Received June 26, 2020, accepted July 12, 2020, date of publication July 17, 2020, date of current version July 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3010209

Security Orchestration and Enforcement in NFV/SDN-Aware UAV Deployments

ANA HERMOSILLA, ALEJANDRO MOLINA ZARCA, JORGE BERNAL BERNABE, JORDI ORTIZ, AND ANTONIO SKARMETA, (Member, IEEE)
Department of Information and Communications Engineering, University of Murcia, 30100 Murcia, Spain

Corresponding author: Jorge Bernal Bernabe (jorgebernal@um.es)

This work was supported in part by the European Research Projects H2020 INSPIRE-5Gplus GA 871808 and H2020 CyberSec4Europe project GA 830929; in part by the AXA Postdoctoral Scholarship awarded by the AXA Research Fund (Cyber-SecIoT project); in part by the Spanish Government through the FPI Programme under Grant PRE2018-083731; and by the PERSEIDES project under Grant TIN2017-86885-R.

ABSTRACT Software Defined Network (SDN) and Network Function Virtualization (NFV) are bringing many advantages to optimize and automatize security management at the network edge, enabling the deployment of virtual network security functions (VSFs) in MEC nodes, to strengthen the end-to-end security in IoT environments. The benefits could exploit in mobile MEC nodes on-boarded in Unmanned Aerial Vehicles (UAV), as the UAVs would carry on-demand VSFs to particular physical locations. To that aim, this paper proposes a novel NFV/SDN-based zero-touch security management framework for automatic orchestration, configuration and deployment of lightweight VSF in MEC-UAVs, that considers diverse contextual factors, related to both physical and virtual conditions, to optimize the security orchestration. Our solution aims to deploy on-demand VSFs, such as virtual Firewalls (vFirewalls), vProxies, vIDS (Intrusion Detection Systems) and vAAA, to assist during emerging situations in particular physical locations, protecting and optimizing the managed IoT network, as well as replacing or supporting compromised physical devices like IoT gateways. The proposed solution has been implemented, deployed and evaluated in a real testbed with real drones, showing its feasibility and performance.

INDEX TERMS IoT, cybersecurity, SDN/NFV, architecture.

ACRONYMS

AP	Access Point.
CAPEX	Capital Expenditure.
CoAP	Constrained Application Protocol.
GCS	Ground Control Station.
HSPL	High-level Security Policy Language.
IOS	Intelligent On-board System.
IoT	Internet of Things.
LCM	Life-Cycle Manager.
MANO	Management and Orchestration.
MEC	Multi-Access Edge Computing.
MSPL	Medium-level Security Policy Language.
NBI	Northbound Interface.
NFV	Network Function Virtualization.
NFVO	NFV Orchestrator.
NSD	Network Scenario Descriptor.
OPEX	Operational Expenditures.
OSM	Open Source Mano.
OVS	Open vSwitch.
RO	Resource Orchestrator.
SDN	Software Defined Network.
UAV	Unmanned Aerial Vehicle.
vAAA	virtual Authentication Authorization Accounting.
VCA	VNF Configuration and Abstraction.
vIDS	virtual Intrusion Detection System.
VIM	Virtualized Infrastructure Manager.
VM	Virtual Machine.
VNF	Virtualized Network Function.
VNFM	VNF Manager.
VSF	Virtual Security Function.

The associate editor coordinating the review of this manuscript and approving it for publication was Prabhat Kumar Upadhyay.

I. INTRODUCTION

As the Internet of Things (IoT) expands, the security issues derived by this technology are increasing accordingly [1]. To cope with these issues, Software-Defined

Networking (SDN) and Network Function Virtualization (NFV) can bring many benefits to automatize and optimize the security management and incident handling in diverse scenarios related to IoT such as smart-cities or smart-agriculture. On the one hand, NFV exploits virtualization to disconnect the network and security functions from the hardware, reducing both, capital expenditures (CAPEX) and operating expenses (OPEX). On the other hand, SDN enables the network control and management softwarization by decoupling control and data-plane, which increases network control and management flexibility. SDN and NFV are the pillars to realize a truly zero-touch security orchestration of softwarized and virtualized security appliances, implemented as Virtual Security Functions (VSF), such as vFirewalls [2], vAAA [3], vIDS, vChannelProtection or vIoTHoneynet [4].

Meanwhile, Multi-Access Edge Computing (MEC) can enlarge the cloud computation towards the edge of the network, not only to improve spectral efficiency, data locality, or reduce latency, but also to support context-specific functionality in certain deployments and locations in the IoT network. The possibilities and benefits increase exponentially in MEC nodes on-boarded in Unmanned Aerial Vehicles (UAVs) [5]. These kind of vehicles are aircrafts that can be operated either under remote control or autonomously by onboard computers. In this way, they can be delivered to specific locations on demand, thereby assisting on emerging situations, bringing virtual network functions (VNF) and VSF to specific physically compromised or demanding IoT deployments. MEC-UAVs could allow elastic provisioning of security appliances to counter cyberattacks and threats, depending on the context, offering self-protection and resiliency capabilities on the managed systems and networks. Nonetheless, despite the existence of some initial works that onboard MEC-nodes in aerial vehicles [6]–[8] while leveraging SDN/NFV to focus on network management and control aspects there is a lack of security management work in the field.

To fill this gap, this paper proposes a NFV/SDN-based security orchestration and enforcement framework of Virtual Network Functions (VNFs) at the edge of the network in MEC-powered UAVs, that allows achieving a cloud continuum and end-to-end security management between end IoT devices, the mobile edge, and the data center. Our UAVs are intended to assist during emergency situations, carrying dynamic and elastic lightweight VNFs and lightweight virtual security functions (VSFs) to replace or support potentially compromised physical network elements such as gateways, proxies, or access points.

The UAVs act as sentinels that carry security functions, including vFirewalls to counteract the malicious IoT traffic, vIDS to inspect suspicious traffic, vProxies to encapsulate the traffic end-to-end, vAccessPoints to provide wireless connectivity (e.g. LoRa, WIFI), in places where malicious or suspicious network activity has been detected. Diverse lightweight virtual network security functions are orchestrated, deployed and commissioned, dynamically, on-demand in the UAVs, to strengthen security and privacy in compromised remote locations.

The UAVs and their virtual network functions are controlled through a cognitive and policy-based security management framework that relies on diverse network controllers such as MANO, SDN Controller, and IoT controllers [9], to orchestrate the security end-to-end.

Our MEC-powered UAVs have been successfully implemented, deployed and validated in a real scenario. The solution demonstrates its feasibility and performance to deploy and configure different numbers of lightweight VNFs in the UAV as well as managing the network connectivity through SDN (re)configuration.

The main contributions of this paper are the following:

- This paper proposes and implements a MEC architecture for UAV vehicles
- It provides an SDN/NFV orchestration mechanism for the security management of IoT networks considering UAV vehicles, acccounting several contextual physical and virtual conditions and metrics for the orchestration in UAVs.
- The work analyzes the challenges and applicability of different kinds of virtual security functions in MEC UAVs and possible scenarios.
- The viability and performance of the implemented security framework to dynamically orchestrate security VNFs in MEC UAVs have been addressed and demonstrated.

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 the 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

On the last years, NFV has received significant consideration by the research community; as a result, there are many publications related to it, primarily due to being identified as a key enabler to the 5G technologies. Besides, being able to deploy virtual network functions on UAVs [10], has become increasingly popular thanks to the major advantages offered, such as portable deployments (no physical restrictions linked to the location of previously deployed centers) and the entailed flexibility and OPEX reduction. Being able to dynamically deploy components in literally any location expands the NFV scenario even further, raising dynamic deployments to a new level. Therefore, it is not surprising the attention this topic has created onto the research community.

Likewise, SDN can decouple control from UAVs and network programmability, as highlighted in [11], where authors review the applications in SDN-enabled drones equipped with base stations, and identify the associated cybersecurity aspects in these applications.

However, even though there are multiple works related to NFV, SDN, orchestration, security, and dynamic virtual deployments in mobile nodes, little attention has been given to the holistic coordinated usage of prior concepts when it comes to the security management and orchestration in UAVs. Regarding NFV deployments on aerial devices, in [6]–[8] authors propose a solution on where lightweight VNFs are deployed on UAVs.

The aforementioned works are limited since they do not perform automatically the deployment but rely on human operators performing the deployments on-demand. Moreover, those works do not consider deploying security functions since the proposed VNFs are focused on routing traffic, or just offering services to end-users.

Authors in [8] also provide with a complete analysis on how different transport-layers limit NFV Orchestration and how they can become a limiting factor. However, the analysis performed for multi-hop scenarios where UAVs need to act as relaying agents. This works assumes direct connectivity between the UAV and the infrastructure through 4G or 5G. It is true that wireless connectivity may affect communications but direct connection avoids the snowball effect pointed out in the referenced work.

We can also find more UAV NFV-based deployments in [12]. However, its purpose is to improve telemetry in UAVs, enhancing the communication with the pilots to reduce accident rates. Besides, NFVs are deployed on land, over GCSs (Ground Control Stations, that serves as a base station to manage and communicate with the UAVs), not in the UAVs themselves. There is neither an implementation of the solution; they show its feasibility, as it is able to detect failures faster than in standard communications. Finally, they do not offer any type of orchestration, since it is an operator who deploys VNFs on demand.

Likewise, in [13], authors propose NFV deployments on vehicles, but to facilitate the updating and improvement of Intelligent On-board Systems (IOS). In their proposal, they do not consider security deployments. Also in vehicle area, we can find in [14] deployment examples of network functions both in vehicles and in locations close to them. However, although these deployments are performed over constraint devices and using NFV, no orchestration or security aspects appear on it, as the objective of the deployed VNFs is to improve communications.

In [15] authors identify some challenges for the design of a system of multiple collaborative small UAVs, and propose a multi-UAV system intended to assists during disasters, search and rescue as well as aerial monitoring. They address aspects such as communications, coordination and sensing. However, that paper does not deal with automatic deployment of virtual network functions in MEC-UAVs nodes as described herein.

In [16], Bekkouche *et al.* propose an architecture for enabling UAV enhanced network Services on 5G. The archtiecture relies on NFV technologies to deploy, as ligthweight containers, the UAV services, which are intended to control and monitoring the UAVs. In their approach the UAV holds edge capabilities and network equipment with radio and backhaul connections. However, they do not consider security orchestration aspects as proposed in our work.

