A First Step Towards Security Extension for NFV Orchestrator

Montida Pattaranantakul[1,2], Yuchia Tseng[3], Ruan He[4], Zonghua Zhang[1], Ahmed Meddahi[1]
[1]Institut Mines-Télécom/TELECOM SudParis, Évry, France
[2]IMT Lille Douai, Institut Mines-Télécom, Lille, France
[3]Paris Descartes University, Paris, France
[4]Orange Labs, Châtillon, France

ABSTRACT

Network Functions Virtualization (NFV) has recently emerged as one of the new networking paradigms to significantly change the way that the networks and services are deployed, managed, and operated. One of the major advantages of NFV is to reduce hardware cost, meanwhile increasing service agility and scalability. Recently, there are many platforms for NFV management and orchestration (MANO) are available, however few of them contains dedicated modules or components for security management. This paper is intended to study the feasibility of extending the current NFV orchestrator to have the capability of managing security mechanisms. To do that, we propose a security extension module based on TOSCA data model which is commonly used by NFV MANO architecture. We then develop an access control use case to illustrate the usage of our proposed security extension. Specifically, we integrate the security extension into the Moon framework, which can automatically verify security attributes, generate access control policies, and further enforce the policies through the underlying infrastructure according to the high-level security policies. The preliminary results show that our security extension can work together with the NFV orchestrator to enable fine-grained access control to protect resources and services.

Keywords

Network Functions Virtualization (NFV), data model, service orchestration, security management

1. INTRODUCTION

NFV has recently emerged as one of the novel networking paradigms that offers a variety of network functions to be rapidly and dynamically deployed in agile, scalable, and adaptive ways [9, 14, 17]. Also, the Capital Expenditure (CapEx) and Operational Expenditure (OpEX) are significantly reduced by using NFV. The nature of NFV allows to create and deploy new services more quickly, by decoupling network functions from proprietary hardware appliances, and implement them using software based approach. To do that, orchestration module is required to dynamically coordinate the underlying infrastructure resources. Therefore, NFV orchestration plays an important role in maintaining full lifecylce management of Virtual Network Functions (VNFs) and end-to-end network services, such as instantiation, configuration, termination, scale-in/out, resource and policy management, performance measurement, event correlation, validation and authorization of resource requests.

Although NFV has generated significant interests in the market place and academia, security concerns remain to be significant barrier to the wide adoption of NFV [20, 8]. In addition to the fact that the operators have to deal with non-trivial service complexities, spanning from configuration and maintenance to management and orchestration, in order to achieve NFV goals [13, 7, 18, 6]. While the European Telecommunications Standard Institute (ETSI) has published a document about NFV Management and Orchestration (NFV MANO) [10], a detailed description about NFV specification, service deployment, and security management is still missing. In particular, two critical issues are highlighted.

First, as a matter of fact, the current research activities are mainly focused on how to migrate network functions from dedicated hardware to virtualization environment, and how to manage and orchestrate the virtualized network functions on demand. The core concept of deploying and operating network services in NFV is based on service templates. These templates define the attributes and requirements to allocate resources and initiate network services, so that NFV orchestrator can use these templates as reference for further deployment. Therefore, the service templates must be modeled clearly and should be defined based on model-driven approach, which allow operators to obtain a clear view about the structure of network services, nodes, and links to be deployed. An appropriate data model can also facilitate the operators to gain visibility and controllability over the workloads, the deployed VNFs, and the underlying infrastructure resources. It is worth noting, however, many existing NFV frameworks from industrial sectors like [2, 5] do not support model-driven NFV orchestration, neither TOSCA data model standard [22].

Second, along with the orchestration of network functions and services, appropriate security mechanisms are expected to be put in place, which however is believed to be a ma-

SDN-NFV Sec'17, March 22-24, 2017, Scottsdale, AZ, USA

http://dx.doi.org/10.1145/3040992.3040995.

jor challenge considering the complexity of the processes of NFV management and orchestration. Moreover, the current NFV orchestrator is mainly based on ETSI NFV reference architecture, which aims at managing VNF lifecycle and orchestrate infrastructure resources for supporting end-to-end network services. That says, a typical NFV orchestrator does not necessarily contain the capability of managing security mechanisms, even though some of them reserve specific fields for specifying security attributes. To date, a best practice recommendation for implementing NFV based security management and orchestration has not yet appeared. Nevertheless, many existing NFV orchestration platforms especially the open source ones are based on TOSCA data model, none of them has been defined from security perspective.

