-cost IoT devices, which make current host-based and static perimeter-oriented defense mechanisms unsuitable for dynamic IoT environments. Accounting for all these considerations, we reckon that the novel softwarization capabilities of Telco network can fully leverage its privileged position to provide the desired levels of security. To this aim, the emerging software-defined networking (SDN) and network function virtualization (NFV) paradigms can introduce new security enablers able to increase the level of IoT systems protection. In this paper, we design a novel policy-based framework aiming to exploit SDN/NFV-based security features, by efficiently coupling with existing IoT security approaches. A proof of concept test bed has been developed to assess the feasibility of the proposed architecture. The presented performance evaluation illustrates the benefits of adopting SDN security mechanisms in integrated IoT environments and provides interesting insights in the policy enforcement process to drive future research.

1 | INTRODUCTION

A huge amount of smart devices is expected to drastically change industrial and home environments by enabling new advanced services for human beings. The Internet of Things (IoT) vision aims at seamless integrating the sensing and actuation features of common objects by leveraging their network capabilities to create pervasive information systems.[1] To this aim, the sensing measurements generated by IoT devices can provide contextual and valuable information of the surrounding environments. The relevant data analysis systems can then derive appropriate control decision, which can be enforced in the physical world through the actuation features of smart devices. The envisioned benefits are boosting the adoption of IoT solutions in a broad range of application scenarios.

On the other hand, the increased connectivity can be exploited by malicious attackers to exploit devices vulnerabilities.[2] Indeed, accounting for the heterogeneity of IoT devices, ranging from smart industrial machinery to simple wearable sensors, it results extremely complex to ensure the same desired protection over different programming environments.

Furthermore, the majority of host-centric security mechanisms do not typically fit into the resource constraints of IoT devices. The absence of automated software updates, as well as users' misconfiguration, can notably increase the potential vulnerabilities, especially because of the unavailability of vendors' support along the whole IoT product life cycle. Last but not the least, current network security solutions present low responsiveness and can unlikely cope with the dynamic IoT environments. All these security vectors claim for new advanced mechanisms able to meet the desired defense levels.

We strongly believe that network environments can exploit their privileged positions in providing capillary connectivity to provide the desired defense mechanisms for IoT ecosystem. In this vein, Telco networks are progressively facing a drastic transformation by embracing software-defined networking (SDN) and network function virtualization (NFV).[3] Software-defined networking introduces a new level of network programmability, by decoupling control and data plane. This network model can enable novel security defense mechanisms, such as promptly managing malicious traffic and enabling secure network zone. Network function virtualization leverages virtualization technologies to deploy network elements as software instances, thus allowing an increased level of flexibility and elasticity in service provisioning. Furthermore, NFV can enable remarkable reduction in CAPEX/OPEX costs, by replacing dedicated hardware with commodity servers able to host software-based network appliances, including virtual security functions. Leveraging the SDN and NFV features, notable benefits can be achieved by off-loading the security of IoT systems within network.

In this paper, we aim at investigating the opportunities that software-based network mechanisms jointly offer in coping with security threats against IoT systems. The paper evolves a previous position paper published in a conference,[4] extending the related work and detailing the policy refinement and enforcement processes, as well as the implementation and performance evaluation of the solution. The main contributions of the paper are the following:

- We provide a comprehensive analysis of the security features introduced by SDN and NFV technologies. Thus, we emphasize their benefits to cope with IoT security threats.
- We design a novel framework to provide security protection mechanisms through the envisioned software-based network enablers and to create added-value services accounting for potential integration with existing IoT security mechanisms. To account for the heterogeneity of the security enablers, the security policies management include different levels of abstractions, so to decouple the desired defense intent from the low-level configuration of the underlying components and to enable a technology-agnostic refinement process. Specific focus concerns the orchestration features, which need to enforce the desired security controls over heterogeneous domains, such as SDN, NFV, and IoT networks, by interfacing with relevant control and management modules.
- We present a proof of concept (PoC) implementation of the proposed framework to assess its feasibility and performance. The envisioned modular environment include the main component and can be easily extended to allocate further security mechanisms and to perform quick, scalable, and automated testing.
- We provide a comprehensive analysis of the policy enforcement, pointing out the benefits of SDN-based security mechanisms in integrated IoT scenarios with respect to conventional countermeasures.

The paper is organized as follows. In Section 2, we analyze the novel security features introduced by SDN and NFV, providing an overview of relevant state-of-the-art security solutions for IoT systems. Section 3 describes the envisioned framework, highlighting the main components and policy-based refinement processes to enforce the desired security mechanisms. In Section 4, two promising use cases are presented to assess the introduced security features. In Section 5, we illustrate the PoC implementation of the envisioned architecture, reporting the experimental results to enforce SDN security mechanisms in integrated IoT environments. Conclusions and future research activities are drawn in Section 6.

2 | BACKGROUND ON SDN/NFV-BASED SECURITY ENABLERS FOR IOT NETWORK

Network softwarization is drastically modifying the telecommunication industries, potentially enabling added-value services. Software-defined networking and NFV can play key roles as security enablers, ensuring the network environments to cope with a broad range of security threats of IoT systems. In the following subsections, we will briefly present these two paradigms, emphasizing their relevant security features.

2.1 | SDN security mechanisms

Software-defined networking is a network architecture that aims to guarantee enhanced network programmability by decoupling the control and forwarding functions. In this way, network management can be done separately, without affecting data flows network, and can be performed by a centralized controller. As a consequence, the complexity of the underlying switching devices is notably reduced in comparison with traditional networks. The derived SDN network results into a simpler programmable environment, allowing external applications to define the network behavior.

Open Networking Foundation (ONF), a nonprofit consortium dedicated to development, standardization, and commercialization of SDN, has suggested a reference SDN architecture model composed of three layers. The data plane includes network elements (eg, switches and routers) that are exploited to process packets based on the rules provided by a controller and for collect network status information, such as network topology and traffic statistics. The control plane bridges the application plane and the data plane, translating applications requirements into appropriate forwarding rules to be enforced over the underlying network switches. To this aim, the southbound interface (eg, OpenFlow) allows the SDN controller to access functions provided by the switching devices. These functions may include reporting network status and managing packet forwarding rules. On the other hand, the northbound interface provides service access points in various forms, eg, application programming interfaces (APIs), to the application plane. In this way, SDN applications can communicate their network requirements to the SDN controller, so to dynamically modify network behavior and request new packet forwarding strategies.

Software-defined networking networks are gaining high momentum within the IoT ecosystem, and several recent works have investigated its effective adoption for interconnecting IoT devices. Sensor OpenFlow[5] has been specifically devised to extend the use of OpenFlow to sensing nodes, by addressing the technical challenges of adopting SDN in wireless sensor networks. SDN–WISE, proposed by Galluccio et al,[6] goes beyond providing stateful mechanisms for the definition and handling of flow tables. In De Oliveira et al,[7] an architecture with multiple controllers is proposed within the wireless sensor network to better manage the control traffics.

This increased network programmability is boosting the use of SDN also in the security research communities.[8] In the following, we comprehensively analyze the major SDN features that can be explored to provide advanced security countermeasures for IoT systems.

- Dynamic flow control: By leveraging the decoupling of control and data planes, a network application can manage network flows dynamically. Indeed, when an SDN switch does not have a flow rule to process a specific packet, a relevant request is forwarded to the controller that can decide the relevant packet processing based on specific application policies. This feature can enable a dynamic access control function, which is commonly implemented to protect a network according to the specified privileges and policies.
- Traffic isolation: SDN can be exploited to enable forwarding of different network traffics over the same physical network infrastructure, while guaranteeing the desired level of isolation and network abstraction.[9] This feature can drastically limit the propagation and damages of security attacks between different network domains. This represents a fundamental feature in IoT system, where sensitive operations can depend on data generated by other objects. In this scenario, the prompt isolation of compromised IoT nodes becomes compelling and SDN flexibility can be used to separate malicious (or suspicious) network flows dynamically.
- Network-wide visibility and monitoring: All SDN data traffic are managed by a centralized controller that is in charge of flow rule configuration. In addition, through the control plane, network status information can be collected from each data plane by sending statistics query messages. Therefore, a network application running on the control plane can have updated status of relevant data plane and flow request messages through the northbound APIs. In this way, SDN can ease the network-wide monitoring and the detection/defense of network-wide attacks. For example, the network administrator can implement anomaly analysis to identify network-wide attacks by monitoring the network state changes, for example, to detect anomalous traffic peaks generated by distributed botnet of compromised IoT devices.
- Network programmability: Since data forwarding in an SDN network can be controlled by a network application program, SDN provides an enhanced flexibility to enable new network security functions. In this vein, the OpenFlow features have been exploited in Shin et al[10] to develop the FRESCO framework to simplify the development of security network functions through the modular composition of OpenFlow-enabled detection and mitigation modules.