Regarding security orchestration deployments, authors in [2], [3], [9] propose a framework where the orchestration is performed dynamically and depending on the context. Such orchestration deploys Virtual Security Functions (VSF) on demand while enhancing security by applying SDN rules. A VSF is a specific VNF dedicated to improve security or to solve any incident related to security. Nevertheless, in that work, there is no consideration of VSF deployments inside constraint environments, being mobile or not.

The opposite situation appears in [17], where the deployment of security functions is performed over constraint devices, but without NFV. Additionally, this work relies on Docker orchestration without delving into the intelligence required in the process, usually provided by means of a higher-level intelligence entity.

Also concerning security orchestration, authors in [18] define a context-aware security framework, focused on IoT scenarios. Nevertheless, the solution is not implemented yet (the framework is only theoretically defined, with simple use cases). Besides, the solution considers constraint deployments, but it does not specify how to perform them. The emphasis on security is highlighted in a recent important survey [5], which shows very few solutions that present the UAV as a MEC node, in addition none of them use it for the purpose of enhancing security.

In [19] we find an orchestration solution focused on securitizing fog computing scenarios. It develops a real-time intrusion detection algorithm. To do this, it monitors the network and when it detects anomalous traffic, it isolates the affected nodes. However, it is also a theoretical solution in which NFV is not taken into account. Furthermore, the paper does not explain the solution in detail, since it does not give any details about how it would be implemented. The work focuses mainly on explaining the network monitoring and traffic isolation methods, which are the main points of their solution.

Other paper dealing with similar topics is [20], which attempts to address vulnerabilities in NFV systems but focusing on improving the security of the NFV frameworks themselves. Thus, it does not propose to use them as a security orchestration method.

Finally, several proposals use SDN to manage security, mainly based on rules. Authors in [21] define a framework to detect and mitigate attacks using Access Control Rules. That framework contains multiple components that deploy mitigation actions when a security incident occurs. However, these rules merely work with components already deployed in the scenario, as they do not dynamically deploy any new components to enhance the countermeasures.

To conclude the related work, table 1 shows a summary with a comparison of all the aforementioned papers on which

TABLE 1. Related work comparison.

Proposal	NFV	Mobile	Constraint	Orchestration	Impl / valid	Security Functionalities
[6] [7] [8]	YES	YES	YES	NO	YES	VoIP, DNS, Telemetry, AP
[12]	YES	-	YES	NO	NO	Improve UAV connectivity
[13]	YES	YES	-	NO	NO	N/A (improve IOS)
[14]	YES	YES	YES	NO	-	N/A (remote control of a toy car)
[3] [9]	YES	NO	NO	YES	YES	AAA, Channel Protection
[17]	NO	NO	YES	NO	YES	Deploy security elements to Factory 4.0 and Smart home
[21]	NO	NO	NO	YES	NO	N/A (SDN)
[18]	YES	NO	NO	YES	NO	N/A
[19]	NO	NO	NO	YES	NO	N/A
[16]	YES	YES	YES	NO	YES	N/A

we can see the current state of the art, as well as the differences between each of them.

III. SECURITY HANDLING IN NFV/SDN-ENABLED IoT SCENARIOS

The rise of SDN and NFV has changed the approach whereby services and networks are managed. Those technologies not only improve network control and management but also provide the ability to react much quicker to certain events, such as attacks or service failures. SDN proposes the decoupling of the control and forwarding plane, simplifying and optimizing its management. NFV offers the ability to deploy network functions in a virtualized way, thus avoiding the requirement to use specific hardware.

By using these two technologies in conjunction, it is possible to design and deploy solutions that otherwise would be unfeasible, such as virtual security functions (VSF) that serve as enablers to deal with certain events.

In [9] some examples of enablers are shown, which will improve the security in our scenario. Nevertheless, not all enablers are suitable to be deployed on an UAV MEC-node, as they are not technically feasible (due to the limited UAV MEC-node computing resources) or not worthwhile. As part of the solution, some of them are proposed to improve the security or to react against certain attacks, taking advantage of their deployment on top of UAVs.

Table 2 shows different IoT threats, attacks, and possible countermeasures as well as the benefits of using UAVs as part of them. This is, some kind of threats can be mitigated by deploying specific VNFs as near as possible of the source of the issue but, it can occur that edge allocations are overloaded, not available, or even compromised. In this case the UAV can be deployed as a mobile node in order to provide/restore the required service or functionality in specific locations. Specific use cases are shown below.

A. USE CASE 1: COMPROMISED OR BROKEN WiFi/CELLULAR AP

Network communications on IoT scenarios are mainly wireless. A single access point usually serves multiple IoT devices, therefore making it a single point of failure and susceptible to be attacked. When that happens, the typical countermeasure is shutting down the AP, in order to isolate devices from the network and prevent that the infection spreads further.

TABLE 2. Countermeasures available on the UAV.

IoT Attack / Threat	Countermeasure	Benefits using a SUAV
Unathorized accesses	vAAA, vProxy, etc.	Deploy the countermeasures at the same place as the devices in situations on where an Edge node is not available or geographically inconvenient
Compromised AP	Isolating attacked network and deploy a new AP in place with an integrated IDS	This scenario is not possible without a SUAV
Infected devices	deploying a VNF which wirelessly reflashes the infected devices	If there is no posibility to perform the reflashing manually, the UAV could be able to do it.

In this context, a solution highly interesting would be not only isolating but also replacing the AP by deploying a new one alongside an IDS. As a result, the attacker would not notice he has been detected, while the attacker's behavior could be monitored and analyzed, extending the knowledge database. This kind of countermeasure can be performed by deploying an UAV as near as possible of the compromised/broken AP. This UAV will mount the same wireless physical technology as the AP, as well as a MEC-node able to deploy and configure VNFs on demand, depending on the orchestration process. In this case, the orchestration process could deploy and configure an AP VNF, trying to replicate as much as possible the real one. It could also deploy an IDS VNF in order to monitor the current behavior. Finally, it could configure the network through the SDN, in order to connect the AP and the IDS providing connectivity among the AP clients and the services.

B. USE CASE 2: AP AND MEC NODE UNAVAILABILITY

In this scenario, the previous use case is extended. It is assumed that not only the AP is compromised or turned off, but also the attack has affected an entire MEC node. In those circumstances, not only the AP needs to be lifted, but also the additional services that were deployed on it (as AAA entities, proxies, etc). As a result, all the original services will be provisionally provided on an UAV MEC-node, until the original compute node is available and operational again.

In such a situation, due to the limited resources of the UAV MEC-node, it may be necessary to assign a priority to each of the services. In this way, it will be possible to differentiate between those services that are indispensable and those that may be dispensable. The reason for this is none other than to maximize the service time since the available resources in the UAV MEC-node might not be enough for accommodating them all.

The following list shows a set of services that are likely to be instantiated in case of a MEC shutdown, and for which the Security Orchestrator would have configured enablers for the UAV-MEC:

- vNetwork: the mobile MEC-node must be able to provide network connectivity by using different wireless technologies and protocols (e.g., Wi-Fi, LoRa, 6LowPAN). These technologies are then combined with specific VNFs in order to provide connectivity.
- vAAA: the deployment of AAA services is really important (sometimes mandatory), since there may be scenarios in which devices must authenticate to access the network. If a device is detected trying to authenticate, but no service is available to perform the authentication, the usual procedure is trying to deploy one to the nearest node. Nevertheless, it could be possible that there is no compute node available nearby or, as it has been previously described, the node on which the service was deployed is currently offline. In this situation, part of an AAA infrastructure (for example, a PANA agent) would be deployed over the UAV MEC-node, so the devices would be able to authenticate themselves and access the network without the need to wait for the nearby compute nodes to become available again.
- Proxy DTLS: another interesting enabler is the deployment of an intermediate proxy. This may be necessary (e.g. a DTLS proxy), either because of constraint devices that are unable to secure their communications or due to compromised existing proxies. In that situation, the localization of the proxy becomes very important, since it must be as close as possible to the IoT devices to ensure the communications security. If there is no compute node nearby, being able to deploy a temporal proxy that establishes a secure connection could make a difference, since it would continue to offer the service until the problem is solved, or until another secure channel is established through a persistent mechanism.
- IoT Controller: the UAV can also be used to reflash infected devices in those situations where there is no possibility to perform the reflashing remotely. In this case, the UAV MEC-node will contain an IoT Controller VNF. Thus, the UAV needs to be deployed as near as possible of the devices to perform specific IoT command and control operations through IoT wireless technologies and protocols (e.g. CoAP over 6LowPAN).
- IDS: as mentioned in use case one, deploying an IDS is highly desirable either as a preventive measure (to check that nothing unusual happens) or to ensure that the origin of any failure is not due to an attack. If we assume a downed AP or an area where there are no nearby compute nodes, the UAV would move to that area and, replacing the original AP, would take care of passing all the traffic through the deployed IDS.
- IoT Monitoring Agent: the UAV is used to monitor passively the wireless network by using specific IoT wireless technologies (e.g. 6LowPan) in order to verify if there is any issue. This allows retrieving important information beyond the ethernet network.

By applying different combinations of the aforementioned enablers, the system can react in different ways when an incident occurs. These countermeasures are deployed dynamically over the UAV, providing the benefits of mobility. The next sections detail the required architecture as well as the UAV orchestration processes.

IV. SDN/NFV-BASED SECURITY ARCHITECTURE FOR UAVs DEPLOYMENTS

This section describes the proposed functional architecture and how it is deployed for the proof of concept that serves to evaluate this work.

A. FUNCTIONAL ARCHITECTURE

Our architecture leverages our previous general security management architecture [9] that allows context-aware dynamic orchestration and management of IoT systems. This architecture is able to deploy diverse kinds of countermeasures to counteract an attack without any operator action. These countermeasures are based on the use of SDN and the deployment of virtualized security functions on demand, using for that purpose a set of policies, monitoring agents, and a Security Orchestrator (among others). Our proposal extends that architecture endowing the platform with the ability to manage, orchestrate and deploy mobile Edge nodes and associated VNFs, not only in small data centers located at the network edge but also in remote or not easily accessible places – due to the use of UAVs.

The complete functional architecture is shown in figure 1. Its main elements are the following:

- Security Orchestration Plane: it is composed of two main components. 1) *The Policy Interpreter* receives a set of high-level security policies, defined in High-level Security Policy Language (HSPL), and refines them into medium-level security policies, defined in Medium-level Security Policy Language (MSPL), as well as translates from medium-level to final enabler configurations. 2) *The Security Orchestrator* receives the medium-level policies, chooses which enablers will be deployed or reconfigured (depending on security requirements, available resources…) and asks the Policy Interpreter to translate them into low level for the specific enabler to be deployed. On this plane, there are also other modules such as the *Security Policies Repository* (which stores the policies), the *Security Enablers Provider*, aimed to provide Security Enablers, and the *System model* which gathers all the information about the current infrastructure and services.