To address the aforementioned issues, our contribution in this paper is three-fold.

- First, we conduct a comprehensive study on the existing NFV orchestration platforms, especially the open source ones, with a purpose to abstract their data models, analyze their operations and workflows, and eventually identify the capability of managing security mechanisms.
- Second, based on the understanding of NFV orchestration data models, we extend the ones using standard TOSCA data model with security aspect, by incorporating security attributes, enabling administrators to define security policy and allowing users to specify security attributes for each VM/VNF.
- Third, we develop a TOSCA based security extension by employing a well-developed security policy engine (Moon framework that is developed for access control) to achieve the capability of managing security mechanisms. Our extension can automatically verify security attributes specified in the extended TOSCA data model, generate corresponding access control policies, and enforce them to protect the NFV infrastructure.

The rest of paper is organized as follows. Section 2 discusses related work. Section 3 presents the architecture of our security extension for NFV orchestrator. In Section 4, we develop a use case for illustrating the role of our security extension for access control, as a first step towards extending security capability in NFV orchestrator. Section 5 reports our implementation, along with the preliminary results. This paper is concluded in Section 6.

2. RELATED WORK

Nowadays, there are some available platforms for NFV management and orchestration, most of which are built and aligned with ETSI NFV MANO specification [10] for managing and orchestrating end-to-end network services and cloud infrastructure resources. Some existing platforms have been developed by industrial sectors like [2, 5], which are implemented as non-model driven based approach. Meanwhile, some of them are developed under open source platforms as model-driven NFV orchestration using TOSCA standard. This paper is specially focused based on this matter. For examples, OpenMANO [3] is a framework that provides practical implementation of management and orchestration using ETSI standard, enabling the creation of complex virtual network scenarios. Also, Tacker [21], a TOSCA-based NFV orchestrator, has emerged as official OpenStack project to build NFV orchestration software supporting both VNF Manager (VNFM) and NFV Orchestrator (NFVO). Heat [15], the most popular orchestrator in OpenStack, which functioned as Tacker. It launches multiple services based on Heat Orchestration Template (HOT). OpenBaton [4] is another open source platform which provides a complete environment for NFV orchestration, focusing on the basic orchestration of NFVO and VNFM that enables VNF deployment on the top of multiple cloud infrastructures.

After a careful study of the data models of these orchestration frameworks with respect to several core functions, e.g., network topology, node specifications, service deployment, we found that they lack a dedicated module or component which can provide holistic security management, especially in the NFV orchestrator. For example, the service templates defined in OpenMANO, Tacker, and OpenBaton generally describe network topology, node specifications, and link relationships between the nodes. One of the key observations is that these typical TOSCA based service templates do not well-define their data model from security perspective, nor clearly specify security attributes for each VM/VNF. This allows unauthorized requests to gain access to the resources and is eligible to use the services. As a result, the operators may lose control over the deployed VMs/VNFs. More seriously, it may allow attackers to control the infrastructure resources and compromised VNFs, leading to unauthorized configuration and theft of services. In addition, despite it is possible to define security groups in Heat, it still lacks dynamic and centralized control with high level security policy specification. It is therefore extremely important to develop a dedicated module of security extension based on the existing NFV orchestrator that allows security functions to be dynamically, centrally, and holistically managed and orchestrated, providing seamless integration for protecting the underlying network assets. To the best of our knowledge, none of the available NFV orchestration platforms provide such this security capability.

3. EXTENDING NFV ORCHESTRATOR FOR SECURITY MANAGEMENT

As aforementioned, in the current ETSI documents about NFV security [11, 12], specific implementations of security management based on NFV orchestration are not yet available. Although the author of [16], pointed out security concerns in NFV orchestration, a detailed data model about security management and related use cases are missing. In this Section, based on TOSCA data model, we propose a security orchestrator as an extension to the NFV orchestrator.