In Shin et al,[11] several examples of SDN-based security applications are described, while the feasibility of deploying various security functions at the SDN control level has been investigated in Yoon et al.[12] Another survey[13] focused on SDN's strengths and weaknesses against distributed denial-of-service (DDoS) attacks in cloud computing environments.

Software-defined networks are starting to be exploited in IoT environments to deal with cyberthreats. Xu et al[14] describe an smart security mechanism to defend against new-flow attack in SDN-based IoT, through using a dynamic access control prior adding new flows to the SDN switch. Choi[15] suggests a network architecture in which the DDoS defense is performed by limitation of flow amount for the same application per data source. Chakrabarty et al[16] present an SDN architecture for secure IoT networking and communications, securing both the metadata and the payload within each layer of an IoT communication packet while using the SDN-centralized controller as a trusted third party for secure routing and optimized system performance management. In Bull et al,[17] the authors propose the use of an SDN gateway as a distributed means of monitoring the traffic originating from and directed to IoT-based devices, enabling the detection of anomalous behavior and then performing an appropriate response (blocking, forwarding, or applying quality of service). Flauzac et al[18] propose a multidomain-distributed SDN security solution for wired, wireless, ad hoc, and sensor networks, showing how to distribute the security rules across different security controllers in order not to compromise the security of one domain. Choi and Kwak[19] propose strategies for establishing a security framework based on a software-defined IoT environment and efficient provision of security services such as authentication, access control, and lightweight encryption. The aforementioned works do not exploit the NFV benefits to increase on-demand scalability and dynamic deployment of virtual security functions within the network.

2.2 | Virtualized security network functions

The deployment of virtualized network services provides remarkable benefits in terms of increased flexibility, improved capital efficiency, and enhanced operational efficiencies in Telco networks. ETSI industry specification group NFV has designed a high-level functional architectural framework for the management of virtualized network functions,[20] which includes three domains. The NFV infrastructure (NFVI) comprises the hardware resources providing necessary processing, storage, and network capabilities, as well as the virtualization software components, to create the virtualization environment. The virtualized network function (VNF) domain refers to the VNFs that are executed leveraging the virtualized resources offered by the underlying NFVI. The management and orchestration (MANO) block is in charge of all the resources in the infrastructure layer for the efficient deployment of the VNFs. To this aim, it interacts with both the NFVI and the VNF blocks.

By leveraging the cloud delivery models of on-demand service provisioning, the NFV paradigm can drastically enhance the the Telco network embracing the concept of security as a service.[21] In this vein, the Cloud Security Alliance has defined guidelines for cloud-delivered defense solutions to assist enterprises and end user to widely adopt this security paradigm shift. The NFV approach presents remarkable advantages with respect to the hosting in remote cloud data centers, since the virtualized security functions can be deployed along the forwarding path, avoiding inefficient traffic detouring. Furthermore, the provisioning of security functions towards the edge of the network can better scale with the expected huge amount of traffic generated by IoT devices.

In this regard, we provide a critical analysis of the main features that boost the adoption of NFV paradigm for supporting novel security enablers and to improve the management of virtualized security functions for dynamic IoT environments.

- Decoupling security functions from hardware: The increased abstraction introduced by virtualization technologies is boosting the replacing of dedicate middlebox hardware with equivalent software instances running on top of commodity servers. This aspect can be introduce notable benefits in the network security domain, introducing virtualized instances of firewalls, DPIs, etc. The on-demand provisioning of network-based security mechanisms enables the off-loading of security functions from constrained IoT devices.[22,23]
- On-demand scalability: Another key feature of virtualized security function is represented by the elasticity to scale instances up/down according to the current workload. This higher level of scalability is also strictly dependent on the security application development, which can notably impact the overall performance. To this aim, a characterization of security functions is required to achieve the desired performance in a virtualized environment.
- Mobility support: Different IoT application scenarios, such as wearable sensors and smart vehicles, are characterized by mobility features. In this vein, the off-loading of security functions towards the network edge allows to process and monitor the traffic near to the devices. To keep the desired performance, the support of mobility will guarantee the desired security performance features, avoiding traffic rerouting and enabling short response time.
- Flexible network service provisioning: The software-based deployment allows for increased efficiency in the deployment of services over a shared physical infrastructure. In Basile et al,[24] an approach towards the adoption of security policies management with dynamic network virtualization is presented, although this work does not specifically deal with IoT features. Furthermore, leveraging software-based networking, different components can be dynamically integrated along the forwarding paths. This can enable the creation of appropriate security service chains where user traffic is appropriately processed according to security policies.

To sum up, SDN and NFV features are extremely promising to improve the security of IoT systems, but their joint use is currently at a preliminary stage and significant efforts are still required to fully exploit their benefits. Our research work proposes a holistic policy-based security management framework for the IoT that relies on SDN and NFV security enablers to strengthen the overall security and mitigate IoT cyber attacks.

3 | AN SDN/NFV-BASED SECURITY FRAMEWORK FOR IOT

3.1 | Architecture overview

The envisioned security framework aims at exploiting the features of SDN/NFV-based security enablers to ensure self-protection, self-healing, and self-repair capabilities in IoT systems, complementing conventional security approaches. To this aim, security policies are defined according to different level of abstraction, so ensure higher flexibility and manage security controls over heterogeneous networks. The required security actions can be enforced in different kinds of physical/virtual appliances, including both IoT networks and software-based networks. The proposed architecture includes three main planes, as shown in Figure 1.

3.1.1 | User plane

The user plane provides interfaces and tools allowing end users to specify the desired policy definition, service monitoring, and management. Its policy editor provides an intuitive and user-friendly tool to configure security policies in high-level security language, governing the configuration of the system and network, such as authentication, authorization, filtering, channel protection, and forwarding.

3.1.2 | Security orchestration plane

The security orchestration plane enforces policy-based security mechanisms and provides run-time reconfiguration and adaptation of security enablers, thereby providing the framework with intelligent and dynamic behavior. It is an innovative layer of our architecture and provides dynamic reconfiguration and adaptation in case of deviation from the expected behavior.