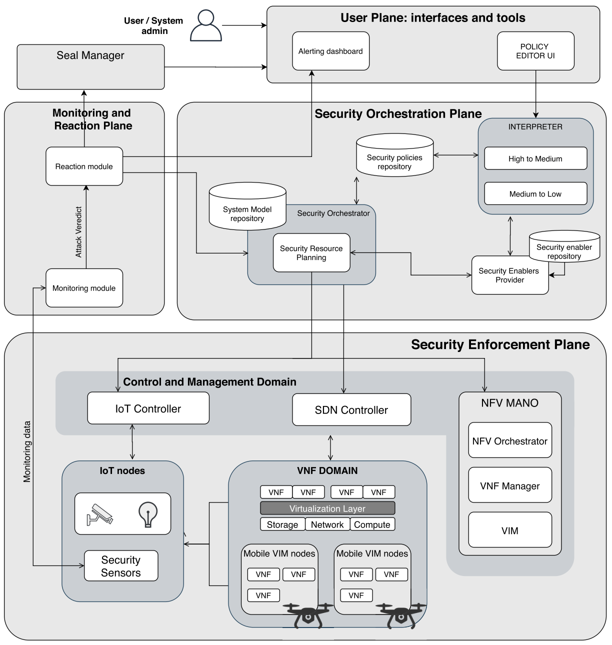


FIGURE 1. Proposed functional architecture leveraging [9] to accommodate UAV-MECs functions.

- User Plane, where there is an *Alerting Dashboard* and a *Policy Editor tool*, which allows the edition of the policies.
- Monitoring and Reaction Plane: it is formed by the *Monitoring Module*, in charge of monitoring the network in search of security failures, and the *Reaction Module*, in charge of offering countermeasures based on the data obtained by the monitoring module.
- Security Enforcement Plane: it contains the *Control and Management Domain* (the module that supervises the deployed Security Enablers and where the SDN, IoT and NFV-MANO controllers are located); *Infrastructure and Virtualization Domain* (all the physical machines that offer the computer, storage, and network resources to deploy the enablers); *VNF domain* (the deployed VNFs), and the *IoT domain* (the IoT devices that will be controlled, where the security sensors controlled by the monitoring module are also located).
- And finally, the *Seal Manager*, which provides a graphical representation of the security/privacy system status.

B. PROPOSED DEPLOYMENT ARCHITECTURE

Therefore, the purpose of this paper is to offer a solution capable of dynamically deploying virtual security functions depending on the context of the scenario, improving existing solutions by adding support to deploy the functions in a dynamic localization. The proposed deployment architecture for achieving this purpose is shown in figure 2. The deployment architecture is divided into two main areas, placing each component in the most accurate zone depending on its function. These areas are the *datacenter*, where the high-performance, energy, and bandwidth-intensive components

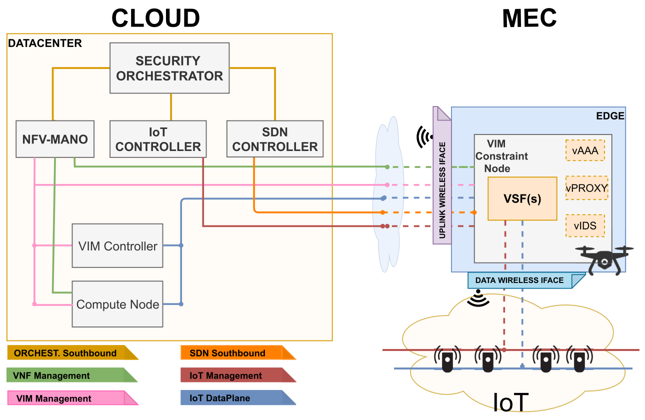


FIGURE 2. Proposed deployment architecture.

are deployed – as has been the trend in recent years – and the mobile *UAV-MEC node*, which is the key component in our solution, as it allows dynamic deployment of virtual security enablers anywhere, regardless of the presence or not of a previously deployed MEC node.

1) DataCenter

The items in the Cloud zone, where there is the datacenter, are the following:

- The *Security Orchestrator*, the element which communicates with the controllers to perform the appropriate tasks depending on the context. For that purpose, it takes into account all the available resources, the security capabilities, and the policy requirements and then requires the responsible controller to perform the countermeasures.
- The *NFV-MANO*, the system which manages the infrastructure upon which the security components required by the Security Orchestrator are deployed. The NFV-MANO is defined in ETSI's NFV Architecture Framework [22], and it is formed by the NFVO (NFV Orchestrator) and the VNF-M (VNF Manager). It is also in charge of communicating with the VIM (Virtualized Infrastructure Manager) controller to deploy the computing resources. Two different management networks are present, one intended for VIM internal communications and another for NFV coordination. Both networks need to be extended to the UAVs and should be as isolated as the wireless technology selected for the UAV may afford.
- *SDN Controller*, the component which manages the network in a centralized way by providing a common northbound interface and multi-southbound interfaces in order to implement the communication for different SDN protocols. This component allows the orchestrator to enforce network configurations like filtering, forwarding, or traffic mirroring along with the SDN components in an optimized way.
- *IoT Controller* as an entity responsible for interacting with the IoT devices providing centralized intelligence.

Similar to what an SDN controller does to the network, it implements different kinds of IoT southbound technologies depending on the deployment. It provides a common interface to actions available on the devices, such as reboot, power off, or more complex ones as activating authentication mechanisms.

2) MEC

At MEC zone, where UAVs and IoT devices are located, we highlight the UAV design. Every UAV has (at least) two wireless interfaces: one providing connectivity to the datacenter control plane (e.g. 4/5G), and others to perform the connection with the IoT devices data plane (e.g. LoRaWAN, 6LoWPAN, Zigbee). Thus, the infrastructure control communications are separated from the data plane traffic belonging to the IoT network. Specifically, we differentiate the following interfaces in the control and management plane:

- *VIM management interface*: this interface aims to manage the constraint VIM compute node (deployed on the UAV), connecting it to its controller and the rest of the VIM infrastructure, located at the datacenter. It will be used to order for operations such as the instantiation of the VNFs, terminating, … and all the VIM-related operations over them.
- *Orchestration southbound interface*: these interfaces comprise the set of operations needed by the security Orchestrator to manage the SDN Controller, IoT controller, and NFV-MANO components of the architecture. It is basically the set of operations offered by the orchestrated components of northbound interfaces. The SDN Controller is managed through a northbound API, supported by most of the current controllers such as ONOS or OpenDaylight. The IoT Controller is managed by the Orchestrator through a tailored interface, using protocols such as CoAP, and supporting operations such as reboot-flash the device and configure it. Finally, the NFV-MANO is managed by the Orchestrator through a provided northbound interface that supports methods related to tenant and datacenter management or VNF lifecycle – among others.
- *IoT Management interface*: IoT devices are usually capable of receiving command and control messages that modify their behavior. The source of these commands must be limited to trusted entities such as the IoT controller and must be as isolated as possible from other traffic, even though on the last mile they share the medium with the IoT Dataplane and are probably routed over it.
- *SDN southbound interface*: this interface is used by the SDN Controller to communicate with all the SDN-enabled devices over the network. Specifically, on the UAV it will be used for managing the onboard OVS using the OpenFlow protocol [23]. This protocol allows to modify the SDN switch flows according to the desired configurations. In this way we can apply dynamically network operations like filtering, forwarding, mirroring, or even field rewriting over the traffic which passes through the OVS.

  OpenFlow was designed assuming a stable connection between the network elements and the controller, even so, the research community has already paid attention to the possibility of using it in non-stable environments as addressed in [24]. Therefore, wireless links may introduce unexpected events like flapping links. Also, some controllers (e.g ONOS) flush the tables on switch connection, producing some data drop when the switch is reconnecting. Thus, a special SDN controller configuration for the boarded SDN switches needs to be enforced. Once the service is installed, the data plane continues behaving even if the controller is disconnected (unless reactive behavior is intended), so a proactive design must be prioritized. It is worth mentioning that the UAV flight control is independent of the SDN Data Plane.
- *VNF-M management interface*: as defined by the ETSI in [22], the VNF-Manager needs an interface to communicate with the deployed VNFs, for managing its lifecycle, exchanging the configuration information, and retrieving their state information. This communication will occur through a dedicated network that links the VNF-M with all the deployed VNFs.
- *UAV management interface*: it allows performing UAV command & control tasks (for example, specifying a set of GPS way-points in order to trace a route), as well as gather telemetry information as battery, altitude, gyro or accelerometer measurements by using a lightweight messaging protocol for communication with UAVs like MAVLINK. Of course, these kinds of operations and measurements can be also provided by the Ground Base Station or the transceiver, depending on the nature of the operation.

All the interfaces aforementioned are subject to the limitations related to wireless communications. In particular, when speaking about the orchestration, most of the protocols rely on TCP and in particular HTTPS for communication in the form or REST API (*Representational State Transfer Application Programming Interface*). Therefore most of the transport layer limitations will traduce onto delays in enforcing the actions intended to be taken by the on-boarded MEC and the need for retrial mechanisms. For more information about the effects of different transport-layer alternatives over wireless control layers for UAVs, the reader is directed to [8].