3.1 Design Architecture

As our purpose is to extend the current NFV orchestrator to enable dynamic security management and orchestration, the first yet essential step is to examine the data models that are commonly used for NFV service management and orchestration. We finally select TOSCA as the basic data model considering its popularity in today's NFV orchestrators, and extend it for our security purpose. The TOSCA based security extension integrates Moon framework [19], a security policy engine, to enforce the security policies over the deployed infrastructure resources and network services. Fig. 1 presents a high level view of our security extension,

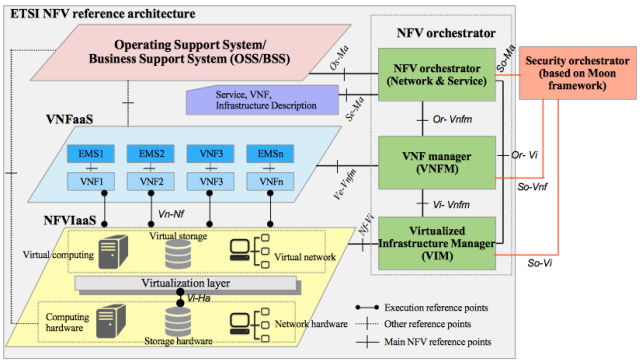

Figure 1: High level architecture of security extension for NFV orchestrator aligned with ETSI NFV MANO specification

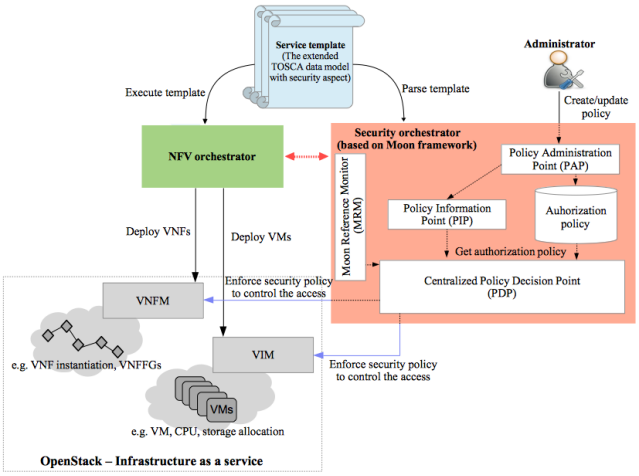

Figure 2: The detailed model of security extension for NFV orchestrator

which complies with ETSI NFV MANO architecture and works along with NFV orchestrator to provide automatic security control, verify security attributes, and enforce security policies.

3.2 Design Principle

The development of security extension contains two major steps. First, we extend TOSCA data model (simply known as service template) to use it for specifying high-level security policy and achieving fine-grained access control. For example, network nodes can be specified with certain security attributes and a group of security policies, so that only authorized requests that belong to the same group of security policy can gain access. Second, security functions such as access control are provided on demand, ensuring that the deployed resources/services are protected from unauthorized access based on the specified security policy.

For better illustration, Fig. 2 shows the detailed model of security extension, which contains three main components,

- *TOSCA data model (or service template):* which defines data structures for NFV orchestration by using YAML [1]. It describes the deployment, operational behavior requirement, and link connection for each network service in NFV. Its structure contains nodes (e.g, VNFs) and relationships (e.g, Virtual Links (VLs)) defined together to deliver end-to-end lifecycle operations of network services. This template is treated as a YAML file, and stored in a catalog during the service on-boarding process for service instantiation. An example TOSCA data model is shown in Fig 4: the left side of the figure represents a general structure, and the right side is about deploying VNFs. Our contribution here is to extend the typical TOSCA data model with security policy specification, in which all nodes presented in the template are defined with their security attributes or a group of security policies (highlighted in grey).