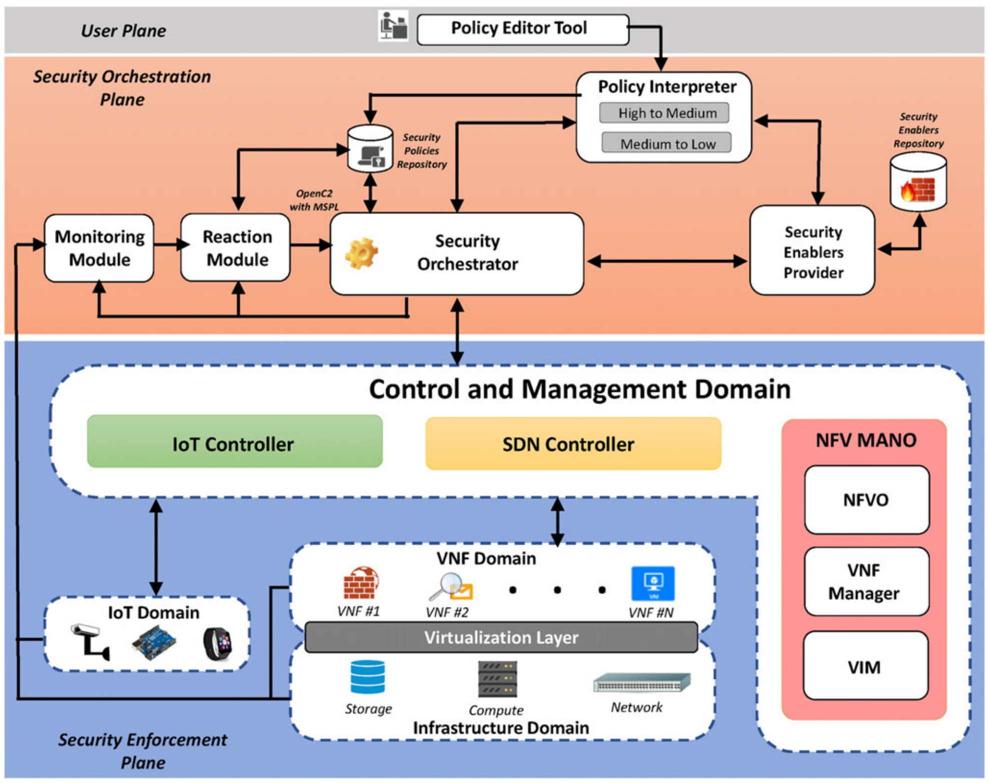

FIGURE 1 Architecture high-level overview

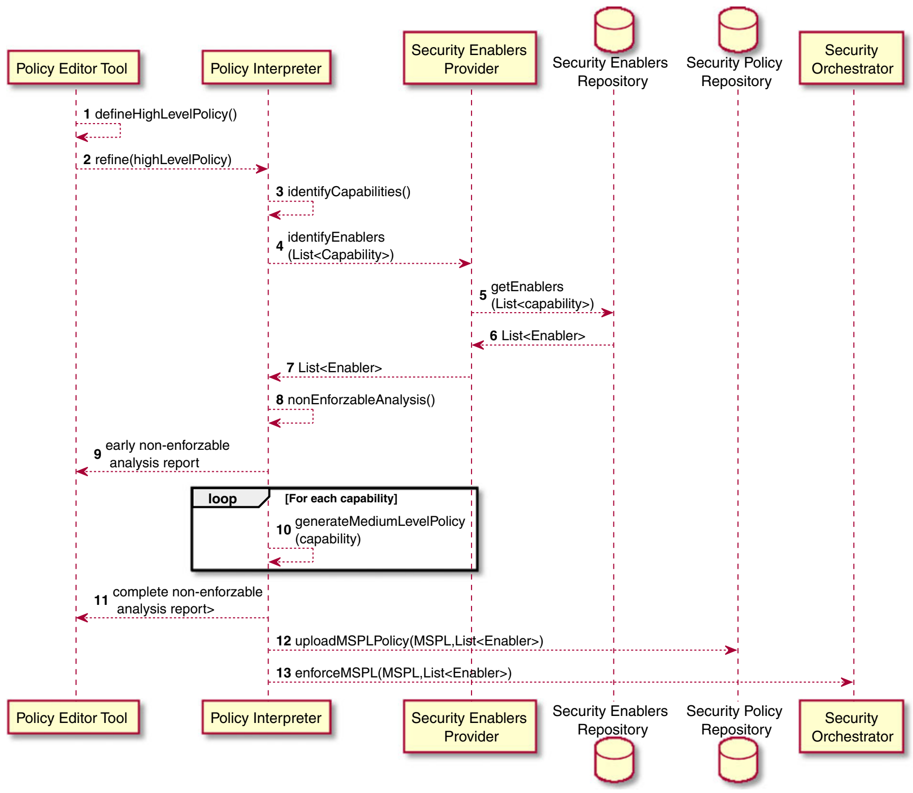
Policy Editor Tool
Policy Interpreter
Security Enablers Provider
Security Enablers Repository
Security Policy Repository
Security Orchestrator
1 defineHighLevelPolicy()
2 refine(highLevelPolicy)
3 identifyCapabilities()
4 identifyEnablers (List<Capability>)
5 getEnablers (List<capability>)
6 List<Enabler>
7 List<Enabler>
8 nonEnforzableAnalysis()
9 early non-enforzable analysis report
loop
[For each capability]
10 generateMediumLevelPolicy (capability)
11 complete non-enforzable analysis report>
12 uploadMSPLPolicy(MSPL,List<Enabler>)
13 enforceMSPL(MSPL,List<Enabler>)

FIGURE 2 High to Medium Security Policy language translation

The *Policy Interpreter* module plays a key role in the refinement of security policies. The high-level policies are first translated into medium-level security policy language, which allows to specify security mechanisms in a technology-agnostic way. Then these policies are refined in specific low-level configurations according to the selected enablers. The policy refinements process are further detailed in Section 2.

The *Security Enablers Provider* identifies the available security enablers according to the required capabilities and their relevant resource requirements. It also manages the security enabler plug-ins to generate low-level security configurations.

The *Monitoring* component collects security-focused real-time information related to the system behavior from physical/virtual appliances. Its main objective is to provide alerts for the reaction module in case something is misbehaving. Security probes are deployed in the infrastructure domain to support the monitoring services. Then the *Reaction* component is in charge of providing appropriate countermeasures, eg, by selecting policies stored in the relevant repository and by requiring reconfiguration of the security enablers to cope with the detected threat.

The *Security Orchestrator* supervises the orchestration of the security enablers to be deployed into the security enforcement plane according to the policy requirements. In addition, at run time, it analyses the reaction outcomes and applies the corresponding countermeasures. In this way, the overall framework can guarantee self-healing and resilience abilities, by constantly ensuring the satisfaction of the security requirements defined in the end-user policies.

Although it is not shown in Figure 1, the envisaged architecture is also endowed by a transversal plane called *Seal Management Plane* that combines security and privacy standards. This plane provides users with a run-time indication of the overall level of trust in the system, combining normative approaches and run-time monitoring. Its normative approaches include analysis and integration with international standards, such as the regulation of the European General Data Protection, security-related ISO standards, and methodologies for security and privacy labeling.

3.1.3 | Security enforcement plane

The security enforcement plane manages the resource usage, network connectivity, and real-time operation of the security enablers over heterogeneous environments. It also includes the components required for their management.

The *Control and Management domain* modules supervise the usage of resources and run-time operations of security enablers deployed over software-based and IoT networks. A set of distributed SDN controllers takes charge of communicating with the SDN-based network elements to manage connectivity in the underneath virtual and physical infrastructure. Network function virtualization ETSI MANO–compliant modules support secure placement and management of virtual security functions over the virtualized infrastructure. As the envisioned framework aims to cover legacy IoT scenarios, different IoT controllers can be used to manage IoT devices and low power and lossy networks (LoWPANs). These IoT controllers are usually deployed at the network edge (eg, gateways) to enforce security functions in heterogeneous IoT domains.

It leverage NFV and SDN by considering additional components of the IoT control component an the physical network functions component that embrace the functionality required to support management of IoT devices and covers legacy scenarios that are not NFV/SDN-enabled. Special kind of IoT controllers are used to manage devices in more heterogeneous scenarios that include IoT or LoWPANs.

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 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.

Virtualized network function domain accounts for the VNFs deployed over the virtualization infrastructure to enforce security within network services. Specific mechanisms will be developed to verify the trustworthiness of VNFs and to continuously monitor their key parameters. Specific attentions will be addressed to the provisioning of advanced security VNFs (such as virtual firewall, IDS/IPS, and channel protection), capable to provide the defense mechanisms and threat countermeasures requested by security policies.

Internet of Things domain comprises the IoT devices to be controlled. This includes the security enablers, actuators, or software agents needed to enforce the security directives coming from the orchestration plane and managed, at the enforcement plane, by the IoT controller. For instance, a special kind of local security agent can be deployed in IoT devices to protect the communications between two devices. In this regard, the Constrained Application Protocol (CoAP)–Extensible Authentication Protocol (EAP)[25] protocol can be used as lightweight authentication service that uses EAP transported by means of CoAP messages, with two purposes: authenticate two CoAP endpoints and derive cryptographic material to protect the exchanges between them to bootstrap security associations at different levels of the protocol stack. In this way, if a Datagram Transport Layer Security (DTLS) channel has to be established, a premaster secret can be derived from the Master Secret Key that results from the EAP authentication.

3.2 | Policy refinement and enforcement workflows

In this section, we provide a more detailed overview of the policy refinement and enforcement processes. The first policy refinement process is depicted in the sequence diagram of Figure 2. As it can be seen, the Policy Interpreter module firstly receives as input the policies specified in High-level Security Policy Language (HSPL). High-level Security Policy Language remains at a high abstraction level, specially though for nontechnical users, or coarse-grained operations. It can be generated by the user through the policy editor. Once the Interpreter has received the HSPL policies, it identifies the capabilities needed to enforce them. To this aim, the Interpreter interacts with the *Security Enablers Manager* to identify the available SDN/NFV-based enablers, performing a matching among the discovered capabilities required by the HSPL policies and the one supported by each available enabler (capability matching). Currently, the security policy models are evolving, making the framework supports capabilities like authentication, authorization, network filtering/forwarding, and channel protection (OpenVPN, IpSec, and DTLs). Once the Interpreter obtains the list of security enablers capable to deal with the discovered capabilities, it verifies if the received list contains enough security enablers in order to cover all capabilities. If it is not the case, the Interpreter returns a non-enforzable analysis notification.

On the other hand, if the refinement process is possible, the Policy Interpreter generates a set of policies defined in a Medium-level Security Language (MSPL) from HSPL policies. Medium-level Security Language policies provide enough information in order to apply the security policies in independent way of the underlaying technology. An example of the correspondence among HSPL and the generated MSPL policy can be observed in listings 1 and 2. Listing 1 shows an example of HSPL policy definition, which aims to deny access to the internet to a certain IoT sensor. Moreover, Listing 2 shows an example of policy refinement for the previous HSPL policy. In this case, we can see the Interpreter has generated some specific infrastructure information from a most general one (eg, get the IP address from the sensor name). This is happening because the policy refinement process requires the resolution of different high-level terms by using different contextual information. The proposed policy refinement process needs to take into account some extra parameters that fill the lack of information of human concepts defined at HSPL level. To this aim, the set of extra parameters is defined in what is called context information that, together with the HSPL policy, forms the input to the translation process that will generate the medium-level security policy.