Regarding IoT wireless interfaces, since the solution needs to be able to integrate with the IoT existing infrastructure, the UAV will mount (at least) an IoT wireless bridge or access point. That access point must be compatible with the existing wireless networks (e.g. LoRaWAN, 6LoWPAN, Zigbee), as it will serve as the link between the IoT devices and the enablers deployed in the UAV. Moreover, UAVs may have multiple access points and use only the one(s) required according to the scenario, thanks to the use of an on-board OVS, managed

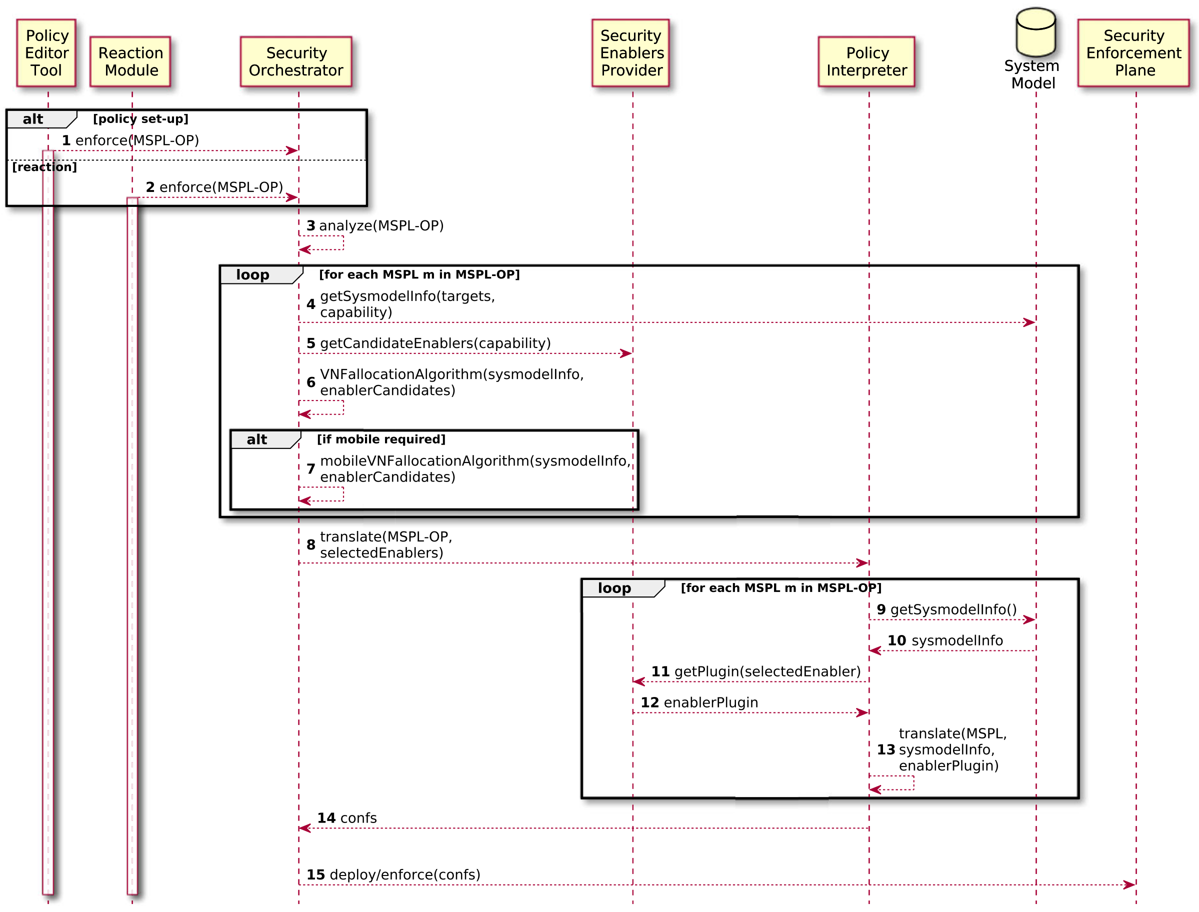

Policy Editor Tool
Reaction Module
Security Orchestrator
Security Enablers Provider
Policy Interpreter
System Model
Security Enforcement Plane
alt [policy set-up]
1 enforce(MSPL-OP)
[reaction]
2 enforce(MSPL-OP)
3 analyze(MSPL-OP)
loop [for each MSPL m in MSPL-OP]
4 getSysmodelInfo(targets, capability)
5 getCandidateEnablers(capability)
6 VNFallocationAlgorithm(sysmodelInfo, enablerCandidates)
alt [if mobile required]
7 mobileVNFallocationAlgorithm(sysmodelInfo, enablerCandidates)
8 translate(MSPL-OP, selectedEnablers)
loop [for each MSPL m in MSPL-OP]
9 getSysmodelInfo()
10 sysmodelInfo
11 getPlugin(selectedEnabler)
12 enablerPlugin
13 translate(MSPL, sysmodelInfo, enablerPlugin)
14 confs
15 deploy/enforce(confs)

FIGURE 3. **VNF orchestration in UAVs general flow diagram.**

by SDN Controller, that will handle the APs, dispatching the traffic through the indicated one.

The extension of the interfaces to the UAV through the Uplink Management Interface as shown in Figure 2 implies some level of slicing. If the connection employed is able to provide network slicing, that procedure may be used; if that is not the case, some kind of tunneling mechanism can be employed, e.g, EoIP, GRE, vxlan, etc.

All of these management and network interfaces are connected to the UAV-onboard compute node, where all the virtual security functions required by the security orchestrator will be deployed. The main characteristic is that, since it is a constraint node, its resources are limited. Therefore, it will use lightweight virtualization, allowing VNFs to be deployed as containers and thereby using fewer resources. Another important advantage of lightweight virtualization is the instantiation time [25], as the VNFs deployment as containers is considerably faster in opposite to traditional VM-based deployments.

V. SECURITY ORCHESTRATION IN UAVs

This section describes the orchestration process focusing on security and also defines the algorithm employed to place VSFs taking into account UAVs as MEC nodes.

A. SECURITY ORCHESTRATION

The enforcement process that transforms orchestration policies into VSFs on top of the deployment, and in particular in the UAV constraint node, is shown in Figure 3. Since the orchestration can be initiated as part of a proactive plan as well as part of a reactive countermeasure, the security orchestrator can receive MSPL Orchestration Policy (MSPL-OP) enforcement requests from different sources (Fig. 3 alt steps-1,2). Once the orchestrator receives the enforcement request, it analyzes the orchestration policy (which in turn it can be composed by multiple security policies) in order to identify the involved entities and capabilities (Fig. 3 step 3). For each security policy included in the orchestration policy, the orchestrator gathers relevant information like system model information (e.g. paths, locations) as well as the available security enablers which could enforce the security policy capability (Fig. 3 steps 4,5).

Once the orchestrator knows the physical/logical location of the involved entities, as well as the candidate security enablers, it executes a VNF allocation algorithm to find a set of suitable existing locations where the security policy could be enforced, if any. Otherwise, it calculates the most suitable allocations where a new security enabler VNF could be deployed. In both processes, we understand suitable allocations like those with sufficient resources which are deployed as close as possible of the involved entities (Fig. 3 step 6). If there is not suitable allocation place (e.g. the required security function should be enforced beyond our static infrastructure), the orchestrator executes a mobile VNF allocation algorithm in order to find a suitable allocation among the available mobile assets, like UAVs (Fig. 3 alt step 7). Specific mobile VNF allocation algorithm for UAVs is covered in section V-B. After the orchestrator has identified the specific security enabler and its location for each security policy, it requests the orchestration policy translation to the policy interpreter. The policy interpreter translates each security policy by using specific enabler plugins according to the security enablers selected by the orchestrator (Fig. 3 steps 8-13). Finally, the security orchestrator receives the enablers' configurations and it enforces them in the required enforcement points of the security enforcement plane according to the VNF allocation algorithm results. Thus, depending on the algorithm results, it can be required to deploy and configure new VNFs through the NFV-MANO as well as performing additional actions. For instance, re-configuring the network through the SDN Controller, configuring IoT devices through IoT Controller or even re-configuring directly already existing security enablers in order to enforce the new configurations. (Fig. 3 steps 14,15).

Regarding the framework, security policies that serve to represent important security capabilities are translated in specific security enablers. For instance, authentication and authorization (AAA, network or resources), data analysis (UTRC agent) and network traffic analysis (MMT agent) [26], filtering, forwarding, mirroring (ONOS), channel protection (DTLS-Proxy), data privacy (CPABE-Proxy) and IoT-honeynets (Cooja simulator). For further information about security policy management, translation and enforcement the reader is referred to our previous works [2], [3], [3], [9].

B. SECURITY VNF PLACEMENT ALGORITHM FOR UAV

This section shows the proposed algorithm aimed to select the most suitable UAV to deploy a required VNF, as demanded by the Security Orchestrator. The algorithm is structured in two phases. The first one aims to select the subset of available UAVs which would be able to allocate the VNF. For that purpose, we define some *hard-constraints*, which refer to the minimum required characteristics or current status that are mandatory to accomplish for the allocation task. Once the list of UAVs which could perform the flight is obtained, the second part of the algorithm calculates the suitability of each one of the UAVs, choosing the most adequate one to perform the task. In order to calculate these kind of constraints, different services are used for gathering information over time about the current status of the infrastructure. Specifically, UAVs telemetry, VIM status and SDN status information updates are retrieved from UAVs, VIM and SDN Controller respectively and stored in the system model continuously. The considered hard and soft constraints as well as the orchestration algorithm are detailed below:

1) HARD-CONSTRAINTS

- *Operating capacity*: the UAV must be operative and ready for the mission. It is important to note the possibility of choosing a UAV that is currently being used in a location close to the desired one, as long as that UAV fulfills the rest of the required characteristics and the current performing task is not affected. Also, even if it is affected but that task has a much lower priority than the new one, it could be "sacrificed" for the benefit of the new one, but this is suggested for a future revision.
- *Battery*: the UAV must have enough remaining battery to perform the complete route and to allow the enabler to perform this task. The battery needed to the flight is obtained by calculating the cost of the route from the UAV's origin location to the destination place, plus the cost of returning from that point to the origin base – if required. It is important to note it might not be necessary to include the cost of returning to the base station in those scenarios where an operator would go to collect the UAV once it is no longer useful (e.g. after repairing the access point the UAV was replacing). Besides, and as mentioned above, highlight the fact that the remaining battery once the UAV arrives at its destination (and subtracting the battery needed to return to the base if necessary) must be enough for the enabler to perform its task.
- *Hardware constraints*: the UAV must have the required hardware technologies to deploy the enabler. It includes specific radio transceivers for fronthaul and backhaul connectivity. For instance, if it is needed to deploy a LoRaWan Access Point as MEC-UAV, the UAV must be endowed with a LoRaWan antenna.
- *Atmospheric conditions*: the current atmospheric conditions will also be taken into account to determine whether the UAV is compatible with them. For example, wind gusts above a certain threshold, humidity, or rain.
- *Computing resources*: furthermore, the UAV MEC-node must have enough available computing resources, like RAM, disk, CPU... to deploy the required enabler. Also, the resources that will be available after the deployment of the enabler will be checked, since they may not be sufficient or the instantiation could affect any other enabler already in service, due to an overload.
- *Flight route*: another aspect to be considered is the route the UAV will follow to reach its destination, to ensure that it does not try to pass through any area that it is not able to cross, either because of legislation or because of risk to the integrity of the UAV it might happen that a particular drone would not be available to accomplish a particular task.