- *NFV orchestrator:* which maintains the lifecycle management of infrastructure resources and network services. NFV orchestrator uses the extended TOSCA data model (service template together with security policy and attribute specification) to allocate infrastructure resources (e.g, VMs, VLs), and instantiate virtual components (e.g, VNFs) at runtime based on the predefined relationship between these components. As indicated in Fig. 2, the NFV orchestrator interacts with Virtualized Infrastructure Manager (VIM) to allocate VMs, and interacts with VNF Manager (VNFM) to instantiate the VNFs. In particular, the extended TOSCA data model provides a complete environment to startup the service, such as the properties and parameters with respect to resource allocation, software installation, service deployment, and network configuration. According to the data model, NFV orchestrator allocates infrastructure resources, orchestrates the services, and deals with several processes during their lifecycle.

- *Security extension:* which works together with NFV orchestrator to provide monitoring, control resource access, and enforce security policy to the deployed VMs/VNFs. Specifically, it uses the extended TOSCA data model to obtain security attributes of each VM/VNF, creates security policy, and enforces access control. To enforce the security policies, security orchestrator works with Moon framework - a security policy engine with attribute-based policy specification. If an authorized entity (*subject*) meets with the policy rules defined in a security policy, it is then allowed to access (*action*) the requested entity (*object*). Otherwise, the request is rejected. For example, if the deployed VNF is specified with a group of security policy, then only legitimate requests in the same security group are allowed to access to it. Other requests that are not from the same security group will be rejected. In doing so, the access can be dynamically controlled in a fine-grained way, and the deployed VMs/VNFs are therefore well protected.

3.3 Moon framework

Our security orchestrator is based on Moon framework [19], which is designed to monitor, control, and manage VMs/VNFs over the OpenStack infrastructure. The core part of current version Moon framework relies on an access control model, which specifies high-level security policies to determine authorization requests. As shown in Fig. 2, Moon framework

based security orchestrator contains four major components,

- *Policy Administration Point (PAP):* which allows administrators to create or update authorization policies.
- *Authorization policy:* defines the template of access control model. It contains a set of rules that are used to determine whether a request is authorized.
- *Policy Information Point (PIP):* holds policy and entity-data assignment table that defines a many-to-many relationship between attributes and entities, based on *subject*, *object*, and *action*.
- *Policy Decision Point (PDP):* which receives authorization requests from Moon Reference Monitor (MRM), validates the requests based on information gathering from the PIP and the authorization policy, and finally enforces security policy.

4. USE CASE: ACCESS CONTROL

Although security extension aims to have the capability of managing various security mechanisms and provides security as a service (e.g., access control, security monitoring, network isolation, and data protection), in this Section we develop a realistic use case of access control as one of basic security functions to illustrate the application of our security extension.

4.1 General description

As shown in Fig. 2, the orchestration processes starts with NFV orchestrator, which parses the extended TOSCA data model from service on-boarding catalog, and launches infrastructure resources (e.g, VMs) and services (e.g, VNFs) accordingly. In the meantime, the extended TOSCA data model is passed to security extension, which performs two main functions: (1) generates access control policy based on the security attributes of each VNF defined in the extended TOSCA data model; (2) implement the access control policies.

Specifically, in order to generate access control policy, the security extension needs to parse the node-template part of the extended TOSCA data model to get security attributes of each VM/VNF, and invokes the API of Moon to assign these security attributes to each subject and object. During the execution phase, access requests to the VM/VNF in OpenStack infrastructure will be redirected to the security extension, which verifies whether the VM/VNF of concern has been specified with certain security attributes or a group of security policy. If so, the security extension uses the specified security policy to control the access. Otherwise, no security control will be enforced for the deployed VM/VNF. In the rest of this Section, we give a more detailed description and comparison between the two cases.

4.2 Scenario 1: Data model without security policy specification

In this scenario, we define TOSCA data model without any security policy specification. Thus, NFV orchestrator simply reads the descriptions of VNFs and deploys them accordingly. That says, security extension will not perform any access control, hence the predefined TOSCA data model does not specify any security policy for each VM/VNF. As a result, any access requests will be granted to access the