The context information maintained and managed by the *Control and Management Domain* layer of the framework provides business-dependent data needed to interpret the concepts expressed in the HSPL. The context information encompasses the environmental data retrieved from the enforcement plane, the monitoring information, along with the real-time instantiated system model, that can be defined in a common language such as Common Information Model (CIM)[26] from DMTF. Common Information Model provides a modeling mechanism to model different types of information regardless the implementation or repository, indeed, some operating systems and platforms already support retrieving the current and instantiated status of the system in CIM model, providing detailed information about the system being managed. This description can be used to retrieve information about which capabilities are provided by different system components, as well as particular network configurations, in order to perform the policy refinement from HSPL to MSPL. The DTMF also provides other standard models to represent specific components of the underlying virtual environment, such as OVF, VMAN, CIMI, and hardware infrastructure (eg, SMASH and Redfish). For example, a filtering policy should be applied by a firewall element that has network traffic–filtering capabilities, and the needed extra information such as network IP addresses associated to a user or device identifier can be resolved and obtained from the instantiated system model.

Listing 1 High-Level Policy example

```
    <tns:hspl subject="SensorA" id="HSPL0" >
        <tns:action>no_authorise_access</tns:action>
        <tns:objectH>Internet_traffic</tns:objectH>
    </tns:hspl>
```

Listing 2 Medium-Level Policy example

```
 <configurationRule>
    <configurationRuleAction xsi:type="FilteringAction">
        <FilteringActionType>DENY</FilteringActionType>
    </configurationRuleAction>

    <configurationCondition xsi:type="FilteringConfigurationCondition">
        <packetFilterCondition>
            <SourceAddress>192.168.1.12</SourceAddress>
        </packetFilterCondition>
    </configurationCondition>
</configurationRule>
```

The second policy refinement process is related the translation from MSPL policies to specific enablers configurations or tasks, as sketched in Figure 3. In this case, after receiving the MSPL policies to apply, the Security Orchestrator selects the enablers to be effectively deployed and configured, according to the enablers' requirements, the available resources in the underlying infrastructure, and optimization criteria. Then the Security Orchestrator requests to the Policy Interpreter for a refinement process of the MSPL policies into enabler configurations. In this sense, the Policy Interpreter invokes specific plug-ins, offered by the security enabler provider, to translate MSPL policies to specific low-level configuration/tasks for the specific enabler. When all configurations/tasks capable to enforce the policy are obtained, they are populated and returned to the Security Orchestrator.

Listing 3 Low-Level Conf Example-IpTables conf

```
*filter
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
-A FORWARD -p TCP -s 192.168.1.12 -j DROP
-A FORWARD -p UDP -s 192.168.1.12 -j DROP
COMMIT
```

Listing 4 Low-Level Conf Example-SDN conf

```
<flow xmlns="urn:opendaylight:flow:inventory">
<flow-name>Dropper</flow-name>
<match>
<ethernet-match>
<type>0$\times$0800</type>
</ethernet-match>
<ipv4-source> 192.168.1.12 </ipv4-source>
</match>
<id>Dropper</id>
<table_id>0</table_id>
<apply-actions>
<action>
<order>0</order>
<drop-action/>
</action>
</apply-actions>
</flow>
```

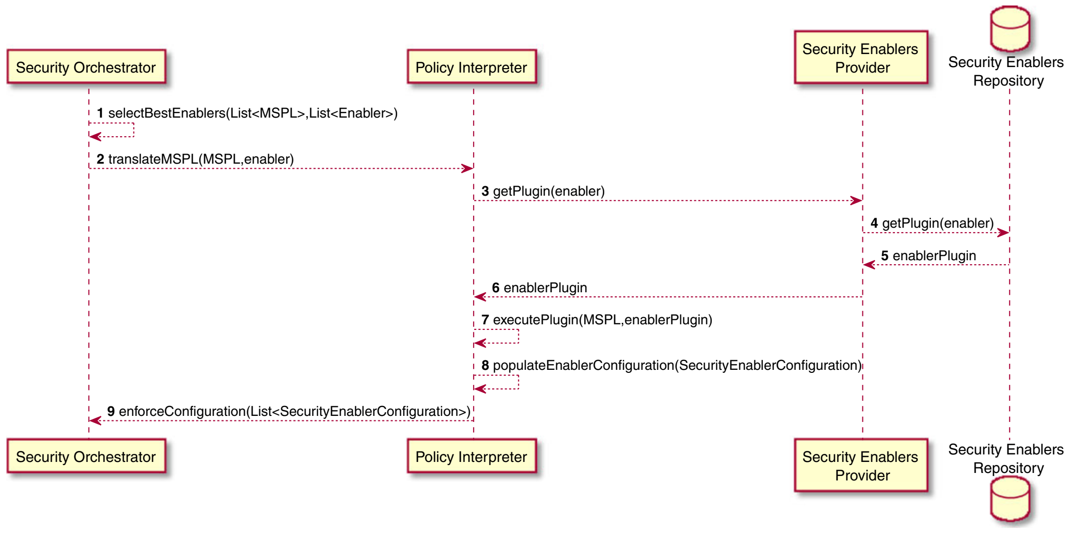
Security Orchestrator
Policy Interpreter
Security Enablers Provider
Security Enablers Repository
1 selectBestEnablers(List<MSPL>,List<Enabler>)
2 translateMSPL(MSPL,enabler)
3 getPlugin(enabler)
4 getPlugin(enabler)
5 enablerPlugin
6 enablerPlugin
7 executePlugin(MSPL,enablerPlugin)
8 populateEnablerConfiguration(SecurityEnablerConfiguration)
9 enforceConfiguration(List<SecurityEnablerConfiguration>)

FIGURE 3 Medium Security Policy language to low-level configurations

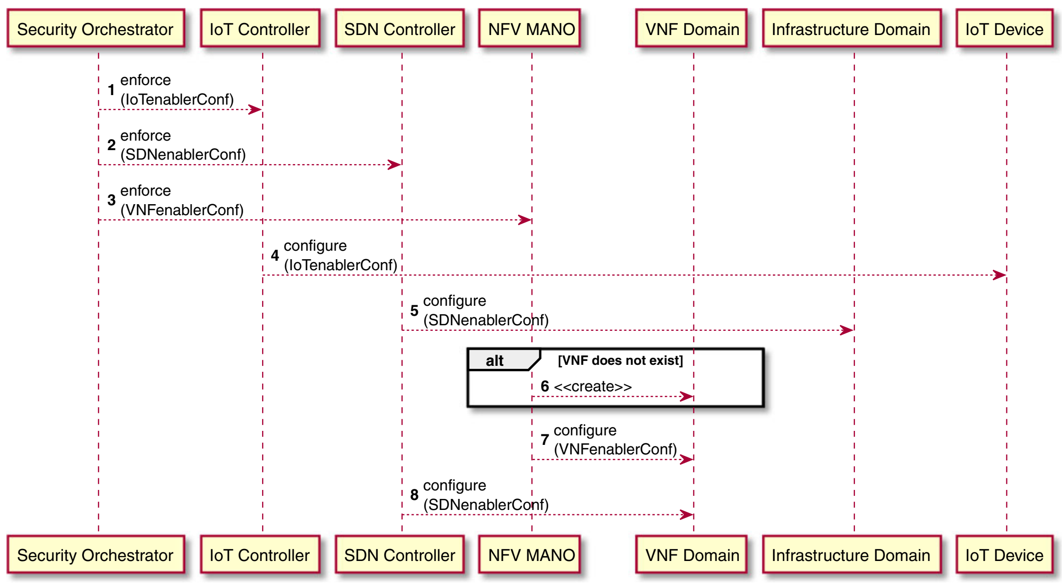
Security Orchestrator
IoT Controller
SDN Controller
NFV MANO
VNF Domain
Infrastructure Domain
IoT Device
1 enforce (IoTenablerConf)
2 enforce (SDNenablerConf)
3 enforce (VNFenablerConf)
4 configure (IoTenablerConf)
5 configure (SDNenablerConf)
alt [VNF does not exist]
6 <<create>>
7 configure (VNFenablerConf)
8 configure (SDNenablerConf)

FIGURE 4 Security enablers orchestration

Listings 3 and 4 provide examples of the low-level configuration obtained by means of the IPTABLES plug-in and SDN plug-in for OpenDayLight Controller, respectively. Both plug-ins take as input the MSPL policy defined in Listing 2, generating different outputs depending on the specific technology.