2) SOFT-CONSTRAINTS

- *Battery*: UAV MEC-node characteristics, such as the remaining battery (needed to calculate how much battery will be available at the landing, which will be needed by the enabler to accomplish its task). This is important, since it is one of the most crucial information: knowing how long the enabler must be deployed (one hour, two hours, the maximum possible). With this in mind, the aim will be to select the UAV with the minimum resources required for the tasks whose duration is known in advance. Thus, more powerful resources will be available for those use cases where the enabler must be deployed for the maximum possible time. This value will be obtained from the UAV management interface (telemetry), as it is a resource from the UAV itself.

- *UAV Velocity*: another UAV characteristics, such as the UAV maximum velocity that could be very important in those scenarios on which the timing is crucial, or the wireless range of the antenna, to ensure it will cover all the required area. This value will be obtained from the technical characteristics of the UAV, and will be into account depending on the deployment requirements of the mission.
- *Computing resources*: the compute resources, such as available RAM, disk or CPU are considered also as soft-constraint. In this case we will take into account the status of the resources after the instantiation to maximize the performance of deployment in different drones. This value is obtained from the VIM, which in turn obtains it from the node via the VIM-mgmt interface. The VIM automatically monitors the current status of all active nodes, in terms of CPU, RAM, etc, so all the algorithm function has to do is requesting them to the VIM when needed. As said before, the assigned cost will depend on the minimum resources needed, trying to assign the most accuracy UAV to not misuse available resources.
- *Network metrics*: network aspects, such as bandwidth and latency. It will depend on the kind of implemented technology in the UAV (5G, LTE, 802.11, etc.). These metrics can be retrieved from monitoring agents as well as from the SDN Controller.

3) VNF PLACEMENT ALGORITHM IN UAVs

As explained before, the security orchestrator relies on an algorithm to decide in which drone (compute node) should be allocated the VNF/VSF, according to the actual security enabler to deploy, physical/virtual conditions and several contextual aspects. The VNF/VSF placement problem follows an optimization procedure starting with hard-constraint verification that comes up with a subset of potential UAV candidates.

First lines of Algorithm 1 describe that hard-constraint verification process. It first checks the inequality (line 8), that verifies hard requirements $\forall_r \in R_d$, where R_d is the set of hard requirements to be fulfilled by a given drone d as described in above in section V-B1. Thus, $r \in R$, includes, for instance, a battery requirement, radio technology requirement, resource storage requirement or a RAM requirement of the VNF v when deployed in d.

If the inequality is met the drone d can be considered as a potential candidate for the optimization process, as it satisfies the requirements to allocate the VNF enabler in the drone d. For each requirement the inequality considers the different kinds of VNFs V, where V_d denotes the VNFs belonging to V that are already allocated in a drone d.

$C_r(v)$ refers to the constraint function that obtains the normalized value of a requirement r, for a given VNF v, as defined by the administrator. $C_r^a(d_v)$ is the constraint function that gets the actual normalized value of a particular type of requirement r currently available a in a given drone d for a given v. For instance, available RAM in a virtual machine, or current available battery in the drone. This value is obtained by monitoring the physical drone, obtaining telemetry data, retrieving data from Virtual Infrastructure Manager in the core, and the compute node in the MEC.

Likewise, $C_r^n(v')$ refers to the constraint function that calculates the value for the requirement r, needed n by a virtual enabler to be allocated, as demanded by the orchestrator, for a given new v' VNF to be allocated. For instance, a particular kind of vIDS like snort might need specific RAM requirements, or for a particular flight to a target land position this function would get the specific amount of battery needed.

It should be noticed that $C_r(v)$ will be usually positive and will define the minimal requirement or margin in the inequality of line 8, between what is available $C_r^a(d_v)$ in the drone and what is needed $C_r^n(v')$ by the orchestrator. For boolean hard requirements, such as the one related to the existence or not of a valid route to the target physical location, $C_r^a(d_v)$ will be given 0 or 1, and $C_r(v)$ will be fixed to 0.

As a result, the set of Available Drones $AD = \{d_1..d_n\}$ that satisfies the hard requirements is held in vector $AD\{\}$.

Thereafter, for each available drone, the algorithm computes the gain that could be obtained by the fact of deploying the VNF in such a drone. To that aim, it relies on two functions needed to obtain soft-constraints, as defined above in section V-B2. Lines 14-19 of Algorithm 1 defines this process. A first Soft-Constraint function $SC_s^a(d_v)$ gets from monitoring the normalized value of a soft-constraint s available a for a given VNF in a device d_v (e.g. RAM available or disk space). The summation for all $v \in V_d$ will denote the total overall available value in the drone. The second Soft-Constraint function $SC_s^n(v')$ holds the normalized value of a given soft-constraint s as "needed" n by the orchestrator. The subtracted value of both functions is weighted w_s by the importance given by the administrator for that kind of soft-constraint. As a result, the algorithm returns the list of drones ordered by their suitability to allocate the VNF.

VI. ARCHITECTURE INSTANTIATION AND DEPLOYMENT

In this section we will describe the deployment of the infrastructure required, as well as the instantiation of the Use Case previously explained. Specifically, Editor tool, Reaction Module, Security Orchestrator, Security Enablers provider, Policy Interpreter and System Model were deployed in docker containers inside a VM over a physical Dell poweredge

Algorithm 1 General VNF Allocation Algorithm for MEC-Enabled UAVs

```
1:  $D = \{UAVs\}$
2:  $V = \{VNFs\}$
3:  $R = \{Hard - Requirements\}$
4:  $S = \{Soft - Constraints\}$
5:  for $d \in D$ do
6:      for $r \in R$ do
7:          for $v \in V$ do
8:              if $C_r(v) \leq (\sum_{v \in V_d} C_r^a(d_v)) - C_r^n(v')$ then
                    $AD[d] \leftarrow d$
9:              end if
10:         end for
11:     end for
12: end for
13:
14: for $d \in AD$ do
15:     for $s \in S$ do
16:         for $v \in V$ do
                $sum \leftarrow sum + ((\sum_{v \in V_d} SC_s^a(d_v)) - SC_s^n(v')) * w_s)$
17:         end for
18:     end for
        $gain_{AD}[d] \leftarrow sum$
        $sum \leftarrow 0$
19: end for
20: return $sort(gain_{AD}[])$
```

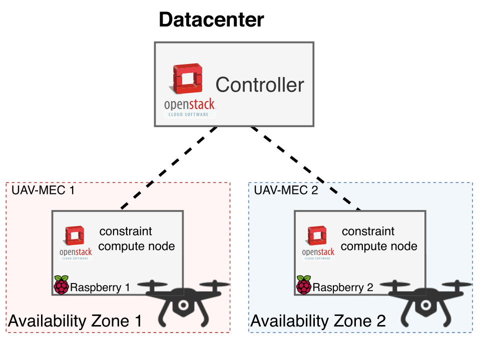


FIGURE 4. Representation of UAVs in different availability zones.

R640 2x Intel Xeon Platinum 8160 256 GB RAM RAID-5 SATA SSD server. All modules were developed in Python3, providing different REST APIs through django and falcon frameworks features. Databases use Mysql v5.6.

Regarding the enforcement plane, firstly, the VIM deployment will be explained, followed by the instantiation of the use case and finally, the UAV deployment.

A. VIM DEPLOYMENT

As mentioned above, the deployment infrastructure has been implemented over constraint devices, which will allow deploying our VSFs in any place or device, such as in a UAV (as was done in [6] and [7]).

For that purpose, some technologies have been selected to implement the proposed solution. The MANO Orchestration

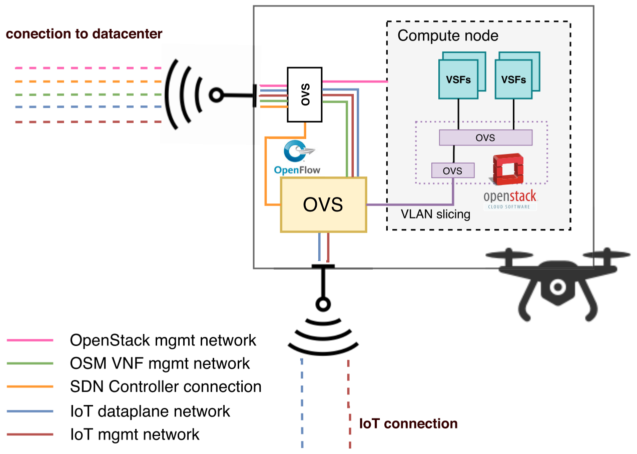


FIGURE 5. Network design of the constraint UAV-MEC node.

platform chosen for the proof of concept is OSM[1] Release SEVEN for its stack alignment with the ETSI NFV MANO reference framework. For the VIM, OpenStack Stein [27] has been chosen partly for the simplicity of integration with OSM. Both OSM and OpenStack controller services are instantiated onto virtual machines.

The underlying network isolation is provided with VLAN segregation provided by Cisco Catalyst 3750G Version 12.2 (25r) SE1. Wireless is provided with Cisco Aironet 1140 which maps VLANs to different ESSIDs, therefore isolating the management channels with WPA2 pre-shared keys.

SDN is spread over the experiment using OpenFlow 1.3 based on OpenVSwitch on the UAV-MEC nodes and the VNFs and HPE Aruba 2920 on top of the underlying network. The controller chosen is ONOS version 2.4.0.

A relevant aspect related to the solution is how we indicate to the VIM on which UAV (the one the security orchestrator chose using the algorithm previously defined) the security functions will be deployed. For that purpose, our solution is based on the use of a mechanism called, in ETSI terminology, resource zones [28]. These zones are labeled with different names depending on the VIM involved.

In OpenStack, they are called availability zones [29] and, as the majority of commercial VIMs, are used as a method to segment the cloud resources according to location, network type, power, etc. The concept of how the availability zones have been mapped to MEC UAVs is shown in Figure 4. The idea is to differentiate each UAV in a specific Availability Zone (as shown in the figure), thus having an availability zone for every existing UAV. In this way, when deploying the enablers, the security orchestrator will indicate in which zone (UAV) to deploy them.

Regarding which computer devices will be chosen to perform the deployment, and since our solution aims to place them into UAVs, it is required to use devices with restricted

[1]Open Source Mano, *https://osm.etsi.org/*

hardware capabilities and compact dimensions. For this, the selected devices are Raspberry Pi 4 Model B.