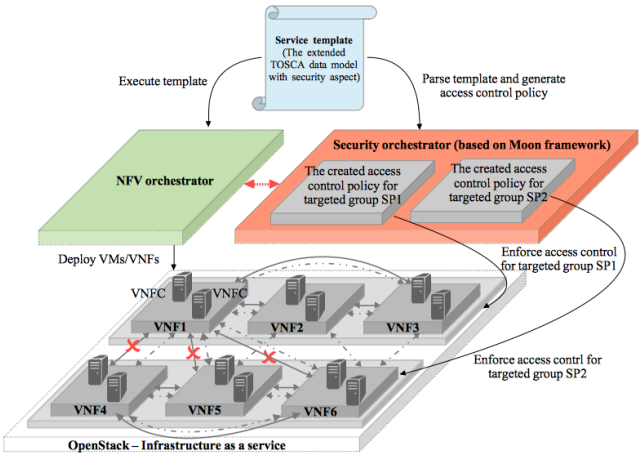

Figure 3: Data model of security extension for NFV orchestrator, with "security policy specification" and with "no security policy specification" scenarios, respectively represented as dotted and solid lines

infrastructure resources. An example scenario is given in Fig. 3 with dotted line. Clearly, this scenario does not ensure a trustworthy environment, and the fine-grained access control to the deployed resources/services is not available.

4.3 Scenario 2: Data model with security policy specification

In this scenario, TOSCA data model is extended to include security policy specification. Like the first scenario, NFV orchestrator deploys VM/VNF according to the template description of the extended TOSCA data model. Moreover, the security extension invokes the API of Moon to generate an access control policy based on security attributes extracted from the extended TOSCA data model. The predefined security attributes of each VM/VNF are assigned to the *subject* and *object* according to the specification of attribute-based access control policy. When a VM/VNF instance is executed, access request will be redirected to the security extension, which then enforce the corresponding access control policies.

This scenario is illustrated in Fig. 4, in which the data model defines six different types of services, respectively running on VNF1 – VNF6. In particular, each VNF consists of multiple VNF Components (VNFCs), provided by two service providers. Thus, two groups of security policy (highlighted in grey) are specified as follows, (1) SP1 provides VNF1, VNF2, and VNF3, and, (2) SP2 provides VNF4, VNF5, and VNF6. With such security policy enforced, VNF1 can gain access to VNF2 or VNF3, and vice versa, while none of the VNFs provided by SP1 can access to VNF4/VNF5/VNF6. Similarly, VNF4 can access the VNF5 or VNF6 and vice versa, while it can not access to VNF1/VNF2/VNF3.

When the extended TOSCA data model is passed to the security extension, a verification process is initiated to check whether the deployed VNFs have been specified any security attributes. Then the security extension invokes the API of Moon framework to generate appropriate access control policy by assigning security attributes of each VNF with *subject*, *object*, and *action*. The results are shown in Fig. 3 with solid

Figure 4: The extended TOSCA data model with security policy specification (highlighted in grey)

line, which indicate that the deployed VNFs are permitted to access to each other if they have the same group of security policy, while the cross group requests are denied. The example clearly demonstrates that thanks to the security extension, the operators are able to enforce fine-grained access control to their assets by simply defining security attributes in the VNFs.

5. IMPLEMENTATION

In our implementation, we use Heat [15], a core part of OpenStack platform, as a service orchestrator to deploy infrastructure resources, and launch multiple applications and services over OpenStack infrastructure. Although Heat itself provides an option to define security attributes, it does not support dynamic and centralized access control. In addition, we leverage Moon policy engine for centralized access control specified with high level security policies. Specifically, three functional modules are implemented for our security orchestration.

- *MoonClient:* the user interface that is used to interact with Moon via CLI. The communication with MoonClient is done via Python subprocess module.
- *Parameter parser:* which is designed for two purposes. First, it extracts the description of VNF nodes, translates the security policy defined in the extended TOSCA data model, and parses these parameters to *MoonClient* for generating high-level security policies in Moon policy engine. Second, it fills all the information in the extended TOSCA data model and translates them to HOT (Heat Orchestration Template) for being interpreted by Heat, because the information contains in the TOSCA fields are not natively interpretable by Heat. Based on the previous example shown in Fig 4, the extracted data that is translated by the parser is shown in Table 1.
- *Network hook:* which works as a lightweight proxy server in each VNF by coordinating with Moon. It is written in Python and maintains the connection between *subject* and *object*. By default, this network hook disables all the accesses to the *object* by setting the rule in iptables as `iptables -A INPUT -j DROP`. The network hook continues listening to an assigned port, and it checks the permission based on the source IP with Moon as soon as it receives the request. If the request is authorized, this network hook inserts a temporary rule with specific source IP in the iptables, e.g, `iptables -I INPUT 1 -s 10.0.0.116 -j ACCEPT`, so that the *subject* whose IP address is `10.0.0.116` can start to communicate with the requested *object*.