Regarding the enforcement phase, currently, ANASTACIA encompasses different security enablers that can be enforced in the system either through the SDN controller, NFV MANO, or the IoT controller. As virtual security appliances, our proposed framework considers different VNFs deployed through the NFV MANO. Namely, virtual firewall, virtual IDS, virtual IPS, virtual switch/router, virtual honeypot/honeynet, virtual secure web proxy, virtual VPN, virtual bandwidth control. Additionally, the SDN controller is in charge of dealing with basic security mechanisms for SDN tasks, such as traffic flow forwarding, traffic flow dropping, traffic flow mirroring, traffic flow bandwidth reduction. Finally, the IoT controller offers different APIs to deal with basic security mechanisms for IoT, such as power management, interface management, and traffic protection management. The security enablers provider supports different plug-ins to generate automatically configurations for those security enablers.

When the Security Orchestrator receives the configuration for one of the aforementioned security enabler, it must start an orchestration process accounting for the whole vision of the underlaying infrastructure. The main security orchestration process is shown in Figure 4. Thus, the Security Orchestrator is in charge of efficiently managing the enforcement

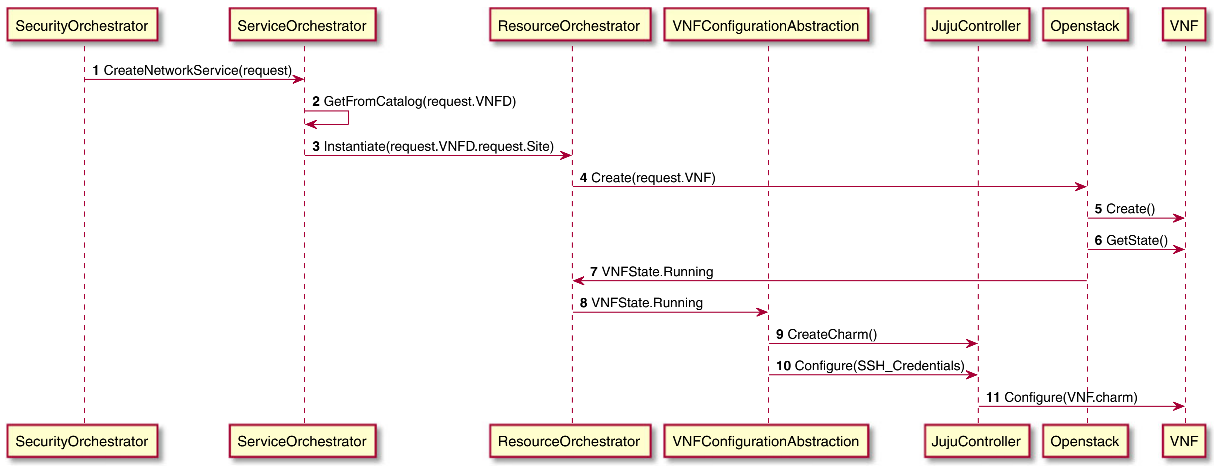

FIGURE 5 Virtualized network function (VNF) instantiation and configuration with network function virtualization management and orchestration

over different environments, such as IoT, NFV, and SDN, by interacting with relevant control and management modules. In case of SDN, the northbound APIs of the SDN controller can be exploited to require the enforcement of relevant flow rules either in the infrastructure domain or in the VNF domain. The SDN controller will interact with physical/virtual SDN switches to install the required traffic flow rules. In case that the enforcement affects the IoT domain, then the IoT controller manages directly the IoT devices by means of different interfaces accessed through CoAP over DTLs.

The process performed to instantiate and configure the virtual appliance through the NFV MANO is showed in Figure 5. The service orchestrator is responsible for end-to-end service orchestration and provisioning. The service orchestrator stores the VNF definitions and NS catalogs, manages workflow of the service deployment and can query the status of already deployed services. The resource orchestrator is used to provision services over a particular infrastructures as a service provider in a given location. It supports Openstack, OpenVIM, Vmware, and AWS. Finally, the VNF configuration and abstraction module performs the initial VNF configuration using Juju Charms (SSH access to the VNFs).

4 | USE CASES

The huge amount of IoT devices may cause network congestion, thereby making the system prone and vulnerable to DDoS attacks when it has been compromised. Besides, the centralized nature of SDN can bring and stress those kinds of availability and other routing security threats when used in IoT scenarios, as the controller might become overwhelmed and main point of failure and hacked. In this regard, distributed and redundant deployment of SDN controllers can minimize the impact of those attacks. In addition, the massive data exchange required in IoT, the authentication and management of heterogeneous devices and gateways, and the convergence of security policies across different domains are some others inconveniences that aggravates the security problems in IoT.

In this regard, Gao et al[27] provides an analysis of security threats and vulnerability for cyber-physical systems. They identified different attacks at the three main layers, ie, physical , network, and application layer. Our framework aims at mitigating mainly networks security threats in IoT. The following attacks and threats can be identified[27] at that layer: DDoS, routing attack, sink node attack, direction misleading attack, black hole attack, flooding attack, trapdoor, Sybil attack, sinkhole attack, wormhole attack, routing loop attack, HELLO flooding attack, spoofing attack, selective forwarding, tunnel attack, and false routing information.

To cope with those attacks, the main countermeasures at network layer focus nowadays on ensuring confidentiality, integrity, and availability. To this aim, novel end-to-end encryption mechanism specially devised for IoT at different levels (eg, 6LowPANs encryption, IPsec tunnels, and DTLs), including peer-to-peer authentication and key negotiation management, can be orchestrated and configured on demand as VNFs through our security framework.

Regarding availability and DDoS attacks mitigation, vIDS deployed dynamically as VNFs by our framework can be used to detect infected bots and then dynamically drop the malicious traffic through new SDN flow rules and vFirewalls, enforced directly in the IoT network by the controller. At the same time, as part of the reaction plan, different operational policies, directly executed over the IoT devices through the IoT controller, such as switch off the compromised IoT devices, can be enforced to mitigate the attacks.

The following subsections illustrate two use cases to describe the potential benefits of the proposed framework in realistic scenarios.

4.1 | Exploiting multiaccess edge computing ecosystem against IoT-based attacks

A broad range of IoT-based applications, such as autonomous cars, industrial automation systems, and Tactile Internet, present demanding requirements in terms of tolerable latency and traffic generation. To face these challenges, the multiaccess edge computing paradigm is gaining high momentum, boosting increased processing and storage capabilities towards the network edge.[28]

Edge environments can also represent a strategic position in the network infrastructure to enforce security features. Indeed, accounting for the increased number of attacks related to IoT devices, the IoT system administrators are interested to gain a higher level of protection. To guarantee the required security features, the Telco operator will adopt the proposed framework within its system, by appropriately integrating it with the existing network and service mechanisms, such as SDN, NFV, and cloud edge computing technologies. In this way, the Telco provider will be able to offer advanced Security-as-a-Service solutions, exploiting its capillary and flexible cloud-based network infrastructure. To meet the security requirements of the IoT systems, virtualized security network functions, such as Intrusion Detection Systems, can be deployed on-demand over edge nodes. These virtual network probes can monitor the traffic generated by the IoT devices with increased scalability and send valuable information to the Monitoring module, which triggers security alerts in case of potential threats. Then, appropriate security countermeasures, such as the isolation of the compromised IoT devices, can be applied by the Security Orchestrator exploiting SDN capabilities to dynamically reconfigure the devices' connectivity. By interacting with the SDN controller, secure network zones are created enforcing proper rules in the physical/virtual SDN switches deployed at the network edge.

This exemplary use case aims at highlighting how the joint management of NFV and SDN approaches can bring remarkable benefits to provide on-demand security features in software-based networks. The increased capabilities of Edge infrastructure can even augment the efficiency of the envisioned security solutions, by enabling prompt reactions near the IoT devices.

4.2 | Building management system use case

In smart buildings, all the electrical and mechanical devices are controlled and monitored by a centralized building management system (BMS). As part of the supported services, the building usually is equipped with a heating, ventilation, and air conditioning system. The building allows to control remotely sensors, controllers, actuators, and equipment via internet connectivity. In this way, multiple services, such as remote monitoring, reporting, billing, predictive maintenance, and remote control, can be enabled by leveraging networked cyber-physical solutions.

However, the BMS system can be subject to many attacks of cybercriminals, breaching commercial buildings such as data centers and supermarkets. The proposed security framework provides new methodologies and tools to increase the resilience of BMS upon cyber attacks. Various scenarios of cyber attacks on the network of embedded systems, software systems, and internet-connected devices that are part of the diverse building operations can be envisaged.