The Operating System running on the constraint compute nodes for the VIM is Ubuntu 18.04LTS for ARM architecture, due to the requirements of OpenStack and the raspberry CPU. In terms of communications, as it was foreseen in the previous section, the UAVs MEC-node has two wireless interfaces, one dedicated to performing the communication with the datacenter (VIM mgmt and NFV-MANO VNFs-mgmt networks) and another one with different wireless technologies, used to connect with heterogeneous IoT devices. Generally, and specifically here to simplify the scenario, there are two networks: management and data plane. Figure 5 shows all the items and connections which form the UAV-MEC node. The networking inside is formed by several switches that provide the necessary connectivity. Among the switches deployed, it is needed to highlight the presence of a specific Open-vSwitch [30], which is controlled from the external SDN controller via OpenFlow. This OVS is the one in charge of diverting traffic to the Constraint Compute Node embedded in the drone or directly to the datacenter. Thanks to OVS capabilities it is possible to perform some advanced actions, such as dropping based on matching actions or rewritting packet headers. The OVS facing the Uplink wireless interface is the one in charge of tunneling the traffic isolated for the different networks towards the datacenter.

Regarding the remaining switches, there are two bridges managed by OpenStack itself, and another one to operate with the management networks. As the VSFs are required to be connected to one of them (NFV-MANO VNF's mgmt), this connection will be managed too for the main OVS to be offered to openstack. It could be possible to merge one of the OVS managed by openstack with the external one - managed by the SDN Controller – but it would require the SDN controller to have a deeper understanding of the inner workings of OpenStack (since it would have to manage its internal connections), so the choice was to separate the internal OVS corresponding to openstack from the one managed by the SDN controller. The use of an OVS differs from the work done by [6], on which linux bridges are used, which were not controlled by any external SDN Controller. Adding an OVS provides the UAV with increased versatility, as it is a powerful resource. Since it enables dynamic control of the flows that pass through it, it allows the option of isolating the desired networks easily and transparently. In this way, scenarios like those defined in section III-A are easy to implement (those in which the idea is to isolate certain devices), and not only that, but it is even possible to redirect traffic to a honeypot (a replica of the real scenario) so that the potential attacker does not notice any difference.

B. USE CASE DEPLOYMENT

To verify the feasibility of the proposed solution, one of the use cases previously defined is used. In particular, an Access Point with an IDS (specifically, a Snort) is deployed, as it is defined in Use Case 1. The idea is offering an AP replacement

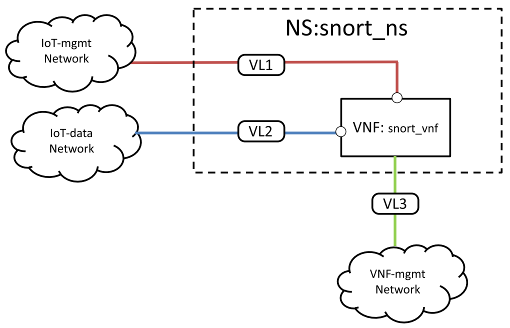


FIGURE 6. Network scenario descriptor (NSD) for the IDS use case.

with sniffing capabilities, to check whether the malfunction of the access point is due to normal circumstances or caused by some kind of attack. So, for that purpose, the NSD showed in figure 6 is defined.

As we can see in figure 6, the instance will be connected to three networks: the VM-mgmt, the one through which OSM will inject the configuration, and the IoT networks IoT-mgmt and IoT-data, the ones the AP will offer and on which we want to monitor the traffic.

The instantiation process is shown in figure 7 and it occurs as follows:

- First, the Security Orchestrator decides to deploy the enabler on our UAV, as seen in section V-A, so it requests OSM to instantiate it. The three main components of OSM are LCM (Life-Cycle Manager, the component in charge of performing the orchestration), RO (Resource Orchestrator, the one who communicates with the VIM) and VCA (VNF Configuration and Abstraction, who fulfill the VNF-Manager tasks in terms of injecting configuration). The NBI (Northbound Interface) is who receives all the petitions, so it is the component that receives the Security Orchestration claim.
- The Security Orchestrator's request is processed by the LCM, who asks the RO to deploy the required scenario. At the same time, the LCM also requests the VCA to create the charms which will inject the configuration to the instances. Currently (REL SEVEN), the injection is made by ''proxy charms'', a series of LXC containers located on the OSM host and powered by juju which communicates with the instances.
- The RO communicates with the VIM (Openstack in this scenario) and asks for the creation of the required networks (if needed), as well as the instances.
- Openstack receives the order and starts the instantiation. Once the connection points are ready, it creates the instance.
- Meanwhile, the VCA has created the proxy charms. When they are ready (and so is the instance), they inject the required configuration (if any).
- If subsequently required, the SO could inject some configuration (called Day-2 configuration) throughout the VCA. It would ask OSM for it, and the charms will inject the required configuration to the instance. That new

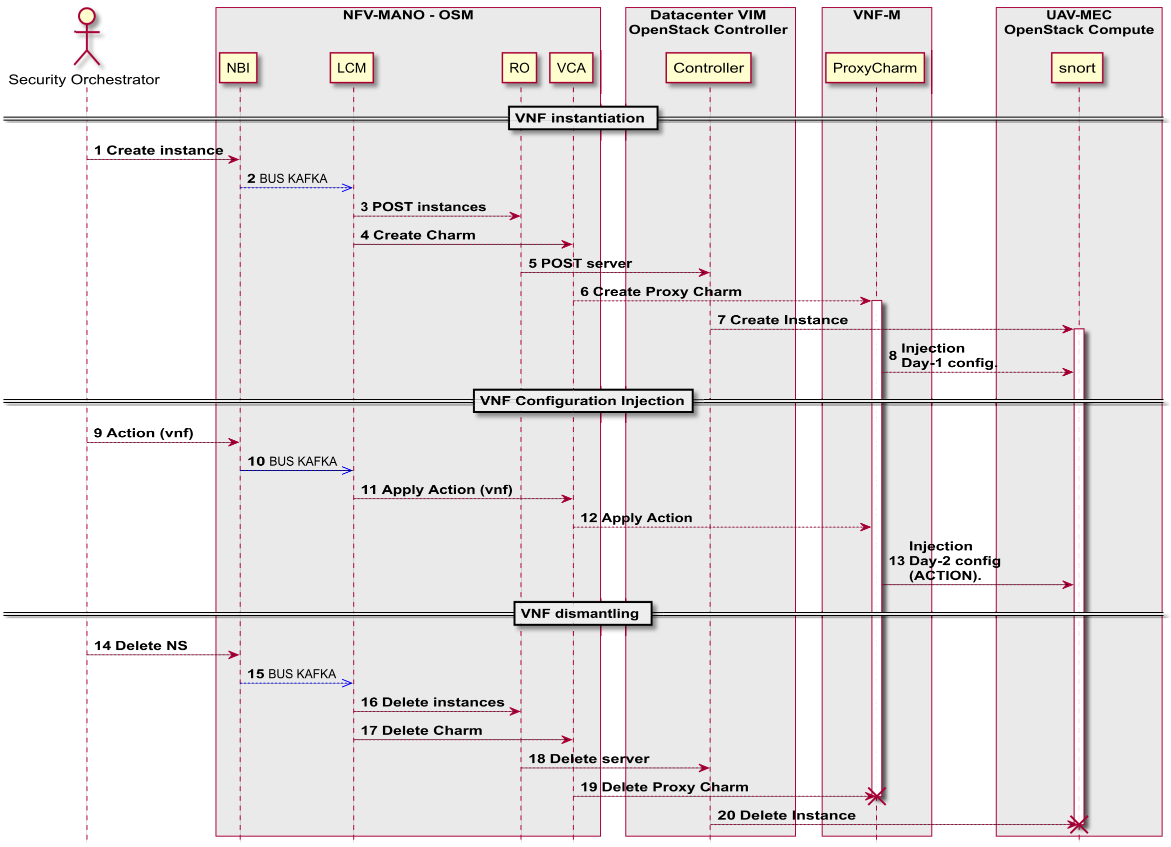


FIGURE 7. Sequence diagram of the VSFs' deployment.

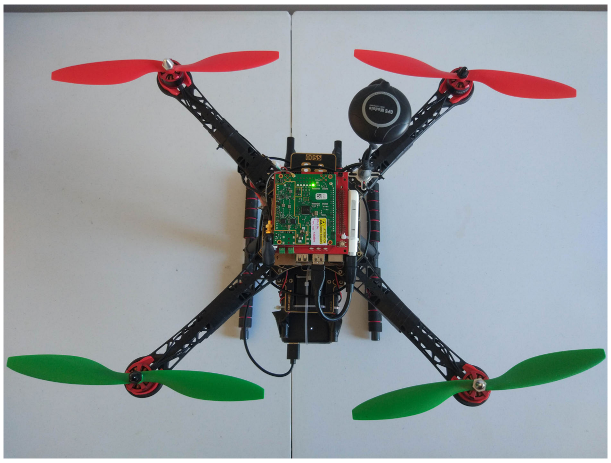


FIGURE 8. Implemented UAV as MEC node, using raspberry and Lora transceiver.

configuration could be a new set of rules to SNORT, for example.

C. UAV DEPLOYMENT

Figure 8 shows the quadcopter UAV assembled by ourselves from commercial components on purpose for experimentation. All components are supported by a 500mm frame which also works as Power Distribution Board (PDB). Four axis converges in a four floor platform. First floor (the bottom one), supports a 4-cell 2500 mAh LiPo battery in order to provide power supply to the UAV. The second floor contains a 10400 mAh LiPo battery for feeding the UAV-MEC node. The flight controller, STM32-F722, is located in the center of the third floor. This floor also mounts the axis which provides support for the propulsion system components, these are, 30A Electronic Speed Controllers (ESC) and 920Kv brush-less engines with 10in diameter and 4.5 in pitch propellers. Finally, on the last floor, the constraint compute node as well as the GPS antenna are located. In this specific build, the UAV-MEC node is compounded by a Raspberry Pi acting as OpenStack node, able to provide connectivity through LoRaWAN, 4G and WiFi technologies. The Take-Off Weight (TOW) of the UAV also including the payload is 1490 g. However, despite the fact that we built our UAV solution per components and we used open source software (e.g., INAV), similar solutions can be achieved by using similar Commercial Off-The-Shelf (COTS) alternatives. For instance, our MEC node could be on-boarded in a DJI matrice series.[2]

VII. PERFORMANCE EVALUATION

In order to evaluate the proposal, different kinds of experiments have been performed. Specifically, the focus is on the time the UAV-MEC takes to deploy new VNFs and on

[2]https://www.dji.com/es/matrice-200-series-v2

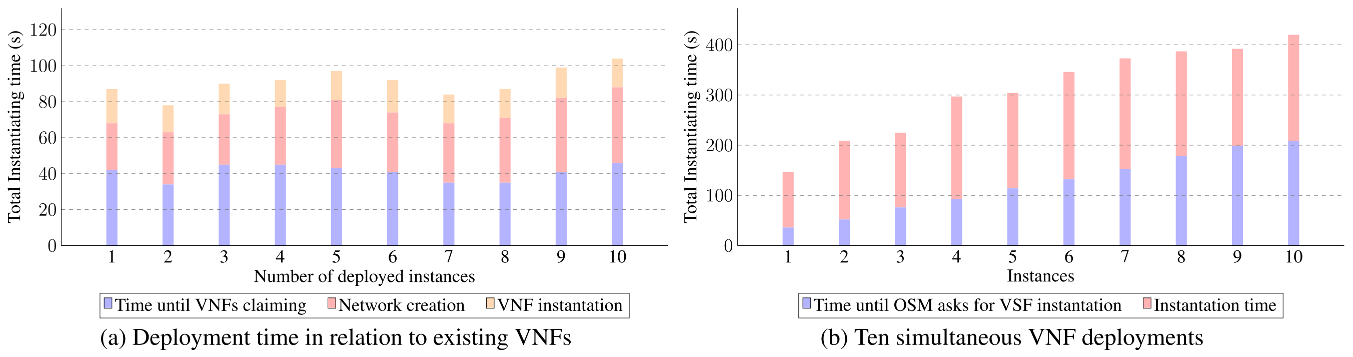


FIGURE 9. Times obtained by VNF deployments in the UAV-MEC through OSM.