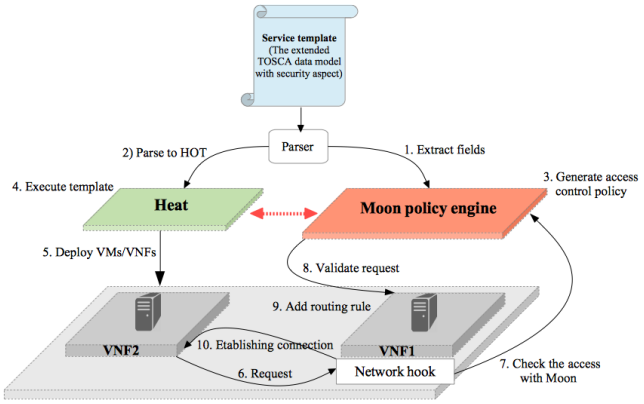

Figure 5: The operational flow of security extension for access control

Table 1: The extracted data based on the extended TOSCA data model

Nodes	VNF1, IP = 10.0.0.122
	VNF2, IP = 10.0.0.116
	VNF3, IP = 10.0.0.104
	VNF4, IP = 10.0.0.123
	VNF5, IP = 10.0.0.113
	VNF6, IP = 10.0.0.121
Security Policy:	SP1: [VNF1, VNF2, VNF3]
	- Subject: VNF1, Object: VNF2, Access=OK
	- Subject: VNF2, Object: VNF1, Access=OK
	- Subject: VNF1, Object: VNF3, Access=OK
	- Subject: VNF3, Object: VNF1, Access=OK
	- Subject: VNF2, Object: VNF3, Access=OK
	- Subject: VNF3, Object: VNF2, Access=OK
	SP2: [VNF4, VNF5, VNF6]
	- Subject: VNF4, Object: VNF5, Access=OK
	- Subject: VNF5, Object: VNF4, Access=OK
	- Subject: VNF4, Object: VNF6, Access=OK
	- Subject: VNF6, Object: VNF4, Access=OK
	- Subject: VNF5, Object: VNF6, Access=OK
	- Subject: VNF6, Object: VNF5, Access=OK

Fig 5 illustrates the operational flows of security extension for NFV orchestrator based access control. Once the VNF/VNFC receives a request, the corresponding network hook sends an authorization request to Moon for validation. The network hook will add a new rule into its `iptables` to allow this connection. If the network hook does not receive any request during a predefined period of time, the corresponding rule will be removed from its `iptables`, while the

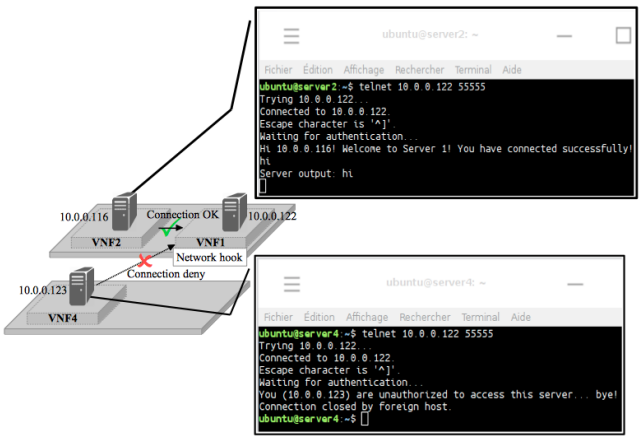

Figure 6: A result of testing network connection with Telnet

connection between *subject* and *object* will be disconnected. The testing result from implementation is presented in Fig 6. As shown in the two telnet consoles, VNF2 successfully establishes network connection with VNF1, while the connection requests from VNF4 to VNF1 are denied. This well illustrates the security policy that the network connections can be established only if the *subject* and *object* belong to the same group of security policy.