For instance, in the scope of the heating, ventilation, and air conditioning, our framework will effectively deal with man-in-the-middle attacks, where the attacker manipulates some sensors introducing corrupted temperature values. This kind of attack in BMS targets might produce long-term financial impact, because of the imagery loss in reaching the set point. Our proposed framework can detect uncommon temperatures, and the system can react and enforce security policy to isolate the compromised sensor from the rest of the BMS system, for a time period until further investigation takes place. In addition, as a result of that attack detection, the framework can react, improving the security between certain IoT devices or within devices in some networks, enforcing a security policy for isolate the sensor or enforce a network channel protection. For instance, DTLS could be dynamically established to protect the communication among the devices.

5 | PERFORMANCE EVALUATION

5.1 | PoC implementation and testbed description

To validate the feasibility of our proposal, we have implemented and deployed a PoC of the architecture defined in Section 3. This PoC allows performing tests to evaluate the performance of the solution. In our analysis, we have focused on

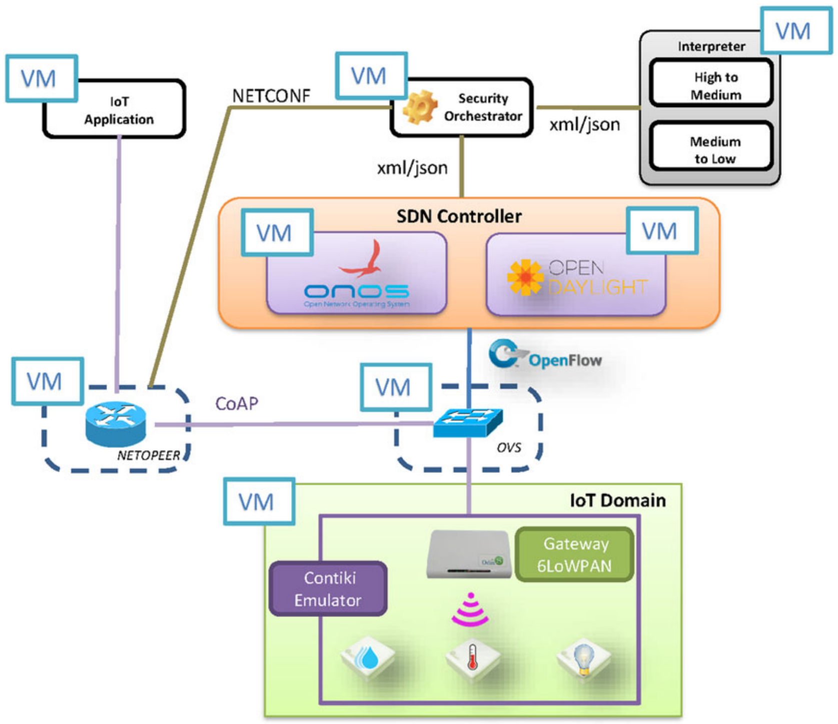

FIGURE 6 Virtual testbed deployment

the policy-based approach for provisioning SDN-based security mechanisms. Since the enforcement could be done using different technologies, we have also compared the SDN approach against a more traditional approach, using a virtual firewall VNF appliance configured through NETCONF and iptables. By means of the aforementioned technologies, the testbed starts from a detected man-in-the-middle attack as described in the BMS in Section 4.2 and deals with it mitigating the attack by isolating the compromised sensors.

As it can be seen in Figure 6, the deployment encompasses several interconnected virtual machines covering the functionality for policy refinement and enforcement in the scope of IoT, NFV, and SDN integration. Specifically, it has been performed using virtual machines as well as LXC and Docker containers, thereby promoting the reusability of the following components:

- The *Security Orchestrator* has been developed in Python, and it is in charge of driving the policy refinement through the Policy Interpreter, indicating the enablers to use and then requesting the enforcement of the configurations obtained in the MSPL to low-level configurations refinement process.
- The *Policy Interpreter* also has been developed in Python, extending the policy models defined on the European project SECURED.[29] It performs two different levels of policy refinement, from HSPL to MSPL and from MSPL to specific enablers configurations or tasks. For this PoC deployment, we have also developed three different plug-ins. Two of them are responsible of the translation from MSPL filtering policy to SDN controllers northbound API configuration (both ONOS and OpenDaylight), and the third one is used to get the IPTables configuration from an MSPL filtering policies. The proper plug-in is chosen by the Security Orchestrator on runtime.
- As *SDN controller*, it has been deployed two different virtual machines with ONOS and OpenDaylight. Both SDN controllers receive from the Orchestrator the configuration rules via northbound API using their corresponding JSON models.
- For the *enforcement* of the filtering policy through SDN, we are using a virtual machine running Open Virtual Switch (OVS), which has been configured connecting with the SDN controller using OpenFlow as SDN protocol.
- For the enforcement through a more traditional approach, the testbed includes a virtual machine as *virtual router* running a NETOPEER NETCONF server. Besides, the Orchestrator has been endowed with a NETCONF client that introduces the iptables configuration in a YANG model and performs the request to the server. In addition, the NETCONF server has been equipped with a function that receives the IpTables configuration and applies it over the virtual router.

- Regarding the *IoT domain*, a virtual machine which contains a Cooja Contiki emulator has been deployed, running several IoT sensors and a border router. The IoT sensors are connected through 6LoWPAN protocol with the border router, which is connected to the OVS, in the same way as the NETCONF server.
- Finally, The IoT application is deployed in a different IPv6 network and can reach the IoT domain through the virtual router. In the testbed scenario, the IoT application is a virtual machine that performs recurrent requests to the IoT sensors.

Once deployed the architecture, the main use case that has been addressed consists on the enforcement isolation policies over sensor. Thus, when a security alert about an IoT sensor raises (eg, the sensor is exceeding the data rate), the reaction countermeasure enforces a filtering policy with the aim of isolating the compromised sensor. Namely, the HSPL filtering policy, enforced by the Orchestrator in real time, indicates to drop all packets coming from the specific sensor. The Orchestrator triggers the refinement process through the Interpreter, generating a refined MSPL policy. Then the last translation step from MSPL to lower configuration is requested, indicating one of the three kinds of enforcement plug-ins, either through ONOS, ODL, or directly with iptables and NETCONF. Thus, depending on the selected enabler, the Orchestrator will request the corresponding enforcement to the specific point (SDN controllers or the virtual router). The testbed ends up with either an installation of a new IPTABLES filtering rule using NETCONF or an installation of a new SDN filtering flow with OpenFlow, thereby blocking the communication between the specified sensor and the IoT application.

5.2 | Performance results

Since it is important to react as soon as possible when we are facing an attack, the proposal has been validated by analyzing the times required for each processes involved in the attack mitigation once the attack has been detected. These are (1) the time taken since the Policy Interpreter receives an HSPL to MSPL refinement requests; until the refinement process has finished providing the MSPL. (2) The time taken since the Policy Interpreter receives the MSPL policy translation requests; until the translation process has finished providing the security enabler configuration/task. (3) Finally, the time taken since the Orchestrator requests the enforcement of the configuration to the Enabler, until the device applies it. While the first two points are measured by taking the times the APIs take to perform their processes, the last point is different depending on the technology used. In the NETCONF case, the virtual router responses to the enforcement request once the policies have been enforced. On the other hand, in the SDN cases, the SDN controller responses when it receives the flows modifications request, so in this case, we decided to record the flow modification times directly in the virtual switch in order to take the enforcement time as the difference among the enforcement request and the virtual switch flow modification time stamp. It is also important to highlight the fact that we have configured both the northbound API and the southbound API of the SDN approaches to use SSL in order to set up a fair comparison with Netconf that also uses SSH.

Regarding the methodology and infrastructure, the tests have been performed enforcing progressively different amount of policies multiple times for each technology. Specifically, the testbed performs 100 iterations of policy refinement, translation, and enforcement from 1 to 1000 (increments of 200) filtering policies in pro of analyzing the time taken by the main identified processes and also for verifying the scalability of the solution. The aforementioned tests have been performed using separated virtual machines with Ubuntu 16.04.2 as operating system, Open vSwitch 2.5.2 version as virtual switch, NETOPEER 0.8.0 version as NETCONF server in the virtual router, ONOS 1.9.0 (Junco) version, and Openday-Light (Carbon) version as SDN controllers. Regarding the IoT domain, the official instant Contiki 3.0 virtual machine has been used for the emulated sensors. All virtual machines have been configured with one CPU, excepting the Interpreter virtual machine that was configured with two. The virtual machines executing OVS, the virtual router and monitoring have been configured with 256 MB of RAM, while the SDN controller virtual machines have been configured with 1 GB. The Contiki virtual machine was executed with 1.5 GB of RAM and the Orchestrator and the Interpreter with 1.5 GB RAM. The whole testbed environment has been deployed on top of a workstation equipped with Intel Core i7-2600 and 8 GB of RAM.