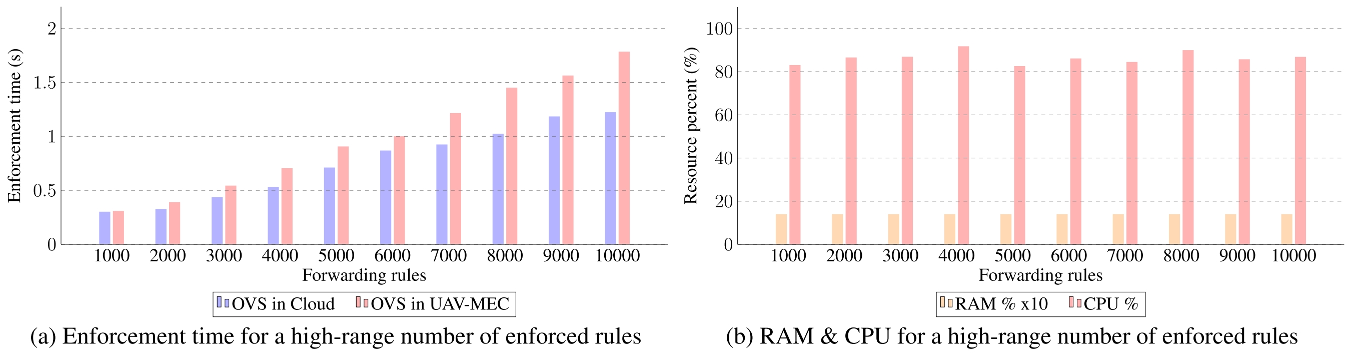


FIGURE 10. SDN enforcement evaluation in Cloud and UAV-MEC. RAM and CPU usage.

the SDN configuration. CPU and RAM resources are also measured in different scalability tests, as well as the duration of the two batteries of the solution. These last values are important as they represent flying operative time and MEC-node operative time until they drain their respective batteries. This is specially important (from a security point of view) when replacing full MEC nodes or deploying VSFs with intensive resource usage.

This last value is important as it represents both UAV flying time and MEC node operative time until it drains its independent battery, which in turn and from a security point of view is of the utter most important when replacing full MEC nodes or deploying VSFs with intensive resource usage.

A. UAV-MEC CONSTRAINT NODE PERFORMANCE

The first measurement is the performance of the UAV-MEC node. To conduct the tests the VSF of the use case has been used, configured for using a 1Gb disk, three network interfaces, and 256Mb of RAM.

The results can be consulted in figure 9. The first test performed aims to check how the number of VNFs already deployed in the UAV-MEC node affects the time needed to instantiate a new one. The graph shows the times obtained when deploying a single instance (i.e. when there is no previously deployed instance), up to the number ten, when there are nine instances deployed and the new one is the tenth. The first time is the time elapsed since OSM receives the request to launch the instance until openstack starts instantiating it. The second time is the time until all the network interfaces are ready, and finally the third one represents the time taken until the VNF has been correctly instantiated and it is active in the UAV-MEC.

The results show how the number of instances previously deployed hardly affects the deployment time. It is noted that it increases slightly, especially in the last instances, although it is not a significant increase.

The second test aims to note the instantiation time with simultaneous instances, it is, when the scenario to be deployed contains more than one VNF. It can be observed how, in contrast to the previous test, the deployment time here is significantly affected by launching the instances simultaneously. This is mainly due to two causes: the first one, that OSM requests the instances one by one, and until OpenStack does not respond that it is deploying it, OSM does not send the next one. This adds an important delay since it takes more than three minutes to request the last instances. The other cause is due to the limited resources of the constraint node, because even though the instances already deployed do not affect the performance, instantiating multiple instances simultaneously does. This can be noticed starting from the fourth instance, where the instantiation times are double that of a sole instance. It is noted, too, how the instantiation time of the first VNF is also higher than the time required when launching a single instance, as seen in previous tests.

Additionally, we have evaluated the memory consumption in the UAV-MEC when deploying simultaneously a large number of light-weight VNFs. The results, shown in Figure 11, indicate that the RAM consumption follows a

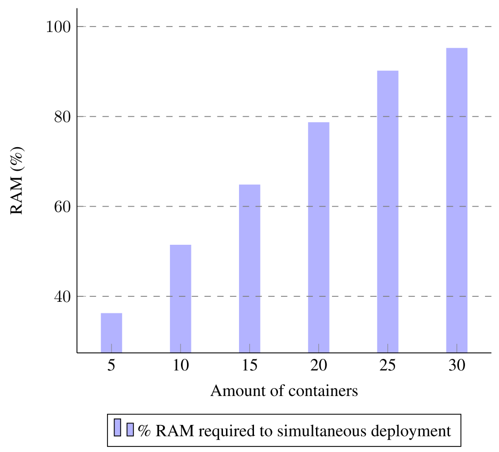


FIGURE 11. Variation of the used RAM depending on the number of containers deployed simultaneously.

TABLE 3. UAV TOW and battery times.

UAV TOW (g)	UAV+Payload TOW (g)	UAV Flying time (min)	UAV+Payload Flying time (min)	Busy MEC time (h)
893	1490	~13	~ 7	~ 5

TABLE 4. UAV-MEC node script consumption.

CPU (%)	IO Read (MB/s)	IO Write (MB/s)	Network Traffic (Mbps)	RAM Used	RAM Free
74.93	51.36	25.60	19.92	3038.35	187.73

linear (non-exponential) progression trend as the number of simultaneous containers are deployed.

B. UAV PERFORMANCE

Regarding the UAV performance, we focused on flying time (with and without payload) as well as the MEC-node availability time by using an independent battery.

Table 3 shows the Take-Off Weight (TOW) and battery times for our UAV solution both, empty and loaded. In our experiments, the UAV flying time falls almost half when we apply the payload. Finally, last column shows the MEC battery time when the constraint compute node is busy. In order to simulate the load of the system, we developed an script which maintains the UAV-MEC node in a high-performance profile during the experiments.

Table 4 shows the average consumption of the script for the UAV-MEC node resources. During the tests, we record the *dstat* command values while the script maintained four processes above 70% of CPU usage. It also alternated write and read for 10 seconds continuously from an usb HDD. In order to generate wireless traffic, the script performs an iperf at 20Mbps. Finally, it consumes up to 90% of the available RAM memory which in this case was near to 3 GB. With this workload, the script drained the battery in less than 5 hours. However, this time can be increased depending on the required access to the wireless network as well as the required access to the usb HDD up to 10-12 hours (the theoretical calculation based on the power consumption features).

In order to evaluate the SDN enforcement scalability in the MEC, we enforced a different number of forwarding policies. Specifically, we deployed ONOS SDN Controller remotely and we obtained the average time the controller taken to enforce from 1k to 10k forwarding policies up to 30 times. These policies were generated by a script we developed which uses the same source IPv6 address (aaaa::1/128) but it increases the destination address (up to aaaa::2710/128) in order to create different matches. After the policy translation into SDN flow rules, we measured the time taken since the controller received the SDN flows enforcement request until it received the Openflow barrier reply message. Barrier messages allow the controller to request the SDN switch that all messages sent before the barrier must be processed prior to handling any new messages. When the SDN switch completes all the requested operations it sends the barrier reply to the controller. In order to ensure the flows were properly enforced, we also developed a script that retrieves the flows from the SDN Controller and verifies that each one of them appears like "ADDED" into the system.

Figure 10a shows the timing results for the SDN enforcement in both, a OVS VM in the cloud and an OVS located in the UAV MEC node. For 1k and 2k the results are similar but the more rules the more different between local and mobile MEC enforcement. This is due to two main factors, the wireless network latency and the computation speed. Regarding resource consumption (Figure 10b), we recorded CPU and RAM results from *top* command each 100 ms and we realised the rules magnitude is not overloading the UAV-MEC node. The rules enforcement requires an average of one CPU between 80-90% of loading. On the other hand, the RAM consumption is always under the initial value of 1.4%. This makes sense since each OVS row table is about 570 bytes,[3] this means 570 bytes * 10k entries = 5.43 MB, while the 1.4% of the total RAM of the device is 57.344 MB.

VIII. CONCLUSIONS

This paper has presented a novel security management framework that enables the deployment, orchestration and management of lightweight security VNF in UAV-MECs nodes leveraging NFV/SDN. In this regard, the paper has provided and UAV allocation algorithm that differentiates hard-constraint and soft-constraint, and considers diverse contextual aspects for allocation, such as operating capacity, battery, atmospheric conditions, computing resources (RAM, disk, CPU), network metrics. It allows to select, without human intervention, the most suitable UAV-MEC to perform the tasks and allocate the VNFs, thereby optimizing the available resources.