6. CONCLUSION

Despite the advantages of NFV based management and orchestration that enable network resources and services to be managed and provided in agile, dynamic, scalable, and adaptive ways, the benefits to security management remain unclear. We have examined a set of available NFV orchestrators and found that none of them provides dedicated modules or components for security management purpose. This observation motivates us to propose and develop a security extension for NFV orchestrator which are based on TOSCA data model. We implemented the security extension based on our Moon framework, which is an available platform for dynamic access control to cloud resources. As a result, the high-level access policies can be automatically enforced to protect underlying NFV infrastructure. To illustrate the usage of our proposed security extension model, we developed a realistic use case of access control with two different scenarios: one data model without security policy specification, and another one with predefined security policy specification. The preliminary result shows that it is feasible and effective to create fine-grained access control rules, e.g., blocking illegal access, grouping or isolating the protected resources, thanks to our security extension. For our future work, we will specifically evaluate the performance impact of our proposed model, further analyzing key factors which influence communication overhead and latency. Also, we will extend the capability of our security extension to have other security functions (e.g., security monitoring, network isolation, and data protection) for achieving a holistic security management orchestration.

7. REFERENCES

[1] YAML Ain't Markup Langauge. `http://yaml.org/`, May 2001.

[2] Cloudnfv. `http://www.cloudnfv.com/`, Jan 2014.

[3] OpenMANO. `https://github.com/nfvlabs/openmano`, Mar 2015.

[4] OpenBaton. `https://openbaton.github.io/`, Jan 2016.

[5] Alcatel-Lucent. Cloudband. `http://www.tmcnet.com/redir/?u=1010632`, 2014.

[6] Z. Bronstein and E. Shraga. NFV Virtualisation of the Home Environment. In *CCNC' 14*, pages 899–904, Jan 2014.

[7] J. Carapinha et al. Network Virtualization - Opportunities and Challenges for Operators. In *FIS'10*, pages 138–147, 2010.

[8] A. Dutta. Security Challenges and Opportunities in SDN/NFV Networks. `http://www.isr.umd.edu/sites/default/files/Dutta.pdf`, Nov 2016.

[9] ETSI. Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for Action, Oct 2012.

[10] ETSI. Network Functions Virtualization (NFV); Management and Orchestration, Dec 2014.

[11] ETSI. Network Functions Virtualization (NFV): NFV Security, Security and Trust Guidance, Dec 2014.

[12] ETSI. Network Functions Virtualization (NFV): Security Report, Security Management and Monitoring for NFV, Nov 2015.

[13] B. Han et al. Network Function Virtualization: Challenges and Opportunities for Innovations. *IEEE Communications Magazine*, 53(2):90–97, Feb 2015.

[14] H. Hawilo et al. NFV: State of the Art, Challenges, and Implementation in Next Generation Mobile Networks (vEPC). *IEEE Network*, 28(6):18–26, Nov 2014.

[15] Heat. Heat - Openstack Orchestration. `https://wiki.openstack.org/wiki/Heat`, May 2014.

[16] B. Jaeger. Security Orchestrator: Introducing a Security Orchestrator in the Context of the ETSI NFV Reference Architecture. In *IEEE TrustCom' 15*, pages 1255–1260, Aug 2015.

[17] R. Jain and S. Paul. Network Virtualization and Software Defined Networking for Cloud Computing: a Survey. *IEEE Communications Magazine*, 51(11):24–31, November 2013.

[18] R. Mijumbi et al. Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2016.

[19] OPNFV. Moon - Security Management Module. `https://wiki.opnfv.org/display/moon/Moon`, Apr 2016.

[20] M. Pattaranantakul et al. SecMANO: Towards Network Functions Virtualization (NFV) based Security MANagement and Orchestration. In *IEEE TrustCom' 16*, Aug 2016.

[21] Tacker. Tacker - OpenStack NFV Orchestration. `https://wiki.openstack.org/wiki/Tacker`, 2013.

[22] TOSCA. TOSCA Simple Profile for Network Functions Virtualization (NFV version 1.0), Mar 2016.