The main performance results are shown in Figures 7, 8, and 9. They show the average time for the HSPL to MSPL policy refinement process, the translation from the MSPL to the Enabler configuration process, and finally the policy enforcement for each different considered technology. As it can be seen, since the HSPL to MSPL refinement process is independent of the underlaying technology, the measures are similar in the different cases, getting the policy refinement for the most expensive case (1000 policies) in about 10 seconds. On the other hand, since the MSPL to Enabler configuration translation depends on the complexity of the translation for the specific Enabler technology, the ONOS (Figure 8)

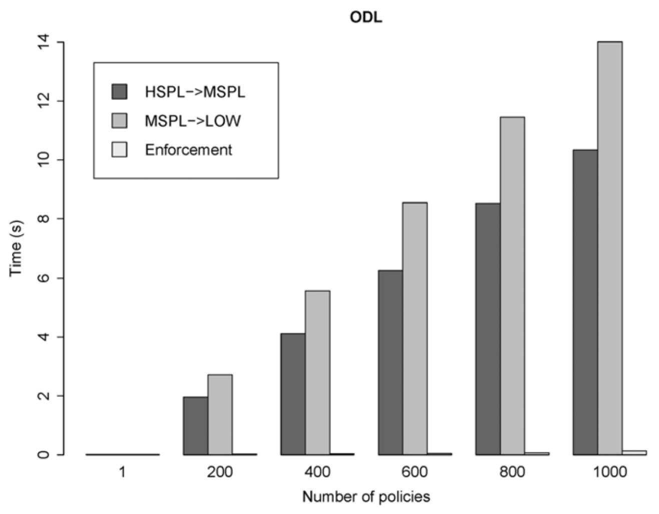

FIGURE 7 Testbed results. Enforcement with ODL

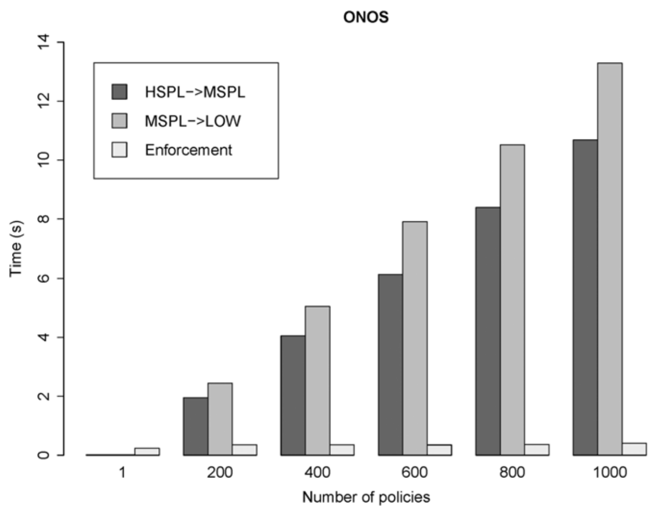

FIGURE 8 Testbed results. Enforcement with ONOS

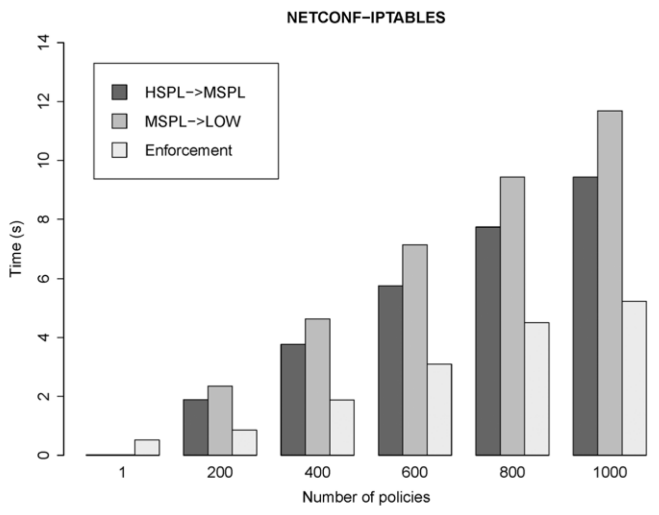

FIGURE 9 Testbed results. Enforcement with classical IpTables

TABLE 1 One thousand filtering policies deployment times by technology

Technology	Refinement	Translation	Enforcement	Total(s)
ODL	10.33	14.01	0.12	24.47
ONOS	10.68	13.28	0.40	24.38
NETCONF	10.43	11.68	5.23	26.35

and ODL (Figure 7) cases are quite similar (both use JSON with similar structures), getting results near to 14 seconds. The iptables translation process (Figure 9) is faster because iptables sentences are lighter than the JSON generation for the northbound APIs of the SDN controllers, getting results near to 12 seconds. As it can be observed, the translation step represents the most heaviest part of the policy deployment workflow. Finally, regarding the enforcement time in the SDN case, both ODL and ONOS are getting results below a second, offering ODL a slightly better handling of the flow management than ONOS. In the NETCONF case, the most expensive use case obtains results near to 5 seconds, which represents a considerable difference with the previous ones. At this point, it is important to highlight the NETCONF option uses a SSH channel for each request, whereas in the SDN case, once the TLS channel is established in the southbound, it is maintained.

Table 1 provides the average results for each process involved on the policy enforcement and for each underlaying selected technology. As it can be observed, regardless of the chosen technology, the testbed allows to deploy and enforce a big amount of filtering security policies in less than 30 seconds. When it comes to the enforcement, the results are also providing a clear difference among the SDN approaches and the traditional approach, offering the first one a significant improvement. However, as a global result, the time impact in the last case is absorbed by the translation phase, which provides better results.

The framework results can be considered promising, as the solution provides high level of abstraction and governance over the full infrastructure through the policy enforcement. It can be considered important desirable properties in order to face challenges like the vast heterogeneity of the IoT domain.

However, the proposal has also different challenges to face due to its complexity. Firstly, from the user perspective, the administrators should be aware that in return for the benefits provided by the framework, they should now pay attention to the deployment and maintenance of new modules that did not previously exist in their architecture. Secondly, while high-level policies are designed to model aspects of coarse grain security, avoiding high technical knowledge, mid-level policies requires a good knowledge of the models to take advantage of their features. Thirdly, from the architectural point of view, one of the challenges and our ongoing work is the management of the IoT heterogeneity across different domains. The framework allows managing inter IoT domain scenarios, with IoT devices located in remote places. Nonetheless, special adapters and monitoring agents will need to be implemented and deployed to in particular IoT scenarios to populate a common system and context data models in order to enable the policy-based network security management described in this paper.

6 | CONCLUSIONS

The increased adoption of IoT solutions raise notable security threats, which can bring dramatic consequences over manifold domains. Accounting for the challenges in terms of scalability and heterogeneity, new security mechanisms are required to efficiently cope with IoT security vectors. We have presented a novel security framework that aims at providing automated and self-configurable SDN/NFV-based security mechanisms in IoT scenarios able to complement conventional approaches. In particular, the security orchestration plane has been devised to include different abstractions in the security policies management, thereby improving notably the flexibility in the configuration of the selected security enablers. We have developed a PoC of the envisioned architecture to demonstrate its feasibility. The performance assessment has highlighted the benefits of enforcing SDN-based security countermeasures with respect to conventional approaches.

In our future research activities, we will extend the evaluation considering additional VNFs and IoT security mechanisms, so to assess an integrated policy-based orchestration solution. We will also analyze the impact in the selection of the enablers according for the status of the underlying infrastructure and application criteria. Finally, accounting for the potential vulnerabilities of SDN and NFV environments,[30,31] we also aim at enhancing the proposed framework with mechanism to guarantee the inherent security of the envisioned SDN and NFV security enablers.

ACKNOWLEDGEMENTS

This work was partially supported by the ANASTACIA project that has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement 731558. The research has been also supported by a postdoctoral INCIBE grant within the "Ayudas para la Excelencia de los Equipos de Investigacion Avanzada en Ciberseguridad" program, with code INCIBEI-2015-27363.

CONFLICT OF INTEREST

The authors declare no potential conflict of interests.

FINANCIAL DISCLOSURE

None reported.