The proposed framework has been successfully implemented and deployed using open-source tools and it has been validated in a real testbed and use case, where the UAV-MEC, endowed with vIDS capabilities, is intended to replace a compromised Access Point. To evaluate the performance of the framework, it has been stressed in terms of SDN rules to be allocated in the dron, number of VNFs instantiated in

[3]https://software.intel.com/content/www/us/en/develop/articles/the-open-vswitch-exact-match-cache.html

UAV-MEC, whereby checking the scalability and behavior in extreme situations that would require the maximum performance of the UAV-MEC. The tests prove the suitability of the solution to face those situations, since it reacts and handles the workload successfully in worst-case scenarios. As future work, we envisage to improve the proposed algorithm, so it is capable to adapt to different and more complex scenarios, such as the ability to consider the simultaneous collaboration and chaining among UAVs-MEC, and migration of VNFs between UAVs.

REFERENCES

[1] M. B. M. Noor and W. H. Hassan, "Current research on Internet of Things (IoT) security: A survey," *Comput. Netw.*, vol. 148, pp. 283–294, Jan. 2019.

[2] A. M. Zarca, J. B. Bernabe, I. Farris, Y. Khettab, T. Taleb, and A. Skarmeta, "Enhancing IoT security through network softwarization and virtual security appliances," *Int. J. Netw. Manage.*, vol. 28, no. 5, p. e2038, Sep. 2018, doi: 10.1002/nem.2038.

[3] A. M. Zarca, D. Garcia-Carrillo, J. B. Bernabe, J. Ortiz, R. Marin-Perez, and A. Skarmeta, "Enabling virtual AAA management in SDN-based IoT networks," *Sensors*, vol. 19, no. 2, p. 295, Jan. 2019.

[4] A. M. Zarca, J. B. Bernabe, A. Skarmeta, and J. M. A. Calero, "Virtual IoT HoneyNets to mitigate cyberattacks in SDN/NFV-enabled IoT networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1262–1277, Jun. 2020.

[5] O. S. Oubbati, M. Atiquzzaman, T. A. Ahanger, and A. Ibrahim, "Softwarization of UAV networks: A survey of applications and future trends," *IEEE Access*, vol. 8, pp. 98073–98125, 2020.

[6] B. Nogales, V. Sanchez-Aguero, I. Vidal, and F. Valera, "Adaptable and automated small UAV deployments via virtualization," *Sensors*, vol. 18, no. 12, p. 4116, Nov. 2018.

[7] C. Tipantuña, X. Hesselbach, V. Sánchez-Aguero, F. Valera, I. Vidal, and B. Nogales, "An NFV-based energy scheduling algorithm for a 5G enabled fleet of programmable unmanned aerial vehicles," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–20, Feb. 2019.

[8] L. F. Gonzalez, I. Vidal, F. Valera, B. Nogales, V. Sanchez-Aguero, and D. R. Lopez, "Transport-layer limitations for NFV orchestration in resource-constrained aerial networks," *Sensors*, vol. 19, no. 23, p. 5220, Nov. 2019.

[9] A. M. Zarca, J. B. Bernabe, R. Trapero, D. Rivera, J. Villalobos, A. Skarmeta, S. Bianchi, A. Zafeiropoulos, and P. Gouvas, "Security management architecture for NFV/SDN-aware IoT systems," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8005–8020, Oct. 2019.

[10] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[11] F. Al-Turjman, M. Abujubbeh, A. Malekloo, and L. Mostarda, "UAVs assessment in software-defined IoT networks: An overview," *Comput. Commun.*, vol. 150, pp. 519–536, Jan. 2020.

[12] K. J. S. White, E. Denney, M. D. Knudson, A. K. Mamerides, and D. P. Pezaros, "A programmable SDN+NFV-based architecture for UAV telemetry monitoring," in *Proc. IEEE CCNC*, Jan. 2017, pp. 522–527.

[13] M. Zhu, J. Cao, Z. Cai, Z. He, and M. Xu, "Providing flexible services for heterogeneous vehicles: An NFV-based approach," *IEEE Netw.*, vol. 30, no. 3, pp. 64–71, May 2016.

[14] R. Vilalta, S. Vía, F. Mira, R. Casellas, R. Muñoz, J. Alonso-Zarate, A. Kousaridas, and M. Dillinger, "Control and management of a connected car using SDN/NFV, fog computing and YANG data models," in *Proc. 4th IEEE Conf. Netw. Softw. Workshops (NetSoft)*, Jun. 2018, pp. 378–383.

[15] E. Yanmaz, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter, "Drone networks: Communications, coordination, and sensing," *Ad Hoc Netw.*, vol. 68, pp. 1–15, Jan. 2018.

[16] O. Bekkouche, K. Samdanis, M. Bagaa, and T. Taleb, "A service-based architecture for enabling UAV enhanced network services," *IEEE Netw.*, early access, Apr. 22, 2020, doi: 10.1109/MNET.001.1900556.

[17] A. Boudi, I. Farris, M. Bagaa, and T. Taleb, "Assessing lightweight virtualization for security-as-a-service at the network edge," *IEICE Trans. Commun.*, vol. E102.B, no. 5, pp. 970–977, May 2019.

[18] N. E. Petroulakis, E. Lakka, E. Sakic, V. Kulkarni, K. Fysarakis, I. Somarakis, J. Serra, L. Sanabria-Russo, D. Pau, M. Falchetto, D. Presenza, T. Marktscheffel, K. Ramantas, P.-V. Mekikis, L. Ciechomski, and K. Waledzik, "SEMIoTICS architectural framework: End-to-end security, connectivity and interoperability for industrial IoT," in *Proc. Global IoT Summit (GIoTS)*, Jun. 2019, pp. 1–6.

[19] M. Vahabi, H. Fotouhi, and M. Björkman, "FIREWORK: Fog orchestration for secure IoT networks," in *Proc. SPNCE*. Cham, Switzerland: Springer, 2019, pp. 311–317.

[20] M. Pattaranantakul, R. He, A. Meddahi, and Z. Zhang, "SecMANO: Towards network functions virtualization (NFV) based security MANagement and orchestration," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 598–605.

[21] M. Dimolianis, A. Pavlidis, D. Kalogeras, and V. Maglaris, "Mitigation of multi-vector network attacks via orchestration of distributed rule placement," in *Proc. IFIP/IEEE IM*, Apr. 2019, pp. 162–170.

[22] *Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV*, document ETSI GR NFV 003 V1.5.1, ETSI, Dec. 2014. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf

[23] The Open Networking Foundation. *OpenFlow Switch Specification*. Accessed: Jun. 26, 2020. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf

[24] G. Araniti, J. Cosmas, A. Iera, A. Molinaro, R. Morabito, and A. Orsino, "OpenFlow over wireless networks: Performance analysis," in *Proc. IEEE BMSB*, Jun. 2014, pp. 1–5.

[25] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[26] D. Rivera, E. Cambiaso, I. Vaccari, J. Villalobos, and M. Bagaa, "Final monitoring components services implementation report," Montimage, CNR, ATOS, AALTO, UTRC, Anastacia H2020 Eur. Project Deliverable D4.4, Montimage, Paris, France, Tech. Rep. D4.4, Oct. 2019. [Online]. Available: http://www.anastacia-h2020.eu/deliverables/ANASTACIA-D4.4-FinalMonitoringComponents-v1.0.pdf

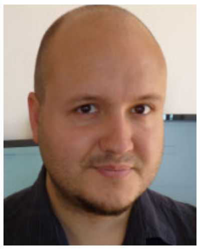
[27] *OpenStack STEIN*. Accessed: Jun. 26, 2020. [Online]. Available: https://www.openstack.org/software/stein/

[28] *Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV*, document ETSI GR NFV 003 V1.5.1, Jan. 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/NFV/001_099/003/01.05.01_60/gr_NFV003v010501p.pdf

[29] (2020). *Availability Zones*. [Online]. Available: https://docs.openstack.org/nova/latest/admin/availability-zones.html

[30] *Open vSwitch*. Accessed: Jun. 26, 2020. [Online]. Available: https://www.openvswitch.org/

ANA HERMOSILLA received the B.Sc. and M.Sc. degrees in computer engineering from the University of Murcia, where she is currently pursuing the Ph.D. degree with the Department of Information and Communication Engineering. She is also a Researcher with the Department of Information and Communication Engineering, University of Murcia, where she is involved in the H2020 Inspire5GPlus project. Her research interests include NFV/SDN, 5G, trust management, and virtualization platforms.



ALEJANDRO MOLINA ZARCA received the M.Sc. and master's degrees in computer science from the University of Murcia, Spain, in 2012 and 2017, respectively, where he is currently pursuing the Ph.D. degree. He is also a Researcher with the Department of Information and Communication Engineering, University of Murcia. He has worked on different H2020 European research projects, such as ANASTACIA and Inspire5Gplus. He has authored several articles. His research interests include 5G, the IoT security, UAVs, network virtualization, and softwarization.

JORGE BERNAL BERNABE received the M.Sc., master's, and Ph.D. degrees in computer science from the University of Murcia. He is currently a Postdoctoral Researcher with the University of Murcia funded by the AXA Research Fund. He has published over 50 articles in international conferences and journals. During the last years, he has been working in several European research projects, such as DESEREC, Semiramis, Inter-Trust, SocIoTal, ARIES, OLYMPUS, ANASTACIA, and CyberSec4Europe. His scientific activity is mainly devoted to the security, trust, and privacy management in distributed systems and the IoT. He has been involved in the scientific committee of numerous conferences and served as a Reviewer for multiple journals.

JORDI ORTIZ received the B.S., M.Sc., and Ph.D. degrees in computer science from the University of Murcia, Spain, in 2008, 2009, and 2018, respectively. Since 2007, he has been a full-time Researcher associated with research projects with International Projection, such as DAIDALOS, SCALNET, SMARTFIRE, OPENLAB, GEANT 3 4, STORK2, ANASTACIA, SURROGATES, MIGRATE, or Inspire5GPlus, among others. He is currently a part-time Professor with the University of Murcia. He has published international articles and serves in the Technical Program Committee in some conferences and journals. His main research interests include identity federation, video streaming, SDN networks, NFV, network orchestration, and the IoT networks.

ANTONIO SKARMETA (Member, IEEE) received the B.S. degree (Hons.) from the University of Murcia, Spain, the M.S. degree from the University of Granada, and the Ph.D. degree from the University of Murcia, all in computer science. He has been the Head of the research group ANTS, since its creation on 1995. Since 2009, he has been a Full Professor with the University of Murcia. He has worked on different research projects in the national and international area in the networking, security, and the IoT area, such as Euro6IX, ENABLE, DAIDALOS, SWIFT, SEMIRAMIS, SMARTIE, SOCIOTAL, IoT6 ANASTACIA, and CyberSec4Europe. He has published over 200 international articles and being a member of several program committees. His main research interests include the integration of security services, identity, the IoT, and smart cities.