ORCID

Jorge Bernal Bernabe http://orcid.org/0000-0002-7538-4788

REFERENCES

1. Atzori L, Iera A, Morabito G. Understanding the internet of things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks*. 2017;56:122-140.
2. Sicari S, Rizzardi A, Grieco LA, Coen-Porisini A. Security, privacy and trust in internet of things: the road ahead. *Comput Networks*. 2015;76:146-164.
3. Taleb T. Toward carrier cloud: Potential, challenges, and solutions. *IEEE Wirel Commun*. 2014;21(3):80-91.
4. Farris I, Bernabe JB, Toumi N, et al. Towards provisioning of SDN/NFV-based security enablers for integrated protection of IoT systems. In: 2017 IEEE Conference on Standards for Communications and Networking (CSCN); 2017; Helsinki, Finland. 169-174.
5. Luo T, Tan H-P, Quek TQS. Sensor OpenFlow: enabling software-defined wireless sensor networks. *IEEE Commun Lett*. 2012;16(11):1896-1899.
6. Galluccio L, Milardo S, Morabito G, Palazzo S. SDN-wise: design, prototyping and experimentation of a stateful SDN solution for wireless sensor networks. In: 2015 IEEE Conference on Computer Communications (INFOCOM) IEEE; 2015; Kowloon, Hong Kong. 513-521.
7. De Oliveira BT, Gabriel LB, Margi CB. TinySDN: enabling multiple controllers for software-defined wireless sensor networks. *IEEE Lat Am Trans*. 2015;13(11):3690-3696.
8. Ali ST, Sivaraman V, Radford A, Jha S. A survey of securing networks using software defined networking. *IEEE Trans Reliab*. 2015;64(3):1086-1097.
9. Sherwood R, Gibb G, Yap K-K, et al. Flowvisor: A network virtualization layer. OpenFlow Switch Consortium, Tech. Rep; 2009. 1–13.
10. Shin SW, Porras P, Yegneswaran V, Fong M, Gu G, Tyson M. Fresco: Modular composable security services for software-defined networks. In: 20th Annual Network & Distributed System Security Symposium NDSS; 2013; San Diego, CA United States.
11. Shin S, Xu L, Hong S, Gu G. Enhancing network security through software defined networking (SDN). In: 2016 25th International Conference on Computer Communication and Networks (ICCCN); 2016; Waikoloa, HI, USA. 1-9.
12. Yoon C, Park T, Lee S, Kang H, Shin S, Zhang Z. Enabling security functions with SDN: a feasibility study. *Comput Networks*. 2015;85:19-35.
13. Yan Q, Yu FR, Gong Q, Li J. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: a survey, some research issues, and challenges. *IEEE Commun Surv Tutorials*. 2016;18(1):602-622.
14. Xu T, Gao D, Dong P, Zhang H, Foh CH, Chao HC. Defending against new-flow attack in SDN-based Internet of Things. *IEEE Access*. 2017;5:3431-3443.
15. Suh J, Choi HG, Yoon W, You T, Kwon T, Choi Y. Implementation of content-oriented networking architecture (CONA): a focus on DDoS countermeasure. In: Proceedings of European NetF-PGA Developers Workshop; 2010.
16. Chakrabarty S, Engels DW, Thathapudi S. Black SDN for the Internet of Things. In: 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems; 2015; Dallas, TX, USA. 190-198.
17. Bull P, Austin R, Popov E, Sharma M, Watson R. Flow based security for IoT devices using an SDN gateway. In: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (ficloud); 2016; Vienna, Austria. 157-163.
18. Flauzac O, González C, Hachani A, Nolot F. SDN based architecture for IoT and improvement of the security. In: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops; 2015; Gwangiu, South Korea. 688-693.
19. Choi S, Kwak J. Enhanced SDIoT security framework models. *Int J Distrib Sens Netw*. 2016;12(5). https://doi.org/10.1155/2016/4807804
20. ETSI ISG NFV. Etsi gs nfv-sec 003 nfv; architectural framework v1.2.1; 2014.
21. Furfaro A, Garro A, Tundis A. Towards security as a service (SecaaS): on the modeling of security services for cloud computing. In: 2014 International Carnahan Conference on Security Technology (ICCST) IEEE; 2014; Rome, Italy. 1-6.

22. Yu T, Sekar V, Seshan S, Agarwal Y, Xu C. 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 ACM; 2015; Philadelphia, PA. 5.
23. Hafeez I, Ding AY, Suomalainen L, Kirichenko A, Tarkoma S. Securebox: Toward safer and smarter IoT networks. In: Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking ACM; 2016; Irvine, California, USA:55-60.
24. Basile C, Lioy A, Pitscheider C, Valenza F, Vallini M. A novel approach for integrating security policy enforcement with dynamic network virtualization. In: 2015 1st IEEE Conference on Network Softwarization (NetSoft); 2015; London, UK. 1-5.
25. Garcia-Carrillo D, Marin-Lopez R. Lightweight coAP-based bootstrapping service for the Internet of Things. *Sensors*. 2016;16(3):358.
26. Common Information Model (CIM), DMTF. http://www.dmtf.org/standards/cim
27. Gao Y, Peng Y, Xie F, et al. Analysis of security threats and vulnerability for cyber-physical systems. In: Proceedings of 2013 3rd International Conference on Computer Science and Network Technology; 2013; Dalian, China. 50-55.
28. Farris I, Taleb T, Flinck H, Iera A. Providing ultra-short latency to user-centric 5G applications at the mobile network edge. *Trans Emerg Telecommun Technol*. 2017;29.
29. SECURity at the network EDge. https://www.secured-fp7.eu/
30. Li W, Meng W, Kwok LF. A survey on openflow-based software defined networks: security challenges and countermeasures. *J Network Comput Appl*. 2016;68:126-139.
31. Lal S, Taleb T, Dutta A. NFV: security threats and best practices. *IEEE Commun Mag*. 2017;PP(99):2-8.

Alejandro Molina Zarca is currently a PhD student and researcher in the Department of Information and Communication Engineering University of Murcia (Spain), involved in the H2020 EU project Anastacia. He received his MSc and Master degree in Computer Science from the University of Murcia in 2012 and 2017, respectively. His research interests include Internet of Things, Cybersecurity, and network virtualization and softwarization.

Jorge Bernal Bernabe received the MSc, Master, and PhD in Computer Science from the University of Murcia. Currently, he is a postdoctoral cybersecurity researcher in the University of Murcia. Author of several book chapters and more than 35 papers in international conferences and journals. He has been involved in the scientific committee of numerous conferences and served in the editorial board and reviewer for several journals. During the last years, he has been working in several European research projects FP7, H2020 , such as DESEREC, Semiramis, Inter-Trust, SocIoTal, ARIES or ANASTACIA. His scientific activity is mainly devoted to the security, trust, and privacy management in distributed systems and IoT networks

Ivan Farris is currently a researcher at Aalto University, Finland, in the Department of Communications and Networking, School of Electrical Engineering. He received his MSc degree in Computer and Telecommunications Systems Engineering and his PhD in Information Engineering from the University of Reggio Calabria, Italy, in 2013 and 2017, respectively. His research interests include Internet of Things, Edge computing, and network softwarization.

Yacine Khettab is currently a research assistant at Aalto University, Department of Networking and Electrical Engineering. He received his Master's degree at the University of Science and Technology Houari Boumedinne, Algeria, in Networking and Distributed Systems. His current field of research focuses on providing Security-as-a-Service using SDN and NFV.

Tarik Taleb is currently a professor at Aalto University, Finland, leading the MOSA!C Lab. He worked as a senior researcher at NEC Europe Ltd until April 2015. Prior to that, he worked as an assistant professor at Tohoku University, Japan. He received his BE degree in Information Engineering with distinction, and his MSc and PhD degrees in Information Sciences from Tohoku University in 2001, 2003, and 2005, respectively. His research interests lie in the field of mobile core, mobile cloud networking, network function virtualization, software-defined networking, mobile multimedia streaming, and social media networking.

Antonio Skarmeta received the MS degree in Computer Science from the University of Granada and BS (Hons) and the PhD degrees in Computer Science from the University of Murcia Spain. Since 2009, he is a Full Professor at the same department and university. Antonio F. Skarmeta has worked on different research projects in the national and international area in the networking, security, and IoT area, like Euro6IX, ENABLE, DAIDALOS, SWIFT, SEMIRAMIS, SMARTIE, SOCIOTAL and IoT6. His main interested is in the integration of security services, identity, IoT, and Smart Cities. He has been head of the research group ANTS since its creation on 1995. He has published over 200 international papers and being member of several program committees.

How to cite this article: Molina Zarca A, Bernal Bernabe J, Farris I, Khettab Y, Taleb T, Skarmeta A. Enhancing IoT security through network softwarization and virtual security appliances. *Int J Network Mgmt*. 2018;28:e2038. https://doi.org/10.1002/nem.2038